Universidade de Aveiro Departamento de Engenharia Mecânica
2013

PROCÓPIO SILVEIRA STEIN

NAVEGAÇÃO EM AMBIENTES DINÂMICOS TIRANDO PARTIDO DE AGENTES MÓVEIS

NAVIGATION IN DYNAMIC ENVIRONMENTS TAKING ADVANTAGE OF MOVING AGENTS

Universidade de Aveiro Departamento de Engenharia Mecânica
2013

PROCÓPIO SILVEIRA STEIN

NAVEGAÇÃO EM AMBIENTES DINÂMICOS TIRANDO PARTIDO DE AGENTES MÓVEIS

NAVIGATION IN DYNAMIC ENVIRONMENTS TAKING ADVANTAGE OF MOVING AGENTS

Tese apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Doutor em Engenharia Mecânica, realizada sob a orientação científica de Vítor Manuel Ferreira dos Santos, Professor Associado do Departamento de Engenharia Mecânica da Universidade de Aveiro e de Anne Spalanzani, Professora Associada da Universidade Pierre-Mendès-France e Investigadora no Instituto de Pesquisas INRIA

to my grandparents

o júri / the jury

presidente / president	**Prof. Doutor José Carlos da Silva Neves** professor catedrático da Universidade de Aveiro
vogais / committee	**Prof. Doutor Luis Merino** professor associado da Universidad Pablo de Olavide, Espanha
	Prof. Doutor Luís Seabra Lopes professor associado da Universidade de Aveiro
	Prof. Doutor Paulo José Cerqueira Gomes da Costa professor auxiliar da Universidade do Porto
	Prof. Doutor Jorge Augusto Fernandes Ferreira professor auxiliar da Universidade de Aveiro
	Profª. Doutora Anne Spalanzani maître de conférences da Université Pierre-Mendès-France, França (coorientadora)
	Prof. Doutor Vítor Manuel Ferreira dos Santos professor associado da Universidade de Aveiro (orientador)

agradecimentos

O caminho para a conclusão de uma Tese é longo e difícil. Felizmente, eu pude contar com muitas pessoas especiais que ajudaram-me nesta jornada. Eu gostaria de agradecer a Vítor Santos, pela sua ética e imensa dedicação na orientação desta tese, e pelo seu constante apoio e motivação em tempos difíceis. Também quero agradecer a Anne Spalanzani por receber-me em sua equipa, pela sua orientação, discussões e por ajudar-me a encontrar meu caminho no campo da robótica social.

À Christian Laugier, por haver me convidado a um estágio no grupo de pesquisas e-Motion, o que fez toda a diferença na minha carreira, e permitiu uma incrível experiência de vida em outro país.

Ao longo destes quatro anos, eu trabalhei com muitas pessoas e também gostaria de reconhecer que elas contribuíram de uma forma ou de outra para o meu trabalho. Um agradecimento especial aos meus colegas de laboratório de Portugal, Miguel Oliveira, Ricardo Pascoal, Jorge Almeida, Marco Santos e da França, Arturo Escobedo, Jorge Ríos, Mathias Perrolaz, Dizan Vasquez, Chiara Troiani e Stéphanie Lefèvre, pelo agradável tempo em que passamos juntos.

Também gostaria de agradecer aos professores Dr. Luis Merino e Dr. Dominique Vaufreydaz, que se disponibilizaram como relatores europeus deste documento.

À minha mãe, pai, irmã e irmão, que sempre apoiaram minhas decisões e com os quais sempre pude contar. E à minha família portuguesa, Ernesto e Né, que sempre estiveram prontos à ajudar.

À minha esposa Ariana, que sempre esteve ao meu lado nesta longa jornada.

Finalmente, ao apoio financeiro da FCT e do FSE no âmbito do III Quadro Comunitário de Apoio, que financiou esta pesquisa através da bolsa $SFRH/BD/46604/2008$, e ao Projeto PAL (Personal Assisted Living) e INRIA, que também proporcionaram suporte financeiro.

acknowledgements

The path to the creation of a PhD Thesis is a long and difficult one. Fortunately, I could count with many special persons that helped me along this journey. I would like to thank Vítor Santos, for his work ethics and great dedication in the supervision of this Thesis and for his constant encouragement and support throughout difficult times. To Anne Spalanzani for receiving me in her team, for her supervision, insightful discussions and for helping me to find my way in the social robotics field.

To Christian Laugier, for inviting me to join the e-Motion research group, which made all the difference in my career, and allowed a great life experience in another country.

Throughout these four years, I have worked with many persons, and I would like to acknowledge that they contributed in one way or another to my work. Special thanks to my labmates, from Portugal, Miguel Oliveira, Ricardo Pascoal, Jorge Almeida, Marco Santos and from France, Arturo Escobedo, Jorge Ríos, Mathias Perrolaz, Dizan Vasquez, for all the enjoyable time spent together.

I also would like to thank professors Dr. Luis Merino and Dr. Dominique Vaufreydaz, that were promptly available as European reporters of this document.

To my mother, father, sister and brother, who always supported my decisions and with whom I could always count with. And to my Portuguese family, Ernesto and Né, who were always ready to help.

To my wife Ariana, who always stood by my side along this journey.

Finally, to Fundação para a Ciência e Tecnologia (FCT) that financed this reasearch with the scholarship $SFRH/BD/46604/2008$ and to the PAL Project (Personal Assisted Living) and INRIA, which also provided financial support.

palavras-chave

navigation in dynamic environments, human-aware navigation, social robotics, leader selection, leader following, machine learning

resumo

Esta tese propõe uma forma diferente de navegação de robôs em ambientes dinâmicos, onde o robô tira partido do movimento de pedestres, com o objetivo de melhorar as suas capacidades de navegação.
A ideia principal é que, ao invés de tratar as pessoas como obstáculos dinâmicos que devem ser evitados, elas devem ser tratadas como agentes especiais com conhecimento avançado em navegação em ambientes dinâmicos. Para se beneficiar do movimento de pedestres, este trabalho propõe que um robô os selecione e siga, de modo que possa mover-se por caminhos ótimos, desviar-se de obstáculos não detetados, melhorar a navegação em ambientes densamente populados e aumentar a sua aceitação por outros humanos.
Para atingir estes objetivos, novos métodos são desenvolvidos na área da seleção de líderes, onde duas técnicas são exploradas. A primeira usa métodos de previsão de movimento, enquanto a segunda usa técnicas de aprendizagem por máquina, para avaliar a qualidade de candidatos à líder, onde o treino é feito com exemplos reais. Os métodos de seleção de líder são integrados com algoritmos de planeamento de movimento e experiências são realizadas para validar as técnicas propostas.

keywords navigation in dynamic environments, human-aware navigation, social robotics, leader selection, leader following, machine learning

abstract

This thesis proposes a different form of robotic navigation in dynamic environments, where the robot takes advantage of the motion of pedestrians, in order to improve its own navigation capabilities.
The main idea is that, instead of treating persons as dynamic obstacles that should be avoided, they should be treated as special agents with an expert knowledge of navigating in dynamic scenarios. To benefit from the motion of pedestrians, this work proposes that the robot selects and follows them, so it can move along optimal paths, deviate from undetected obstacles, improve navigation in densely populated areas and increase its acceptance by other humans.
To accomplish this proposition, novel approaches are developed in the area of leader selection, where two methods are explored. The first uses motion prediction approaches while the second uses a machine learning method, to evaluate the leader quality of subjects, which is trained with real examples. Finally, the leader selection methods are integrated with motion planning algorithms and experiments are conducted in order to validate the proposed techniques.

Contents

List of Figures

Acronyms

ANN Artificial Neural Network

CBR Case Based Reasoning

CCA Cooperative Collision Avoidance

DW Dynamic Window

FRP Freezing Robot Problem

GP Gaussian Process

GHMM Growing Hidden Markov Model

GNG Growing Neural Gas

GPRF Gaussian Process Regression Flow

HMM Hidden Markov Model

HRI Human Robot Interaction

IGP Interacting Gaussian Processes

IRL Inverse Reinforcement Learning

KF Kalman Filter

NLVO Non-Linear Velocity Obstacle

PCA Principal Component Analysis

PMP Partial Motion Planning

PRM Probabilistic RoadMap

PVO Probabilistic Velocity Obstacles

ROS Robot Operating System

RiskRRT	Risk Rapidly Exploring Random Tree
RRT	Rapidly Exploring Random Tree
RVO	Reciprocal Velocity Obstacle
SF	Social Filter
SFM	Social Forces Model
SVM	Support Vector Machine
VO	Velocity Obstacle

Chapter 1

Introduction

Among several challenging issues in the robotics field, one of the key problems, still subject of intense investigation, is how to create motion plans for robots that share space with humans, in dynamic environments. This is a relevant problem, as robots are already part of our life. An important group of robots are the ones that provide everyday services to humans, also called service robots, because they are able to directly interact with humans and therefore have a high potential of directly improving our quality of life.

Examples of service robots are floor sweeping robots, that help on household tasks, as iRobot Roomba and Neato Robotics XV-12, interactive museum and exposition tour guide robots (Rhino, Minerva, Enon), avatars robots (TOURBOT) and also robots that help impaired persons (nursebot, wheelchairs). Some of these robots can be seen in Figure 1.1.

As service robots share the same environment with humans, they must face a difficult task, which is how to navigate in those environments, because the state of the environment is constantly changing and humans move and interact according to complex social rules.

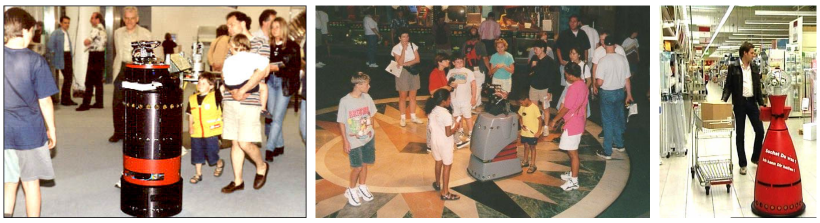
Besides that, in populated environments, the dynamism of the agents pose several difficult issues for robotic motion planning: sensor measurements are likely to be accurate only for short time periods; static features, that could guide a navigation algorithm, might not be detected; moving agents may move in unexpected ways; and algorithms have to be fast enough to provide solutions on-line.

1.1 Previous Approaches

The most relevant developments in this area approached the problem of navigation in dynamic environments with predictive models. In these models, the motion of agents and objects are observed and, according to different techniques, their future states are predicted. Within these approaches, everything that moves in those environments is normally regarded as **moving obstacles** to the robot movement and, therefore, should be **avoided**, since moving entities may be constantly interfering with planned trajectories. So the motion planning

(a)

(b)



(c)

Figure 1.1: Examples of service robots that must interact with humans: (a) health aid robots, the MAid wheelchair and the GuideCane (Prassler et al., 2001; Borenstein and Ulrich, 1997). (b) cleaner robots: roomba and neato robotics XV-11 (iRobot, 2013; Robotics, 2013). (c) guide robots: RHINO, Minerva and Toomas (Burgard et al., 1999; Thrun et al., 1999; Gross et al., 2009).

algorithms create plans where, based on predictions, the robot will avoid all the moving "obstacles".

Besides that, very few approaches actually take into account the reaction that the robot's motion may cause in other moving agents (Snape et al., 2009; van den Berg et al., 2008a; Kluge and Prassler, 2004; Abe and Yoshiki, 2001). This means that most techniques assume that either the moving objects are passive and will not change their trajectories or that agents share the same algorithm, so they will avoid collision in the same way as the robot will.

Even in the cases where the reaction of others are taken into account, the agents try to independently reach their goal, treating other moving agents as adversaries. These types of approaches can be seen as **egocentric** solutions to the dynamic environment navigation problem: the robot has an objective that has to be accomplished and creates a plan, and everything that interferes with that plan are **obstacles** that should be avoided, and no benefit can come from them.

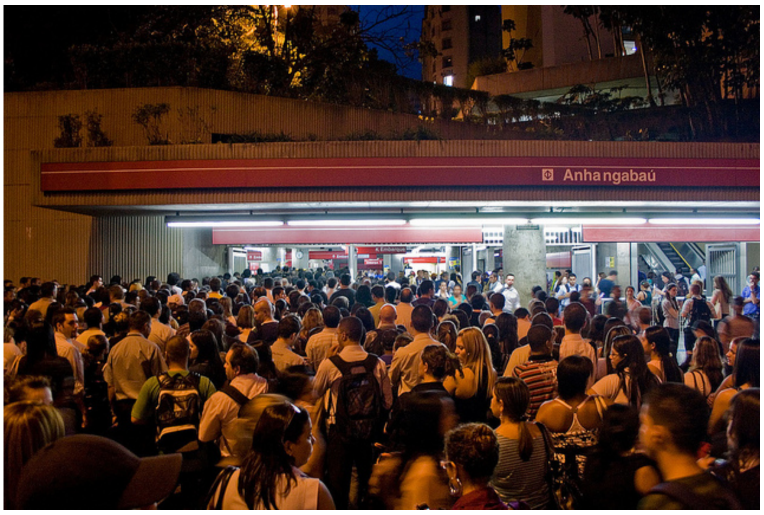


Figure 1.2: The crowded entrance of a metro station, poses a difficult situation for the navigation of robots, but persons can easily move in such environments. Credit: Fernando Stankuns.

1.2 Motivation

The motivation of this Thesis is the hypothesis that **it is possible to take advantage from the motion of moving agents, in dynamic environments, in order to improve the robot navigation in such environments**. This idea comes from the fact that pedestrians move in efficient ways (Helbing et al., 2000; Mombaur et al., 2009; Alexander, 1984), and are able to move in very complex situations, with minimal effort.

So if a robot is able to properly select a person to guide it, the robot will be able to benefit from the high level reasoning and social skills of the person being followed. When we observe an environment occupied by humans, it is inevitable to take notice of the myriad of different motions performed and trajectories traveled by persons. Even so, humans manage to safely move in such environments to reach their destination, in a seamless manner, and with many interactions with the other persons that are present.

In this sense, it can be assumed that humans often engage in group navigation, similar to swarms. For example, when a large group of commuters enter a metro station, it is often difficult for some of them to have a clear view from the wall, stairs or signs, or any feature that could help someone find his/her way in that environment, as shown in Figure 1.2.

But normally, what can be observed in those situations is that persons do not move individually, but rather as members of a larger group. Even if the goal of such group of people is not the same, the involved agents profit from the motion of each other until they are not useful anymore, either because there is more space to move individually or due to a distinct objective. This group navigation is closely linked to studies of animal behavior, as swarm intelligence:

Inside a group of animals like a flock or a school, or an insect society, multiple simple direct or indirect interactions between animals lead the whole group to adopt a specific spatial (and/or temporal) configuration (...) such kinds of systems are commonly called "Collective Intelligence" (CI) Systems or "Swarm Intelligence Systems". (De Schutter et al., 2001)

When pedestrians are moving, they are constantly informing themselves based on the motion of others. Individuals may take detours, start or stop to move, and even run based solely on how people around them move. This is important, as it can prevent accidents and also save time, as every single individual does not need to see an obstruction to adapt his/her path to avoid it. They are also constantly dealing with large amounts of high-level information and reacting to it, while pursuing an objective, effectively providing indirect information about their surroundings.

1.3 Proposed Approach

According to these observations, this Thesis proposes a conceptual shift, treating the so called *moving obstacles* as *intelligent moving agents*. Instead of seeing them as perturbations, as undesired obstacles, they should be seen as guides, that can provide very useful information about their surroundings. Pedestrians can anticipate collisions and are able to make complex decisions and interaction with others, so it is possible to take advantage of this expertise to enhance the robot's navigation capabilities in dynamic environments.

Two forms of exploring the motion of dynamic agents are proposed: taking advantage of the navigation and interaction capabilities of humans by following them; and benefiting from optimally generated paths that were traveled by humans, moving along them.

The prevailing concept in this work is that the main problem to be addressed here is not the motion planning itself, but rather the identification of agents in the environment that can be used as **leaders**. In this way the robot will be able to take advantage of those agents' navigation expertise. To be able to accomplish this idea, the following steps are proposed:

1. obtain information about the environment and the state of agents, as their velocity, distance, and other;

2. use this information to decide if, among several candidates, a subject can be used as a guide to the robot;

3. decide which type of behavior the robot should adopt to benefit from the motion of a chosen leader, as engaging in path following or direct following of a leader.

1.4 Thesis Outline and Contributions

The organization of this thesis starts with a review, in Chapter 2, about robot navigation among humans, beginning with a study of the evolution of motion planning algorithms in general and then following to the specificities of human aware navigation. This study allows a better comprehension about the current techniques used in this area and their limitations.

Then, Chapter 3 covers the methods proposed for leader selection and how those techniques can be integrated with navigation strategies. It is divided in three parts. The first part describes how existing motion prediction techniques can be used for leader selection, based on the prediction of a candidate's goal. The second part presents a different selection method, based on learning. It describes how an AdaBoost algorithm can be used to implement a learning framework for leader classification, based on human experience, and also describes a methodology for the evaluation of the most relevant features used in that classification process. Finally, the last part of this chapter describes two different strategies for leader selection, using a set of rules for the reasoning-based approach and a voting scheme for the learning-based one. After this, a multi-mode navigation framework is presented, showing how the leader selection methods can be integrated with navigation strategies.

In Chapter 4 several tests are performed, using the reasoning-based approach for leader selection and following. It starts with experiments where real data is used to replicate the motion of humans in a virtual environment, and a virtual robot selects leaders using a probabilistic prediction of candidate's goals. To account for the reaction of agent's to the presence of the robot, a simple pedestrian simulator is created and used in experiments of leader selection and following, using a multi-mode navigation system.

Chapter 5 presents the experiments of leader evaluation using a supervised learning framework. The first part covers the acquisition and labeling of the training dataset. After that, it is conducted an evaluation of the performance of the AdaBoost classification algorithm, trained with different sets of features, and a study of the contribution of different features to the leader evaluation process. The last part of this Chapter shows the experiments of the integration of the classification system with navigation strategies, where real data is used to recreate the motion of agents in a virtual environment.

Finally, Chapter 6 presents the conclusions and the open issues, to be addressed in future works.

1.5 Publications

- Vasquez, D., Stein, P., Rios-Martinez, J., Escobedo, A., Spalanzani, A., and Laugier, C. (2012). Human aware navigation for assistive robotics. *In International Symposium on Experimental Robotics (ISER).*

- Stein, P., Spalanzani, A., Laugier, C., and Santos, V. (2012). Leader selection and following in dynamic environments. *In 2012 12th International Conference on Control Automation Robotics Vision (ICARCV)*, pages 124–129.
- Stein, P., Spalanzani, A., Santos, V., and Laugier, C. (2012). Robot navigation taking advantage of moving agents. *In IROS 2012 Workshop on Assistance and Service Robotics in a Human Environment.*
- Stein, P., Santos, V., Spalanzani, A., and Laugier, C. (2013). Navigating in populated environments by following a leader. *In International Symposium on Robot and Human Interactive Communication (RO-MAN).*
- Stein, P., Santos, V., Spalanzani, A., and Laugier, C. (2013). Learning-Based Leader Classification. *to appear in IROS 2013 Workshop on Assistance and Service Robotics in a Human Environment.*

Chapter 2

Robot Navigation Among Humans: a Review

This Chapter will conduct a review of developments, in different areas, that are important to the work developed in this Thesis. Initially, an overview on the evolution of motion planning algorithms is given, to understand how the developments arrived to the current state and which are their limitations.

The second part of this state of the art is related to different aspects of navigation among humans, which are essential to the development of the proposed work. Initially a revision on works related to the comprehension of populated environments is conducted, regarding motion prediction and proxemic rules.

Following that, algorithms developed for robot navigation in human environments using motion prediction are presented, also showing their limitations. The last part of this chapter covers a review on previous works that explored different aspects of human motion and human-robot interactions.

2.1 Overview of Motion Planning

Motion planning techniques are a crucial and active field of study for the robotics community. For robots to be able to move from one configuration to another, it is expected that they accomplish their movement without collision and in an efficient and safe manner. In different types of robots, ranging from mobile robotic platforms, robotic arms and even virtual robots in video-games, good motion planning is an essential tool for the autonomy and automation of any robot.

In more formal words, the motion planning problem in robotics consists of generating a trajectory that links the initial configuration to a goal configuration and that can be traversed by the robot safely and without collisions.

This section will give an overview about the evolution of motion planning algorithms,

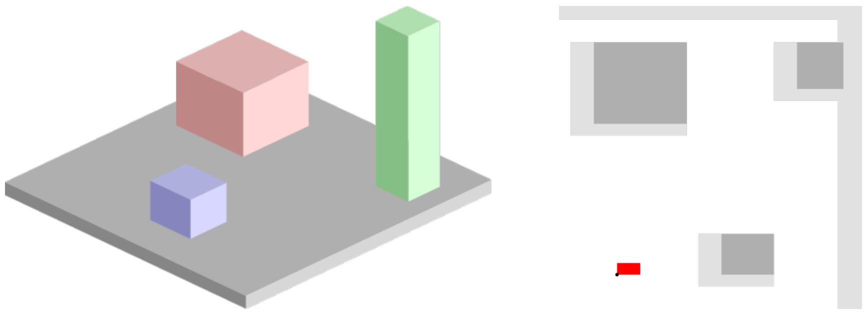


Figure 2.1: Example of an workspace and its respective $\mathcal{C}$-*space*, considering a rectangular robot centered at the black dot that can only translate, the light gray areas would result in a collision of the robot with the static objects.[1]

showing how previous approaches tackled the problem. It will become clear that the motion planning problem has been divided into simpler problems, and difficulties of real world were incrementally incorporated in more complex algorithms.

2.1.1 Motion Planning in Static Environments

The simplest case of motion planning for robots is when the environment structure and restrictions do not change with time, in which case it is called a static environment. This is the case in many industrial applications, where the area of actuation of robots is generally protected, and everything on the surroundings is static and generally closed to humans, due to hazardous conditions.

As this is where the motion planning field for robotics has started, several algorithms designed to create motion plans in static environments have been proposed. Due to the intensive work in this field, nowadays it can be considered as a well studied and solved problem (Latombe, 1991; LaValle, 2006).

The Configuration Space

One initial approach to address the motion planning problem consisted in formulating it as a geometric problem on a special workspace, where all the possible configurations of the robot and obstacles are represented. Such space is called the *configuration space*, or $\mathcal{C}$-*space* (Lozano-Perez, 1983). One example of this representation is shown in Figure 2.1, where objects are projected on a 2D plane and inflated according to the robot's shape, to represent forbidden configurations.

Using such representation, the motion planning problem is simplified into exploring feasible configurations in the space (**construction phase**) and finding a sequence of points in the $\mathcal{C}$-*space* that link the initial configuration to the goal configuration **(query phase)**.

[1]Adapted from the Wikimedia Commons file File:Motion planning workspace 1 configuration space 2.svg `http://commons.wikimedia.org/wiki/File:Motion_planning_workspace_1_configuration_space_2.svg`

The **query phase** is usually conducted by graph search algorithms, as the Dijkstra's shortest path (Dijkstra, 1959) or its variation, the A^* (Hart et al., 1968).

Different approaches can also be used in the **construction phase**. One method is called the *roadmap* and it consists in the creation of a graph that represents feasible robot configurations, and its connections, in the free regions of the $\mathcal{C}$*-space* ($\mathcal{C}_{free}$). The *roadmap* can be understood as a general guide to link different configurations. Among the techniques that can be used to create *roadmaps* are the Visibility Graph and the Voronoi Diagram. Once it is created, the motion planning algorithm *queries* this representation to find a path that will take the robot from a start to a goal configuration in the $\mathcal{C}_{free}$ space.

Another method that can be used in the **construction phase** is called Cell Decomposition. It consists in decomposing the $\mathcal{C}_{free}$ space into cells. Then a graph representing the connection between adjacent cells is built, called *connectivity graph*, which is searched in the motion planning phase to find a connection between the start and goal configuration of the robot.

Sampling Based Approaches

As the complexity and dimension of the problem increases, it becomes more difficult to create a representation of the problem in the $\mathcal{C}$ space, and it may not be possible to find an analytical solution.

In such cases, a different approach is required. Instead of trying to create a representation of a complex $\mathcal{C}$ space and analytically finding a collision-free connectivity, configurations of the robot in the $\mathcal{C}$ space are randomly sampled and then tested for collisions. If a sampled configuration is collision-free, then it is added as a node of a *roadmap*.

Subsequently, the algorithm tries to find a connection between the new *roadmap* node and nodes that had been previously tested. This process repeats until a *roadmap* that satisfies some criteria is created, and then the motion planning algorithm can *query* the *roadmap* for paths solutions, in the same way of previous techniques. Figure 2.2 shows an example of randomly sampled nodes (black dots) in the $\mathcal{C}$ space, with queried path shown in red. A well known example of such algorithm is the PRM method (Kavraki et al., 1996).

Another example of an algorithm that does not require a previously built *roadmap* is the RRT (LaValle and Kuffner Jr, 2001). Similar to the PRM approach, here configurations are again randomly sampled from the $\mathcal{C}_{free}$ space. The difference is that instead of creating a roadmap that will guide the search between any two configurations, here the construction is aimed at providing a link between two specific configurations (initial and goal).

The RRT algorithm is as follows. The initial robot configuration is inserted as the root of the tree T. Then a configuration is randomly sampled from the $\mathcal{C}_{free}$ space and the tree

[2] Adapted from the Wikimedia Commons file File:Motion_planning_configuration_space_road_map_path.svg `http://en.wikipedia.org/wiki/File:Motion_planning_configuration_space_road_map_path.svg`

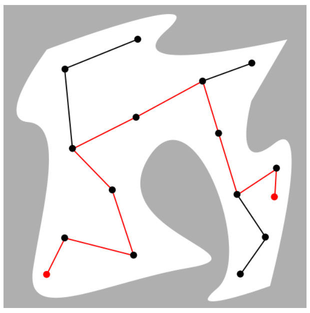


Figure 2.2: Example of PRM. The black dots are configurations and the traces are the connectivity. The queried path is shown in red. [2]

is searched for the closest neighbor x_k of that configuration. A motion control u_k that brings the node x_k closest to the sampled configuration is chosen and a new node $x_{k+1} = f(u_k, x_k)$ is inserted into the tree.

The process repeats until there is a connection between the tree root (the robot initial configuration) and the desired final configuration. Figure 2.3 illustrates this process. The robot is represented by the dark rectangle in the bottom of the images and its goal is the red circle. The images show three iteration steps of the tree construction, which expands until a feasible path to the goal configuration is found, represented by the red line.

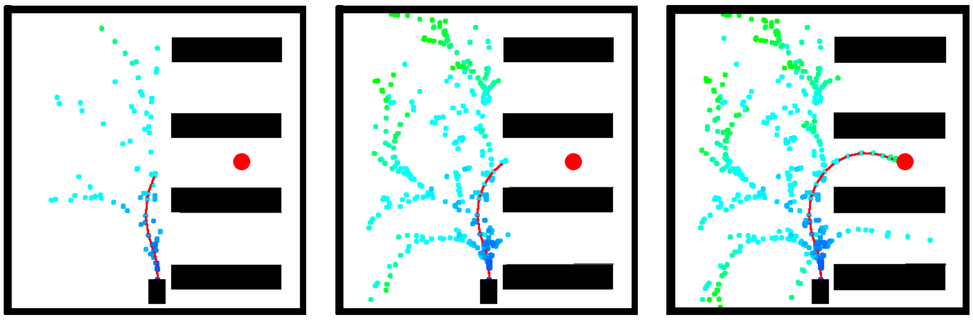


Figure 2.3: Three iteration steps of the RRT algorithm

In order to improve efficiency, there can be a bias toward the goal during the sample phase, which results in a bias in the direction of the tree's growth. Another method is to create two trees, one starting at the robot's initial configuration and the other at the robot's goal configuration, and grow both of them, until a connection between them is found.

Potential Fields

The *Potential Field* method (Khatib, 1986; Brooks, 1986; Koren and Borenstein, 1991) is an approach that does not fall into the previous categories. This method do not require the problem to be formulated using $\mathcal{C}$ approach nor require a preprocessing phase, where a *roadmap* is created.

In this algorithm, artificial forces are created in the environment. The robot's goal produces and attractive force while obstacles produce a repulsive force. The resultant of the sum of these different forces is the force that will drive the robot along the environment, making it avoid obstacles while heading to its goal. The main advantage of this technique is that it is very fast and computer efficient, but it suffers from drawbacks, as the robot can get stuck in a local *minima*.

2.1.2 Motion Planning in Dynamic Environments

Although the motion planning algorithms achieved very good results in static environments, robots implementing these algorithms would suffer from several restrictions to be used in real scenarios. So the next, and natural, step was to incorporate dynamic features into the problem. This better represents real life situations that robots will find, as they move out from the factory floor, in environments where interaction with persons or other moving entities are often required.

Dynamic Environments, however, pose several added difficulties to the motion planning problem. The dynamics of the robot must be taken into account, and there are limitations due to the sensors range and uncertainty in measurements, that must be reflected on the motion plan. Besides it, a motion plan must incorporate time restrictions, meaning the robot will require a certain amount of time to accomplish a task. For example, when crossing a road, the robot must do it fast enough to avoid incoming cars.

To address these restrictions, first approaches tried to adapt existing algorithms developed to be used in static environments to dynamic ones. As the computer power grew, it was expected that if robots could perform the classic sense/plan/act loop very quickly, it would be able to solve problems in dynamic environments, by planning and replanning fast enough.

However, this approach could only be applied in situations where the velocity of other entities were slow compared to the robot's. In other situations, this approach would fail. Take the previous example of a robot crossing a road: if the algorithm assumes the world as static (take a snapshot), it could consider an incoming car as a static object and plan a path crossing the road. If the robot moved slowly, the car could hit it. But if the time dimension is explicitly taken into account in the problem, representing the car's and the robot's future positions, better solutions can be obtained.

Configuration-Time Space

Among initial developments that focused on a solution for the dynamic collision avoidance, one approach is to add the *time* as another dimension to the $\mathcal{C}$*-space*, creating the *configuration-time space*, or $\mathcal{CT}$*-space*. In this case, if we consider a two-dimension $\mathcal{C}$*-space* in the static case with obstacles represented with 2D shapes, in the dynamic environment the elements will appear as 3D volumes that sweep along the time axis. It is important to emphasize that this approach requires the motion of the moving objects to be fully known. If this is not the case, the motion of the objects must be predicted, resulting in a more complex problem.

The same family of algorithms used in the static motion planning problem can be applied here. However they require some restrictions regarding the robot dynamics, as it is not possible to travel back in time or move instantaneously from one configuration to another, so the velocity of the robot must be bounded. To solve the problem of finding a solution in the $\mathcal{CT}$*-space*, new approaches were developed. In Reif and Sharir (1985), a visibility graph is created in the $\mathcal{CT}$*-space*, while Erdmann and Lozano-Pérez (1987) discretized the time dimension into a sequence of slices and solved the path planning problem for each of these slices and Fujimura and Samet (1989) solved this problem using cell-decomposition.

In the case that the motion of objects in the scene is *unknown* a priori, an option is to predict the movement of the dynamic entity and create a plan that will avoid a possible future collision. In Miura et al. (1999) and Miura and Shirai (2000), the authors develop a method that takes into account the predicted motion of dynamic objects. It uses a probabilistic approach to take into account the uncertainties related to the path, velocity and observations of the dynamic object, resulting in improved behavior of the robot. Another example can be found in Zhu (1991), where a HMM is used to model and predict the motion of dynamic objects in order to generate a collision-free *local* motion plan for the robot.

Decomposition-based Motion Planning

It is also possible to find proposals that decompose the motion planning problem: planning a path that would avoid static objects using the previously mentioned techniques and then changing velocities along this path to avoid dynamic objects. Collisions are tested along the precomputed path as the robot speeds up or down to avoid them. This method is implemented in Kant and Zucker (1986) and improved in Fraichard and Laugier (1993), where not only one path is computed, but several adjacent paths, so the algorithm would provide a solution even if a dynamic object stopped over a precomputed path. The authors also take into account the kinematics constraints of the robot in this stage, which is called the path-planning phase. The second stage is the trajectory planning, where the motion of the robot along adjacent paths is computed in order to avoid dynamic objects and comply with kinematic constraints.

This method gives room to a different classification of motion planning algorithms, according to their scope of application: global motion planners and local motion planners (or local collision avoidance) algorithms (Brock and Khatib, 1999; Myers et al., 2005):

- **global motion planning**: here, all the information about the environment is assumed to be known, and the whole planning is done *offline* (takes place before the maneuvers start). The advantages of this approach is that an optimal and complete path can be computed, but it is unsuited for dynamic collision avoidance, as seen before, since information about the environment can change and then a re-planning is required.

- **local motion planning**: the objective is to provide a solution as fast as possible, with only a partial information about the robot's surroundings. The algorithms are usually reactive, work *online* and are not computer intensive, but they provide suboptimal solutions and may result in the robot being stuck in local minima.

With the fusion of these two categories of algorithms, it is possible to overcome their individual limitations and sum their strengths. First, a complete path to the goal is planned in an initial step and then the robot reacts to new measurements and unpredicted obstacles along the path, as new information becomes available to the planner (Krogh and Thorpe, 1986; Choi and Latombe, 1991; Quinlan and Khatib, 1993; Brock and Khatib, 1999; Stachniss and Burgard, 2002).

In this way, if an unexpected object blocks the robot's path or if the object's motion does not coincide with the predictions, the robot can avoid a collision reacting to the new information, temporarily moving away from the original path, and later resuming to follow its original motion plan.

Due to separation of the problem of motion planning between local and global, when addressing the problem of motion planning in dynamic environment, more emphasis was given to collision avoidance on a local frame. The main reason was that planning on the global frame could still be performed using the classic motion planning techniques developed for static environments, specially when the motion of objects was unknown and difficult to incorporate on the problem.

Velocity Obstacles

Different from previous algorithms, the Velocity Obstacle (VO) algorithm (Fiorini and Shiller, 1993, 1998) uses *velocity* information instead of *position and time* to compute possible collisions and generate motion commands to an agent. This implementation maps the environment into the robot velocity space, and calculates all relative velocities that would lead to a collision, this set of velocities is called the VO. Then collision avoidance can be easily performed, if a velocity outside the VO is chosen as the robot desired velocity.

For the velocity choice, several simple heuristics can be used, as the highest avoidance velocity along the line of the goal, the maximum avoidance velocity within an angle from the line of the goal or an avoidance velocity that minimizes the risk according to the perceived obstacle.

To handle more complex trajectories, another variation of the VO was developed by Shiller et al. (2001), called Non-Linear Velocity Obstacle (NLVO), which was used in Large et al. (2002) and Large et al. (2004). This approach overcomes a drawback of the original VO implementation, that represents obstacles' motions at a constant speed and in straight trajectories. According to Large et al. (2005), the VO can be considered as a first order approximation of NLVO at a given time. This approach has the advantage of giving a better representation of the objects true trajectory and therefore require less adjustments of velocities of the robot, when trying to avoid them.

It is important to have in mind that in a realistic environment, the dynamic agents do not move in a completely passive fashion. They will react to the motion of the robot as well as to the motion of other agents. So, in order to develop an effective local motion planner algorithm, *the capability of other agents to take active maneuvers to avoid collisions must be taken into account when performing self maneuvers.* This is a key aspect, that was incorporated in variations of the original VO algorithm (Abe and Yoshiki, 2001; Kluge and Prassler, 2004; van den Berg et al., 2008a; Snape et al., 2009).

In a technique named the Cooperative Collision Avoidance (CCA) (Abe and Yoshiki, 2001), the authors developed a system where agents cooperate to avoid collision with each other. This is accomplished assuming all the agents involved use the same algorithm to choose their respective velocities, but this is impractical in real world situations.

A similar approach is proposed in van den Berg et al. (2008a) and Snape et al. (2009), where it is assumed that the involved agents perform similar avoidance maneuvers, and that agents implicitly share responsibility when taking avoidance maneuvers. By doing so, they prevent oscillations that result from the assumption that other agents are completely passive, a problem found in the original VO implementation.

Finally in Kluge and Prassler (2004), the authors implement an approach called *reflective navigation* based on the idea that the mobile robot should reflect on the intentions of other dynamic agents, in order improve its navigation, as humans do. Also, to address the problems related to the uncertainty in predictions and measurements, they introduce the Probabilistic Velocity Obstacles (PVO), a probabilistic variation of the original VO algorithm.

2.1.3 Analysis

This section provides a study of the evolution of motion planning techniques, so it is possible to better comprehend what are the limitations of the existing approaches.

Along this section, it becomes clear that *anticipation* is a keyword when it comes to motion planning in dynamic environments. Among the different cases and approaches, the lack of information and poor predictions will lead the motion planning algorithms to create plans that are short-lived and that will result in suboptimal trajectories traveled by the robot. It is also evident that a general solution is still out of grasp.

Fortunately, improved results can be expected narrowing the area of application of the intended algorithms and robots. The requirements for an autonomous car are quite different from those for a health-care robot, for example. Throughout this Thesis, the focus will be centered on robots that share the environment with humans, and the following section will discuss the peculiarities and previous work on this area.

2.2 Human-Aware Navigation

When robots share an environment with humans, the classical navigation techniques must be modified to take into account requirements associated with human behavior. Persons must be treated different from obstacles, and rather as agents that can react to situations and interact with the robots. This means that robots must respect their social conventions, guarantee the comfort of surrounding persons, and maintain legibility, so humans can understand the robot's intentions (Kruse et al., 2013).

It is also important to notice that in recent years, a myriad of different robots that share their environments with humans were developed. Examples of these are the tour-guide robots RHINO and MINERVA (Thrun et al., 1999), sweeping floor robots as the iRobot Roomba, robots that aid humans with disabilities as the GuideCane (Borenstein and Ulrich, 1997) and MAid (Prassler et al., 2001), fetch and carry robots as Cero (Severinson-Eklundh et al., 2003), and also robots specifically designed for human-robot interactions as the Robovie (Ishiguro et al., 2001), among others.

This large range of fields illustrates how ubiquitous robots have come to be among us. As a consequence, the study of human-aware navigation is an active topic of research, involving different fields as, human perception, cognitive models and motion planning. However, this is still a challenging task due to the following reasons:

- Humans have very little restrictions to their movements, they may, for example, abruptly stop and change direction, without previous warning, and in an holonomic way;
- Persons abide to social rules, dependent on their culture and context. It is expected that robots behave in a similar fashion, respecting social configurations, in order to improve their acceptance by humans;
- The robot must also be safe, meaning it cannot risk injuring humans or creating dangerous situations to other robots and machines.

2.2.1 Environment Comprehension

Motion Prediction

Humans have a large freedom of motion, being able to change their direction and speed in fractions of seconds. Nevertheless, the prediction of the motion of humans is used in several human-aware navigation algorithms, as they may benefit from that information when creating motion plans for the robot. Besides that, motion prediction can also enhance people tracking algorithms, as the motion model of pedestrians can be used to improve the quality of the tracking. According to Kruse et al. (2013), motion prediction can be separated in two groups:

- **prediction based on reasoning**. According to the current robot state and the environment, different methods can use distinct levels of reasoning to determine the future motion of agents. Examples are linear extrapolation of the current velocity and orientation; stochastic predictions, where the likelihood of transitions are empirically set; and potential fields, that assumes the agents will avoid static obstacles while heading toward a destination (Tadokoro et al., 1995; Hoeller et al., 2007).
- **prediction based on learning**. The environment is first observed for the typical motion patterns of persons, and these patterns are stored. Then this information is used in a learning framework that will perform predictions assuming the motion patterns are likely to be repeated. Techniques that are used in this method are Artificial Neural Network (ANN), Gaussian Process (GP), the Growing Hidden Markov Model (GHMM), and others (Makris and Ellis, 2002; Tay and Laugier, 2007; Ellis et al., 2009; Vasquez Govea et al., 2009).

Although predictions based on reasoning are simple, they may need more computer resources than the learning based approach, where most of the processing is done offline. Besides that, the former method may fail to produce good results in more complex environments, while the latter is able to learn special situations and is also able to improve its performance over time.

Social Spatial Rules

The seminal work in this area was conducted by Edward T. Hall in his very influential work, The Hidden Dimension Hall (1966), where he coined the term **proxemics**. This concept can be understood as the study on the spatial separation among individuals, and the cultural concepts of space. It is also an study on how humans utilize their space, how they maintain separation between other humans, and how they builds the structures around them.

Although the original study had a far reaching scope, the robotics community borrowed some insights from this seminal work to create rules in motion planning that would respect and maintain comfort of humans while navigating among them. To improve the acceptance of robots, the main focus was given to the Personal Space, which is the region around a person that may cause discomfort if invaded. But equally important is the space related to the interaction among humans, that should also be respected by the robot (Ciolek and Kendon, 1980).

Works on this area concentrated on two main lines. The first concerns the metrics involved in proxemics, where real experiments are conducted and persons are interviewed in an attempt to learn how the presence of the robot affects the comfort of the persons (Huettenrauch et al., 2006; Walters et al., 2009; Takayama and Pantofaru, 2009). These experiments are conducted with different groups of people (young, old, familiar with robots), different robots (small, tall, robot-like, human-like) and also distinct situations (angles and velocity of approach, gaze direction, etc.). The main findings point in the direction that the proxemics distances are not fixed, but vary according to the culture, the age, context and others.

The second line of research on this field relates to how the agent's proxemics, position and orientation can be modeled and integrated into motion planning algorithms. These issues are addressed in the work of Sisbot et al. (2007), Rios-Martinez et al. (2011) and Vasquez et al. (2012), where the notions of respecting the Personal Space and interactions zones are incorporated in a motion planning algorithm as a *risk of disturbance*. While in Luber et al. (2012), instead of using proxemic models for navigation, motion prototypes are learned that reproduce the motion behavior of walking persons, which respect the social space of others.

2.2.2 Prediction-Based Collision Avoidance

As shown before, the prediction of the motion of the moving objects may be used when the complete environment information is unknown, in order to improve the global and local motion planning in dynamic environments. So it is interesting to detect and track moving agents so the motion planning algorithm can avoid future collisions or risky situations.

Many methods use predictions about the agents motion in order to compute motion plans that avoid future collisions in a dynamic environment. One example of such work has been conducted by Bennewitz et al. (2003, 2005), where the typical movement of pedestrians are taken into account and modeled using HMMs. During the path planning phase, the algorithm incorporates a probabilistic belief of likely pedestrian trajectories to minimize chances of collision. The planning is performed on a grid, where the cost of traversing a cell on is proportional to the probability of it being occupied by a pedestrian at a given time.

In environments with higher dynamics and time restriction, a different approach is implemented. On Fulgenzi et al. (2008) the typical motion pattern of pedestrians are modeled as GPs and on Fulgenzi et al. (2009) as HMMs. The path planning algorithm is implemented with a variation of the RRT algorithm, called RiskRRT, which introduces probability uncertainty into the problem formulation and allows new information to be integrated *online*.

The evaluation of tree nodes takes into account a probabilistic measure of the risk associated with traversing the generated paths. To compute such risk, several factors are taken into account, as distance (longer paths are riskier, as uncertainty increases), and the probability of a pedestrian to be moving along a typical pattern (which increases the risk for navigation). The result is that areas with a high density of typical patterns are associated with a higher collision risk and are normally avoided by the planner. On these techniques, the online navigation restriction is addressed with a Partial Motion Planning (PMP) architecture (Petti and

Fraichard, 2005), yielding the best partial solution for a given time limit.

The Freezing Robot Problem

Motion Planning algorithms that rely on motion prediction, suffer from a recurrent problem, known as the FRP (Trautman and Krause, 2010), illustrated in Figure 2.4. The problem is that, when making motion predictions, the uncertainty of the agents' position grows with time. The increasing uncertainty, allied to densely populated environments, eventually covers all the open space in front of the robot. Due to this, the motion planning algorithms are unable to compute a feasible plan and as a result, the robot gets stuck, or frozen, even if a solution exists.

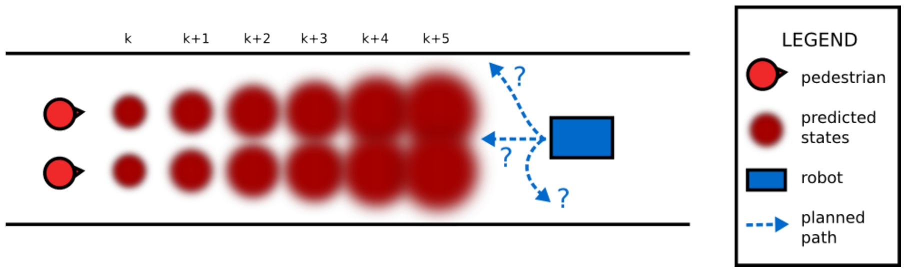


Figure 2.4: The Freezing Robot Problem (FRP): the red blurs represent the growing uncertainty of the position of pedestrians, which eventually covers the empty space in front of the robot, and as a result, the motion planning algorithm cannot find a solution.

Some techniques implemented to address this problem artificially increase the belief in prediction, in an attempt to find a path through moving agents, but the authors show that the FRP still occurs if the environment is crowded enough.

The origin of this problem, nevertheless, is not taking into account that the pedestrians in an environment will react and change their motion due to the presence of the robot, which the authors call *joint collision avoidance*, where participants adapt their motion to each other. Examples of works that do take that into account can be found in crowd simulators (Helbing and Molnar, 1995; Helbing et al., 2000, 2002) and with reciprocal velocity obstacles (van den Berg et al., 2008a, 2011).

To address the FRP, the authors developed the Interacting Gaussian Processes (IGP), which is a method for describing probabilistic interactions among multiple moving agents. This tool is used for navigation purposes, modeling the interactions among the crowd agents and between the robot and the crowd, so the robot can successfully merge with crowds and navigate cluttered environments. In another work, Yu and Wang (2012) address the FRP by implementing a learning framework that learns policies of navigation in dynamic environment from human examples. However training and validation are performed in a very simplistic environment.

2.2.3 Navigation Benefiting from Human Motion

Different methods of robot navigation can benefit in distinct ways from the motion of humans. As seen in the previous section, methods that predict the future position of humans can be used to create motion plans that avoid collisions in the future and are valid for a longer period of time.

But the benefits of human motion go beyond collision avoidance approaches. They can also be used to actually bias the robot motion plan toward areas commonly traveled by humans, benefiting from their high-level planning capacity. The behavior of humans in an area can also provide cues to help service robots to identify potential "clients" in order to assist them, as helping lost persons in a mall or inviting them to try a product. It is also possible to learn how humans behave when moving in populated environments and apply those findings in navigation algorithms, so the robot can behave in the same fashion as humans. The following subsections will give some examples of works in the mentioned fields.

Preferential Paths

Human locomotion is very informative, and assumed to be optimal (Helbing et al., 2000; Mombaur et al., 2009; Alexander, 1984). In this sense, it may benefit the robot to reuse the motion patterns of humans to guide their search of a motion plan. Among the advantages of traveling along paths usually taken by humans, are the avoidance of obstacles difficult to detect and better navigation among groups of people.

An example can be found in the work of Sehestedt et al. (2010). Instead of using the pedestrians' motion predictions to avoid future collisions only, the motion patterns can also be used as preferential roadmaps for an A^* search algorithm. By adjusting two weight factors, the behavior of the motion planning algorithm can be changed, so the robot can follow or avoid common paths on the environment. The weights also control the choice of high or low density paths, where density means how often that path is traveled by humans, so the robot can move along more or less populated paths.

In a similar work, O'Callaghan et al. (2011) use Gaussian Process (GP) to learn a map that stores information about the direction usually traveled by humans, with each map is associated with different destinations. Using these maps, the robot can follow the same patterns and move as people would do in such environment, effectively avoiding static obstacles, even if undetected by the robot's sensors. This idea is shared with Murphy and Corke (2012), where the authors build a map that has a reduced navigation cost in the areas traveled by humans, for later path planning. These maps are built online, while the robot is following humans.

Learning from Human Motion

Learning and imitating motion behaviors of people can bring benefits in terms of navigation efficiency for the robot. Besides that, the robot's motion can became more predictable, improving acceptance by pedestrians (Kruse et al., 2012).

In order to learn how humans move from an initial to a goal configuration, Mombaur et al. (2009) uses inverse optimal control to determine what is the optimization criteria that humans use for locomotion. The technique is applied over a real dataset where humans were asked to move from an initial position and orientation until reaching a marked position on the floor. Then the authors proceeded in creating a model, based on the optimization criteria, that best represents the real motion behavior of humans.

Regarding the human preferences and the way they interact with each other while moving, Henry et al. (2010) use Inverse Reinforcement Learning (IRL) in an attempt to understand how humans decide to switch among different behaviors such as "desire to move with the flow", "avoidance of high-density areas", "preference for walking on the right/left side" and "desire to reach goal quickly".

The IRL is used to learn from demonstrations of typical paths (traces) of persons in crowded environments. A realistic crowd simulator is used in order to generate those motion patterns that contain information about the velocity and density of surrounding pedestrians.

By using the IRL, the algorithm is capable of learning the weights associated with the possible actions in the environment. The planner then uses the A^* search algorithm to compute the best path to the goal given the learned costs of actions and the dynamic features of the environment. The authors validate their approach on experiments, as they show that the robot prefers to move together with the crowd flow, even if that is not the shortest path to the desired goal. The reason is that the IRL algorithm learned that in crowded environments, moving with the flow has a lower cost than moving independently.

Interacting with Humans

Human motion provides cues about the intentions of the subjects. In the situations where robots have been designed to actively interact with humans, with the intention of providing services, this cues can be used to decide when and where to approach a person.

In a shopping center, for example, if a person is constantly looking at several store windows, he/she may be considered a potential client and a robot may start an interaction with that person. While if someone is moving fast and in the middle of a corridor, for example, it is very likely that this person is not interested in shopping and the robot would not profit in interacting with them. Within this scope, the movement of persons, as well as their local behavior in a shopping center is studied by Kanda et al. (2008).

On the same line, Tranberg Hansen et al. (2009) implements a Case Based Reasoning (CBR) framework that takes as input the position and orientation of a person to evaluate if an encounter between a human and a robot is likely to result in an interaction. Based on the predicted outcome of an encounter, a human aware algorithm changes the potential field surrounding a person to reflect the willingness of the human to interact (or not) with the robot, so it can approach them.

Leader Following

One interesting way of demonstrating the capabilities of any robot is to follow someone or something. It is usually an attractive experiment, as it demonstrates in practice several aspects that define a robot, as sensing the environment, taking decisions and moving autonomously.

Leader following is also a relevant topic of research, as it is useful in many human-robot interactions and activities. Studies that have been conducted on how to follow humans address different aspects of the problem, and they can be roughly divided in: how to position the robot w.r.t. the leader; what motion technique is used; how to track leaders; and the purpose of following a leader.

Following Position Normally a virtual target is created that represents the ideal position that the robot should maintain with respect to the person being followed. In Gockley et al. (2007) a study is conducted with two types of virtual targets, one that is the position of the leader and the other is the leader path (a sequence of virtual points). In both cases the robot maintains a minimum distance from the person (Figure 2.5a). In experiments, external evaluators found that the direct leader following results in a more natural, human-like behavior.

In Hoeller et al. (2007), the goal position for the algorithm is a virtual target that is created by partitioning the area close to the leader in three categories: beside, slightly behind/beside and behind. The preferred category is slightly behind and beside the leader, as the leader is still in the field of view of the robot but its front is unobstructed (Figure 2.5b). The algorithm changes the virtual target among these three categories on-the-fly, according to the situation, as an obstruction, for example.

A more distinct technique of virtual point positioning is proposed by Ho et al. (2012), where a robot ”follows” a human leader while actually in front of him/her (Figure 2.5c). The argument of the authors is that it is easier for humans to access or deliver items to a robot in front of them, as in a supermarket, for example. This is a difficult task, because the robot must anticipate the human motion and quickly correct its position. To be able to accomplish this task, an understanding of the human walking behavior is required, as well as the prediction of his/her motion. The authors give special importance to the change of orientation in human walk, as it is during this phase that the biggest effort is required from the robot to keep in front of the human leader.

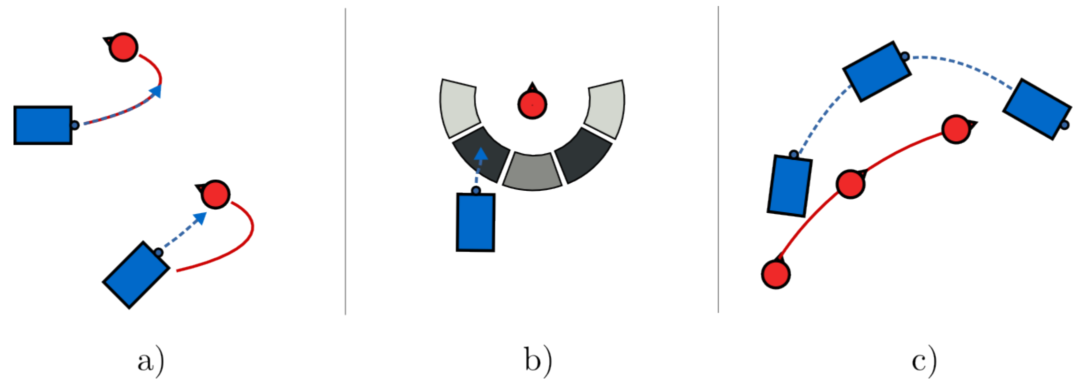


Figure 2.5: a) two types of following behavior: path following and direct following (Gockley et al., 2007); b) virtual targets distributed around the pedestrian being followed, preferred targets are shown in darker colors (Hoeller et al., 2007); c) a robot that "follows" a pedestrian in front of him/her, showing a recovery behavior after the human makes a turn.

Motion Techniques In the case of simple uncluttered environment, with cooperative leaders, the leader following algorithm can be implemented using tools as simple as a proportional controller (Sidenbladh et al., 1999) or the potential field approach (Müller et al., 2008).

With more complex situations, more refined techniques are used, as variations of the RRT algorithm. In Hoeller et al. (2007), the Expansive-Spaces Trees method is used, which is a type of RRT where nodes present in more sparsely sampled areas are more likely to be chosen for extension. In other work, Miura et al. (2010) implements a RRT where the tree nodes are checked for collision with dynamic agents, assuming their velocity and orientation are constant. In this work, two methods of node evaluation are combined in order to find the best path on the tree. The first method is the simplest and tries to minimize the distance to the target as well as the change of orientation required by the robot. The second method implements a more complex evaluation, which also takes into account the dynamics of the robot.

Tracking The most common sensors used for tracking are cameras and laser range finders. In the work of Miura et al. (2010), the tracking system acquires images with a stereo camera and combines a depth template matching with a Support Vector Machine (SVM) to reject false positives. Then a Kalman Filter (KF) is used to cope with occlusion of the tracked person.

A camera is also used in Sidenbladh et al. (1999), where the leader is detected by skin color segmentation and a proportional controller is used to command the robot in order to keep the head of the leader on the central portion of the image frame.

Gockley et al. (2007) uses a laser range finder and measurements are segmented and tracked using a particle filter. Although not directly related to leader following, but important in this subject anyhow, are the works of Arras et al. (2008), that uses machine learning to detect legs in laser scans and of Almeida (2010), that also uses laser scans to track generic moving target, implementing clustering of scan points and using a KF to cope with obstructions.

Purpose Different purposes guide the studies in the field of leader following. In some cases, the objective is to improve the person following, by creating efficient controllers or developing methods for robust tracking. But other works focus on different purposes, as to improve the acceptance of the robot by human counterparts (Gockley et al., 2007).

Acting similar to humans has proven to be a key aspect regarding robot acceptance, and with this purpose, Müller et al. (2008) develop a planning algorithm that selects and follows a leader that moves towards the robot's goal. The argument of this approach is that when robots attempt to avoid regions near pedestrians, usually the resulting behavior of the robot is very unnatural when compared to the motion of humans in the environment. So, the preferred behavior in this algorithm is to move together with groups of humans, instead of searching a path that is risk free or that provides the shortest trajectory.

This is the only work on this review that actually employs a refined method of leader selection, instead of having a person to actively present him/herself as a leader or having a leader chosen by others.

Concerned with the social impacts of a robot in urban spaces, Svenstrup et al. (2009) conducts a pilot study of a robot following random people in a shopping mall that is also a transport station hub. The intention of the work was to evaluate if social robots can add value to urban spaces, so a series of questions were asked to the persons that were followed.

The results show that about 92% of the persons were positive toward the robots and 67% did not feel discomfort when they were followed by the robot (neutral or good), while 10% did not feel good and 23% did not answer to that question. It is interesting to note that among several questions, this was by far the one with the highest percentage of persons not giving an answer. As the authors do not elaborate on this, it is difficult to understand the reason of this outlier.

2.2.4 Analysis

Human-aware navigation have added difficulties, compared to simple dynamic motion planning, as several constraints must be considered, as comfort, safety and the respect of social rules.

To be able to address these restrictions, it is necessary an understanding of the robot surroundings and rules of social interactions. The motion prediction of pedestrians can be used to provide extended information about future position of subjects, as well as the motion patterns that exists in specific environments. While the study and modeling of social rules allow them to be taken into account by motion planning algorithms.

This information about the robot's surroundings can be used by motion planning algorithms to avoid future collisions with humans when creating a motion plan while respecting their social conventions. But it is also possible to benefit from that information in other ways, as moving along preferential paths and learning from human examples of interactions.

Other advantages of following a human leader is that humans naturally abide to the proxemic rules, and are capable of understanding very complex interactions that should be respected. Therefore when a robot follows a human leader, this person will not only help the robot to navigate in difficult environments, but will also help the robot to respect the personal spaces and social rules of other persons in the environment. This results in another advantage that following the motion of leaders can bring to the robot.

The main contribution of this thesis, compared to previous developed works in the area of human following, is a different objective and distinct means to accomplish it. In many works, the objective is to study following behaviors and techniques (Hoeller et al., 2007; Sidenbladh et al., 1999), or the leader following is implemented to improve, or study, the acceptance of robots by humans (Gockley et al., 2007; Müller et al., 2008; Svenstrup et al., 2009). This is an important difference and contribution of this thesis, as it proposes to engage in following behavior with the purpose of enhancing the motion capabilities of the robots, as avoiding collision with other agents or escaping frozen situations. Besides that, the lack of research in the topic of how to choose a leader led to the development of leader selection techniques, which is another important contribution of this thesis and will be presented in the following chapter.

Chapter 3

Methods for Selecting and Following a Leader

In this work it is proposed that when a robot follows a leader it may benefit from high level information and from decisions provided by the subject being followed. As a result, the robot may be able to navigate through difficult scenarios and even avoid obstacles that could not be detected by its sensors alone.

But in order to take advantage of following a leader, one must first be selected. Considering that the robot finds itself in a populated environment, each person can be seen as a leader candidate for the robot to follow. This brings us to one of the key problems addressed by this Thesis, which is how to choose a leader among several candidates.

This chapter will describe two techniques that were developed in order to perform the leader selection, and that can be separated into two categories. The first will be based on reasoning, with a set of rules being used to decide on a leader candidate. And the second will be based on learning from examples of human experience. The last section of this chapter presents the integration of these two different leader selection techniques in a multi-mode navigation framework, that implements different algorithms, according to the environment condition.

3.1 Techniques for Leader Selection Based on Reasoning

For this selection method, the current and future state of the humans surrounding the robot are taken into account. The decision is based on observations and assumptions of the characteristics of a good leader. In the method proposed here, a good candidate will be any person that has a goal or will follow a path similar to the robot's one. If a person has the same destination as the robot (goal similarity), this subject can be considered as a strong leader candidate, because the robot can follow him/her all the way to its destination.

But even if the goals of a candidate and of the robot are not the same, they may follow similar paths to reach their destinations (path similarity). For example, if the robot wants to reach a door in the middle of a corridor and it predicts that a subject will move all along that corridor, this person can be followed until the robot reaches its goal, as both paths are likely to be close to each other.

To determine if a candidate has a similar goal or path to the robot's, the subject's motion must be predicted. This is not a straight forward task, as humans do not normally advertise their intentions and destination, and can suddenly stop or change direction, without previous warning. Once candidates are pre-selected, based on their goal and/or path similarity, then other criteria are used to select the best among several candidates as their distance from the robot, their velocity and others, which are discussed in section 3.3.

The following subsections will present existing methods that can be used in the prediction of the path and probable destination of leader candidates, and how they can be used for leader selection purposes.

3.1.1 Goal and Path Advertisement

In this case, the candidates advertise their planned path and intended goal to other agents in the system. Although this is not common to occur, we already have the technology necessary to advertise our intentions or navigation plan. One example is the software Waze, described as a community-based traffic and navigation app, where users can contribute about road traffic, accidents, and others, while also sharing their position and destination with friends [1].

Thinking about vehicles, most of them now come equipped with GPS receivers that are used to create routes to guide the driver based on his/her position and intended destination. As humans, we are also beginning to use intensively smartphones that also feature GPS receivers. Such devices can be used to guide the user in the same way as performed with a vehicle's driver. In both cases, the goal, plan and current position of a user can be made available to others, and a robot could use that information in order to select a leader, as in the case of similar goals, for example.

Some airports also provide smartphone apps that show the boarding gate to users and other flight information to them. This information can be distributed, so a robot in those airports would be able to select leaders, or at least enhance the prediction of candidates' destination, based on the boarding gates of individuals.

Many of the technologies described here are still not available, and would have to address several privacy issues before becoming functional. Nevertheless the proposed techniques would certainly be useful in enhancing the motion and goal prediction of persons.

[1] `http://www.waze.com/`

3.1.2 Goal and Path Prediction by Extrapolation of Current State

Another method of motion prediction is performing an extrapolation of the candidate's current state (speed and orientation) into the future. This method assumes that the moving subject will maintain his/her direction and speed in the short-term. Eventually, when looking to the future predictions, the subject will reach a wall or the end of a scenario, and that region can be considered as his/her predicted goal.

In order to accomplish this, several assumptions and simplifications must be made. For example, assuming the environment does not play a role on one's path and that every possible position on the environment's limits are equality likely to be visited. Despite these simplifications, goal and path prediction based on extrapolation is very fast and usually good enough for short-term predictions.

Kalman Filter

As simple linear extrapolations are subject to high levels of noise, the predicted velocity and orientation may suffer from a high variance and error. To mitigate these errors, a Kalman Filter (KF) can be applied to the measured data, so the outcome of the predicted path and future positions may be less affected by noise and short-term fluctuations on the measurements.

The KF has two distinct steps, and its workflow can be seen in Figure 3.1. One is the **prediction** step, where the current state variables are estimated based on the previous state. The second is the **update** step, where new (and noisy) measurements are used to update the estimates of the prediction step. The resulting state is an average weight according to the filter's model and the certainty of the measurements, which balances the contribution of each of the two steps. If measurements suffer from a large uncertainty, and therefore a large covariance, more weight is given to the prediction step, while if the measurements have a low covariance, more weight is given to the update step.

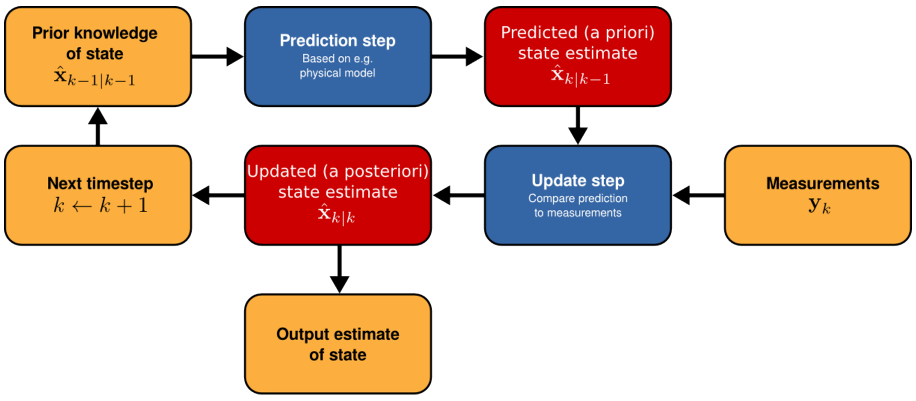

Prior knowledge of state $\hat{\mathbf{x}}_{k-1|k-1}$
Prediction step
Based on e.g. physical model
Predicted (a priori) state estimate $\hat{\mathbf{x}}_{k|k-1}$
Next timestep $k \leftarrow k+1$
Updated (a posteriori) state estimate $\hat{\mathbf{x}}_{k|k}$
Update step
Compare prediction to measurements
Measurements $\mathbf{y}_k$
Output estimate of state

Figure 3.1: Diagram of a Kalman Filter.

As the filter also has an internal model of the target, it can provide information on subjects even in the case of obstructions and lack of measurements. In that situation only the prediction step is used, until a new measurement is available. The same concept can be used to make future predictions, by running the filter without measurement updates in order to identify the possible path and destination of a subject.

Among the advantages of this method is that it does not require a previous knowledge of the environment, it is simple and well suited for short-term predictions. However, it is not practical to use this approach for medium and long-term predictions, as the further predictions are in the future, the higher the uncertainty associated to it will be.

Influence of Environment Structure on Prediction by Extrapolation

The approach of goal and path prediction based on the current state of a subject work well in some types of environments, as corridors and unobstructed rooms. In a corridor, pedestrians do not have much freedom and must move along it, until reaching their objective. Figure 3.2 illustrates this situation.

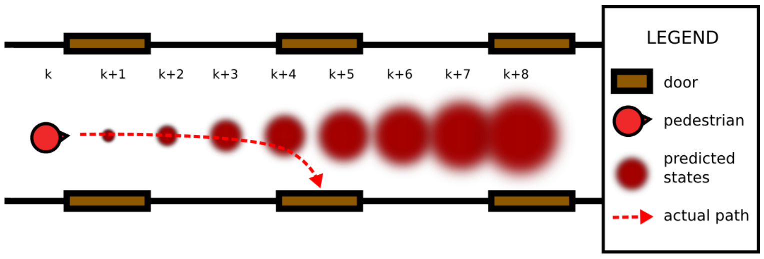


Figure 3.2: Prediction of path and goal by state extrapolation on a corridor.

In the image a person moves from left to right and has the middle bottom door as destination. The blurred blobs represent the position prediction in time steps $k+1, k+2, ..., k+8$. Larger blobs represent the growing uncertainty of future predictions. Based on the extrapolation of the pedestrian state at time k, the goal of the subject would be erroneously predicted at the right-end of the corridor. However the predicted paths do coincide with the actual path until time step $k+4$, so the robot can benefit from that candidate until he/she stops.

In unobstructed rooms, the users will naturally head toward their objectives, as there is no obstacle or constraint impeding a straight motion from their position to their goals, so it is acceptable to assume that their goals are directly linked to their motion direction. This can be seen in Figure 3.3, where the extrapolation of the current state leads to a goal prediction very close to the desired subject destination (upper right door). It can be seen that the prediction has a high level of correspondence with the actual path and goal of the pedestrian.

Another limitation arises in more complex scenarios, because in those environments, pedestrians will move in a more complex fashion, conditioned by the environment structure. Due

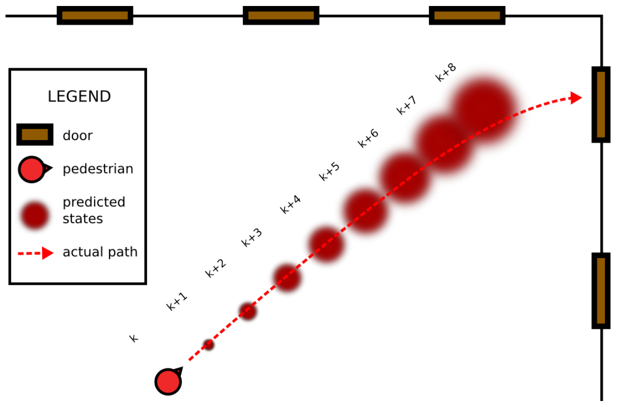


Figure 3.3: Prediction of path and goal by state extrapolation on an open area.

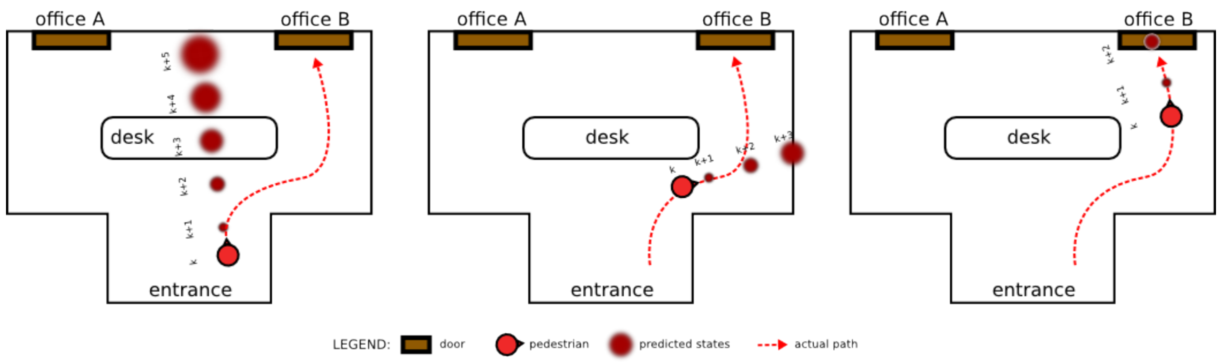


Figure 3.4: Prediction of path and goal on a complex environment, in three different time instants, using extrapolation of current state. Note that due to the structure of the environment, the direction of motion of the agent does not correspond to its intended goal.

to this complexity, predictions based only on the subject's current state will perform poorly, as shown in Figure 3.4.

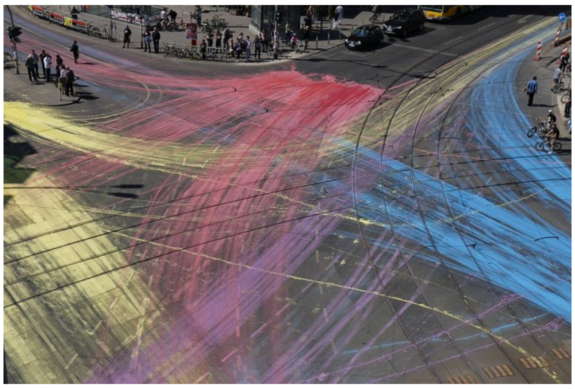
In this Figure, three images illustrate different instants of a pedestrian moving along an office room. The dotted line shows the true path of the robot. Note that due to the structure of the environment and the position of walls, doors and the desk, the subject has to travel along a S-shaped path.

Due to this pattern, the motion prediction algorithm fails to provide a correct solution on the first and second instants. Only on the last image, the algorithm produces an accurate result. This illustrates that linear extrapolation is not well suited to be used in complex environments, providing correct predictions only on short-term.

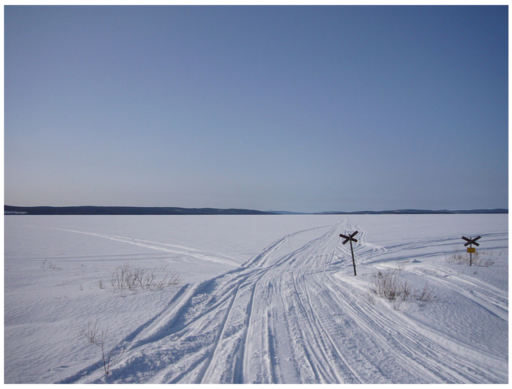
3.1.3 Motion Prediction Based on Learning - Typical Paths

I shall be telling this with a sigh
Somewhere ages and ages hence:
Two roads diverged in a wood, and I?
I took the one less traveled by,
And that has made all the difference.

(Robert Frost, 1916, page 9)



(a)



(b)

Figure 3.5: (a) An artistic intervention on a crossroad in Berlin showing typical paths of cars (*Painting Reality* ©IEPE); (b) a frozen lake, showing snowmobiles' typical paths (*frozen lake in lapland* ©ezioman).

In order to overcome the limitations of motion prediction using linear extrapolation, a different approach, based on learning, can be used. The learning-based method relies on the fact that when observing a populated region, it is possible to notice that moving agents do not move at random. Instead of that, their motion is related with points of interest on the environment (doors, stairs, tables, food, water), with optimal trajectories between those points and with rules of allowed motions (as traffic rules for cars, or dynamic constraints), which result in well defined motion patterns, that can be represented by *typical paths* Helbing et al. (2001), Tay and Laugier (2007).

Definition A typical path is a representation of groups of paths that are similar and connect the same interest points in an environment.

The typical paths can be used to train a motion prediction algorithm, so it learns how agents are likely to move in a given environment, improving its motion prediction capabilities. Figure 3.5 shows a visualization of the motion patterns created by vehicles in two situations. On the left, paint was dropped at the crossroad entrances, marking the path traveled by

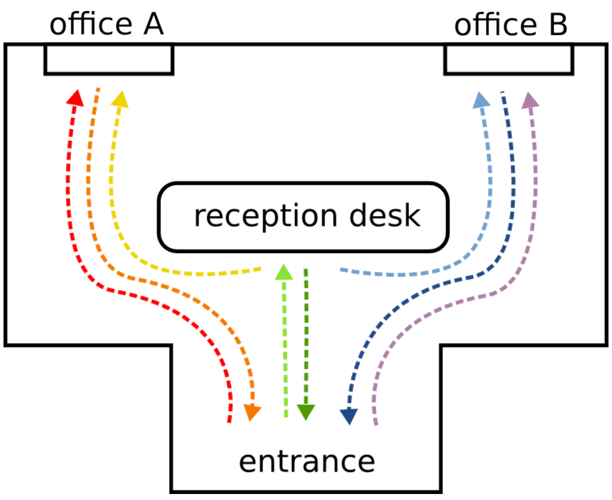


Figure 3.6: Example of typical paths at an office environment.

cars. The right image shows snowmobile tracks on a frozen lake, note that the tracks are not spread, but concentrated in common patterns, forming typical paths.

Figure 3.5a also helps to understand how learning typical paths can improve motion prediction algorithms. Suppose that one wants to predict the motion of a car coming from the right edge of the image. Although the vehicle can theoretically move to any part of the road, the blue paint shows that previous vehicles coming from that area either moved straight or turned right. So, it is reasonable to assume that there is a high likelihood that a new vehicle coming from that area will follow those patterns.

Another example can be seen in Figure 3.6, where there are eight typical paths that link interest points in a room (two office doors, a reception table and the entrance/exit). They represent the group of similar paths traveled by employees going to work and coming back from one of the two office doors, and also the path taken by persons that only reach the reception desk and then leave the room. Although individual paths ought to be unique, they can all be grouped together according to the interest points they connect.

There are two major advantages in using typical paths together with a learning framework, in order to perform motion prediction of pedestrians. First, they are able to encompass the complexity and structure of buildings in the model, improving the accuracy of motion prediction. Second, the learning framework can incorporate probabilistic techniques, which allows a measure of confidence when performing predictions of a goal and path. This is important, as predictions that suffer from a large uncertainty can be treated differently from ones with high levels of confidence.

Data used to create typical paths model can be collected from different sources: robot's sensors; surveillance cameras; an array of lasers and others. It is important to notice that the data collection process only needs to be conducted in a limited period of time, in order to gather the most common motion patterns. Once a model is learned, it will remain constant as long as the environment structure or functionalities of interest points remains unchanged. A new table or a broken elevator, for example, would change the typical paths, requiring an update of the model.

Next subsection will conduct a review on the different methods for typical paths modeling.

Modeling Typical Paths - A Review

The knowledge that humans move following motion patterns, or typical paths, have been used to improve algorithms in many different fields of research. Examples are tracking of persons, detection of unusual behaviors, in the surveillance field, and aiding architectonic projects, to improve access and exit designs. Several methods exist for modeling the typical paths, as interpolated knots, splines, sub-goals, and probabilistic approaches, and they can be roughly divided in two groups: geometric and stochastic modeling.

Geometric Approaches A simple approach for typical path modeling is presented in Makris and Ellis (2002), where they are modeled as a sequence of knots and linear interpolations associated with entry and exit points, a method that is fast and computer efficient, but have restriction in more complex scenes. For a better representation of the human motion path in complex environments, Baiget et al. (2008) cluster the most common entry and exit points and then uses concatenation of simple splines (B-splines) for an improved representation of the typical paths.

Geometric methods are fast and efficient approaches for modeling typical paths. However, they do not incorporate probabilistic information, which is important when the objective is to model the motion of humans and its uncertainties, as it is inherently a stochastic process.

Stochastic Approaches The uncertainties associated with human motion can be incorporated into typical paths models using stochastic approaches. Although more computer intensive then their geometric counterparts, they provide a more accurate description of the motion patterns and their predictions.

In Johnson and Hogg (1996), a neural network is implemented in order to learn from sequences of images and to create a model of the distribution of typical trajectories. Within this work the model is used for surveillance purposes and to detect anomalous behaviors.

The use of Gaussian Process (GP) also allows uncertainties to be incorporated on the model, better representing the human motion (Tay and Laugier, 2007; Ellis et al., 2009). Still using GP as a tool, Kim et al. (2011) create a 3D spatio-temporal flow field using Gaussian Process Regression, modeled as a dense flow field. This approach allows prediction of the path taken and also the detection of anomalous behaviors. The model incorporates not only the position but also the dynamics of the objects that moved on the scene, as accelerations, which improves the detection of different trajectories over the same path. It generates a continuous vector field, called Gaussian Process Regression Flow (GPRF), which provides a representation of motion tendencies.

In Tipaldi and Arras (2011), the time dimension is also incorporated on the model and the authors create a spatio-temporal model called *spatial affordance maps*, which is based on previous space and time information about subjects, and then used to generate paths that maximize the probability of finding certain persons.

On a more recent work, Ikeda et al. (2012) model the typical paths taken by pedestrians using a concept borrowed from human sciences, called *sub-goals*. These are points that are associated with the environment structure and architecture and transition probabilities are associated with them. According to that work, in any environment pedestrians move along a sequence of those points.

Finally, Vasquez Govea et al. (2009) developed an approach called Growing Hidden Markov Model (GHMM). The main advantage of this technique is that the *learning* and *prediction* phases are on-line concurrent processes, resulting in a *learn and predict* paradigm. This is the technique chosen to be used on this work, as it was already integrated on a broader experimental framework developed at INRIA. The next Subsection will give a detailed description of this method.

Growing Hidden Markov Model

One form of representing typical paths is by means of discrete state-space models. In such models, time is represented as a discrete variable and the motion of a subject is represented as several discrete states. A common representation is the Hidden Markov Model (HMM), where each node represent a state and the transition probability among them, $P(S_t|S_{t-1})$, are represented by edges. A characteristic of the HMM is that the states are not directly observable, but rather inferred according to an observation probability $P(O_t|S_t)$. Figure 3.7 shows a typical representation of a HMM.

In order to create a HMM, two learning phases are necessary. One is the structure learning, which consists of determining the number of nodes and the edges that connect them. The other is the parameter learning, where the probability of the prior, transitions and observations are computed. Once the learning phase has been performed, the result is a static HMM, which can be used for predictions, but will not adapt to new information. This condition may bring some limitations to the model, as it is not able to incorporate new observations to its structure or probability parameters.

To tackle these limitations, Vasquez Govea et al. (2009) developed a technique called Growing Hidden Markov Model (GHMM). It implements an approach where the *learning* and *prediction* phases are on-line concurrent processes, resulting in a *learn and predict* paradigm. The structure of the GHMMs are the same as the regular HMMs, with the difference that, as

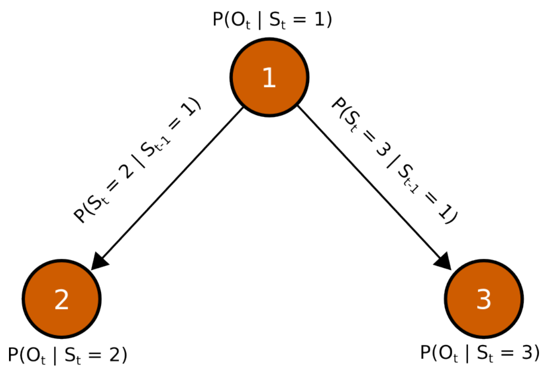


Figure 3.7: A typical three state HMM, with observation probabilities on the nodes and transition probabilities on the edges.

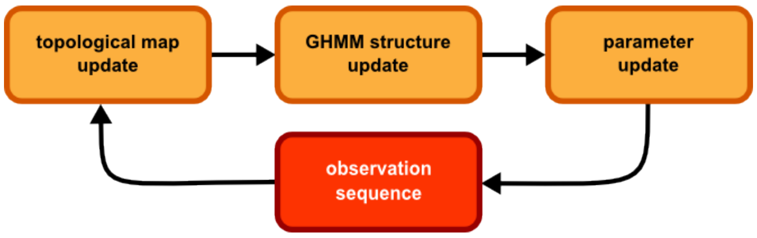


Figure 3.8: GHMM learning algorithm overview (adapted from Vasquez Govea et al. (2009))

new observations sequences are incorporated into the model, the transition structure and the number of states can change, updating the model (Figure 3.8).

The GHMM algorithm consists of the use of the Growing Neural Gas (GNG) algorithm Fritzke et al. (1995), to estimate the model structure as well as the transition probabilities of a Hidden Markov Model (HMM). As the algorithm is adaptive, it is capable of creating or removing states to cope with new observations.

Each goal in the scenario has a different HMM associated to it, representing the path taken to reach such goal. As the intention of modeling a typical path is to compare the robot's goal with a predicted subject's goal, the GHMM inherently provides an elegant solution to that problem.

Clustering of the paths is not necessary, as the model is capable to adapt its state space topology according to the observations. In this way, similar trajectories will automatically be represented by the same sequence of states, which in its turn can change, according to the more frequent observations.

When using the GHMM for the prediction purposes, it is possible to "ask" the algorithm what is the estimated current state and future states of the observed dynamic object, as well as the prediction of its goal. The output of the algorithm is a probabilistic value over a discrete grid, as can be seen in Figure 3.9. In both images, the red bars represent the probability for each of the grid's cell to be the agent's destination, where the agent position is represented by the three orthogonal axis. On the left image, it is possible to see two groups

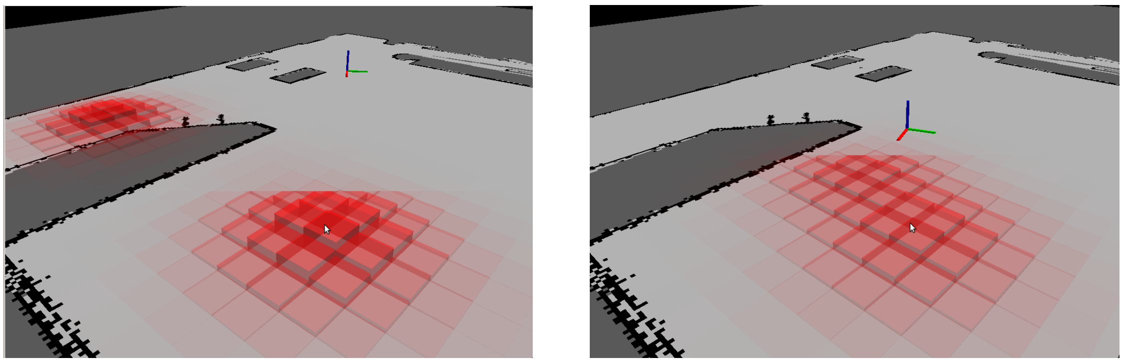


Figure 3.9: Two instants in the goal prediction of a dynamic object. The height of the bars is proportional to the probability of a cell to be the final goal

of bars, which represent two regions on the map that are likely to be the agent's goal. As the agent continues its motion, the goal prediction converges, as shown by the single region with red bars on the right image.

3.1.4 Limitations of the Reasoning-Based Technique

Although the reasoning-based approach provides a solution to the task of leader selection, it suffers from some limitations. One of them is that it relies on future predictions, that may be inaccurate. Another restriction is that the robot must know the map of the environment, in order to compute its own plan, that will be compared to the predictions of leader candidates. Due to these limitations, such methods are not suited to situations where the robot must navigate through an unexplored environment.

Besides that, even in known environments, the presented technique may have some drawbacks. In the situation where several candidates have a common predicted goal, which matches the robot's own goal, there is no clear way to chose one single leader among the several candidates. This technique also does not take into account the reactions of the leaders to the presence of the robot: in a situation where the robot is following a leader that experiences discomfort in being followed by a machine, and becomes annoyed by that. This leader may start moving faster in an attempt of avoiding the robot, for example. If the leader selection is based only on goal or path similarity, the robot may continue to follow this person, increasing the discomfort experienced by the leader.

3.2 Leader Evaluation Technique Based on Learning

As it has been shown so far, a key issue that this work addresses is how to choose a leader among several candidates. With this objective, an innovative technique has been devised, that creates a learning-based approach for leader selection, based on human experience.

The idea to use this approach comes from the fact that humans often encounter themselves in the situation where they engage in leader following behaviors and group formation, usually without much thought or reasoning. In this sense, it is interesting to try to understand what are the underlying criteria used by persons to engage and disengage in leader following behavior, so this can also be applied in the design of robots.

It is important to have in mind that, although robots can benefit from following humans, this cannot be done by the expense of their comfort. Disturbing persons while following them, violates a crucial requirement for robots existing among humans, which is to respect their social conventions. The developed method also proposes to address this issue, as the data used for training will be labeled by persons, whom can easily perceive cues associated with discomfort in other persons, as looking backwards, moving faster or moving out of the way.

It is expected that this approach bring benefits to the leader selection problem, as:

- the robot does not need to create a plan or make predictions about the candidate's intentions, being able to explore unknown environments while following someone;
- it is possible to provide individual scores for leader candidates, allowing a better basis for the decision of who to follow;
- a bad leader can be associated with situations of discomfort experienced by the subjects being followed, so the algorithm can stop following a leader, improving its acceptance.

Although numerous works have been conducted regarding the robot-human interactions and proxemics (Gockley et al., 2007; Takayama and Pantofaru, 2009; Walters et al., 2009), they are normally focused on how the humans react and feel regarding an approaching robot. However there is a lack of study on what happens when the human is also moving and being followed by a robot. In this sense, this work also launches the basis of such study, by proposing a methodology for data acquisition in this area.

Besides the objective of classifying a good or bad leader candidate, it is also important to understand what are the measurements, or features, that are relevant to that classification (distance, velocity, etc.). Because of this, the machine learning algorithm chosen is the AdaBoost, as this algorithm inherently selects the features that contribute the most to the classification process. This choice addresses two issues at once: the leader classification problem and the study of what are the features that are most important to that process. Next section will provide more details on this algorithm and how the information of features contribution can be obtained.

3.2.1 AdaBoost Classifier

Boosting is the name of a category of supervised machine learning ensemble algorithm, where several *weak classifiers* (or *weak learners*), or rule-of-thumbs, are combined in order to create a more accurate predictor, or *strong classifier*. In such category of classifiers, two important issues arise:

- how to choose or modify the training set to train weak classifiers;
- how to combine the weak classifiers.

The adaptive Boosting (or AdaBoost), developed by Freund and Schapire (1997) is a variation of the basic Boosting algorithm which addresses these issues, and its pseudocode is shown in Algorithm 1. As the input, the algorithm receives a training set $ts_n = \{(x_1, y_1), ..., (x_n, y_n)\}$ containing a vector of features $x_i \in X$ and its respective labels $y_i \in \{-1, +1\}$.

Algorithm 1 Adaptive Boosting (AdaBoost)

```
1: $D_1(i) = 1/n$                                   ▷ initialize weights
2: for $t = 1 \to T$ do
3:     $h_t \leftarrow TrainWeakClassifier(D_t)$    ▷ find a classifier $h_t$ that minimizes the error $\epsilon_t$
4:     $\epsilon_t = \sum_{i:h_t(x_i) \neq y_i}^{n} D_t(i)$    ▷ computes the weighted classifier error $\epsilon_t$
5:     $\alpha_t = \frac{1}{2} ln \frac{1-\epsilon_t}{\epsilon_t}$    ▷ computes $\alpha_t$
6:     for $i = 1 \to n$ do
7:         $D_{t+1}(i) = \frac{D_t exp(-\alpha_i y_i h_t(x_i))}{\sum_{i=1}^{n} D_t exp(-\alpha_i y_i h_t(x_i))}$    ▷ update and normalize weights of training set
8:     end for
9:     return $H(x) = sign \sum_{t=1}^{T} \alpha_t h_t(x)$    ▷ return the strong classifier $H$
10: end for
```

As an enhancement over the original Boosting algorithm, the AdaBoost maintains a set of weights on the training set, that are updated after each iteration, where $D_t(i)$ is the weight of a sample set i on iteration t. As the algorithm runs, the weight of each sample of the training is modified, with more weight given to misclassified examples and decreasing the weight of correctly classified ones. As a result, in the next iteration, the algorithm will give more emphasis in making a correct classification of previously misclassified samples.

When initializing, the algorithm gives the same weight to all ts_n samples. The next step is to compute a *weak classifier* h_n taking into account the weight distribution of the training set. The best *weak classifier* is the one that minimizes the error ϵ_t, which is the weighted sum of misclassifications. The restrictions in the choice of the *weak learner* is that $h_t(x) \in \{-1, +1\}$ and that its error rate is better than chance (smaller than 0.5).

It is interesting to choose a *weak learner* that is fast to be trained, as usually a large number of them is required. Examples of *weak learners* are decision trees, multi-layer perceptron and radial basis function. In this thesis, the decision stumps are used, which is a one-level decision tree, making predictions based on a single threshold over a single feature. Figure 3.10 illustrates a decision stump that operates over a dataset that contains two features, x_1

and x_2. In the image, the classifier is operating only over the feature x_1 and the classification is based only on the threshold θ. If the sample value is above this level, it is classified as $+1$ and if below, the class is -1.

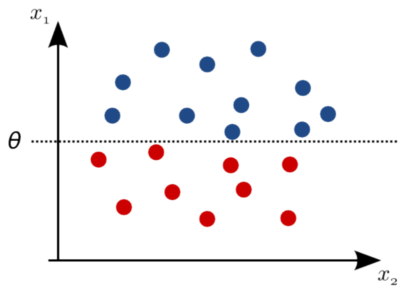


Figure 3.10: Example of a decision stump classifier operating over the feature x_1 and with threshold θ, used as a *weak learner*.

In the algorithm workflow, the next step is the computation of the confidence parameter α, an important factor that is responsible for the adaptive capability of the algorithm and is computed based on the error ϵ_t. This parameter is used in the update of the weights distribution D_t, where:

$$-\alpha_i y_i h_t(x_i) = \begin{cases} < 0, & y_i = h_t(x_i) \\ > 0, & y_i \neq h_t(x_i) \end{cases} \tag{3.1}$$

So if a training sample is correctly classified ($y_i = h_t(x_i)$), its weight is reduced, whereas if the classification fails, the weight of sample i is increased. The new weight distribution will now affect the choice of the *weak classifier* on the next iteration, favoring one that manages to correctly classify the difficult cases.

For each training sample, the weight update is computed based on α_t and then normalized. Finally, once all the iterations were computed, or if an error threshold has been reached, the algorithm returns the final *strong classifier* H. The result is a combined ensemble of *weak classifier* weighted by the factors α_t, which can be seen as a measure of the contribution of each one of the *weak classifiers*, producing a signal weighted linear combination of them.

Among the advantages of the AdaBoost algorithm is the exponential convergence of the training error to zero and good generalization properties, besides the capacity of the algorithm to perform feature selection.

3.2.2 Features Study

It is interesting to hypothesize on which features are the most important in a classification problem, for further investigation and understanding of the underlying phenomenas.

This work is a practical example of this, since learning and understanding what matters in following-leading engagements is essential to the development of a more refined theory on human-following behaviors for robots.

In classification problem, determining which features should be used during learning is not a straight-forward task, as usually it is not known, beforehand, which features are the most relevant for separating classes.

One form of addressing this issue is by creating an extensive list of features, to guarantee that at least some of them are relevant to the problem in question. But this may result in a very large number of feature, elevating computer costs required for training, with many of them being redundant or irrelevant.

Several methods exist that perform the reduction of the features space, in order to obtain more efficient classifiers. When the dimension of the feature vector is small, a brute force method can be used, where all the possible combinations of features are tested and the best combination is selected.

However, as the dimension of the feature vector grows, an automatic selection approach is required. By using the Principal Component Analysis (PCA) method, features are combined to produce new sets of uncorrelated features, or principal components. For an example, if two features are highly correlated, they can be combined in a new single feature that has a low correlation with the remaining. However, in the current work, this method is not interesting, as the new combined features will have no real-world meaning, not contributing to the comprehension of the human-following problem.

According to Susnjak (2009), AdaBoost has a feature selection built into it when using decision stumps as weak learners. As in each iteration, a weak classifier is associated with only one feature, the contribution parameter α_t and error ϵ_t are measurements of the importance of the feature used in a weak learner for the classification problem. This provides a form of quantitative evaluation that can be used as a criteria for feature selection.

As a result of feature selection, a reduction of the number of features used can benefit the time spent for training and predicting classes, which is an important requirement in on-line systems. Besides that, and more relevant to this work, is the possibility to draw conclusions about why some features are more important than others, as this can help the understanding of the criteria used by humans in following behaviors.

According to Tsuchiya and Fujiyoshi (2006) and Drauschke (2008), the contribution of each feature in the general classification problem can be measured by summing the number of times each feature is used, weighted by the α_t of the weak classifier that uses the feature in question. Differently from the aforementioned works, here, this value will be normalized by the sum of all α_t, to obtain the ratio of the contribution of each feature.

$$CR(f_i) = \frac{\sum_{t=1}^{T} |\alpha_t| \delta[F(h_t) - f_i]}{\sum_{t=1}^{T} |\alpha_t|} \quad \text{where} \quad \delta[n] = \begin{cases} 0, & n = 0 \\ 1, & n \neq 0 \end{cases} \tag{3.2}$$

Where $CR(f_i)$ is the contribution ratio of feature $f_i \in \{1, 2, 3, ..., i\}$, α_t is the confidence parameter and $F(h_t)$ is a function that returns the identification (number) of the feature chosen by the weak learner h_t. Finally, δ is the Kronecker delta, which outputs 1 in the case the weak learner h_t was created using feature f_i and 0 otherwise. As a result, $C(f_i)$ will be the weighted sum of the number of times that the feature of interest f_i has been used by a weak learner. And this will be the measurement used to evaluate features contribution.

3.3 Contributions on Leader Selection and Following

So far it has been discussed how existing techniques can be changed and used to qualify a candidate to became a leader for the robot. In this section, it will be discussed how use that information to perform leader selection and following.

The contributions presented here are a rule based system for leader selection based on reasoning, and a voting scheme for leader selection based on learning. Another development is how these methods are then integrated in a navigation system that can take advantage from leaders, when the situation is adequate.

The system setup was built using the Robot Operating System (ROS) Quigley et al. (2009) architecture. This implements a modular system, where several modules run separately and communicate by means of TCP/IP connections. Such architecture gives a series of advantages, mainly due to its flexibility, allowing modules to be easily incorporated or subtracted from the system. Another great advantage is the standardization of messages format, an unified form of recording the data and also several tools for aiding data processing and visualization.

To simulate the robots and agents, the STAGE multiple-robot simulator Vaughan (2008) was used, which is capable of simulating several robots that move and sense in a $2D$ environment. This environment has been integrated with ROS, where all the motion planning algorithms and the heuristics of leader decision were developed.

3.3.1 Leader Selection Methods

This section presents the methods used for leader selection and following in the two types of experiments conducted. In the reasoning-based experiments, the choice of the leader is based on the prediction of the subjects' future state and on a set of rules. While on the learning-base experiments, a classifier together with a voting system give a score for each candidate, and the one with the higher score is selected as a leader.

Reasoning-Based

The workflow of the reasonig-based approach is shown in Figure 3.11, where it can be seen that STAGE publishes the state of the robot and of the agents to the other ROS modules, and receives motion commands that produce a change on those states. The pedestrian simulator module, that will be described in Section 4.2.2, is responsible for generating the behavior of a group of pedestrians.

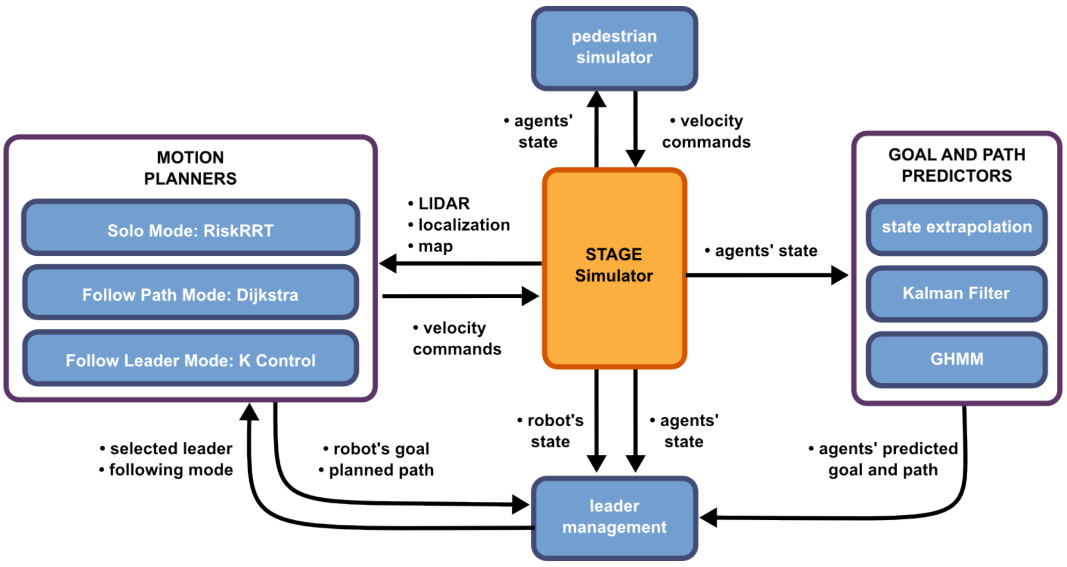


Figure 3.11: Integration of STAGE with ROS modules used in experiments for the reasoning-based approach.

The goal and path predictors group can implement different algorithms, for the prediction of the goal and path of the simulated agents, and that information is passed to the leader management module. This module also receives information about the robot's state, the agent's state, the desired destination of the robot and a precomputed path to that goal.

Based on that information, this module will decide if the robot should engage (or disengage) in a leader following behavior, and send the selected leader identification and the selected following mode to the motion planners group. Finally, one of the modules inside that group will generate a motion plan either to the robot's final destination or for it to engage in a follow-the-leader behavior, according to the following mode chosen by the leader management module.

To decide which, among several leader candidates, is the best one and should be followed by the robot, a number of rules have been defined, which are illustrated in Figure 3.12, and are described below:

- First there is the goal similarity rule, that states that a good candidate must go to a place near the robot intended destination. In the image, the orange candidate is promptly discarded as a leader candidate, as its goal falls out of the goal similarity threshold.

- Second, the leader candidate must be moving, so any subject with velocity inferior to $0.5m/s$ is not considered.

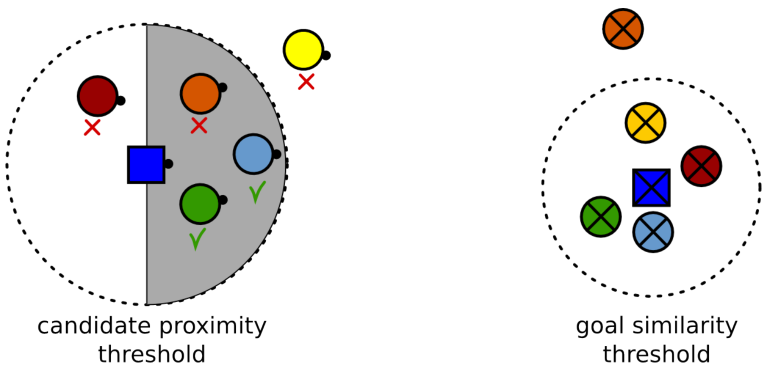


Figure 3.12: Visualization of leader selection rules. The robot is represented by the blue square and its goal by the blue crossed square. The candidates are represented by colored circles and their goals as crossed circles, with the respective color.

- Next, there is the candidate proximity rule, created to guarantee that only subjects close enough for the robot to profit from their motion would be chosen. Again using Figure 3.12 as reference, the yellow candidate would be eliminated according to this rule.
- Finally, the candidate must be closer to the robot's goal, then the robot itself. This is to make sure that the leader is in front, or at least beside the robot, because otherwise the robot would have to turn around to follow a leader. This rule is represented as the shaded semicircle centered on the robot. This would eliminate the dark red candidate, as it is behind the robot.
- Among the remaining candidates, the one that is closer to the robot would be chosen as leader, which is, in this case, the green one.

These conditions are constantly being evaluated, which allows the robot to switch leaders during the motion execution. For example, if a leader stops, or if a candidate appears between the robot and the leader, the robot can adapt to that situation. The following experiments will test these rules, as well as the multi-mode navigation, explained in section 3.3.

Learning-Based: Voting Scheme for Leader Score

The developed framework uses real data acquired from previous experiments and implements a simulated robot that will classify and eventually follow subjects that had their motion recorded. In this way, the simulated robot will extract and classify features from real data, but in a virtual world. The advantages of this proposal is that the data used is not simulated, but real, and measurements are taken with respect to the virtual robot position, so they are always different from the features used in the training of the classifier. The workflow of this framework is presented in Figure 3.13.

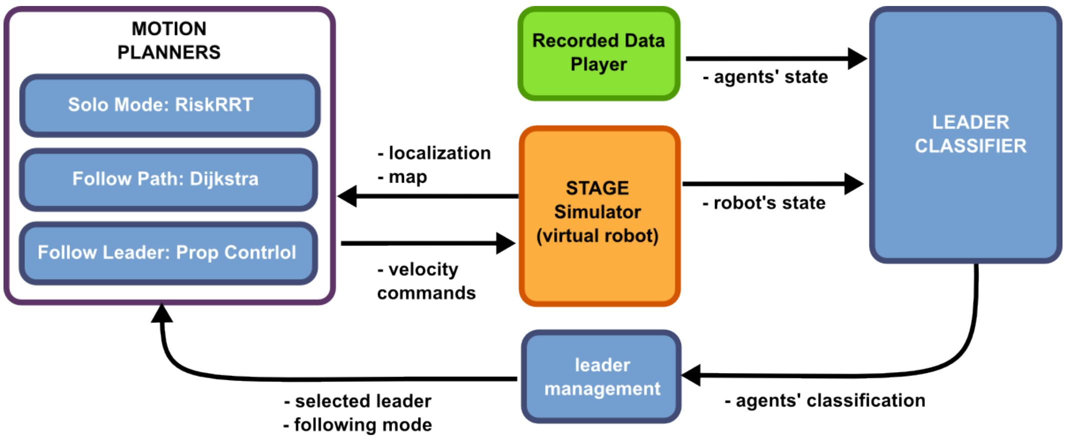


Figure 3.13: Integration of STAGE with ROS modules used in experiments for the learning-based approach.

Differently from the previous case, here there is no simulation of pedestrians, as this data is only played back from recorded experiments. The STAGE simulator is only used to simulate the robot position and motion in the virtual world. A leader classification module receives information about the state of the agents (from the playback) and of the robot (from STAGE), and then extracts features and classifies each subject.

This information is passed to the leader management module that will select the best leader, if any, and informs the motion planning group according to the situation. In the case a leader is found, the robot may engage in a following behavior.

The classifier presented in section 3.2 performs classification only on instantaneous features. In each instant it produces a binary output meaning that a candidate is a good or bad leader. But that behavior is undesirable for online applications, because when a robot follows somenone, it cannot constantly switch between two modes (following navigation/stand-alone navigation). So a type of hysteresis is desired, where past events have influence on the decision of the leader choice.

To address this situation, a voting scheme has been created, that results in a score for each leader candidate, effectively taking into account the past classifications. This is implemented in the leader management module. This new layer of processing produces more stable classifications of leader candidates, so they can be used by a navigation algorithm. In this voting scheme, an instantaneous classification of a subject as a good leader counts as a positive vote with the value 0.01 and a classification of a bad leader casts a vote with the value -0.5. The maximum value of a candidate's score is 1 and the minimum -0.1, while the threshold for considering a candidate a possible leader is any score above 0.

All these values had been empirically configured, so, in the form that this voting scheme is implemented, it takes several instantaneous classifications of a good leader to raise a subject's score. So only a constant, long term, good behavior will make someone a good leader. While, in the other hand, even a reduced number of bad leader classifications can highly penalize subject's score, as shown in Figure 3.14. The meaning of this is that the robot will slowly gain confidence in following someone, but if this subject exhibits a behavior that does not correspond to a good leader, the robot will quickly distrust this person.

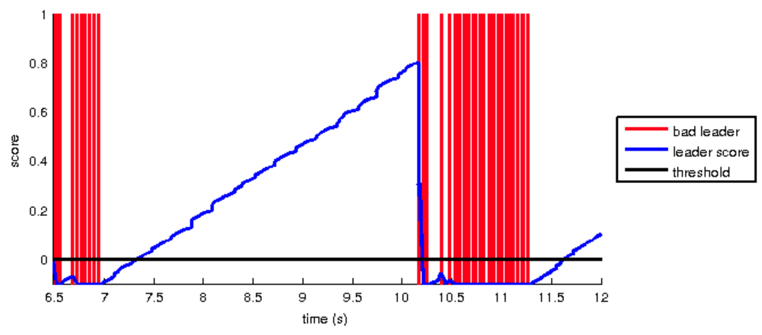


Figure 3.14: Example of voting scheme for leader score.

To consider a subject as a possible leader, his/her score should be higher than 0. In the case that more than one candidate is present, the algorithm would normally give different scores for each subject, and the leader would be the one with the highest score. This system allows the algorithm to automatically switch among leaders and follow the *best* leader in a scenario.

3.3.2 Navigation and Leader Following Methods

The developed system consists on a multi-mode navigation framework, that can use three different navigation strategies according to the situation. When no suitable leader is found, the **solo** mode is active, and the robot uses a motion planning algorithm suited for navigation in dynamic environments called RiskRRT. In the case a leader is found, using one of the previously mentioned techniques, the navigation framework can activate one of the two modes: **path follow** or **leader follow**.

The activation of one mode or another is based on a distance threshold, empirically set at $4m$. Figure 3.15 illustrates how the algorithm switches among the three navigation modes.

In the case of the **path follow** mode, the path traveled by the leader is recorded and sampled according to a fixed time step. These samples are then used as subgoals to the Dijkstra's shortest path algorithm, so the robot can follow the leader's path, as illustrated in Figure 3.16.

In this way, even if not very close to the leader, the robot can avoid obstacles that it is not able to detect, as spilled coffee or holes on the floor, and benefit from the space created by the leader motion.

While on the *leader follow* mode, the robot should follow the leader directly. In this case, two independent proportional controllers are used to control the distance and the heading of

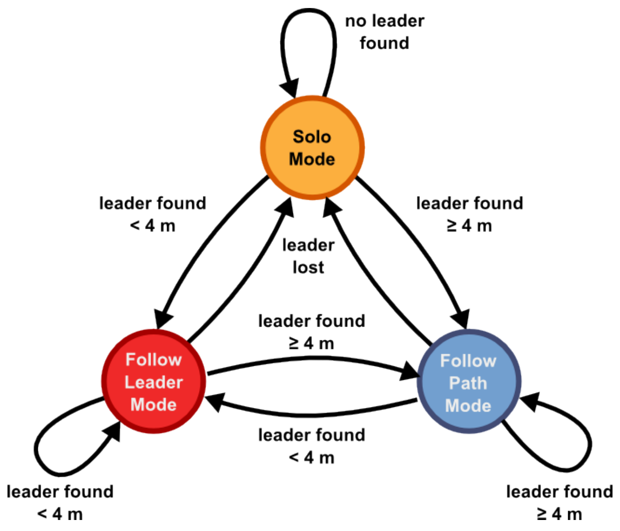


Figure 3.15: Diagram of the multi-mode navigation system.

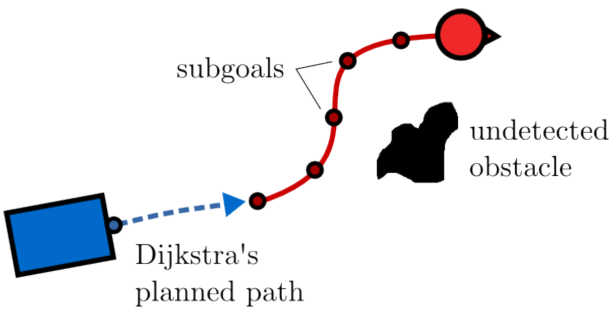


Figure 3.16: Illustration of the path following navigation mode.

the robot towards a leader, as shown in Figure 3.17.

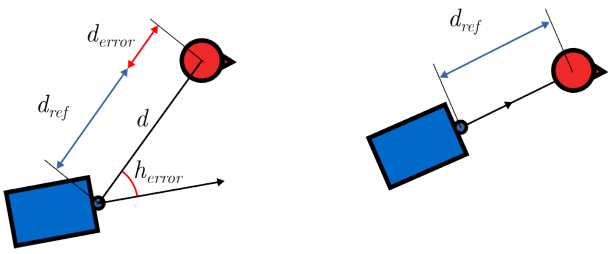


Figure 3.17: Illustration of the leader following navigation mode.

The velocity commands of a differential wheeled robot are computed with the following equations:

$$v_\theta = h_{error} * K_\theta \tag{3.3}$$

$$v_{lin} = (d - d_{ref}) * K_{lin} \tag{3.4}$$

Where v_θ is the angular velocity and v_{lin} the linear velocity of the robot. d_{ref} is set at $2m$, which is a distance that respects the social space of the person being followed (Hall, 1966). In the case that $d < d_{ref}$, v_{lin} is set to 0. This following mode is considered more natural, according to Gockley et al. (2007), and due to the closeness of the robot to the leader, it can also benefit from his/her motion and avoid undetected obstacles.

Navigating Without a Leader: RiskRRT Motion Planner

The motion planner used when the *solo* mode is active is the RiskRRT motion planner, developed by Fulgenzi et al. (2008, 2009). This algorithm has been developed explicitly to be used in dynamic environments, and is based on the classic RRT algorithm (LaValle and Kuffner Jr, 2001). The difference being that it uses the notion of the **risk** of collisions with moving agents to compute the best motion plan. It combines a part dedicated to perception (of static objects and moving agents) with another part for planning trajectories, with navigation and planning performed in parallel.

During the execution of the algorithm, the configuration-time space is searched randomly, and a tree T is grown from the initial configuration until a feasible path to the goal configuration is found. The algorithm randomly chooses a configuration P in the $\mathcal{C}$*-space* and extends an existing node of its current search tree towards that point. To speed up the exploration towards the final goal, once every 100 times, the goal itself is chosen as P.

The node chosen for extension is the one where the resulting partial path has the lowest risk of collision with moving agents. The likelihood of each partial path can be expressed as the multiplication of the independent probability of collision with static (P_{cs}) and dynamic components (P_{cd}), where q_N is the candidate node, for a partial path:

$$L_\pi(q_N) = \prod_{n=0}^{N}(1 - P_{cs}(q_N)) \cdot \prod_{n=0}^{N}(1 - P_{cd}(q_N)) \tag{3.5}$$

The prediction approach for forecasting the position of moving obstacles in the near future can be performed using different approaches, as for example GPs or GHMMs. With the information of probable occupied positions in the future, the algorithm can anticipate the behavior of the agents.

After the tree expansion phase, the motion plan is computed taking into account the sequence of nodes that minimizes the risk of collision with other agents while at the same time bringing the robot closer to its goal. Here, the risk of collision can be seen in this case as a measure of the feasibility of a path, with the maximum accepted risk specified as a threshold.

In more recent developments, Rios-Martinez et al. (2011) created a technique called Social Filter (SF), which consists of modeling the interactions among humans (o-space) and also their personal spaces (p-space) as risks of disturbances, represented by Gaussian distributions as shown in Figure 3.18.

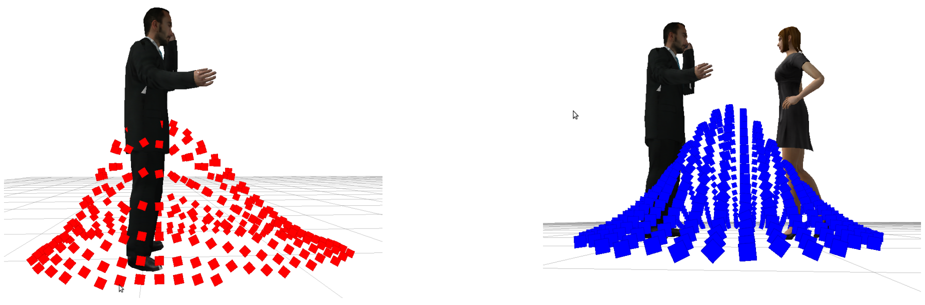

(a) (b)

Figure 3.18: The SF is modeled as a Gaussian distribution, with the risk represented by the height of the points: (a) p-space, the personal space, which is centered in the subject; (b) o-space, the interaction space, with the maximum risk of disturbance located at the center of the interaction.

By integrating such technique with the RiskRRT algorithm, the authors develop an approach that is able to respect social conventions. This is accomplished due to the increased risk of nodes generated close to humans and close to social interaction regions, and as a result, the robot can behave in a socially acceptable way. With this enhancement, the risk function becomes a measure of safety and also of human friendly navigation. The effects of the SF on the RiskRRT motion planning is shown in Figure 3.19.

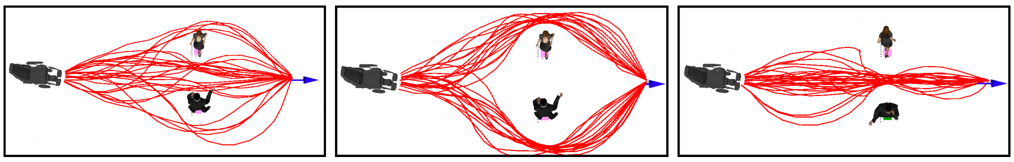


Figure 3.19: Several RiskRRT plans with Social Filter (SF) deactivated; SF activated, with humans facing each other; and SF activated, with humans facing away from each other.

The pseudocode of the RiskRRT is described in Algorithm 2, which runs a loop until the robot reaches its goal.

When the algorithm starts, it enters a loop waiting for a goal to be received. Once that occurs, a RiskRRT object is initialized with information about the current robot state, the received goal, an agents structure, the map and the current time. A flag is changed to signal that the algorithm has been initialized.

In the next loop, with the RiskRRT object already initialized, a verification if the robot has reached its goal is conducted. If this condition is not satisfied, the current robot state is retrieved and a function retrieves the prediction of motion of the agents in the scene. Then RiskRRT tree is updated with the most up-to-date information about the robot's state, the agents, the map and the current time. In this step, the root of the tree is updated according

Algorithm 2 Risk-RRT

```
1: procedure RiskRRT
2:     if $newGoalAvailable$ then
3:         $robotState \leftarrow read()$
4:         $goal \leftarrow read()$
5:         $agents \leftarrow clear()$
6:         $map \leftarrow read()$
7:         $t \leftarrow clock()$
8:         $riskrrt.initialize(robotState, goal, agents, map, t)$
9:         $goalRetrieved = true$
10:    else if $goalRetrieved$ then
11:        $goalReached \leftarrow isInGoal(robotState, goal)$
12:        if $!goalReached$ then
13:            $robotState \leftarrow read()$
14:            $t \leftarrow clock()$
15:            $map \leftarrow read()$
16:            $agents \leftarrow predict()$
17:            $riskrrt.update(robotState, goal, agents, map, t)$
18:            while $timeToGrow$ do
19:                $riskrrt.grow(agents, map, t)$
20:            end while
21:            $riskrrt.checkCollision()$
22:            $riskrrt.findBestPath()$
23:            $riskrrt.publishPath()$
24:        else
25:            $riskrrt.finish()$
26:            $break$
27:        end if
28:    else
29:        wait goal
30:    end if
31: end procedure
```

to the robot's position, and nodes that are no longer accessible are eliminated.

After that, the algorithm enters another loop where it will expand the tree branches during a certain amount of time. Once that time has elapsed, the RiskRRT object performs a collision check, then computes the best path that minimizes the risk of collision and publishes that path for the motion executer. The execution and the planning are two parallel processes. While the robot follows a computed trajectory, the motion planner is already entering a new loop. This process continues until the condition of the robot reaching its goal is satisfied.

Figures 3.20 and 3.21 show experiments conducted with the RiskRRT algorithm. In these cases, the prediction is based on the linear extrapolation of the agents' current state, represented by a sequence of colored squares. Each square represents the estimated position (state) of the human in the future, in a given time-step. The distinct colors represent the amount of time that the agent is expected to take to reach each state. The colored circles are the exploration nodes of the tree built by the RiskRRT algorithm, and they are coded in the same fashion as the colored squares. Each node represents a possible configuration for the robot and its color is the time the robot will spend to reach each configuration. The size of the circles is a representation of the risk associated with each configuration, so they are bigger when circles and squares of the same color are close to each other.

In Figure 3.20 a robotic wheelchair and a simulated human have opposite goals and must pass-by each other in a narrow corridor to reach their destinations. The RiskRRT algorithm plans based on the predicted path that will be taken by the human in the scene.

As can be seen in the images, when the colors of the squares and of the circles coincide, the circles grow in size. The reason is that both human and robot are expected to reach those states in approximately the same time, so they will be close to each other, producing an increased risk of collision, represented by the larger circles.

The RiskRRT algorithm will always choose the sequence of nodes that have the smaller associated risk, to avoid potential collision and interference with humans. In the example of Figure 3.20, it automatically chooses one of the sides of the corridor to pass by the simulated agent. In its turn, the agent's motion is governed by the pedestrian simulator described in section 4.2.2, so it will react to the presence of the robot once close enough to it, slightly moving to the free side of the corridor.

In the experiment of Figure 3.21, the robot must move along a crossing, where two humans pass in front of its way. The RiskRRT algorithm creates a motion plan (red line) to the goal (blue arrow) that leads the robot to pass by the back of the first person, and then moves slower, along a straighter path, to pass after the second person. With this strategy, the algorithm minimizes the risk of collision and the discomfort caused by the robot for the two pedestrians. Frames at the right of the figure show a zoomed image of the future robot positions (increased circles) when close to the predicted position of pedestrians (squares) in the same instant (same color), which translates as an increased risk of collision of discomfort for the persons.

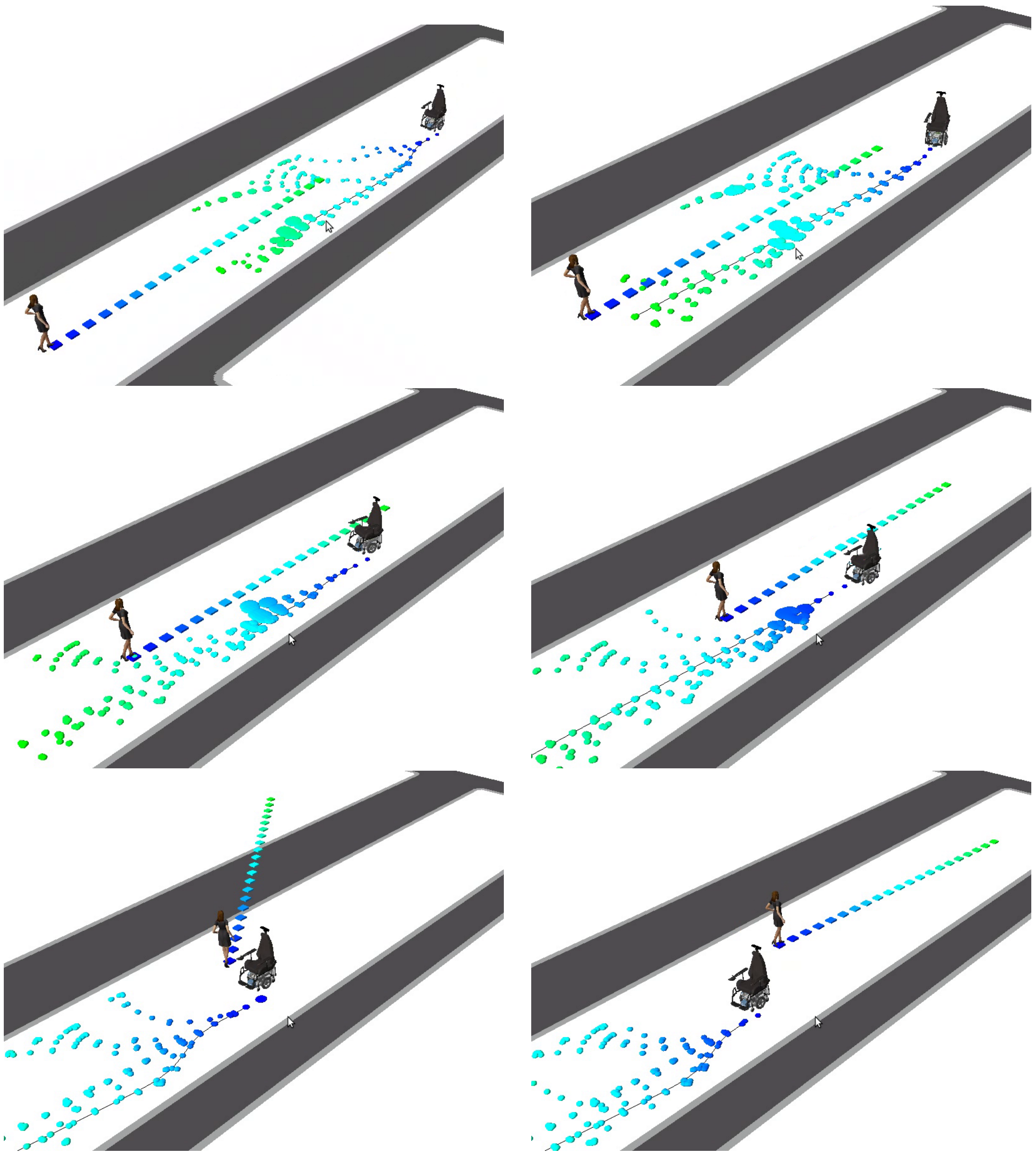


Figure 3.20: Experiment with the RiskRRT algorithm in a narrow corridor, where a robot must cross with a human.

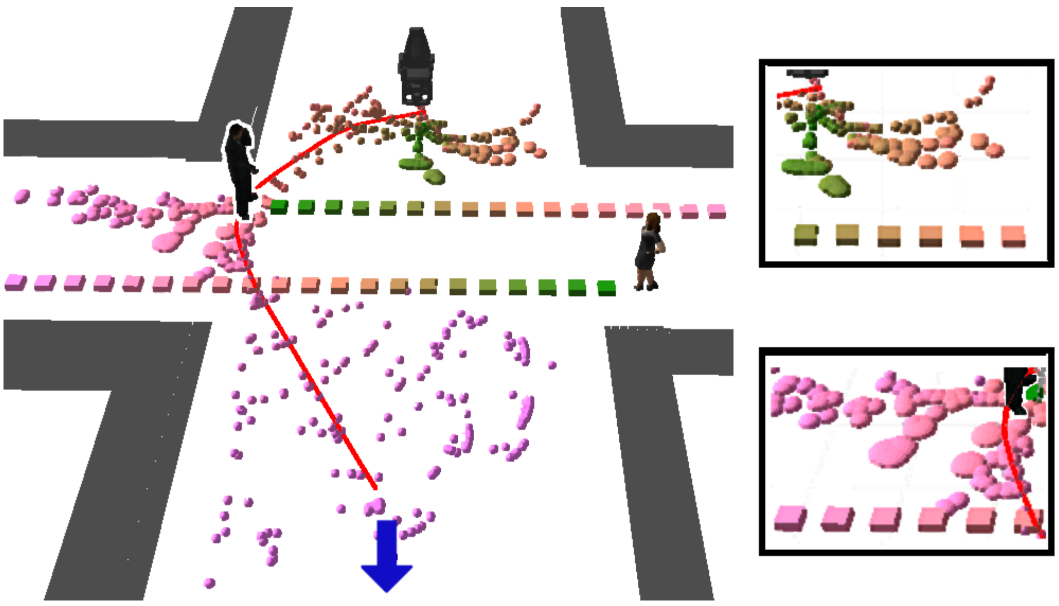


Figure 3.21: Experiment with the RiskRRT algorithm in a crossroad, with two pedestrians crossing in front of the robot. The frames show a zoomed image of exploration nodes where there is an increased risk of collisions, represented as larger circles.

Chapter 4

Experiments on Leader Selection and Following Based on Goal and Rules

This chapter describes the experiments conducted within two aspects of this research: how to select leaders and how to follow them. Here, the destination goal of the subjects is used as a criteria to filter only the candidates that will move to a destination close to the robot's goal, as discussed in Section 3.1. Then, the filtered candidates must satisfy a set of rules, that will result in a leader selection.

Three types of experiments were developed here. In the first, it is demonstrated the use of the GHMM method for leader selection based only on goal prediction, where real data is used train the GHMM and to replicate real agent's behavior in a simulated environment. Following this, the use of a trajectory generator, together with a pedestrian simulator, allows to extend the applications of the GHMM to a virtual world, where the robot identifies and follow a leader among moving agents. Finally, the last experiment merges the rule-based leader selection with the multi-mode navigation framework described in subsection 3.3.2, using the pedestrian simulator to create realistic experiments.

4.1 Experiments Integrating Real Data and Simulation

These experiments will use the GHMM method to model and predict the typical paths and goal of pedestrians. After this step, a virtual test scenario is created, containing a virtual robot and "semi-real" agents, which are actually recorded data from real persons, played back in the virtual world. Although their motion replicates the real way a person would move, they will not be able to react to the presence of the robot. The virtual robot uses a modified version of the RiskRRT algorithm to navigate and follow the *semi-real* leaders.

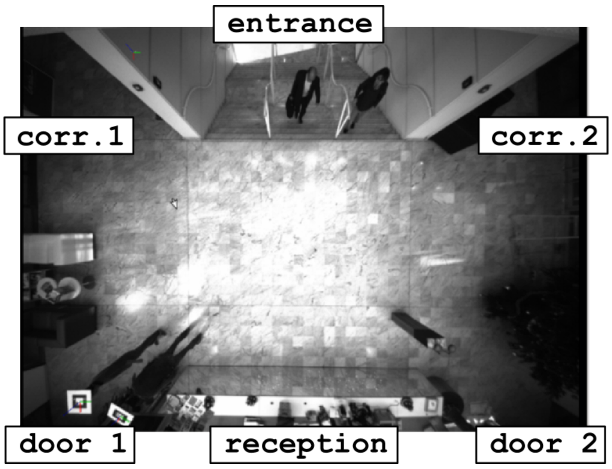


Figure 4.1: INRIA Rhône Alpes entrance hall, with labels over its typical interest points: the doors, corridors, entrance and reception desk.

4.1.1 Data Acquisition

In order to train a GHMM model, real data with the typical trajectories of an environment was logged. Humans are able to detect series of cues that may help them infer the goal of a subject, as the gaze, head and body posture, velocity and others. Based on the information they can gather from someone, and based on previous knowledge of the human behavior and of the scenario, they are able to predict, at some extent, the destination of a persons.

This task is more complicated when relying on sensors attached to a robot and on the interpretation of the measurements. Although there is a growing number of algorithms that can detect body and head orientation, as well as gaze direction, some still lack robustness or are developed for indoor applications only.

In the current experiment, only the velocity and orientation (regarding the motion, and not the body's pose) of tracked subjects will be used for the leader selection algorithm. These were chosen as they can easily be acquired by different types of sensors, as cameras or laser scanners, which provide robust measurements, that are easier to obtain in dynamic scenarios, as in Arras et al. (2008), for example.

The system tracked persons that walked along the main hall of INRIA Rhône Alpes building (Fig. 4.1). This area is an interesting choice as it is the place of passage of a large portion of the building's employees, besides offering a set of interest points, as the reception desk, the building's entrance and the two doors that provides access to the restricted area where offices are located.

In order to provide a robust and fast deployment tracker system, the position and orientation of the subjects in the environment were obtained using the *ARToolKit* augmented reality system (Kato et al., 2000). During this experiment, fiducial markers were worn as hats by subjects, as shown in Figure 4.2 and the images were obtained by an overhanging camera

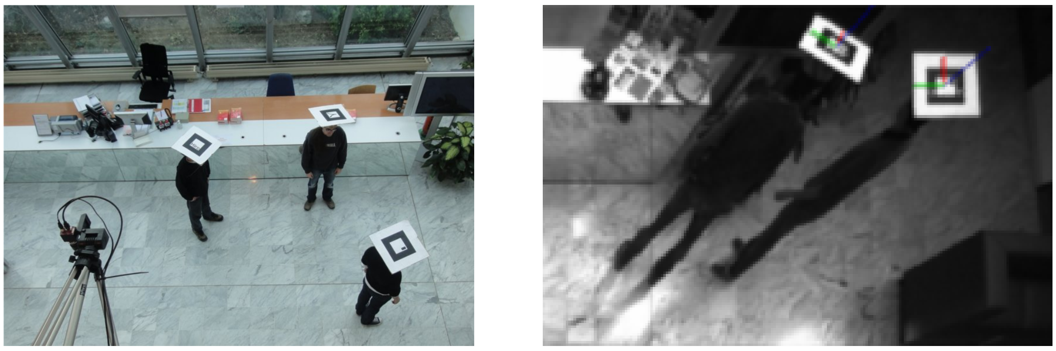


Figure 4.2: (a) Subjects wearing hats with markers and the overhanging camera (left bottom) at INRIA's hall. (b) Detail of the tracking system showing a three-axis reference system drawn over each marker.

fitted with a wide angular lens. This choice of lens allowed a large field of view, but it also produced large distortions on the borders of the images. As the tracking system searched only for squared shaped markers, a calibration process was performed, using the checkboard technique, in order to correct the distortions in the acquired image.

Such approach may seem strange, as it requires participants to be conscious about the experiments they are taking part in. And other methods would have seemed more appropriate, as using laser scanners to perform detection, for example. However, this setup was developed to also serve other researchers and one of the requirements was the detection of the subject's orientation even when they were not moving, which was easily provided by the implemented system.

Among the advantages of such system are the low cost involved (a single camera), the large coverage area, even with the presence of obstructions and the ease of deployment. However such system also suffered some disadvantages, as the large influence of light on the markers detection, the necessity of calibrating the position of the camera with several degrees of freedom (due to its tripod support) with respect to the scenario's map and also the large distortions on the borders of the image.

During tests, volunteers were asked to move naturally among interest points in the environment, as the entrance of the hall and the two doors. Figure 4.3a shows a sample of these trajectories, with the interest points represented as red circles. Each participant received a script of the interest points they were allowed to visit, but they were free to choose when and which one to visit. The objective was that all the interest points were visited and to distribute the task among the participants. The paths traveled by the experiment's participants were processed by the tracking system, which extracted the tracks generated by them as a sequence of points, resulting in a total of 107 tracks of different sizes.

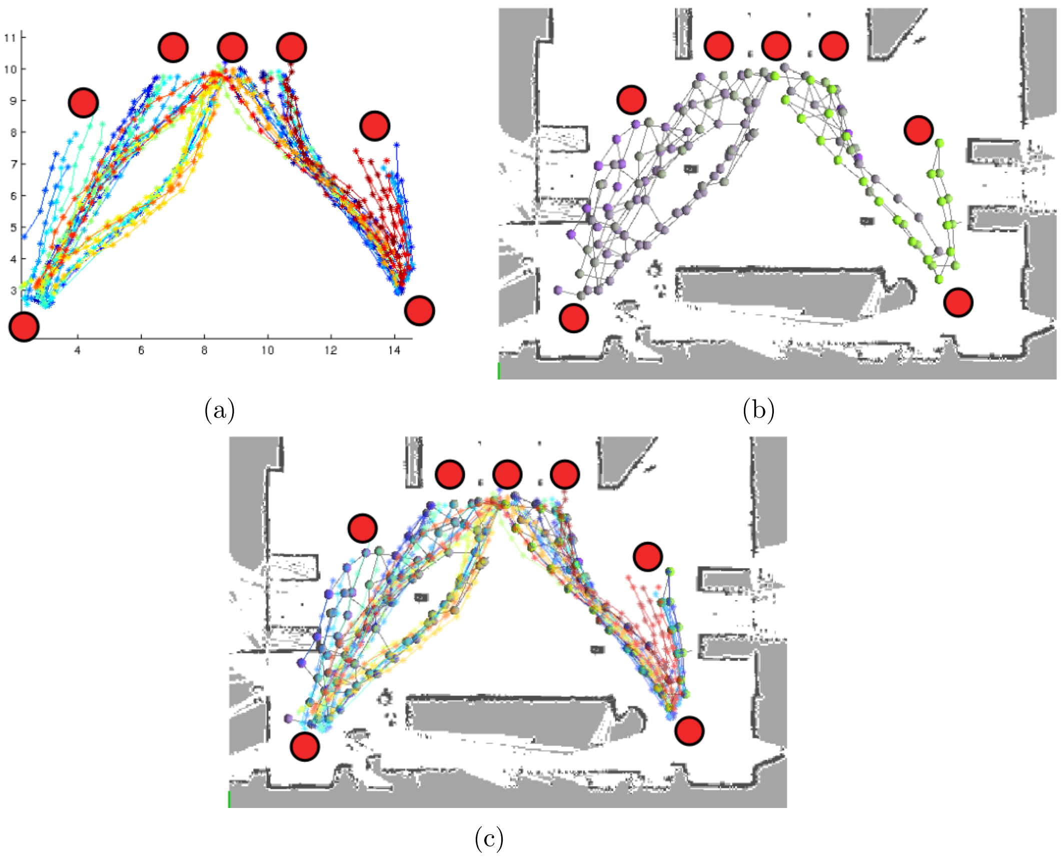


Figure 4.3: Images of the recorded training data and the GHMM structure, the red circles represent the interest points of the scenario. (a) Recorded trajectories of subject's moving among interest points. (b) Resulting GHMM structure, showing states as nodes and transitions as edges. (c) Superposition of the training data set and the learned structure.

4.1.2 Modeling Typical Paths with GHMM

These vectors are then presented to the GHMM algorithm which processes each track and computes the best number of nodes and transitions that represent the received information. As new tracks are processed, the algorithm decides if new states are required to represent the new information or if existing states are sufficient for the representation. It also adapts the connections among nodes and the transition probabilities. Ideally, the algorithm would generate several different HMMs, each one linking a start point to a destination, according to the samples presented. Figure 4.3b shows the resulting GHMM structure, with different colors representing the individual HMM that links a start point to a goal. Figure 4.3c presents a superposition of the training trajectories and the generated structure of the GHMM.

It is important to notice that the GHMM algorithm requires manual settings of some parameters, as the minimum distance to insert new nodes, the observation interval, the grid discretization and others. So, these parameters have to be empirically set, according to the requirements of each problem. A fine discretization of the map, for example, would bring more accurate predictions at the expense of a higher computational load.

4.1.3 Tests

A robot was simulated using STAGE (Vaughan, 2008), while the scenario agents represent real data recorded from the motion of humans, in the INRIA's entrance hall (Figure 4.1). Both in Figures 4.4 and 4.5, the robot is represented as a light gray rectangle. The obstacles are colored dark gray and encompass walls, desks and sofas.

The circles represent persons and the triangles are their respective predicted goals. They have a letter associated to identify their colors (Red, Green and Blue). The robot goal is marked with a cross, located at the lower left of the test area. Finally, the dots represent the RiskRRT exploration nodes and the solid line is the path chosen by the algorithm.

In the first test, shown in Figure 4.4, three humans start to move just in front of the robot, each moving toward a different goal. After some iterations, as the subjects start to move in the scenario, the prediction algorithm gives a goal estimation for two of them (red and green). Based on that estimations, the leader following algorithm makes the choice to follow the red subject, as its predicted goal is similar to the robot's.

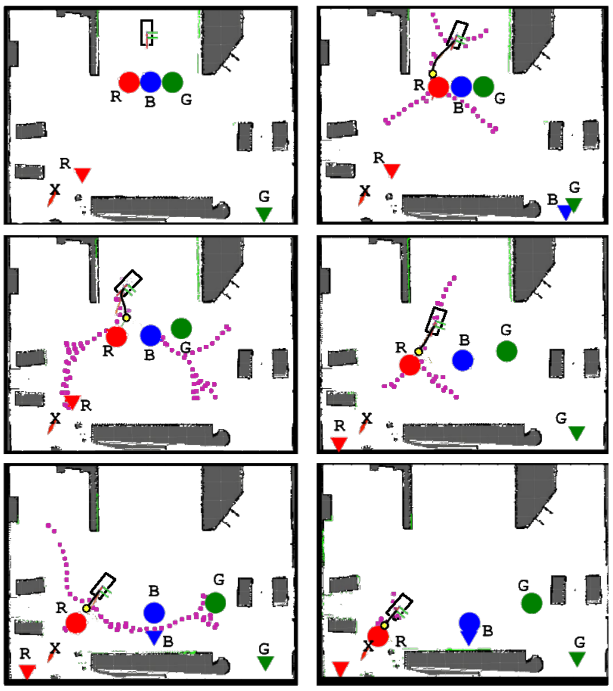


Figure 4.4: Leader selection and following using the goal similarity criteria, predicted by a GHMM. The robot is represented by the light gray rectangle and its goal by a X. Three leader candidates are represented by circles with letters R, G and B. The predicted goals are the triangles with the corresponding letters.

The objective of the second test is to evaluate the benefits of following a leader in order to avoid agents moving in the opposite direction and is shown in Fig. 4.5. The way the robot selects and follows a leader occurs in the same fashion as in the previous test. The robot goal is again in the left bottom corner of the image, but here there are now two humans that move from the door to the stairs, in the opposite direction of the robot's desired trajectory.

After the leader is chosen, the robot starts to follow him/her. As the leader approaches the two humans moving in the opposite direction, they naturally give room for him/her to pass. The robot benefits from this space and is able to continue to move without the need to take evasive measures to avoid the two incoming persons.

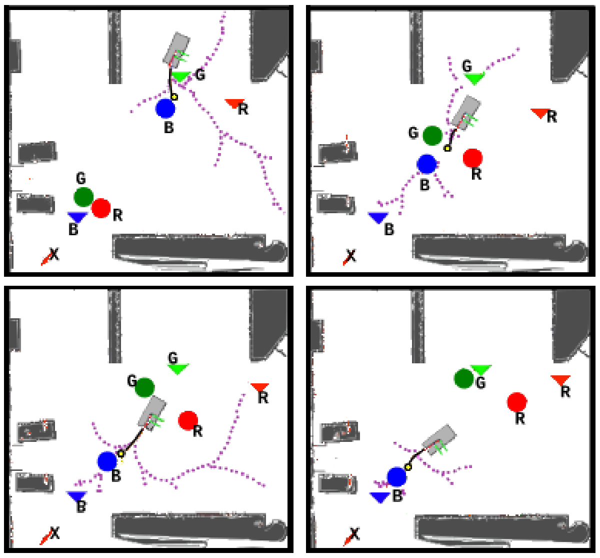


Figure 4.5: Experiment of a leader guiding the robot avoid two incoming persons. The robot is represented by the light gray rectangle and its goal by a X. Three leader candidates are represented by circles with letters R, G and B. The predicted goals are the triangles with the corresponding letters.

4.2 Experiments with Pedestrian Simulator

As in the previous section, these experiments will use the GHMM algorithm to learn and predict the typical goals of subjects in a scenario. The difference is that the motion of the subjects in the experiments is created using a pedestrian simulator framework.

The advantage of this approach is that different configurations of scenarios can be tested, but most important, the pedestrian simulator allows the reactiveness of humans to be taken into account during the experiments.

The reaction of persons due to the presence of the robot is an important situation that is often neglected in studies of robot motion among humans. It is relatively easy to find works where the robot is the sole responsible for all the avoidance maneuvers while humans behave, or are expected to, blindly follow their original path.

Obviously, this does not coincide with reality. In the real world, pedestrians are constantly adapting their motion to the environment structure and to the presence of objects and other humans. As a consequence, realistic experiments in human environments must be conducted on the real world or by means of a simulator that replicates the pedestrians' behavior.

4.2.1 Pedestrian Simulator - a Review

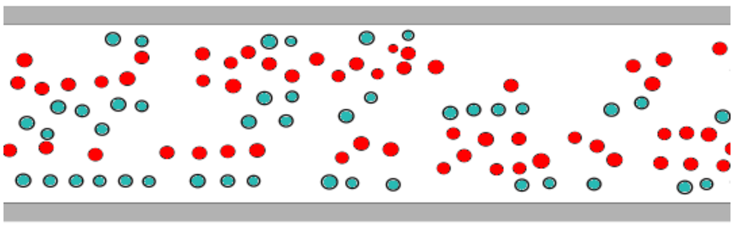
Pedestrians simulation has application in many different areas as architecture, video-games, computer animation, biology, physics, and others. In buildings, the simulated motion patterns of groups of people can provide insights for the architectural project, for example, to guarantee a smooth flow of persons and where easy exit access is essential in the case of panic situations. In many video-games that simulate humans, a realistic behavior from them will improve the players experience. While in the movie industry, the audience expects a realistic behavior of groups of animated characters, which would be very difficult to accomplish without the use of a simulator, when hundreds or thousands of individuals are present.

The Social Forces Model A seminal work on this field can be found in Helbing and Molnar (1995), where the typical formations that pedestrians engage in, and their usual behavior, are successfully simulated. In order to accomplish this task, the authors implement the so called Social Forces Model (SFM). According to this model, the motion of pedestrians is affected by the sum of forces exerted by different sources.

The simulations using the SFM could successfully replicate behaviors found in real situations, such as, the **formation of lanes**, the **oscillatory changes of motion direction** at narrow passages (shown in Figure 4.6) and the emergence of **temporary round-about traffic** in intersections. This simulator also provided interesting insights in the behavior of pedestrians, as new situations and structures could be tested without the need of physically implementing them, which would be difficult and dangerous with hundreds of persons involved.

On following works (Helbing et al., 2000, 2002), the authors explore the SFM within simulations of panic situations. The authors conclude that pedestrians behavior in such situations can be considered *irrational*, as they are more harmful than the pedestrians' behavior in normal situations. Structural improvements that would optimize pedestrian flow on panic situations are suggested in the conclusion.

Collective Behavior The keystone work in the field of distributed behavior has been conducted by Reynolds (1987), where the initial objective was to replicate the emergent collective behavior from groups of animals, as bird flocks and fish schools. The generic simulated indi-


(a)

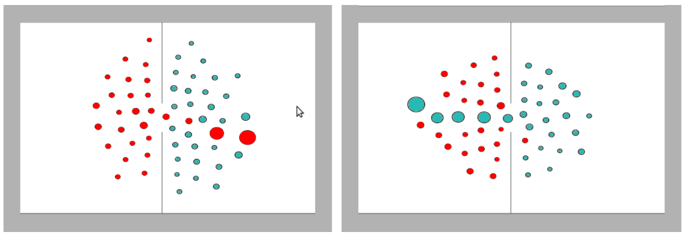

(b)

Figure 4.6: (a) The lane formation phenomena. (b) The oscillatory motion direction in narrow passages. The different colors of the blobs represent their motion direction, while their size represents their velocity (larger is faster). Adapted from Helbing and Molnar (1995).

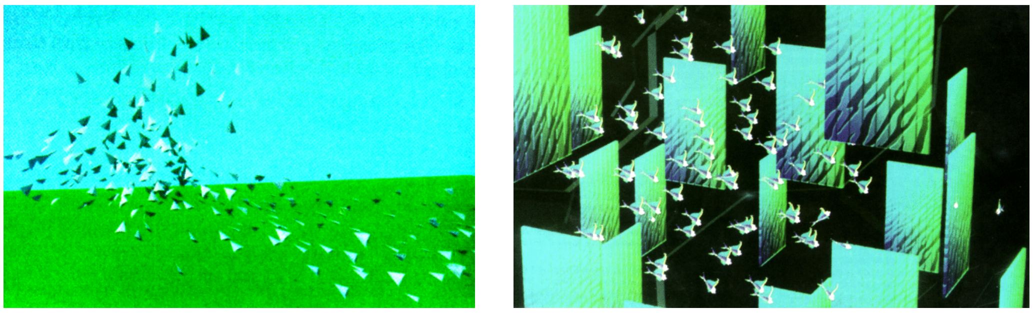


Figure 4.7: Images of **boids** moving together, resulting in life-like patterns of groups of animals. (From Reynolds (1987))

viduals were called **boids**. The flocking behavior is accomplished with three rules: separation, maintaining a certain distance from other individuals; alignment: orient itself towards the average orientation of surrounding individuals; cohesion: moving towards the average position of surrounding individuals.

This flocking model is further developed into a more complex simulator, aimed at animation and games, where simple rules can be combined to allow characters to navigate in a life-like manner (Reynolds, 1999).

Concepts from Robotics In more recent developments, some works borrow concepts from the robotics field, as motion planning theories (Treuille et al., 2006). The authors create a simulation framework where global navigation and moving agents are integrated using a dynamic potential field, without the need of explicit collision avoidance. Another example can be found in van den Berg et al. (2008b), where authors implement a technique that works particularly well in crowded scenarios, as agents do not repel each other, contrary to the case in several works based on the social-forces models. In this implementation, on the global frame, a roadmap is built to guide the agents, while in the local frame the VO method is extended so each agent selects a velocity that brings it closer to the goal while avoiding other agents.

In the work of Shiomi et al. (2012), the authors recognize the importance of using a pedestrian simulator to perform realistic experiments of robots among humans. To create an efficient simulator, they incorporate the most common Human Robot Interaction (HRI) that occurred in experiments with a real robot in a shopping mall (regarding humans): approach to stop, stop to observe, slow down and collision avoidance only.

Summary Although crowd simulations provide very efficient solutions to the problem of navigating in populated spaces, they cannot be readily applied to robot motion planning. The main reason is that in a simulator the states of all agents is known all the time, which is unpractical to a real robot application where only portions of the environment are observable and there is always some degree of uncertainty associated with any real measurement.

So in this Thesis, a pedestrian simulator will be developed with the sole purpose of providing a realistic framework for the evaluation of the developed techniques. The most important characteristics pursued are the capability of agents to react to the presence of each other and to the presence of the robot, and also the lane formation phenomena, where opposing groups of people end up forming lines, that facilitate their passage.

4.2.2 Development of a Pedestrian Simulator

In order to perform realistic experiments, a pedestrian simulator has been developed, that replicates the reactive behavior of pedestrians. This means that even if a robot is standing still, people moving in its direction should actively avoid it.

The developed pedestrian simulator is based on the SFM, which has been extensively validated in several works as a simple and efficient way of replicating pedestrian dynamics (Helbing and Molnar, 1995; Helbing et al., 2000, 2001).

In this model, the resulting velocity and orientation of the robot is the resultant of the action of two types of forces: an attraction force that pulls the agents toward their destination, and also in the direction of other points of interest, as street performers or shop windows; and repulsive forces caused by other pedestrians and by objects, as walls and obstacles.

In the current work, instead of using the resultant force as the acceleration applied to each agent's current velocity, it was used directly as the velocity command for those agents. The reason to use this approach was that it produced less oscillations in the motion of the simulated agents. This was possible only due to the high frequency of the forces computation loop in the main program. For the sake of consistency, however, the same nomenclature and units from the original implementation were maintained.

Only the attraction force towards the agent's goal has been implemented here, together with the repulsion force exerted by other agents and by walls and static obstacles. This system has been developed using the ROS framework, where STAGE provides information about a virtual world, and a **pedestrian simulator module** computes the resulting "force". This is transformed in velocity commands which are sent back to STAGE, so the virtual agents will behave as pedestrians, as shown in Figure 4.8.

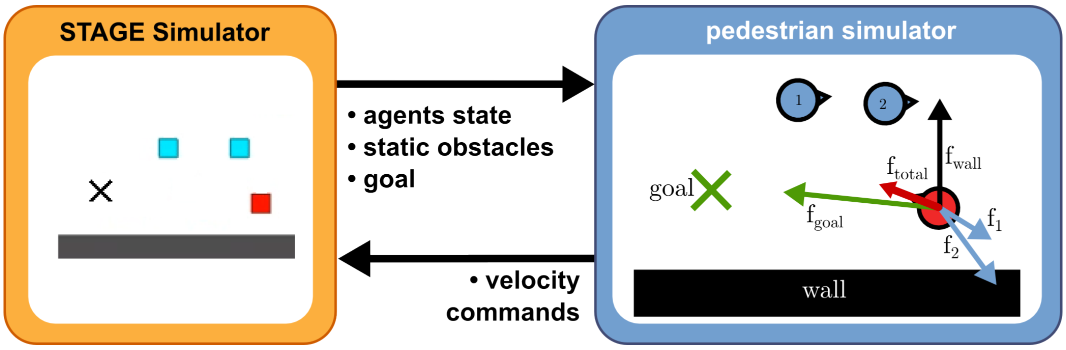


Figure 4.8: Workflow and description of the forces used in the pedestrian simulator.

Following the original notation, with minor modifications, these forces are described below:

- every agent α wants to reach its destination goal r^0_α as directly and as fast as possible. The desired direction of travel is computed based on the agent's position $r_\alpha(t)$ and represented as $\vec{e}_\alpha(t)$. So a constant force $F^0_\alpha(r_\alpha)$ is applied to the agent in the direction of its desired goal and modulated according to the agent's preferred velocity v_α:

$$\vec{F}^0_\alpha(r_\alpha) = v_\alpha \vec{e}_\alpha \tag{4.1}$$

- the motion of each pedestrian is also affected by the presence of other pedestrians. In the case where a two pedestrians get too close to each other, their discomfort increases, as they invade each other's personal space. To simulate the tendency of keeping some distance from each other, a repulsion force $F_{\alpha\beta}(\vec{r}_{\alpha\beta})$ between agents α and β is implemented, where $\vec{r}_{\alpha\beta}$ is the distance between the two agents:

$$\vec{F}_{\alpha\beta}(\vec{r}_{\alpha\beta}) = V_{\alpha\beta} e^{-\vec{r}_{\alpha\beta}/\sigma} \tag{4.2}$$

Here $V_{\alpha\beta} = 10m^2/s^2$ and $\sigma = 0.3m$. Both these values were set empirically and differ from the values from Helbing and Molnar (1995), to reduce the number of collision among agents. The motion of pedestrians will also be affected by the presence of the robot in scenario. As the robot is a strange entity among groups of people, it is expected that persons keep a larger distance from it than from other persons. To model this, the repulsion force produced by the robot is 50% stronger then the repulsion force among agents.

- finally, the walls and static objects also influence the motion of the pedestrians, as they must adapt to the environment structure. Pedestrians tend to become uncomfortable when too close to the walls, for example, as they have their freedom restricted and also can get hurt. In this way, walls and objects' border B produce a repulsive force $F_{\alpha B}(\vec{r}_{\alpha B})$, with $\vec{r}_{\alpha B}$ being the distance between the agent's position r_α and the closest border portion r_B^α, according to the equation

$$\vec{F}_{\alpha B}(\vec{r}_{\alpha B}) = U_{\alpha B} e^{-\|\vec{r}_{\alpha B}\|/R} \tag{4.3}$$

  With $U_{\alpha B} = 10m^2/s^2$ and $R = 0.2$, which are values obtained from the original paper, which provided good results in the simulations.

- to account for situations where dead-locks could happen, due to symmetrical generated forces, as agents with opposing goals facing each other, a fluctuation term $\vec{\xi} \in [-0.5, 0.5]$ was randomly generated and also incorporated into the model.

The resulting equation has the form

$$\vec{F}_\alpha(t) = \vec{F}_\alpha^0 + \sum_\beta \vec{F}_{\alpha\beta} + \sum_B \vec{F}_{\alpha B} + fluctuations \tag{4.4}$$

In summary, according to this model, there is a constant force that pulls the agent towards its goal and two repulsive forces that decrease exponentially, one according to the distance to the other agents, and the other according to the distance between the agent and the borders of walls and obstacles.

The result is a very useful tool to perform experiments that explore the reaction of groups of people in populated environments, providing a realistic validation framework. It can be shown that even if the robot navigates following a straight line in the presence of people, it will likely succeed because, as in real life, pedestrians will adapt their motion to avoid the incoming robot, as long as they notice it. However that will be accomplished with the cost of an impolite behavior that may cause discomfort to the pedestrians.

Regarding the scope of this Thesis, an important phenomena that arises in real human populated environments is the lane formation. This naturally occurs when groups of people cross each other in a high density environment, which is the result of a motion pattern that is more efficient, as it reduces avoidance maneuvers. This phenomena is also important for the proposed approach, as if the robot can enter one of the lanes forming, it will be able to navigate "with the flow", efficiently moving in congested spaces. Figure 4.9 illustrates this process and also validates the developed simulator in the sense this phenomena naturally arises in it.

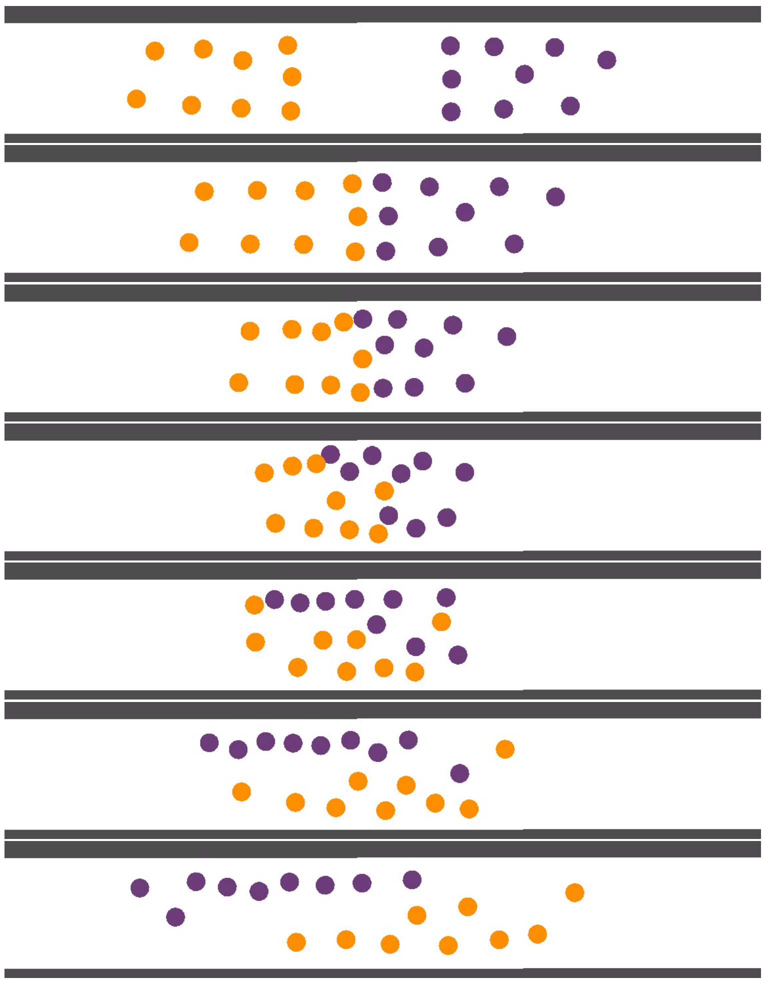


Figure 4.9: Lane formation phenomena: in densely populated environments, crossing groups of people naturally arrange themselves into crossing lanes, which result in a more efficient navigation.

Another interesting experiment consists of incorporating an "impolite" robot in the simulation, that moves in a straight line, disregarding the moving agents, shown in Figure 4.10. This experiment serves two purposes: first, to show that the simulated agents properly react to the presence of the robot during their motion; second, to illustrate that sometimes a large part of the avoidance is performed by persons surrounding a robot, and not by its navigation algorithm. Although this is a very simple perspective, it is often overlooked in many works, where the sole responsibility falls on the robot's motion planner. These results support the idea that in some situations a very simple algorithm outperforms more complex ones, although this is accomplished at the expense of an impolite behavior.

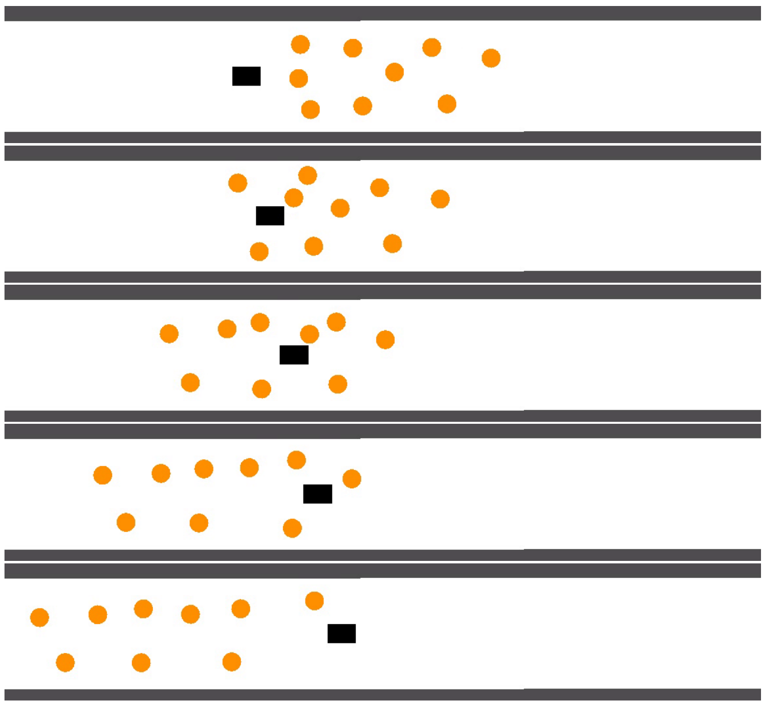


Figure 4.10: A robot using a simplistic motion algorithm moves in a straight line disregarding the moving agents. In order to avoid collisions, the agents move away from the robot.

Limitations of Classical Approaches Still using the developed pedestrian simulator, it is interesting to explore the limitation of the classical approaches when creating motion plans in densely populated corridors. In Figure 4.11 the A^* algorithm is used to create a plan that takes the robot from the left to the right side of the corridor.

As this algorithm have been developed to work in a static mode, its planning phase takes place at ”snapshots” of the current environment state. The problem is that in the next instant, the plan is invalidated by the motion of the agents, prompting the algorithm to create a new plan (represented as the black line in the figure). This sequence of plan/replan steps virtually bring the robot to a halt, as there is no plan that is valid for more than a coupe of seconds. Only when the density of agents reduces, the algorithm manages to create a plan that the robot is able to follow, until its goal.

Regarding algorithms that have been developed for navigation in dynamic environments, a common approach is to plan based on the prediction of future states of other agents. This is the case with the RiskRRT algorithm, used in the simulation depicted in Figure 4.12.

In this simulation, the robot is able to navigate reasonably, until it is faced with motion prediction of incoming agents. The problem is that, due to the high density of agents in the corridor, at some point, the predicted future states of the agents cover all the region in front of the robot, which causes the algorithm to fail in finding a solution. This is known as the Freezing Robot Problem (FRP), and it is a recurrent problem in prediction-based dynamic

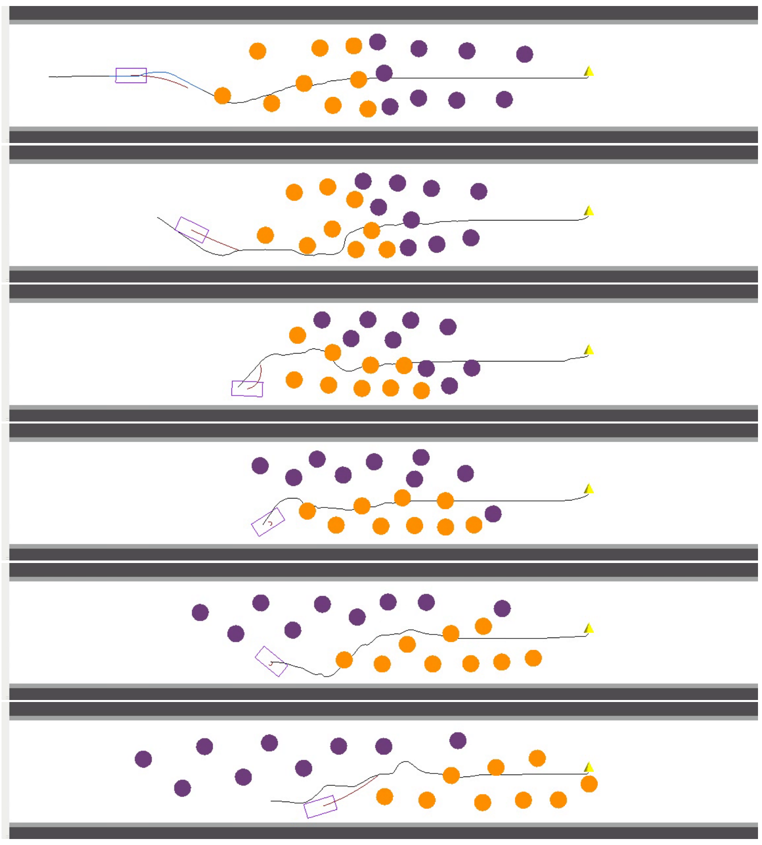


Figure 4.11: Performance of the A* motion planner in a densely populated dynamic environment.

motion planners.

4.2.3 Pedestrian's Motion Generation and Modeling

Differently from subsection 4.1, no real data is used in this experiments. Instead of that, a software is responsible for generating the trajectories that will be used to train the GHMM algorithm. This software uses a file that describe checkpoints and connections to generate the virtual trajectories. The visualization of the configuration file used in this experiment is shown in Figure 4.13a.

The nodes are checkpoints where the simulated agent must pass. It is possible to configure the direction and velocity, as well as their variance, that should be respected by the virtual agent. The green lines represent the sequence of checkpoints that must be respected and the yellow lines are the environment walls, that aid the creation of this map. In this example, agents enter the corridor coming from the left area and then can move to three different destinations, straight ahead, and left or right at the intersection.

Once more, the GHMM algorithm has been used to capture the information about the typical paths, this time generated by the trajectory generation software. It can be visualized

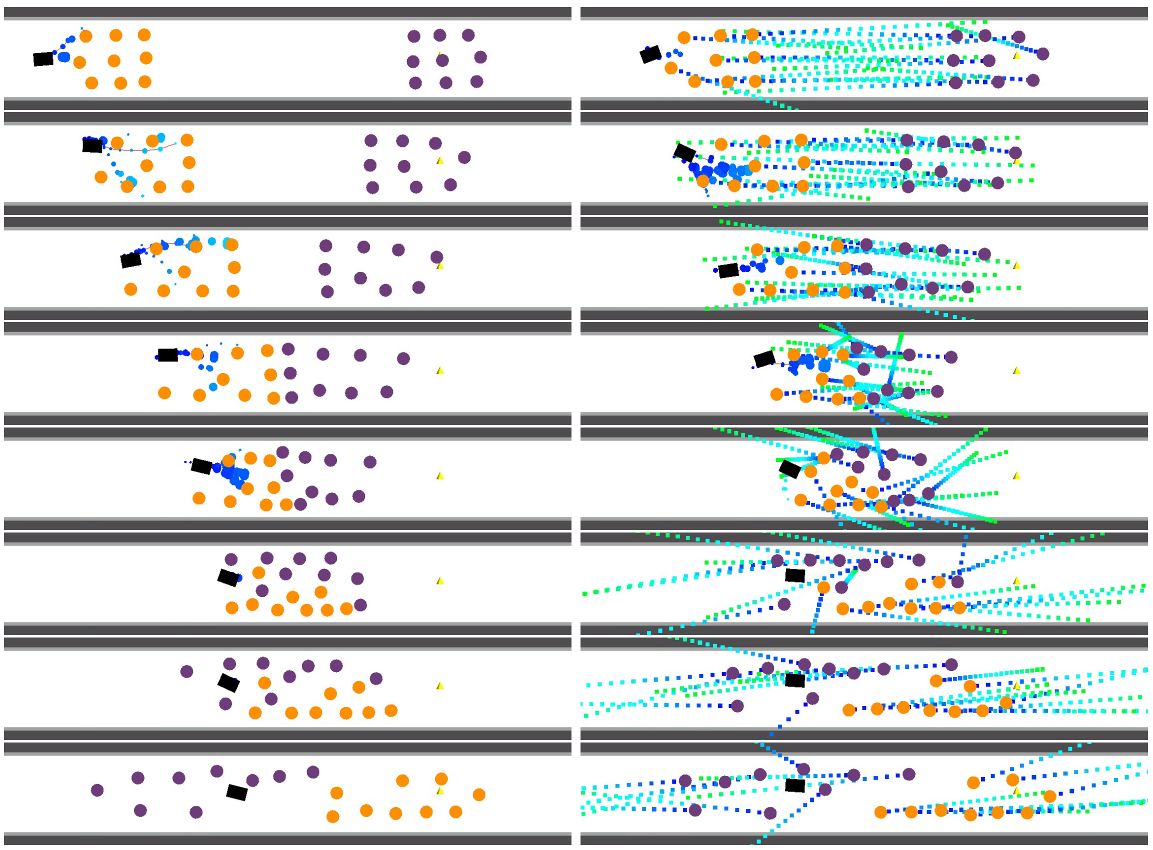


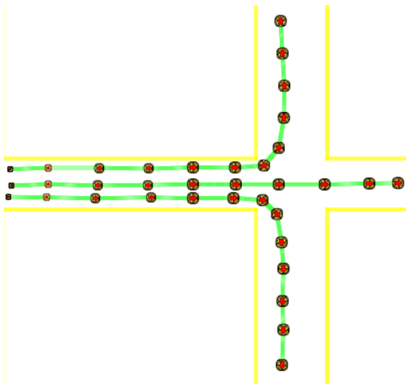
Figure 4.12: Performance of the RiskRRT motion planner in a densely populated dynamic environment. The left column shows the navigation without the predicted states of the agents, for the sake of clarity. In the right column, where the predictions are depicted, it is possible to see that they cover all the space in front of the robot, and the planner cannot find a solution.

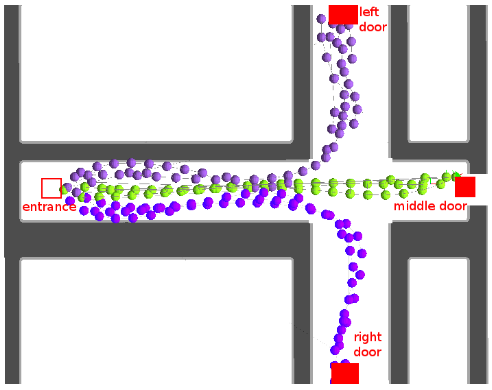
in Figure 4.13b

4.2.4 Tests

In this experiment, there are three leader candidates, represented by the blue squares, that are moving in the same direction intended by the robot. As shown in previous figures, there are three possible destinations (top, middle and bottom). Using the GHMM algorithm and the typical paths learned (see Fig. 4.13b), the algorithm predicts the likely goal for each one of the candidates, represented as the three different lines drawn from each candidate. The robot's goal is located in the bottom of the image (Fig. 4.14), and is marked with a cross. The candidate that has a goal similar to the robot's is chosen, and the robot starts to follow it.

In this experiment, there is also a group of red agents, and some of them move in the opposite direction of the robot's motion, which would normally create difficulties to the movement of the robot. But as the robot selects and follows a leader, it manages to negotiate passage


(a)



(b)

Figure 4.13: (a)Visualization of the file loaded by the trajectory generation program. The nodes represent checkpoints the virtual agents must pass, with a given orientation and velocity. (b) The resulting GHMM structure that models the typical paths of the generated agents.

with the incoming agents, and is able to successfully pass through the incoming group of people until reaching its destination.

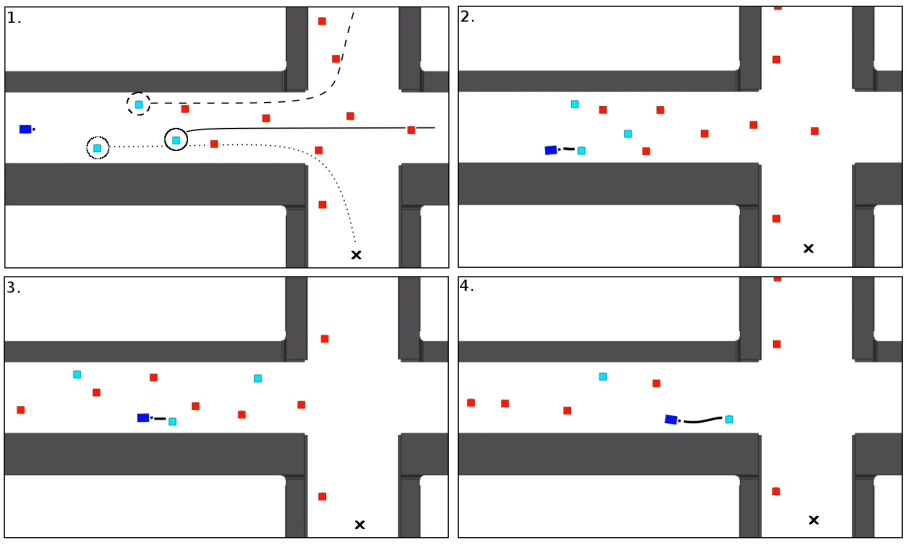


Figure 4.14: The robot is represented as the blue rectangle and the leader candidates as light blue squares. The first figure shows the predicted typical path and goal for each evaluated candidate. The candidate at the bottom is chosen as it is predicted to move to the same destination as the robot.

4.3 Experiments with Pedestrian Simulator in Narrow Corridors

In narrow corridors, the prediction of candidate's goal is simplified, as it can be assumed that subjects will move to one of the ends of the corridor. In this scenario, it will be assumed that any agent moving in the same direction of the robot, are good leader candidates. This allow a greater emphasis on the leader selection rules and on the type of navigation that the robot will use.

4.3.1 Tests

Multi-mode Navigation One aspect of the navigation module created in this work is the capacity of change the motion planner according to the presence and distance to the leader. This system has been tested using a single moving agent and a robot in a narrow corridor, as shown in Figure 4.15.

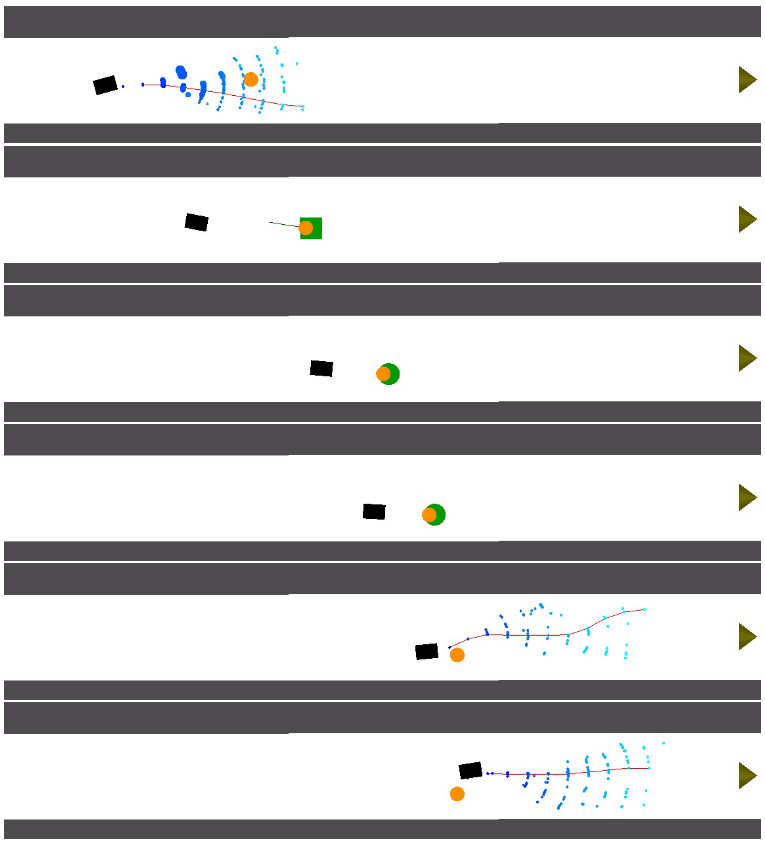


Figure 4.15: Multi-mode navigation, the robot is represented by the black rectangle, while the leader candidate as the orange circle. Both have the same destination at the right end of the corridor. The green square means leader path following mode is active, while the green circle means the direct leader following mode is active.

Here the robot starts its navigation in the *solo* mode, using the RiskRRT motion planner to reach the right end of the corridor. Once a leader candidate satisfies the leader selection rules, as being under a certain distance threshold from the robot, it is selected as leader and the multi-mode navigation switches to the *path following* mode. In this mode the path of the leader is recorded (represented as the green line in the figure) and the navigation system uses the Dijkstra's shortest path algorithm, using subsamples of the recorded path as subgoals. While in this mode, the leader is marked with a green square.

Once the robot catches up with the leader and the distance between them is less then $4m$, the *leader following* mode is activated and the robot directly follows the leader, which is marked with a green circle. Near the end of the experiment, the leader suddenly stops, having reached his/her destination. According to the leader selection rules, this agent is no longer a good leader, and the navigation system switches back to the *solo* mode, where the RiskRRT algorithm takes over, and the robot resumes its motion toward the end of the corridor.

Navigating Among Crowds A narrow corridor poses a big challenge for robots to navigate in the presence of several agents, as shown in subsection 4.3. It has been shown that both classical and modern techniques developed for dynamic environment navigation would produce suboptimal solutions. This experiment, shown in Figure 4.16, demonstrates one of the main arguments of this work, that taking advantage from the motion of leaders, the robot can navigate through difficult situations, as the one found in the narrow corridor.

In this experiment, two groups of nine agents each, that move to opposite directions in a corridor, with the robot just behind one of these groups. According to the leader selection rules, the robot soon chooses a leader and start to follow him/her. As the opposing groups encounter each other, the phenomena of lane formation starts to occur, which is a natural form of arrangement that improves the navigation of all agents. As the robot is closely following a leader, it manages to enter one lane and effectively becomes part of the group moving to the right side of the corridor.

It is interesting to notice that between images 6 and 7, due to the constant rearrangement of the agents, a subjects appears between the robot and its leader. Due to the form that the leader selection rules were created, the robot switches leader, and starts to follow the closest good candidate, continuing to move smoothly among the crowd. This test clearly shows the benefits of taking advantage of moving agents in a scenario, as they engage in complex formations and the robot is able to profit from this.

4.4 Summary

This chapter presented three types of experiments that evaluated the capacity of a robot to select a and follow a leader using different setups. In the first case, real data of human motion was used to train a GHMM and to recreate the motion of humans in a virtual environ-

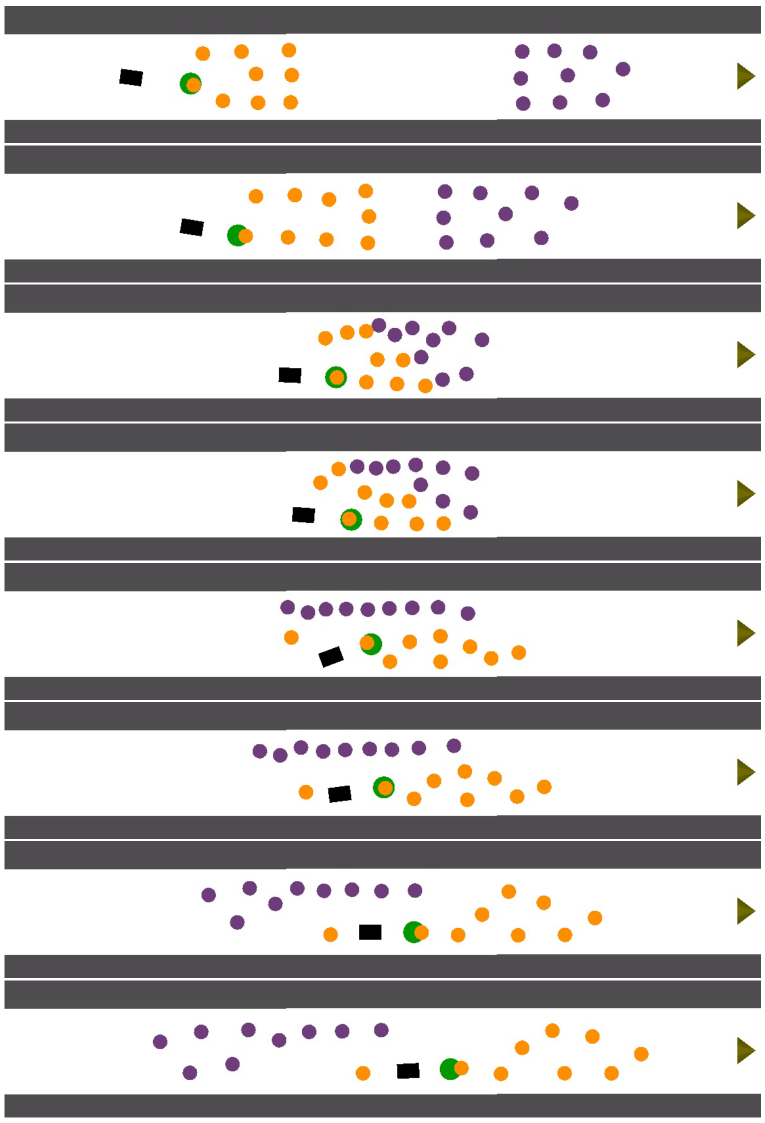


Figure 4.16: A robot is able to navigate in a difficult situation, along a populated corridor, by selecting and following a leader. In this way the robot becomes part of a group of pedestrians, and benefits from their motion interactions.

ment. The leader selection was accomplished using goal predictions provided by the GHMM algorithm. In the second experiment, the GHMM was also employed to predict candidate's goals, but this time the motion of persons was simulated, by means of a pedestrian simulator. In the last experiment the robot had to move in a crowded narrow corridor. Again a pedestrian simulator was used, and the leader selection was accomplished using a set of rules, together with the motion direction of agents. A multi-mode navigation framework was used for the leader following, where different strategies of navigation were activated, according to the situation. The contributions of this chapter are the following:

- Different methods can be used for goal prediction, depending on the complexity of the environment. Here, it was demonstrated the use of a probabilistic approach, which learned from previous data what were the typical paths traveled by persons in the environment, enhancing the goal prediction.

- A pedestrian simulator was created, which allowed realistic experiments where agents reacted to the presence of the robot and to the conditions of the environment.

- A multi-mode navigation framework was put in place, which used a state-of-the-art technique for navigating without a leader, and two other navigation strategies for the leader following, one that followed the leader's path and other that followed directly the leader.

- Experiments in crowded environment showed the benefits of taking advantage of the motion of other agents. The robot successfully navigated through difficult situations, where other algorithms provided suboptimal solutions.

Chapter 5

Experiments in Leader Evaluation and Following Based on Learning

It takes years to build up trust, and only seconds to destroy it.

anonymous

This chapter will conduct experiments associated with the methods described in Section 3.2, using a learning framework implemented with AdaBoost, to classify the quality of a leader candidate. Tests will be divided in two parts, where first the objective is to evaluate the classification capabilities of a leader evaluation algorithm, comparing the results of the classification framework with a ground truth. In this part, a study about the features' importance to the classification process is also performed. The second part will cover experiments of leader selection and following using real data to recreate the motion of humans. For that, the voting scheme described in subsection 3.3.1 is used, which will give a score for each candidate, associated with their quality as a leader to be followed.

5.1 Data Acquisition Using a Following Robot

Data collection was performed with a small car-like robot (Figure 5.1), measuring approximately $0.4 \times 0.75m$ built at the Laboratory of Automation and Robotics (LAR). A laser scanning range finder was installed on the robot, providing distance measurements up to $30m$ and with a field of view of 270^o. Also a firewire camera fitted with a wide-angle lens was present at the robot, providing a way to acquire images with a large field of view. In this way, videos could be taken during tests and then associated with the laser scans, as shown in Figure 5.2. This allowed a better understanding of the experiment situations, together with precise measurements of the surrounding area.

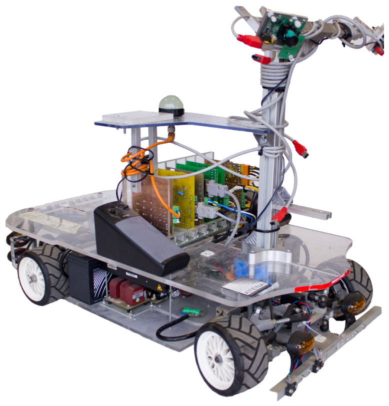
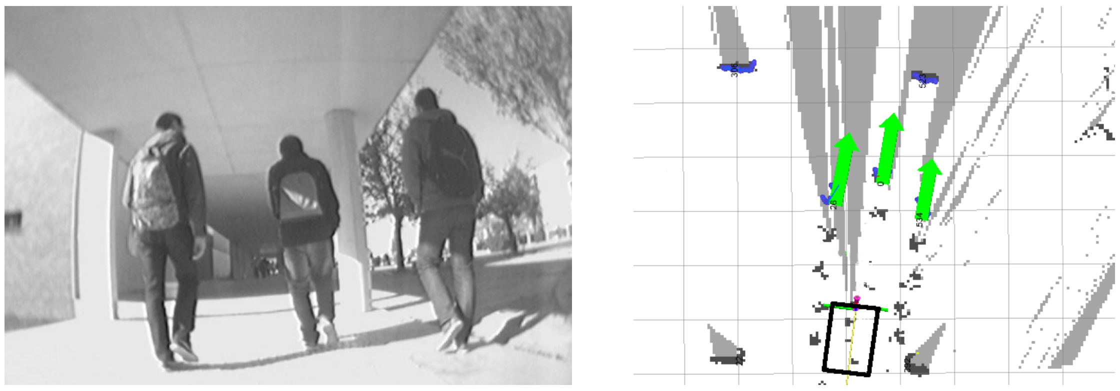


Figure 5.1: The robot used for data acquisition when following pedestrians.



(a) (b)

Figure 5.2: Two images of the same instant, captured by the camera and the laser range finder installed over the robotic platform. (a) Image from a wide-angle camera. (b) Representation of the laser measurements, showing the three tracked subjects and their orientation as the green vectors. The robot is represented as a black rectangle.

At this stage of the research, the mentioned robot was still not capable of navigate safely in dynamic, populated environments. So, as the purpose of the experiment was to evaluate the reactions of persons being followed by an autonomous robot, it was teleoperated using a wireless gamepad by a person that was hidden from the subjects being followed, in order to create the illusion that the robot was autonomous.

In total 47 tests of the robot following persons or group of persons were recorded, with the mean duration of about 20 seconds each. Out of the total, interviews were conducted with the leaders in the end of 10 tests, in an attempt to understand how the robot affected the leaders and how the leaders reacted to the presence of the robot.

Tests were conducted in an open corridor, about 3m wide, that links several University buildings. The robot operator had as objective to move along this corridor from one building to another, covering a distance of about 20m in each pass. Whenever persons were present

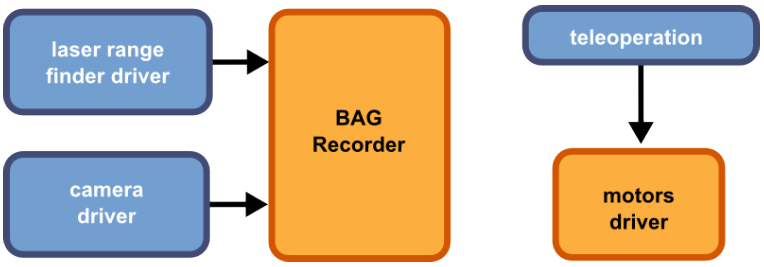


Figure 5.3: Setup of modules used for data acquisiton.

while the robot was moving between its objectives, the operator positioned the robot just behind the pedestrians and tried to maintain the same speed and a fixed distance. In the case the leader stopped, moved aside or to fast, the operator should resume the stand-alone navigation between the objective points, simulating the situation where an autonomous robot would have abandoned a leader.

The setup used for data acquisition is sketched in Figure 5.3. There is a module that receives the laser range finder measurements and another for the camera images. Readings from both sensors are stored in a *bag* file, which is the standard structure used in ROS for data recording. Isolated from these modules, there is also a program that receives commands from a gamepad and passes them to a module responsible for a low level interface with the robot's motors.

5.1.1 Features Extraction of Followed Persons

In order to obtain the descriptors of each leader subject, only the laser range finder measurements was used. Theoretically, features could also be extracted from the acquired camera images, as face detection and posture measurements, for example. These features could improve the training of the classifiers, as they could be associated with the quality of the leader. Someone that looks backward too often, is potentially feeling discomfort towards a robot following him/her.

Moreover, humans can communicate and interact in very complex ways, which may be very difficult for an algorithm to detect. During some tests, for example, persons being followed were very friendly towards the robot, sometimes inviting it to follow them as if the robot was a pet (Figurer 5.4).

However, due to the high complexity to obtain such information, they were not used in the feature extraction process. But certainly, a robot that could detect such interactions would greatly improve its classification capabilities and this is a topic that deserves future investigation.

The system setup used during the feature extraction process can be seen in Figure 5.5, which shows which modules are used an how they are connected in order to extract the features from the recorded *bag* file.

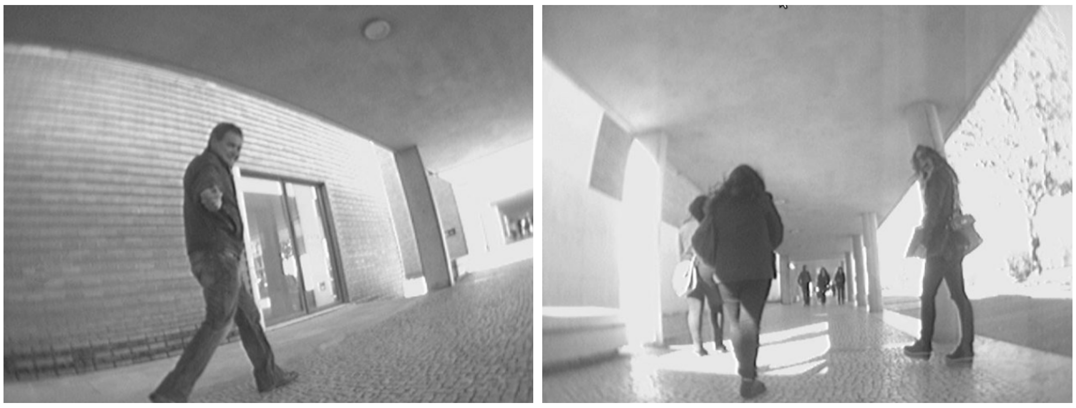

(a) (b)

Figure 5.4: Images of leaders inviting the robot to follow them, in both situation, they were unaware of what was the objective of the robot, but noticed they were being followed by it. (a) A man makes hand gestures inviting the robot to follow him as if it were a dog. (b) The rightmost woman makes head gestures and tries to communicate with the robot, asking it to follow her: "C'mon, c'mon..."

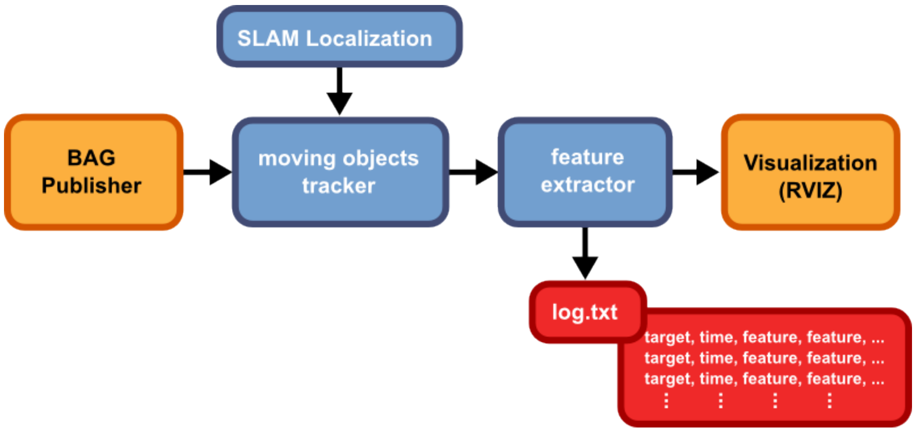

SLAM Localization
BAG Publisher
moving objects tracker
feature extractor
Visualization (RVIZ)
log.txt
target, time, feature, feature, ...
target, time, feature, feature, ...
target, time, feature, feature, ...

Figure 5.5: Setup of modules used for data extraction.

Initially the *bag* is read and its messages are published to a module that clusters and tracks moving objects using the motion tracker developed by Almeida (2010), which implements clustering by the **break point** detection method and tracking using a Kalman Filter (KF). As the laser measurements are obtained from the robot's point of view (robot's frame), static objects will appear as moving ones, as the robot moves. To account for that, the measurements need to be transformed to the global frame, in order to differentiate static object from moving ones. One way of performing this task is by finding the transformation of the robot with respect to the global frame (or map). Once this transformation is known, there will be a transformation chain: *global frame* $\rightarrow$ *robot frame* $\rightarrow$ *laser measurements*, which allows the laser measurements to be transformed to the global frame. The task of finding the transformation *global frame* $\rightarrow$ *robot frame*, is accomplished using the Hector SLAM tool

(Kohlbrecher et al., 2011). This is a powerful tool, which is able to localize the robot in an environment based only on laser measurements, without the necessity of a map or odometry measurements.

Once the moving objects are tracked, the information associated with them is passed to another module that proceeds to the feature extraction. This module provides information regarding the robot and subject's positions, their orientation and also their speed. Based on these measurements, other features can be computed, as distances and relative angles. Such features are then stored in a *.txt* file that will be further used during the training process. Besides that, all the current information of each test can be visualized using a ROS tool called *RVIZ*, to verify the consistency of the whole process.

With the purpose of generating features that can be understandable and visualized, in order to better understand human-following strategies, the following features were extracted for the training of the classifier algorithms:

- candidate's absolute velocity v_{c_i}: if a candidate moves too fast, the robot may not be able to get close and begin to follow him/her, while if a leader is moving slowly, he/she may not be suited to be a leader.

- relative heading between the robot and the candidate α_{rc_i}: this feature relates to the difference in the direction of movement between the robot and the candidates. A value of $cos(\beta_{c_i})$ close to 1 means both the robot and the candidate are moving along a similar direction, while values close to -1 indicates motion in opposite directions.

- candidate's relative velocity w.r.t. the robot reference frame vx_{rc_i} and vy_{rc_i}: relates to getting closer to or further from the leader and also to the variation of the lateral displacement.

- angle between the robot heading and the candidate's position β_{c_i}: this feature is associated to the lateral displacement of the subject, where a $cos(\alpha_{rc_i}) \approx 1$ means the leader is right in front of the robot and a $cos(\alpha_{rc_i}) \approx 0$ means the candidate is by the robot's side.

- distance between the candidate and the robot d_{c_i}: this is another important feature, as we usually keep a constant distance when following someone. The robot must maintain a minimum distance from the leader and, at the same time, a leader that is too far away may not be suitable to be followed.

- lateral displacement of the candidate ld_{c_i}: it is a measure of the lateral distance of each candidate w.r.t. the robot motion direction. This feature may be related to leaders giving space for the robot to pass them, which may be associated with the leader's discomfort.

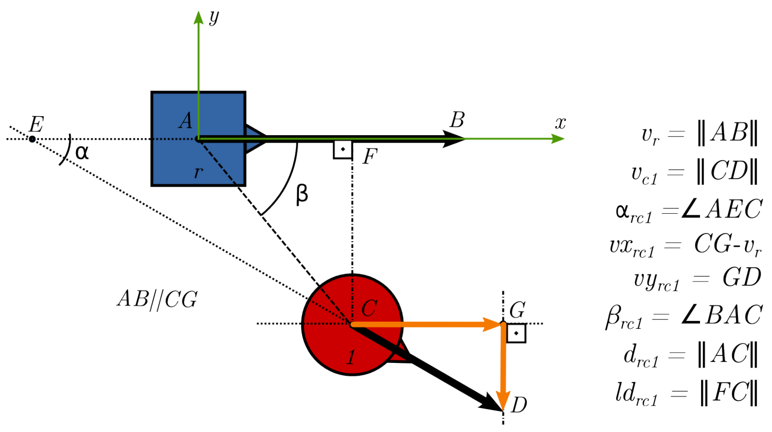


Figure 5.6: Graphical representation of the features extracted during the following procedure. The robot is represented as the blue square while the human being followed as the red circle.

The listed features are illustrated in Figure 5.6. The velocities of the robot and the candidate 1 appear as vectors $\vec{AB}$ and $\vec{CD}$, respectively. Besides the robot and candidate absolute velocity (v_r and v_{c_i}), all other features are measured w.r.t. the robot reference frame (with positive x pointing in the direction the robot's velocity vector). The relative heading α_{c_i} is the difference in the orientation of the velocity vectors the candidate and the robot. This feature is used to compute the projection of the candidate's relative velocity over the robot's axis, vx_{rc_i} and vy_{rc_i}, represented by the vectors $\vec{CG}$ and $\vec{GD}$, with $\vec{CG} \parallel \vec{AB}$ and $\vec{CG} \perp \vec{GD}$. The relative position of the candidate w.r.t. the robot is used to calculate the angle between positions β_{c_i}, the distance d_{c_i} ($\overline{AC}$) and the lateral displacement ld_{c_i} ($\overline{FC}$).

As the candidates are tracked by a KF, all the measurements actually incorporate a temporal dimension, as the filter uses previous states to compute the current one. Besides the filtered features, their derivative and standard deviation were also computed. The derivative provides information on the tendency of a measurement, for example if the distance between the a candidate and the robot is growing or diminishing. The standard deviation is associated with the stability of the measurements. For example, it might be preferable to follow a leader that has a constant velocity instead of an oscillatory one. Due to this, the total number of features is 21.

5.1.2 Data Labeling of Leader Quality

Most machine learning algorithms require a good dataset of examples to learn from. So one important issue is how to associate the measurements obtained, as explained in the previous section, to a good or bad leader situation. A labeling that is not properly performed, may result in two main problems. First, it may provide contradictory information during the training phase of the classifiers. Second, it can produce a bad ground truth dataset, which would result in poor evaluations of good classifiers.

5.1.3 Labeling Methodology

In many data labeling situations, the decision to create a label on the dataset is clear and objective, for example, faces in images, or pedestrians in laser scans. However, this is not the case in this work. Here the intention is to capture how humans decide when to start or stop following someone, or in other words, when someone is a good or bad leader. In order to do so, participants must create labels based on their *feeling* about someone being a leader or not. Of course there is a great subjective factor in creating such labels, as this *feeling* can change from person to person.

The labeling process also needs to be simple, as the input of persons that were unfamiliar with the process was required. This lead to a decision of creating a binary labeling system, with candidates identified either as good or bad leader. Besides that, for the sake of simplicity, the data presented to the volunteers should always have a single good/bad leader transition, so tests that had more than one transition were split.

Before the labeling process began, the objective of the experiment was explained to each of the volunteers responsible for the data labeling, where they should press a button whenever they *felt* a transition from good/bad leader occurred.

The only source of information that volunteers had access was a video acquired by the robot's camera, that was displayed to the user from a recorded ROS bag file. Another ROS module captured the keyboard input from the user and creates a new message that gets recorded, together with all the information from the original bag file, to a new one. This workflow is presented in Figure 5.7. All the experiments began with the robot following someone, which characterized a good leader situation. The label had to be created *while* the video was playing, so the labeler could have a better feeling about the dynamics of the experiment.

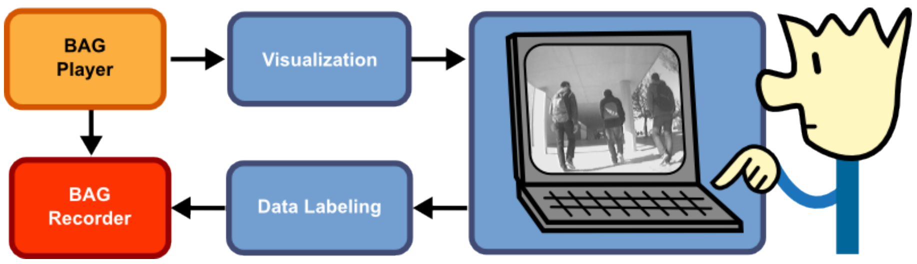


Figure 5.7: Diagram of the data labeling process.

5.1.4 Good and Bad Leader Situations

Before the labeling began, examples of different classes of good and bad leader situations were shown to the volunteers. Later they were asked to tell which of these situations occurred in each experiment. Although each test has its own peculiarities, the experiments can roughly fit one of the following classes:

- **good leader (gd)** - leader(s) maintained their speed and orientation, withouth changing their behavior while being followed (Figure 5.8);

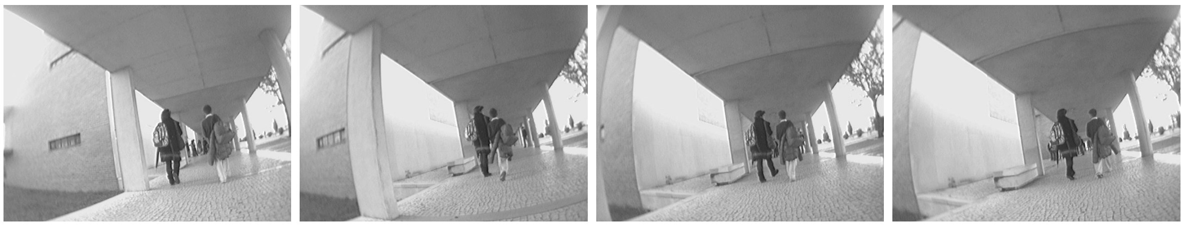


Figure 5.8: Good leader behavior, although the robot is noticed, the leaders kept moving without changing their behaviors

- **bad leader, moved aside (as)** - identifies situations where leaders gave room for the robot to pass, generally moving aside, while keeping their original motion direction (Figure 5.9);

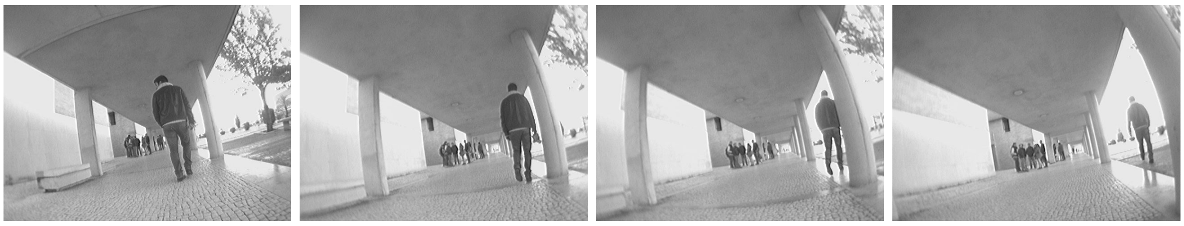


Figure 5.9: Bad leader: the person notices the robot and moves aside, giving room for the robot to pass.

- **bad leader, far or fast (fr)** - this occurred whenever the distance between the leader and the robot grew to a point where it was not advantageous to keep following them (Figure 5.10);

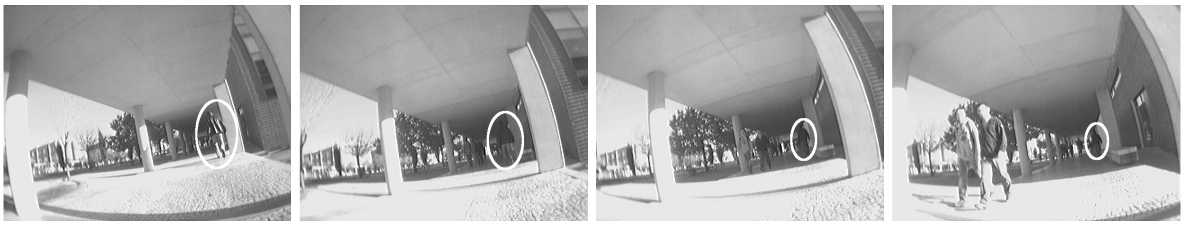


Figure 5.10: Bad leader: the subject marked with a white circle moves to fast compared to the robot's speed and gets too far from it.

- **bad leader, stopped (st)** - when the person being followed stops moving (Figure 5.11).

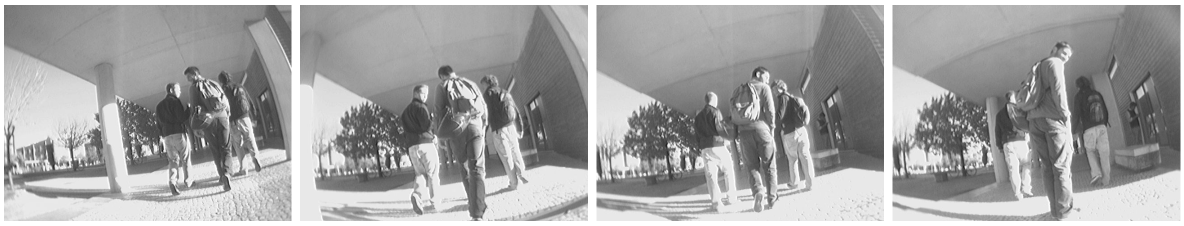


Figure 5.11: Bad leader: the person stops moving.

5.1.5 Combining Labels

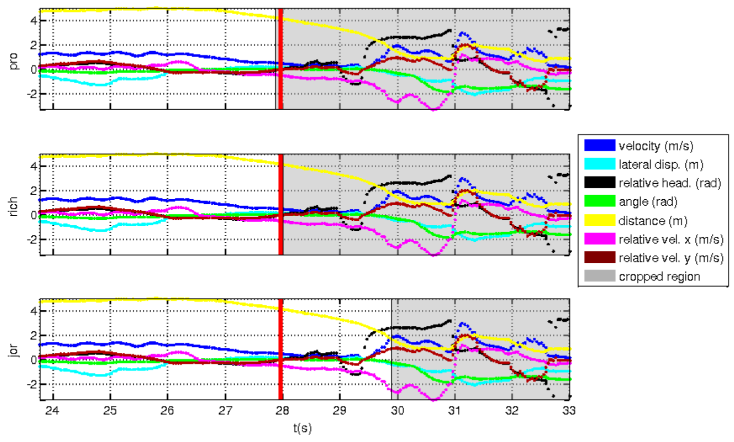
To reduce subjectiveness in the data labeling process, it was performed with three persons. The labels are represented as time tags and, to reduce subjective bias, the mean of the three labels were used. In the case that labels were not close to each other (timewise), only the two closest were used. This analysis was aided by the typical situation identified by the labelers. For example, if one had found that the transition occurred because the leader was too far, but the other two thought that the transition was due to the leader moving aside, only the labels of the latter two were used, as they form the majority.

Table 5.1 shows an example of how the labels scheme works. The first column shows the test number, the next three columns show the identified situation according to each one of the evaluators (pro, rich, jor). The following three columns show the time each label was created. Note that in test 7, for example, the first evaluator identified a different situation from the others, so his time label is discarded (XX) and does not contribute to the final label. If no transition of bad/good leader was found, meaning the leader remained a good one throughout the test, no label was created (—). The remaining columns show the final label, computed as the mean of the individual labels, its standard deviation and finally the consensus of the transition situation, based on the majority vote of evaluators. For example, in test 9, the first evaluator detected a leader far/fast situation, but the other evaluators found out that the leader could still be followed (gd) so that was the consented situation.

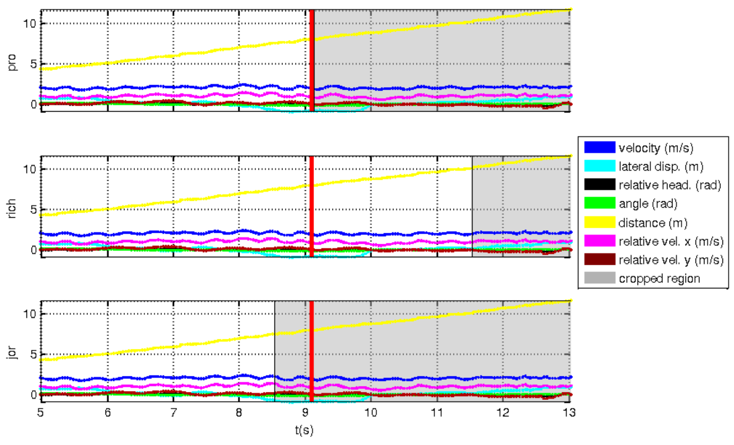
Table 5.1: Labels from each evaluator and the final label

test	id. situation			time labels (s)			final label (s)	consensus
	pro	rich	jor	pro	rich	jor		
1	st	st	st	27.84	27.97	29.84	27.97	st
2	fr	fr	fr	9.08	11.48	8.49	9.08	fr
6	as	as	as	12.02	11.39	12.47	12.02	as
7	as	fr	fr	XX	12.56	12.99	12.78	fr
8	gd	gd	gd	——	——	——	——	gd
9	fr	gd	gd	XX	——	——	——	gd
10	as	st	as	18.33	XX	18.24	18.29	as

To have a better understanding of this process, Figure 5.12 shows a plot of some of the extracted features corresponding to test number 1 and 2 from Table 5.1. The gray area represents bad leader labels, created by each evaluator. The beginning of such area represents the moment the volunteers decided the subject being followed became a bad leader. The red vertical line in the images represent the final label, computed as the mean of individual labels after the voting method described above.



(a)



(b)

Figure 5.12: Comparison between each evaluator's label (beginning of gray region) and final label (vertical red line). The labels are plotted over the features extracted from a person being followed. a) test 1 - leader stopped; b) test 2 - leader too far.

5.1.6 Resulting Dataset

Besides the four typical situations mentioned before (good leader, leader moved aside, leader stopped and leader too far/fast), two other situations were recorded, which did not required manual labeling. The reason is that no transition exists and the subjects are considered as bad leaders along the whole experiment. These situations refer to cases when candidates are not moving, but rather standing close to the robot's path (nm), and the other case is when persons were moving to the opposite direction compared to the robot (od).

It is important to emphasize that the methodology developed for the data labeling system only generated one good/bad leader transition for each test run. Once that transition was marked, the remaining samples of a test retained the *bad leader* label. The main reason for this type of implementation was the simplicity of the labeling process, allowing inexperienced users to participate, but the drawback is that it may result in suboptimal labels. For example, if the robot is following a leader that stops and later resumes his/her motion with the same characteristics, he or she will not be labeled as good leaders after resuming their motion.

To minimize this contradictory information, care was taken in cutting out those *bad* $\rightarrow$ *good* transitions. Also start and end transients caused by tracking initialization or target loss, were also removed from each experiment. Figure 5.13 shows one example of the elimination of transients in the dataset.

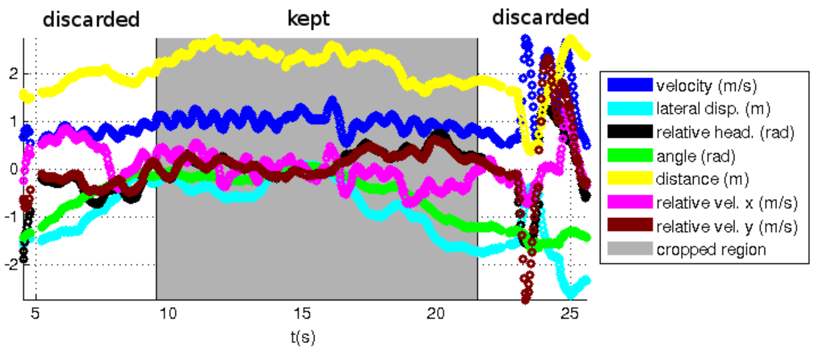


Figure 5.13: Example of cropping an experiment to eliminate the initial and final transients, the gray area highlights the extracted region.

After the labeling and crop processes, each resulting dataset was given a name to identify what situation they represented, together with a number to differentiate them, for example *as05* is the fifth test containing the leader moved aside situation. In total, 12911 samples were obtained, with the proportion of 37% of bad leader labels and 63% of good leader labels. This is equivalent to 451 seconds of tests, divided in 47 experiments and distributed in the following classes:

- 9 gd - good leader
- 7 as - bad leader, leader moved aside
- 5 fr - bad leader, leader too far or fast
- 9 st - bad leader, leader stopped

- 7 nm - bad leader, candidate not moving
- 10 od - bad leader, candidate coming from opposite direction

5.2 Training of a Leader Classifier

For the training process, the data was organized according to the situation they represented (good leader, moved aside, stopped, too far/fast, not moving, opposite direction). By grouping typical leader situations, it was possible to create datasets that contained examples of all the situations, both for training and for evaluation, so the classifier could also be evaluated in terms of the situation presented.

To evaluate the classification, three measurements were made. The **false good leader**, where the classifier labeled a sample as *good leader* but the ground truth has a *bad leader* label. This is the most critical error, because by following someone that should not be followed, the robot may disturb the leader or find itself in unwanted situations. The second measurement is the **false bad leader**, that occurred when the classifier output a *bad leader* and the ground truth is labeled as *good leader*. Although this is also a mismatch in the classification, it is much less critical then the previous measurement. This is because if a robot does not follow a potential leader, it is only loosing an opportunity, which can be associated with a conservative stand-alone behavior, but it is not disturbing or interfering with the potential leader. Finally the total error is the sum of the two previous measurements and represents the total number of misclassifications.

The data was divided according to the following process. Two datasets (except only one far/fast) of each situation were randomly picked and grouped to create a test set to be used to evaluate the classification after training. An extra dataset contained one sample of each situation, also randomly chosen, and constituted the validation dataset (this dataset was used in Appendix B, for the training of an Artificial Neural Network (ANN)). Finally, the remaining sets were used grouped into a training dataset. Table 5.2 describes the composition of each dataset:

Table 5.2: Train dataset composition, with the number of datasets of each class used, the number of samples and the proportion of bad leader labels.

	gd	as	fr	st	nm	od	samples	bad leader label %
train	7	4	3	6	4	7	8504	34.11
train (no odnm)	7	4	3	6	0	0	6626	15.44
validation	0	1	1	1	1	1	1692	47.16
test	2	2	1	2	2	2	2715	39.30
total	9	7	5	9	7	10	12911	36.91

During the Adaboost training, there was no limitation in the number of the weak learners used, and the stop criteria was the error. In all cases, it reached 0 after a different number of iterations (weak classifiers).

5.2.1 Leader Classification Results

The first test, consisted of comparing the feature contribution and the performance of a classifier when two types of training datasets were presented. The first dataset was composed by the situations shown in the first line of Table 5.2 and the resultant classifier is called *complete*. The second dataset was based on the first one, but the not moving (nm) and opposite direction (od) situations were excluded and will be called *stripped* classifier. The objective of this test was to evaluate the impact of having "bad leader only" situations (nm and od) on the features contribution and on the classifier performance. Figures 5.14 and 5.15 show the contribution ratio of each of the 21 features.

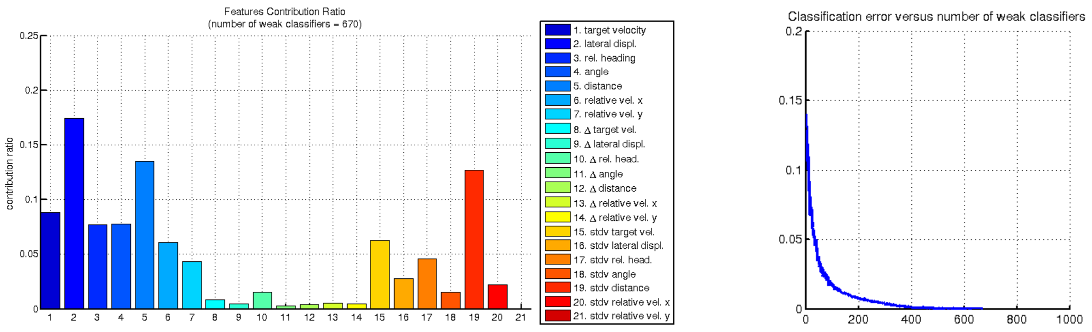


Figure 5.14: Feature contribution ratio using the **complete training dataset**, number of weak learners: 670, features used: 21.

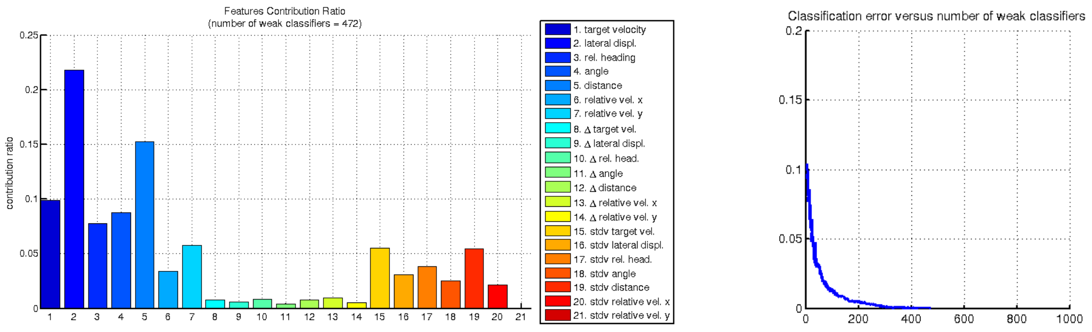


Figure 5.15: Feature contribution ratio using the **stripped training dataset**, without the *not moving (nm)* and the *opposite direction (od)* situations, number of weak learners: 472, features used: 21

Note that in both situations, the most two most important features are the same: lateral displacement and the distance between the robot and the leader. Although there are some differences, the trend of the contributions are similar. In both cases, when looking at the first features, that are the output of the KF in one instant, it is possible to see that the relative velocity, both in x and y is less important then the other features. Also, the derivatives have almost no role at all in the classification, with individual contributions of usually less then 1%.

Looking at the third group of features, it is possible to see that the predominant features are the same in the two graphs, which are the standard deviation of the leader velocity and the standard deviation of its distance. The difference is the much larger contribution of the latter in the case where the *od* and *nm* situations are presented to the classifier during training. The standard deviation is computed over a window of 1 second, so it can be interpreted as the stability of a feature measurement during the last second.

The quantitative results of the classifiers trained with and without the *nm* and *od* situations are exhibited in Figure 5.16 and Figure 5.17, respectively.

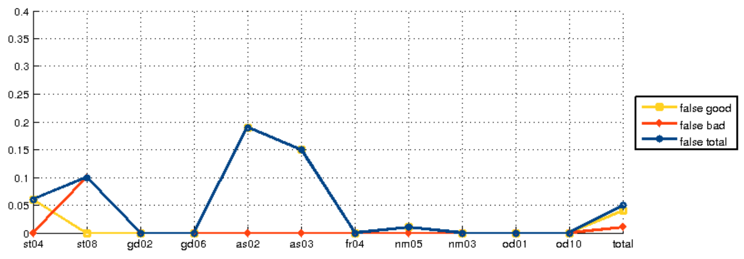


Figure 5.16: Classification error when training with the **complete training dataset**, including the *not moving (nm)* and the *opposite direction (od)* situations

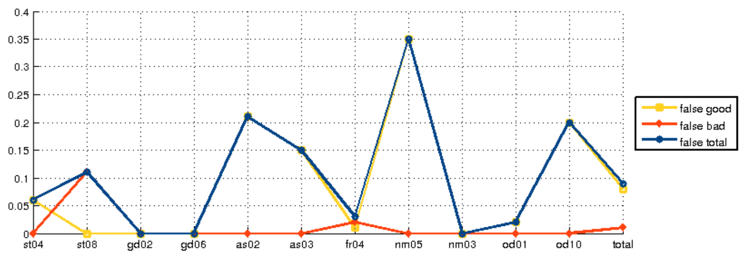


Figure 5.17: Classification error when training with the **stripped training dataset**, without the *not moving (nm)* and the *opposite direction (od)* situations

These Figures show the *false good leader*, the *false bad leader* and the total classification error, using an Adaboost classifier trained with two different datasets. Each entry in the x axis represents the error for one typical situation, so the performance of each classifier can be evaluated not only globally but also according to specific situations. Looking at both images, both classifiers have similar error rates for common situations, with the false good leader error particularly large at the *move aside (as)* situation. The largest difference comes from the *od* and *nm* situations. When these situations are present in the training set, the classifier manages to correctly classify those situations, with minimal error. However when these situations are removed from the training set, the result is a larger error in classifying them, as it should be expected. Nevertheless, in two out of four of those situations (*od01* and *nm03*), the classifier that was not trained with the *od* and *nm* classes, managed to have a very high hit rate, which demonstrates a very good generalization capability.

Following the process of feature selection, the least important ones are removed from the train dataset and the classifiers are trained again (same set with reduced features). In this case the features removed are all the derivative group and the standard deviation of the relative velocity y, due to their low contribution. The results of the training regarding features contribution, error and number of weak learners are shown in Figures 5.18 and 5.19.

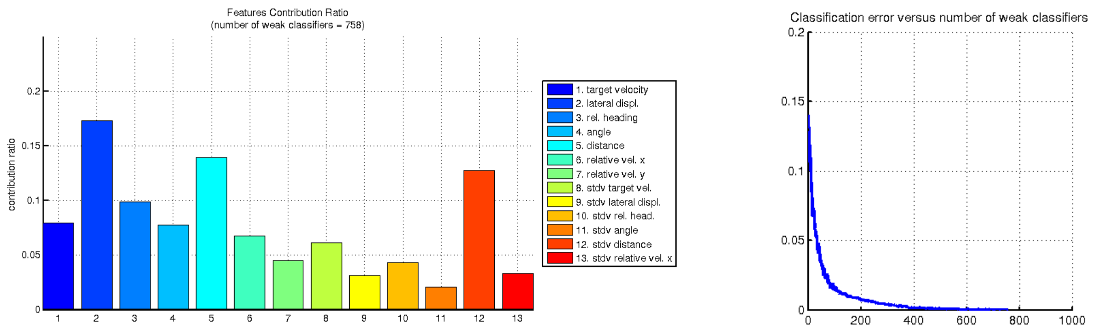


Figure 5.18: Feature contribution ratio using the **complete training dataset**, number of weak learners: 758, features used: 13

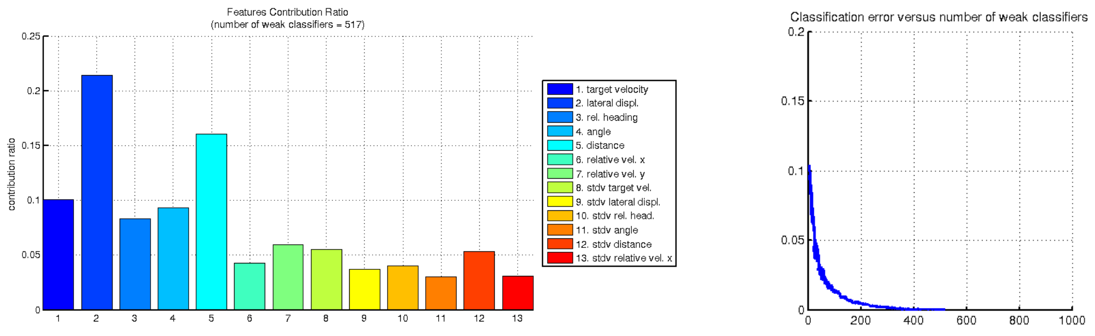


Figure 5.19: Feature contribution ratio using the **stripped training dataset**, number of weak learners: 517, features used: 13

The quantitative results of the two classifiers, trained with a reduced number of features are shown in Figure 5.20 for the classifier trained with the complete dataset and in Figure 5.21 for the classifier trained with the stripped dataset.

Regarding the feature contribution, it can be seen that the same trend from the first test was maintained. When using the complete training dataset, focusing on the standard deviation group, the standard deviation of the distance and of the target velocity are still

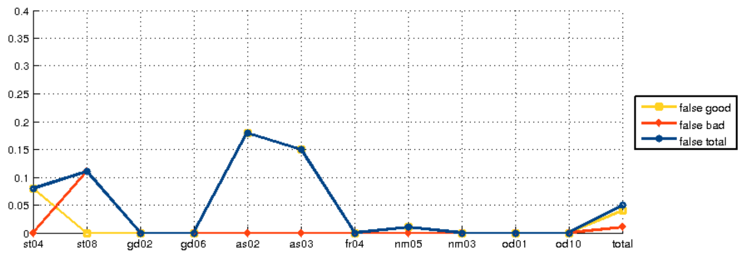


Figure 5.20: Classification error, **complete training dataset**, 13 features

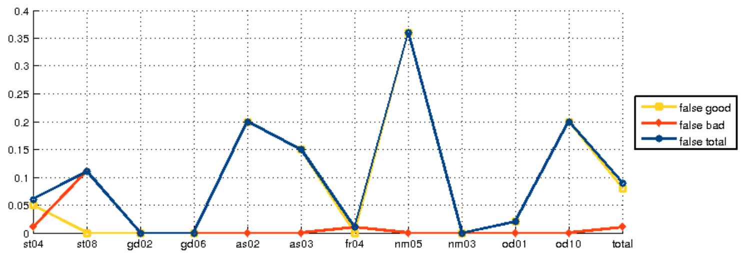


Figure 5.21: Classification error, **stripped training dataset**, 13 features

more relevant than the other features of this group. This is still the case when the classifier was trained with the stripped dataset, but the contribution of the aforementioned features is much lower, only slightly higher then their counterparts. Independently of the training dataset, the two most important features are still the lateral displacement and the distance to the leader.

With respect to the quantitative results, there is no real improvement, with only slightly changes in the classification error. The total error is the same from before the feature space reduction.

Once more, the least relevant features are removed from the training dataset. This time the features removed were from the standard deviation group, except the standard deviation of the distance and of the target velocity. The feature contribution of this third training round can be seen in Figures 5.22 and 5.23.

The quantitative results of the two classifiers, are shown in Figure 5.24 and in Figure 5.25.

5.2.2 Analysis of Contribution of Features to the Leader Evaluation

As the tests were labeled based on human experience, the findings of this chapter will allow a better understanding about how humans decide to stop following someone, and what are the features that we, unconsciously, use to make such decisions.

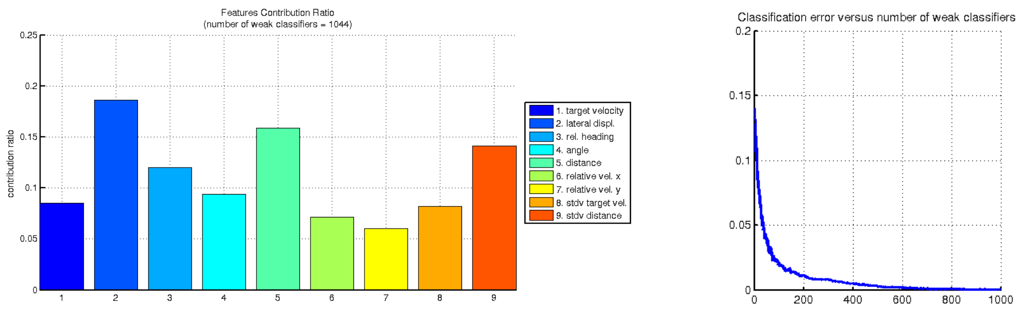


Figure 5.22: Feature contribution ratio using the **complete training dataset**, number of weak learners: 1044, features used: 9

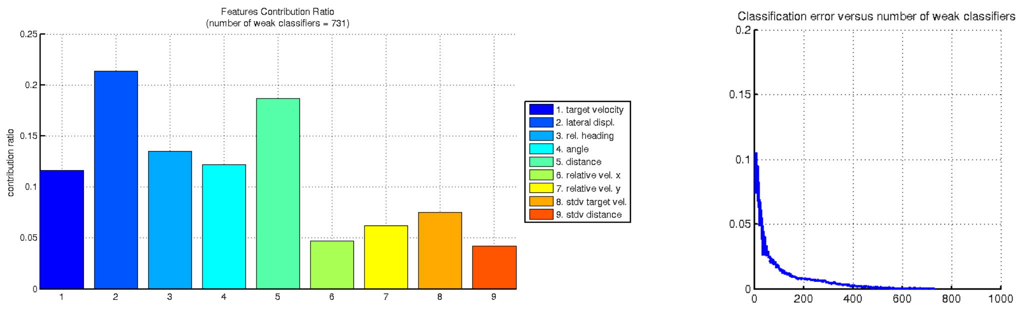


Figure 5.23: Feature contribution ratio using the **stripped training dataset**, number of weak learners: 731, features used: 9

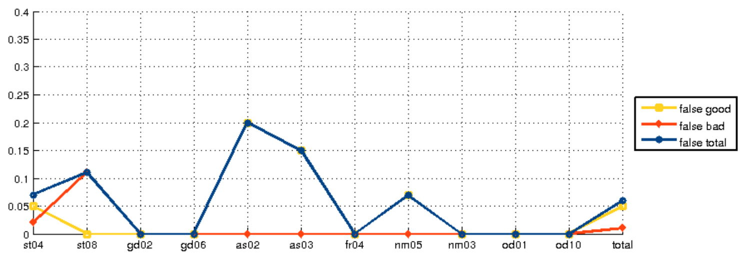


Figure 5.24: Classification error, **complete training dataset**, 9 features

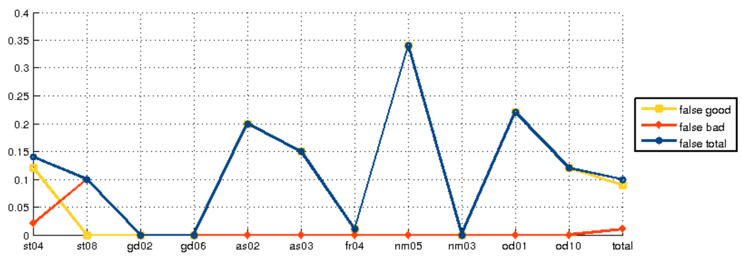


Figure 5.25: Classification error, **stripped training dataset**, 9 features

Importance of the Lateral Displacement and Distance

First, and most important, is the analysis of the contributions of the most relevant features in the classification of a good or bad leader. The tests provided very interesting results about the importance of the lateral displacement and of the distance to the leader, which were the features that contributed more to the creation of the classifier, independent of the training set or the number of features used.

Both the lateral displacement and the distance are spatial features, that do not take into account the dynamics of the leader. This means that the position of someone with respect to the robot plays a very important role when deciding among leader candidates and when deciding to stop following someone.

In the case that situations where candidates are not moving or moving from the opposite direction, the standard deviation of the distance becomes one of the most important features to the classification of good/bad leader. As the standard deviation is computed on a time window, this measurement can be understood as the stability of the measurements during the elapsed time. In this case, the stability of the distance between the robot and the candidate becomes a feature of great importance. To understand why this happens, one can think how the distance evolves in the case of a good or bad leader. In the first case, it is expected to remain stable, assuming both the leader and the robot can maintain similar velocities. But in the case the candidate is not moving, or coming from the opposite direction, the distance between them and the robot will constantly change. Tests showed that this is a feature more important then the velocities or relative heading of candidates.

Relevance of Derivative Features

The lack of importance of the derivative of the used features was also evident, with them having almost no contribution to the results. It was expected that the derivatives provided a measure of the tendency of features to grow or diminish (a growing distance, or an increasing velocity of a leader). A probable cause for the small contribution is that the derivatives ended up amplifying the noise presented in the measurements, resulting in useless features. If this was the cause, a possible workaround would be to compute the derivative over a time window, to reduce the noise amplification.

Relevance of Velocity Features

One expectation that matched the results was the low contribution of the relative velocity vx_{rc_i}, measured along the robot's heading direction, between the robot and the leader. The reason was that, due to the teleoperation, the robot presented a large variation on the velocity, as the operator tried to keep the pace of a leader. So, even if the leader velocity remained almost constant, the variation on the robot velocity, caused by inaccuracies of human teleoperation, would produce a variation on the relative velocity. Another possible cause to the low contribution of this feature is that it was overshadowed by other features as, for example, the standard deviation of the distance or the standard deviation of the velocity of the leader.

This may also be the reason for the low contribution of the relative velocity along y axis, measured perpendicular to the robot's heading, where it may have been overshadowed by the lateral displacement feature, which had a much greater importance.

Quality of the Results

When evaluating the classifier results, several aspects must be taken into account, as the number of features used, the error rate and the number of weak learners used. Although in all the training the classifiers reached error zero for the train dataset, they required a different number of weak classifiers to reach that result. Such number has a direct effect on the time spent for classification. As the objective of the current work is to incorporate such classifiers on robots, the time spent for classifications is a critical constraint. So, even though the classifiers can technically reach error 0, this may not be feasible to implement in practical experiments.

In overall, the trained classifiers produced very good results, with the total error across different situations ranging from 6% to 10%. And the total error for false good leader classifications ranging from 5% to 9%, according to the training set and the number of features used.

By evaluating the error in a situation-by-situation basis, it is possible to notice that it can grow quite large in some situations. When presenting *od* and *nm* situations to the classifier trained with no examples of such situations (*stripped* classifier), the error grew as high as 36% (Figure 5.21). Excepting those situations, both classifier types, *complete* and *stripped*, had similar results. Both classifiers, independent on the number of features used, had no noticeable error in the good leader situation (*gd*), and their performance in other situations were practically the same.

There was no significant differences between the qualitative results among the two types of classifiers and using different number of features. Nevertheless, this stepwise analysis was required, where features are removed, the classifier trained and errors measured. The reason is the form the AdaBoost works, as it uses a combination of weak classifiers over features, the removal of one of them could change the way the others are combined, possibly affecting the results.

Although promising, these results are obtained from a sum of instant classifications, so past information is not taken into account for the classification. This means that the output of the classifiers cannot be taken directly to be used for leader following, with the risk of switching constantly between good/bad leader. Instead of that, it should be used as the input of a higher level program that is capable of computing a leader score, that takes into account previous classifications, maintaining some kind of hysteresis. These experiments will be conducted in the following section.

5.3 Tests of Leader Classifier Integrated with Navigation

These tests evaluated the performance of the classification framework, presented in the previous section, when integrated with a navigation framework, where a robot must classify, select and follow leaders. Simulations were performed using STAGE and the ROS framework and, as in previous experiments, real data was used to recreate the motion of real persons in the simulations.

As a result, the features extracted from the motion of the recreated pedestrian are realistic. And, at the same time, they are different from the ones used in the training of the classifier, as they are measured with respect to the virtual robot configuration, which will be different from the configuration of the real robot, originally used. A drawback, however, is that pedestrians are not able to react and change their behavior due to the presence of this virtual robot.

Tests were conducted with the four typical situations encountered during the data acquisition (see subsection 5.1.2), which are: one with good leader situation, and three with bad leader situations when a leader stops, is too fast or far, and when a leader moves aside. Again, the multi-mode navigation system was used, and the objective of the robot was to move from the left to the right side a corridor, following leaders when they were available, or moving independently (solo mode), if no good leaders were present.

The environment where tests took place is an open corridor (see Figures 5.8 to 5.11), that was mapped with a laser range finder, during the data acquisition process.

Figure 5.26 shows a sample image of a test. The robot is represented as a black rectangle, and leader candidates are represented as small colored circles. The color of the circles represents the instantaneous classification of those subjects, where green means a good leader and red a bad one. The velocity vector of the candidates is represented as a green arrow, and the selected leader is marked with a green square.

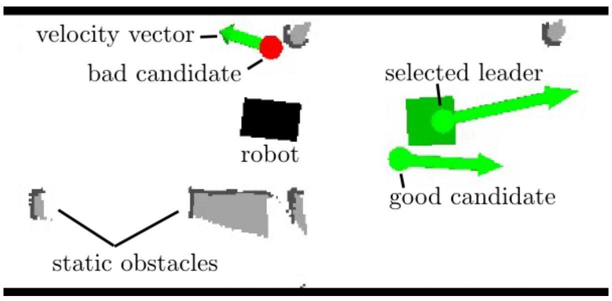


Figure 5.26: Details on a leader following tests showing how the entities are represented.

Good Leader In this first test, illustrated in Figure 5.27, the robot is behind a group of persons that move together. This group comprises several persons, but due to occlusions, not all of them are detected by the tracker system. The subjects that are closer to the robot are correctly classified as good leaders and the candidate with the higher score is selected and marked with a square.

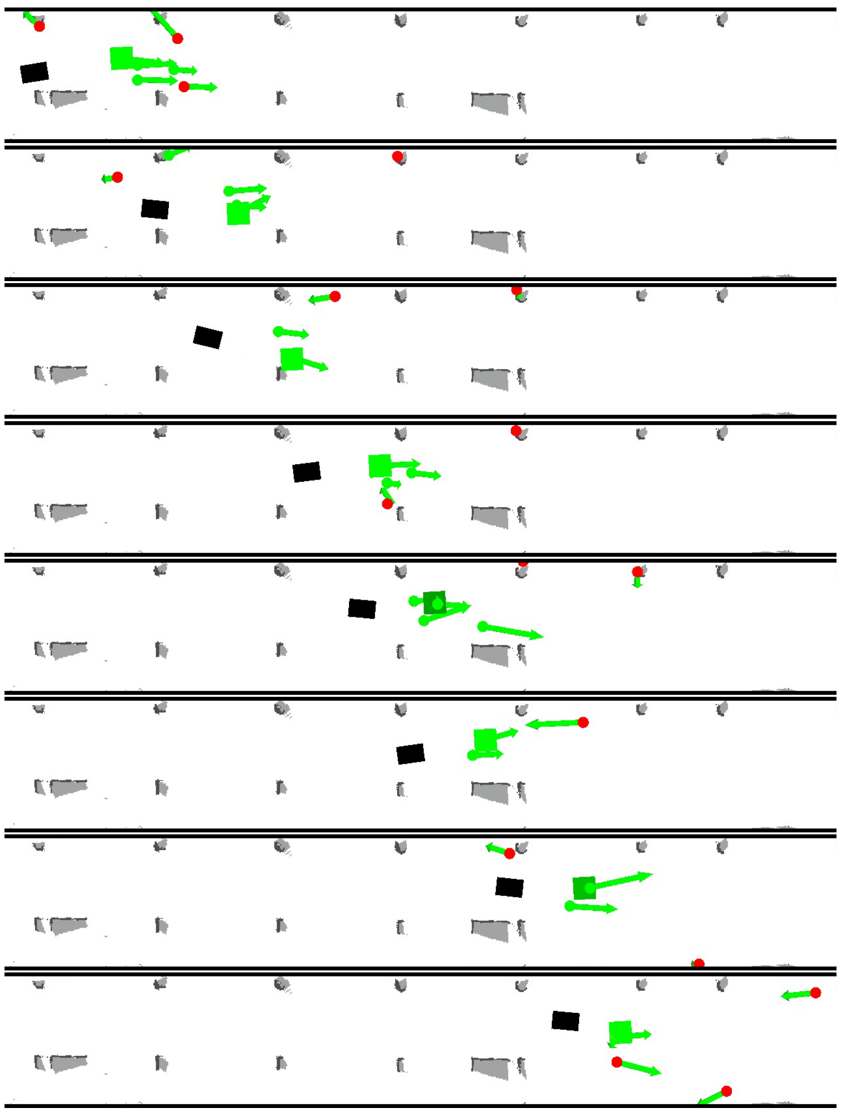


Figure 5.27: Robot following of a good leader, using the learning-based classification framework.

In the other hand, persons moving in the opposite direction are correctly classified as bad leaders, as can be seen by the red circles representing them. Besides this, due to limitations of the tracking system, sometimes static objects are erroneously tracked as persons, but the classification system has no problem in marking them as bad leader candidates, as can be seen by the red circles over some of the static objects.

It is possible to see that, in this test, the closest leader candidate is not always chosen. The reason is that in this framework, the rules system for leader selection was not implemented, using only the voting scheme presented in subsection 3.3.1.

Following a large group of people, as it is the case here, has advantages in the sense that they provide a compact "wave front" that can "protect" the robot from incoming subjects, as others inevitably give room to this group of persons to pass. The robot takes advantage of the space left behind by this group, moving together with it.

Bad Leader - far and/or fast This test shows a situation where the person that was initially chosen as leader is moving faster than the robot's maximum speed and, as a consequence, becomes too far from the robot. Once again, the robot manages to avoid persons moving from the opposite direction by following the path taken by the leader, as shown in Figure 5.28.

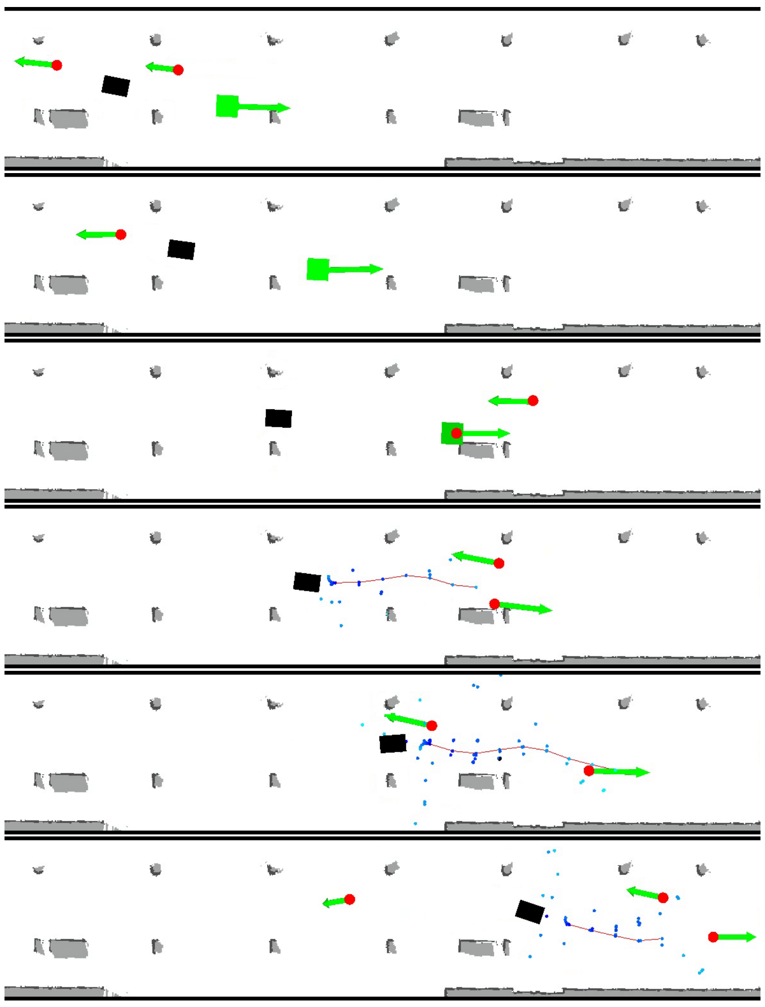


Figure 5.28: Robot following of a leader that is moving too fast, increasing the distance from the robot.

In the third image, the leader starts to be classified as a bad leader, which can be seen by the red circle, but due to the hysteresis of the voting system, it remains for a while as the chosen leader. On the fourth image, the subject is discarded as a leader. From that point on, the *solo* mode navigation is activated and the robot continues to move to the right part of the corridor using the RiskRRT navigation algorithm, still avoiding persons moving from the opposite direction.

Bad Leader - leader stops In this test, illustrated by Figure 5.29, the robot follows a group of three persons that are moving toward the right side of the corridor. The best leader candidate is the person in the right-most part of the group. During this test the corridor was almost empty and there were no persons moving in the opposite direction.

In the third image, the group being followed splits, as they noticed the presence of the robot (on the recorded dataset), and the currently selected leader stops (fourth image). The algorithm is able to quickly select as a new leader another member of the group, that continued to move. After this event, the group reunites and resume their motion towards the end of the corridor, followed by the robot.

This test illustrates the capability of the developed algorithm of maintaining a list of the available candidates and of switching among them according to their specific behavior and circumstances.

Bad Leader - leader moves aside The last test, shown in Figure 5.30, presents a situation where the leaders being followed notice the presence of the robot and move aside, in an attempt to free the path for the robot.

This is a situation that can usually be associated with discomfort of the persons being followed, as they prefer to move out of their typical path than continue along it while being followed. Unfortunately, this test was not successful. The robot kept following the leader even when they have moved out of the corridor.

One possible explanation to why the classifier failed in this situation, is that when this behavior occurred during the data acquisition phase, as the robot was manually controlled, it kept its heading in the direction of the end of the corridor. This resulted in an increase of the lateral displacement feature. As in this test the robot was closely following the leader, when the subject moved aside, the robot changed its heading to match the new leader position, as the robot motion controller always tries to minimize the heading error.

As a result, there is no change in the lateral displacement, because the robot always tries to face the leader position, and this feature is computed based on the difference of the heading of the robot and the leader position, which in this case, is close to zero.

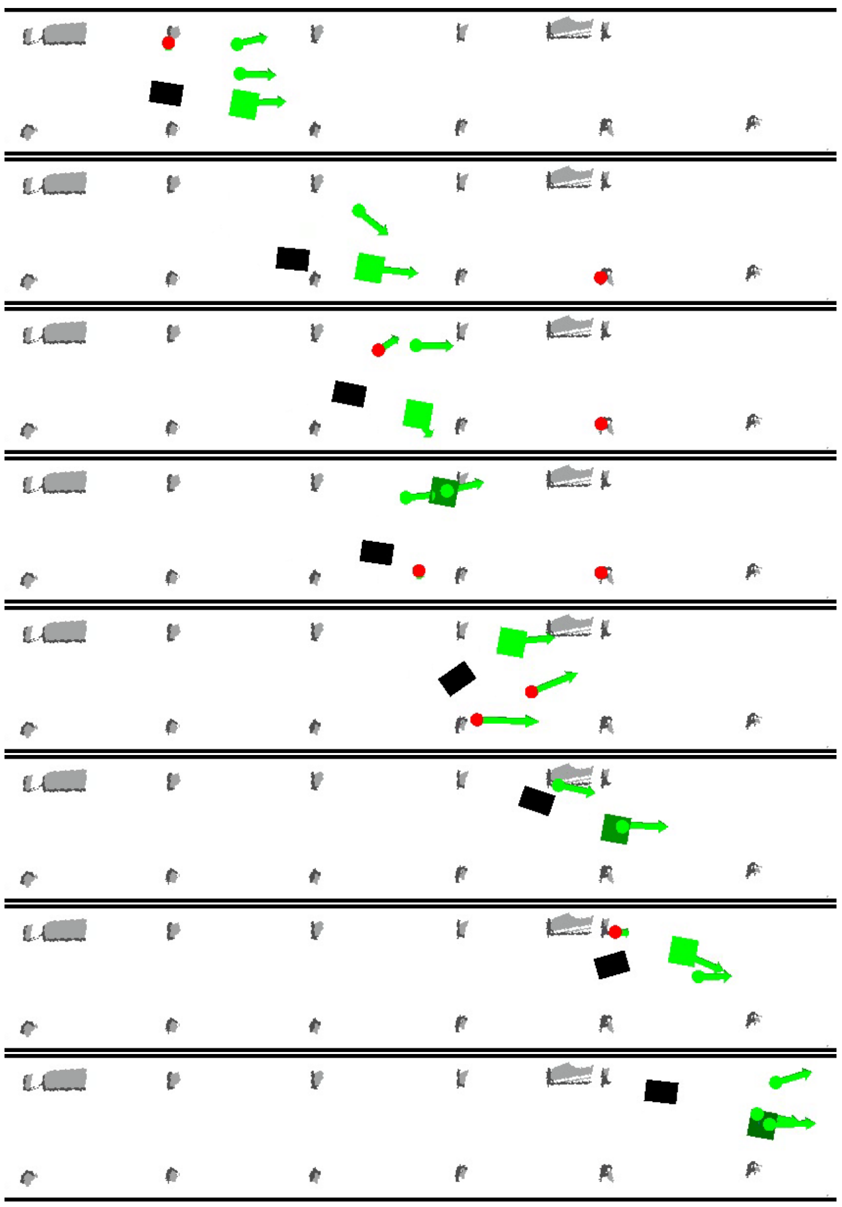


Figure 5.29: Robot following a group of leaders that splits, and some subjects stop moving, the classification system readily selects a new subject that is more suited to be a leader.

5.3.1 Analysis

The performance of the tests varied. In the first three experiments, where the robot was faced with situations of a good leader, and bad leaders that moved too fast or stopped, the algorithm managed to correctly classify the subjects. The classification framework was also able to discard, as leader candidates, agents that were not moving or were moving in the opposite direction of the robot. However, in the case of a bad leader that moved aside, the algorithm failed the classification, and the robot kept following that agent until it left the

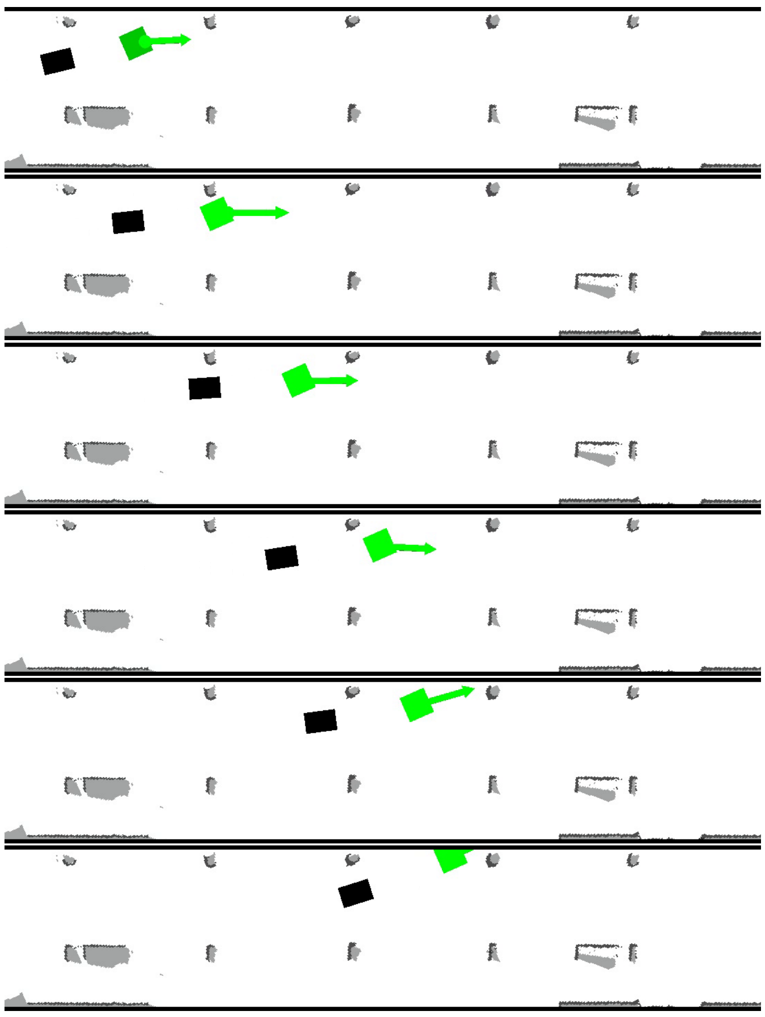


Figure 5.30: Robot following a leader that moves aside to an area that it cannot navigate. The classification algorithm fails to detect the subject as a bad leader.

scenario.

The integration of the navigation system with the leader voting scheme also provided good results. The algorithm maintained a list of possible candidates, with a score associated with each of them. The robot was able to properly follow the best candidate of this list and change the agent being followed, if it presented a bad leader behavior, which penalized its score.

At first glance, these results may suggest that a simple rule-based leader selection algorithm would provide similar results to the ones presented here. But if a rule-based system was to be developed to classify leaders in more complex environments, it would require exhaustive experiments and manual setting of thresholds in an attempt to deal with complex situations. This is where the advantages of a learning framework resides, as it can be extended to incorporate more complex features and learn from more intricate scenarios and examples, without the need of manual setting of parameters.

Future developments should use more complex features, possibly obtained from different sensors, as cameras, microphones and other, to enhance the classification capability of the learning algorithm. Data acquisition in more complex scenarios should also be conducted, where leaders would move according to more complex paths. Features should also reflect that a leader is moving away from a previous planned path of the robot, so it would detect when a leader is moving away from its desired path, and stop following this person.

5.4 Summary

In this chapter, experiments on a new method for leader evaluation were presented. This technique uses the AdaBoost algorithm to learn from labeled data how to classify a person as good or bad leader, according to a set of extracted features. Data was obtained using a laser range finder mounted on a robot that followed persons, while the labels associated with measurements were created using a simple framework where volunteers would watch a video of the tests and mark the moment they felt the robot should stop following someone.

The labeled data passed through a feature extraction process, and then different sets of features were used for training the Adaboost algorithm. This allowed a study of the importance of each of them to the classification process. After this, the voting scheme presented in subsection 3.3.1 was used to transform instantaneous classifications in a score that represented how good a person would be as a leader to the robot. This system was integrated with a multi-mode navigation system and experiments of leader selection and following were conducted in a virtual environment where real data was used to recreate the motion of persons. These are the contributions of this chapter:

- Developed a framework for leader following data acquisition and labeling, using human input.
- Proposed a set of features to be extracted and used in the training of an Adaboost algorithm, to classify persons as good or bad leaders.
- Conducted a study on the contribution of different features to the classification process, which allowed a better understanding of what are the underlying criteria that persons use to decide when start or stop following someone.
- Integrated the classification output with a voting scheme and with a multi-mode navigation, and conducted tests where a virtual robot would select and follow persons, based on their evaluation as leaders.
- Proposed a simulated environment where the motion of persons was recreated using previously recorded data, which allowed realistic features to be extracted and used by a virtual robot.

Chapter 6

Conclusions and Future Work

Robot navigation in human populated environments is a difficult task, due to the unpredictability and large complexity of the interactions that take place in those scenarios. This thesis proposed a new form of robotic navigation in such environments, where instead of regarding moving persons as obstacles, they were considered as entities that could help the robot to move in difficult situations.

It was hypothesized that by selecting and following leaders, the robot could take advantage from the navigation expertise of pedestrians, as moving along optimal paths, respecting social conventions, negotiating rights of passage, and dealing with complex and unexpected situations.

With this approach, the problem to be solved was no longer how to create motion plans in populated dynamic environments, but how to choose persons and how to follow them, according to the robot's objective. This lead to the development of two new methods for leader selection, that were the focus of the study in this work.

6.1 Leader Selection and Following Based on Reasoning

Analysis on the Leader Selection Methods The first method proposed that if a person was moving to the same destination as the robot, or was moving along a similar path initially planned by the robot, he or she would be a good leader to be followed. This meant that the robot could stick to this person and take advantage of his/her motion until reaching its objective.

To develop this technique, the likely goal and path of candidates had to be predicted. This work explored both simple approaches as state extrapolation, and complex ones, as models of typical paths, using probabilistic tools, as the GHMMs. After filtering leader candidates based on their predicted destination, a set of simple rules were used to rule out candidates that were in difficult positions to be followed, as behind the robot, or too far from it, for example.

One of the limitations of this method is that the environment must be known to the robot before the leader selection process can take place. In the case that the typical paths are used for goal prediction, the environment must first be observed, so enough data can be collected to create the models of the common traveled paths. This is not always a feasible situation. Besides that, in dynamic environments the motion of pedestrians may not always fit previous observations and unexpected situations and detours can result in erroneous goal predictions.

Analysis of the Navigation Tests To evaluate this approach, the leader selection method was integrated with a multi-mode navigation algorithm. This approach used three different navigation algorithms that were used in specific situations. On the *solo* mode, this system uses the RiskRRT algorithm, which is well suited for navigation in dynamic environments and that was used when no leader candidates were present or when none of those candidates could be used as leaders. When good leaders were present, the system could activate two other modes, depending on the distance between the robot and the leaders. When this distance was above $4m$, the *path follow* mode was activated, where the robot followed the path of the selected leader. In the case the distance was below $4m$, the *direct follow* mode was chosen, where the robot used two proportional controller to directly follow the leader, controlling the distance and the heading of the robot.

Two types of tests were performed, one that used a mixture of real data with a virtual world and another one that implemented a pedestrian simulator. In the first case, the real data was used to train a GHMM and also to recreate the motion of real persons in the virtual world, where a robot would select and follow these recreated agents. The advantage of this method is the use of real data for the training and prediction processes, but the drawback is that the recreated persons could not react to the presence of the robot, as their motion was only a playback of data acquired earlier.

In the second type of tests, only simulated data was used. In this setup, a pedestrian simulator was created, based on the Social Forces model, where the sum of attractive and repulsive forces determined the resulting motion of simulated pedestrians. Using this tool, it was possible to create realistic experiments where the presence of the robot affected the surrounding agents.

In both types of tests the ability of the robot's algorithm to select and follow leaders were successfully demonstrated, enabling the robot to navigate through difficult situations where other approaches had suboptimal results. In these situations, the robot effectively became part of a group and avoided collision with other agents.

6.2 Leader Selection Based on Learning

Analysis on the Leader Evaluation Framework Instead of using sets of rules and conditions, the second method originated from the desire to understand and replicate how humans perform leader selection. To accomplish this proposition, a machine learning framework was used to learn from real examples how to classify someone as a good or a bad leader candidate. This study can be divided into five parts: data acquisition, data labeling, feature extraction, algorithm training and classification tests.

For the data acquisition, a car-like robot, remotely operated, moved along an open corridor between two objective points. Whenever persons were present in the robot's path, the human operator positioned the robot behind them and engaged in a following behavior, until reaching one of its objective points. While moving, the robot recorded measurements made with a camera and a laser range finder, installed on it.

After this, the labeling part took place. The video of the tests was shown to volunteers, that marked the instant when, according to their opinion, the robot should have abandoned the leader following behavior, and were asked the reason they chose such instant. In the end of this process, the labels of the participants were merged and categorized according to the reason that the robot should have left the leader, according to the evaluator's opinion.

For the feature extraction, an algorithm used the laser measurements to detect and track the motion of persons, so the measurements (or features), were calculated based on the state of the robot and the state of the tracked persons. Examples of the extracted features were the distance between candidate and robot, their relative velocity and others.

The machine learning algorithm used was the AdaBoost, because it allows an evaluation of the contribution of the features used for the classification process. This is an important factor, as it permitted the study of the features that are unconsciously used by humans when deciding to start or stop following someone. Different sets of features were used in the training phase, with the less important features being sequentially removed.

Experiments in a straight corridor showed that the lateral displacement, the distance between the robot and the leader, and the standard deviation of this distance, were the features that mostly contributed to the classification process. It is interesting to notice that among the three most important features, two of them are measurements that are not dependent on the dynamics of the environment (lateral displacement and distance). These findings can be used in future designs of leader following algorithms, to better select leaders and reduce the discomfort caused by the robot.

The study of features importance in leader selection is an important contribution of this Thesis, as no other references of works with similar approaches or objectives, were found. It also launches the base for future studies in a new area of Human Robot Interaction (HRI), related to the human comfort and acceptance when a robot follows them, which is different from current proxemics studies where the focus is in the robot navigating while respecting personal spaces.

It is important to emphasize that these results were obtained using only one type of robot, and in one specific test scenario, where people moved along an open corridor. Also the features extracted are highly dependent on the type of sensors and the algorithm used. Without more tests with different platforms, sensors, scenarios and algorithms for feature extraction, care must be taken when generalizing these findings.

In different setups, the features extracted will have distinct characteristics from the ones used in this work, due to different types of platforms, sensors and algorithms used. In this case, the advantages of using a learning framework become evident. As the presented algorithms learn from examples, new classifiers can adapt to the features obtained in different experiments. In resume, the framework described in this work permits that the same type of experiments to be repeated for different robots and scenarios, while a learning tool allows new classifiers to be trained according to new features and situations.

Analysis of the Navigation Tests This classification framework was also evaluated while integrated with a navigation system, so the performance of a robot that followed persons using this technique could be assessed. As the classification system worked over instantaneous measurements, a method that created hysteresis for the leader selection was developed, so the history of classifications could be taken into account. This was accomplished using a voting system, where positive classifications increased a candidate score and negative classifications reduced it, but by a larger factor. These rules allowed a score to be associated with each leader candidate, that could be used for the leader selection, associated to the quality of each subject as a leader.

For this tests, the same navigation system presented in Section 6.1 was used. Also the same type of setup that mixed real data measurements with a virtual robot was used. The difference here is that the features were extracted based on the virtual robot position, so although the motion of pedestrians had been previously recorded, the features were different from the ones used for training the algorithm.

The tests were conducted in the four different leader situations, previously classified. Although the robot could properly identify, follow and abandon leaders in most of the cases, it failed to detect the bad leader situation where the leaders moved aside. This was a difficult situation, that was associated with the discomfort of the person being followed. The conclusion was that the features used for the training of the classification algorithm were not enough to detect that situations when a robot was closely following a leader.

As a side study, during the data collection process, a small group of persons (N=10) were interviewed to assess how they felt about having a robot following them. In all the interviews, people told they noticed the presence of the robot, and they changed their behavior because of it. This deserves further exploration, with more experiments and a larger number of interviews, to investigate what are the aspects of the robot design and behavior that influence the persons being followed, and how that affects them.

6.3 Comparison of Leader Selection Methods

This thesis presented two methods for leader selection, one that based the decision on the prediction of a candidate's goal together with a set of rules, and other where a learning framework was trained using examples of good or bad leaders labeled by humans.

As experiments were conducted in different setups, it is difficult to establish a quantitative comparison for both methods. It is possible, however, to point what are the strengths and weaknesses of each approach.

The reasoning-based leader selection had very good results in selecting a leader for long-term following and in selecting and following leaders in densely populated environments. However, it is heavily dependent on the goal prediction of candidates and it requires a previous knowledge of the environment to obtain good goal prediction. Besides that, the algorithm only choses the best leader, disregarding other sub-optimal candidates.

The learning-base approach, in the other hand, is able to evaluate each candidate based on a series of features, creating combinations that may be associated with humans behaviors and discomfort. Another advantage is that it can be used in unknown environments, and the learning framework can easily be expanded to account for new features and situations. This method also keeps a list of candidates and give scores to them, associated with their behavior along time, so the task of leader switching becomes easier. The drawbacks of this technique is that, in the form it was implemented here, a leader may lead the robot away from its objective, and it may fail to detect some situations, namely when leaders present the move aside behavior.

As a suggestion of future developments, a technique that integrated the reasoning-based and the learning-based approach could overcome individual limitations and sum the strengths of each method, resulting in a much more robust leader selection algorithm.

6.4 Future Work

This study only entered briefly in a large an unexplored field of how to take advantage of moving entities, for the benefit of robot motion. This thesis focused on only one form of taking advantage, which was by following selected leaders in the environment. But this concept has a broader definition, and other forms of taking advantage from human motion are possible. For example, using the typical motion paths of persons as optimal paths in an environment, to guide the search of motion planning algorithms, even when no persons are present.

Regarding improvements in the work developed here, intensive data collection should be conducted with automated robots following persons, using a variety of sensors and scenarios, so a large database of leader following examples can be constructed and shared with other researchers. This would allow the training of better machine learning classifiers and a real comparison between different methods using the same ground-truth.

This database could also profit from a more complex labeling system. In the current work, only one transition from good to bad leader was present in each test, using a binary label. In future developments, it could be interesting to have many levels of leader quality, to better reflect real situations, where the transition from good to bad leader is a gradual change. One form of accomplishing this, would be to create transient labels, backpropagating instant transitions into gradual ones.

As some of the tests of robot following using the classification framework have failed, it is necessary to explore more features that could be used to improve the performance of classification algorithm. Examples of such features are be body posture, gaze direction, and also relative measurements taking into account a preferential path by of the robot. This ideas come from the fact that in many cases, persons looked back several times before changing their behavior, which could be associated with their discomfort. But this reactions could not be captured using only a laser range finder and more complex algorithms are required for the detection of such features, using camera images, for example.

This work conducted tests in simulated environments and in mixed conditions, using real data together with virtual environments. So an essential future task, is to perform experiments in real situations. Only then the true impact of the developed algorithms can be evaluated. This will allow a deeper study on the reaction of persons to a robot following then, and may lead to adjustments in the algorithms to cope with situations that were not anticipated in this thesis.

Finally, with a better understanding of what characterizes a good leader (Dyer et al., 2009) and of the behavior of pedestrians moving together with a robot, it will be possible to explore other situations of HRI. For example, instead of following persons, the robot would became the guide of groups of humans, or even actively change the organization of a group De Schutter et al. (2001). This could bring benefits in emergency or panic situations, where the default human behavior is actually dangerous to the group (Helbing et al., 2000), but could be altered with the intervention of robots.

6.5 Contributions

Different forms of navigation taking advantage of moving agents were explored in this thesis. The main contributions can be resumed as follows:

1. Implementation of a probabilistic approach of leader selection, base on goal prediction, using the Growing Hidden Markov Model (GHMM) (Stein et al., 2012a,b).

2. Development of a pedestrian simulator, that was integrated on the leader selection framework, which allowed realistic experiments where agents reacted to the presence

of the robot and to the conditions of the environment (Stein et al., 2013b). Experiments in crowded environment, using the pedestrian simulator, showed the benefits of taking advantage of the motion of other agents, where the robot successfully navigated through difficult situations, while other state-of-the-art algorithms provided suboptimal solutions.

3. Implementation of a multi-mode navigation framework, which used a state-of-the-art technique for navigating without a leader, and two other navigation strategies for the leader following, one that followed the leader's path and other that followed directly the leader.

4. Developed a framework for leader following data acquisition and labeling, using human input, and proposed a set of features to be extracted and used in the training of an Adaboost algorithm, to classify persons as good or bad leaders. This framework was also used to conducted a study on the contribution of different features to the classification process, which allowed a better understanding of what are the underlying criteria that persons use to decide when start or stop following someone (to appear in Stein et al., 2013a).

5. Integrated the AdaBoost classification output with a voting scheme and with a multi-mode navigation, and conducted tests where a virtual robot would select and follow persons, based on their evaluation as leaders. Tests were conducted in a simulated environment where the motion of persons was recreated using previously recorded data, which allowed realistic features to be extracted and used by a virtual robot.

References

Abe, Y. and Yoshiki, M. (2001). Collision avoidance method for multiple autonomous mobile agents by implicit cooperation. In *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millenni*, pages 1207–1212, Maui, HI, USA.

Alexander, R. M. (1984). The gaits of bipedal and quadrupedal animals. *The International Journal of Robotics Research*, 3(2):49–59.

Almeida, J. (2010). Target tracking using laser range finder with occlusion. Master's thesis, Universidade de Aveiro, Aveiro, Portugal.

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

Baiget, P., Sommerlade, E., Reid, I., and Gonzalez, J. (2008). Finding prototypes to estimate trajectory development in outdoor scenarios. In *Proceedings of the 1st THEMIS Workshop*, pages 27–34.

Bennewitz, M., Burgard, W., Cielniak, G., and Thrun, S. (2005). Learning motion patterns of people for compliant robot motion. *The International Journal of Robotics Research*, 24(1):31.

Bennewitz, M., Burgard, W., and Thrun, S. (2003). Adapting navigation strategies using motions patterns of people. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, volume 2, pages 2000–2005.

Borenstein, J. and Ulrich, I. (1997). The guidecane-a computerized travel aid for the active guidance of blind pedestrians. In *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, volume 2, pages 1283–1288.

Brock, O. and Khatib, O. (1999). High-speed navigation using the global dynamic window approach. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, volume 1.

Brooks, R. (1986). A robust layered control system for a mobile robot. *Robotics and Automation, IEEE Journal of*, 2(1):14–23.

Burgard, W., Cremers, A. B., Fox, D., Hähnel, D., Lakemeyer, G., Schulz, D., Steiner, W., and Thrun, S. (1999). Experiences with an interactive museum tour-guide robot. *Artificial Intelligence*, 114(1-2):3–55.

Choi, W. and Latombe, J.-C. (1991). A reactive architecture for planning and executing robot motions with incomplete knowledge. In *IEEE/RSJ International Workshop on Intelligent Robots and Systems '91. 'Intelligence for Mechanical Systems, Proceedings IROS '91*, pages 24 –29 vol.1.

Ciolek, T. M. and Kendon, A. (1980). Environment and the spatial arrangement of conversational encounters. *Sociological Inquiry*, 50(3-4):237–271.

De Schutter, G., Theraulaz, G., and Deneubourg, J. (2001). Animal-robots collective intelligence. *Annals of Mathematics and Artificial Intelligence*, 31(1):223–238.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271.

Drauschke, M. (2008). Feature subset selection with adaboost and adtboost. Technical report, Tech. rept. Department of Photogrammetry, University of Bonn.

Dyer, J. R., Johansson, A., Helbing, D., Couzin, I. D., and Krause, J. (2009). Leadership, consensus decision making and collective behaviour in humans. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 364:781–789.

Ellis, D., Sommerlade, E., and Reid, I. (2009). Modelling pedestrian trajectory patterns with gaussian processes. In *2009 IEEE 12th International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 1229–1234. IEEE.

Erdmann, M. and Lozano-Pérez, T. (1987). On multiple moving objects. *Algorithmica*, 2(1-4):477–521.

Fiorini, P. and Shiller, Z. (1993). Motion planning in dynamic environments using the relative velocity paradigm. In *Proccedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 560–560.

Fiorini, P. and Shiller, Z. (1998). Motion planning in dynamic environments using velocity obstacles. *The International Journal of Robotics Research*, 17(7):760–772.

Fraichard, T. and Laugier, C. (1993). Path-velocity decomposition revisited and applied to dynamictrajectory planning. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 40–45.

Freund, Y. and Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139.

Fritzke, B. et al. (1995). A growing neural gas network learns topologies. *Advances in neural information processing systems*, 7:625–632.

Frost, R. (1916). *Mountain Interval.* Henry Holt.

Fujimura, K. and Samet, H. (1989). A hierarchical strategy for path planning among moving obstacles. *IEEE Transactions on Robotics and Automation*, 5(1):61–69.

Fulgenzi, C., Spalanzani, A., and Laugier, C. (2007). Dynamic obstacle avoidance in uncertain environment combining PVOs and occupancy grid. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 1610–1616.

Fulgenzi, C., Spalanzani, A., and Laugier, C. (2009). Probabilistic motion planning among moving obstacles following typical motion patterns. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 4027–4033.

Fulgenzi, C., Spalanzani, A., Laugier, C., and Tay, C. (2010). Risk based motion planning and navigation in uncertain dynamic environment. Research report, INRIA Rhone-Alpes.

Fulgenzi, C., Tay, C., Spalanzani, A., and Laugier, C. (2008). Probabilistic navigation in dynamic environment using rapidly-exploring random trees and gaussian processes. In *IEEE/RSJ 2008 International Conference on Intelligent RObots and Systems*, Nice, France.

Gockley, R., Forlizzi, J., and Simmons, R. (2007). Natural person-following behavior for social robots. In *Proceedings of the ACM/IEEE international conference on Human-robot interaction*, pages 17–24.

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

Hall, E. T. (1966). *The hidden dimension.* Doubleday.

Hart, P. E., Nilsson, N. J., and Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on*, 4(2):100–107.

Helbing, D., Farkas, I., and Vicsek, T. (2000). Simulating dynamical features of escape panic. *Nature, Vol. 407, pp. 487-490, 2000.*

Helbing, D., Farkas, I. J., Molnar, P., and Vicsek, T. (2002). Simulation of pedestrian crowds in normal and evacuation situations. In Sharma, S. D. and Schreckenberg, M., editors, *Pedestrian and Evacuation Dynamics*, pages 21–58. Springer Berlin.

Helbing, D. and Molnar, P. (1995). Social force model for pedestrian dynamics. *Physical Review E*, 51:4282–1286.

Helbing, D., Molnar, P., Farkas, I., and Bolay, K. (2001). Self-organizing pedestrian movement. *Environment and Planning B*, 28(3):361–384.

Henry, P., Vollmer, C., Ferris, B., and Fox, D. (2010). Learning to navigate through crowded environments. In *2010 IEEE International Conference on Robotics and Automation (ICRA)*, pages 981–986. IEEE.

Ho, D. M., Hu, J.-S., and Wang, J.-J. (2012). Behavior control of the mobile robot for accompanying in front of a human. In *Advanced Intelligent Mechatronics (AIM), 2012 IEEE/ASME International Conference on*, pages 377–382.

Hoeller, F., Schulz, D., Moors, M., and Schneider, F. E. (2007). Accompanying persons with a mobile robot using motion prediction and probabilistic roadmaps. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 1260–1265.

Huettenrauch, H., Eklundh, K., Green, A., and Topp, E. (2006). Investigating spatial relationships in human-robot interaction. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5052 –5059.

Ikeda, T., Chigodo, Y., Rea, D., Zanlungo, F., Shiomi, M., and Kanda, T. (2012). Modeling and prediction of pedestrian behavior based on the sub-goal concept. In *Robotics: Science and Systems VIII.*

iRobot (2013). iRobot corporation: Robots that make a difference. Retrieved June 2013, from `http://www.irobot.com/us/`.

Ishiguro, H., Ono, T., Imai, M., Maeda, T., Kanda, T., and Nakatsu, R. (2001). Robovie: an interactive humanoid robot. *Industrial robot: An international journal*, 28(6):498–504.

Johnson, N. and Hogg, D. (1996). Learning the distribution of object trajectories for event recognition. *Image and Vision Computing*, 14(8):609–615.

Kanda, T., Glas, D. F., Shiomi, M., Ishiguro, H., and Hagita, N. (2008). Who will be the customer?: a social robot that anticipates people's behavior from their trajectories. In

Proceedings of the 10th international conference on Ubiquitous computing, UbiComp '08, pages 380–389, New York, NY, USA. ACM.

Kant, K. and Zucker, S. W. (1986). Toward efficient trajectory planning: The path-velocity decomposition. *The International Journal of Robotics Research,*, pages 72 –89 vol. 5:.

Kato, H., Billinghurst, M., Poupyrev, I., Imamoto, K., and Tachibana, K. (2000). Virtual object manipulation on a table-top AR environment. In *Proceedings of ISAR 2000*, pages 111–119.

Kavraki, L. E., Svestka, P., Latombe, J. C., and Overmars, M. H. (1996). Probabilistic roadmaps for path planning in high-dimensionalconfiguration spaces. *IEEE transactions on Robotics and Automation*, 12(4):566–580.

Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. *The International Journal of Robotics Research*, 5(1):90–98.

Kim, K., Lee, D., and Essa, I. (2011). Gaussian process regression flow for analysis of motion trajectories. In *2011 IEEE International Conference on Computer Vision (ICCV).*

Kluge, B. and Prassler, E. (2004). Reflective navigation: individual behaviors and group behaviors. In *Proceedings of IEEE International Conference on Robotics and Automation ICRA*, volume 4, pages 4172–4177.

Kohlbrecher, S., Meyer, J., von Stryk, O., and Klingauf, U. (2011). A flexible and scalable slam system with full 3d motion estimation. In *Proc. IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR).* IEEE.

Koren, Y. and Borenstein, J. (1991). Potential field methods and their inherent limitations for mobile robot navigation. *In Proc. IEEE Int. Conf. Robotics and Automation*, pages 1398—1404.

Krogh, B. and Thorpe, C. (1986). Integrated path planning and dynamic steering control for autonomous vehicles. In *Robotics and Automation. Proceedings. 1986 IEEE International Conference on*, volume 3, pages 1664–1669.

Kruse, T., Basili, P., Glasauer, S., and Kirsch, A. (2012). Legible robot navigation in the proximity of moving humans. In *Advanced Robotics and its Social Impacts (ARSO), 2012 IEEE Workshop on*, pages 83–88.

Kruse, T., Pandey, A. K., Alami, R., and Kirsch, A. (2013). Human-aware robot navigation: A survey. *Robotics and Autonomous Systems.*

Large, F., Laugier, C., and Shiller, Z. (2005). Navigation among moving obstacles using the NLVO: principles and applications to intelligent vehicles. *Autonomous Robots*, 19(2):159–171.

Large, F., Sckhavat, S., Shiller, Z., and Laugier, C. (2002). Using non-linear velocity obstacles to plan motions in a dynamic environment. In *Proceedings of IEEE International Conference on Control, Automation, Robotics and Vision*, volume 2, pages 734–739.

Large, F., Vasquez, D., Fraichard, T., and Laugier, C. (2004). Avoiding cars and pedestrians using velocity obstacles and motion prediction. In *Proceedings of IEEE Intelligent Vehicles Symposium*, pages 375–379.

Latombe, J. (1991). *Robot Motion Planning.* Kluwer Academic, 1 edition.

LaValle, S. M. (2006). *Planning Algorithms.* Cambridge University Press, Cambridge, U.K. Available at http://planning.cs.uiuc.edu/.

LaValle, S. M. and Kuffner Jr, J. J. (2001). Randomized kinodynamic planning. *The International Journal of Robotics Research*, 20(5):378.

Lozano-Perez, T. (1983). Spatial planning: A configuration space approach. *IEEE Transactions on Computers*, C-32(2):108 –120.

Luber, M., Spinello, L., Silva, J., and Arras, K. (2012). Socially-aware robot navigation: A learning approach. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 902–907.

Makris, D. and Ellis, T. (2002). Path detection in video surveillance. *Image and Vision Computing*, 20(12):895–903.

Miura, J., Satake, J., Chiba, M., Ishikawa, Y., Kitajima, K., and Masuzawa, H. (2010). Development of a person following robot and its experimental evaluation. In *Proceedings of the 11th International Conference on Intelligent Autonomous Systems*, pages 89–98, Ottawa, Canada.

Miura, J. and Shirai, Y. (2000). Modeling motion uncertainty of moving obstacles for robot motion planning. In *IEEE International Conference on Robotics and Automation, 2000. Proceedings. ICRA '00*, volume 3, pages 2258–2263 vol.3.

Miura, J., Uozumi, H., and Shirai, Y. (1999). Mobile robot motion planning considering the motion uncertainty of moving obstacles. *Proceedings. 1999 IEEE International Conference on Systems, Man, and Cybernetics*, pages 692–697.

Mombaur, K., Truong, A., and Laumond, J. (2009). From human to humanoid locomotion - an inverse optimal control approach. *Autonomous Robots*, 28:369–383.

Müller, J., Stachniss, C., Arras, K., and Burgard, W. (2008). Socially inspired motion planning for mobile robots in populated environments. In *Proc. of International Conference on Cognitive Systems*.

Murphy, L. and Corke, P. (2012). STALKERBOT: learning to navigate dynamic human environments by following people. In *Proceedings of the 2012 Australasian Conference on Robotics & Automation*.

Myers, T., Noel, T., Parent, M., and Vlacic, L. (2005). Autonomous motion of a driverless vehicle operating among dynamic obstacles. In *Proceedings of IEEE Conference on Decision and Control, European Control Conference*, pages 5071–5076.

O'Callaghan, S. T., Singh, S. P. N., Alempijevic, A., and Ramos, F. T. (2011). Learning navigational maps by observing human motion patterns. In *2011 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4333–4340. IEEE.

Petti, S. and Fraichard, T. (2005). Partial motion planning framework for reactive planning within dynamic environments. In *Proc. of the IFAC/AAAI Int. Conf. on Informatics in Control, Automation and Robotics*.

Prassler, E., Scholz, J., and Fiorini, P. (2001). A robotics wheelchair for crowded public environment. *Robotics & Automation Magazine, IEEE*, 8(1):38–45.

Quigley, M., Gerkeyy, B., Conleyy, K., Fausty, J., Footey, T., Leibsz, J., Bergery, E., Wheelery, R., and Ng, A. (2009). ROS: an open-source robot operating system. In *ICRA Workshop on Open Source Software*.

Quinlan, S. and Khatib, O. (1993). Elastic bands: Connecting path planning and control. In *Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference on*, pages 802–807.

Reif, J. and Sharir, M. (1985). Motion planning in the presence of moving obstacles. In *Proceedings of 26th Annual Symposium on Foundations of Computer Science*, pages 144–154.

Reynolds, C. W. (1987). Flocks, herds, and schools: A distributed behavioral model. *COMPUTER GRAPHICS*, 21:25–34.

Reynolds, C. W. (1999). Steering behaviors for autonomous characters. In *Game Developers Conference. http://www. red3d. com/cwr/steer/gdc99*.

Rios-Martinez, J., Spalanzani, A., and Laugier, C. (2011). Understanding human interaction for probabilistic autonomous navigation using Risk-RRT approach. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2014–2019. IEEE.

Robotics, N. (2013). BotJunkie review: Neato robotics XV-11 - neato robotics. Retrieved June 2013, from `http://www.neatorobotics.com/botjunkie-review-neato-robotics-xv-11/`.

Sehestedt, S., Kodagoda, S., and Dissanayake, G. (2010). Robot path planning in a social context. In *2010 IEEE Conference on Robotics Automation and Mechatronics (RAM)*, pages 206–211. IEEE.

Severinson-Eklundh, K., Green, A., and HÃ$\frac{1}{4}$ttenrauch, H. (2003). Social and collaborative aspects of interaction with a service robot. *Robotics and Autonomous Systems*, 42(3-4):223–234.

Shiller, Z., Large, F., and Sekhavat, S. (2001). Motion planning in dynamic environments: obstacles moving along arbitrary trajectories. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, volume 4, pages 3716–3721 vol.4.

Shiomi, M., Zanlungo, F., Hayashi, K., and Kanda, T. (2012). A framework with a pedestrian simulator for deploying robots into a real environment. In Noda, I., Ando, N., Brugali, D., and Kuffner, J., editors, *Simulation, Modeling, and Programming for Autonomous Robots*, volume 7628 of *Lecture Notes in Computer Science*, pages 185–196. Springer Berlin / Heidelberg.

Sidenbladh, H., Kragic, D., and Christensen, H. I. (1999). A person following behaviour for a mobile robot. In *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, volume 1, pages 670–675.

Sisbot, E., Marin-Urias, L., Alami, R., and Simeon, T. (2007). A human aware mobile robot motion planner. *IEEE Transactions on Robotics*, 23(5):874 –883.

Snape, J., van den Berg, J., Guy, S., and Manocha, D. (2009). Independent navigation of multiple mobile robots with hybrid reciprocal velocity obstacles. In *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*, pages 5917–5922.

Stachniss, C. and Burgard, W. (2002). An integrated approach to goal-directed obstacle avoidance under dynamic constraints for dynamic environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2002*, volume 1, pages 508– 513 vol.1. IEEE.

Stein, P., Santos, V., Spalanzani, A., and Laugier, C. (2013a). Learning-based leader classification. In *IROS 2013 Workshop on Assistance and Service Robotics in a Human Environment.*

Stein, P., Santos, V., Spalanzani, A., and Laugier, C. (2013b). Navigating in populated environments by following a leader. In *International Symposium on Robot and Human Interactive Communication (RO-MAN).*

Stein, P., Spalanzani, A., Laugier, C., and Santos, V. (2012a). Leader selection and following in dynamic environments. In *2012 12th International Conference on Control Automation Robotics Vision (ICARCV)*, pages 124–129.

Stein, P., Spalanzani, A., Santos, V., and Laugier, C. (2012b). Robot navigation taking advantage of moving agents. In *IROS 2012 Workshop on Assistance and Service robotics in a human environment.*

Susnjak, T. (2009). Accelerating classifier training using AdaBoost within cascades of boosted ensembles. Science in computer sciences, Massey University, Auckland, New Zealand.

Svenstrup, M., Bak, T., Maler, O., Andersen, H. J., and Jensen, O. B. (2009). Pilot study of person robot interaction in a public transit space. *Research and Education in Robotics – EUROBOT 2008*, pages 96–106.

Tadokoro, S., Hayashi, M., Manabe, Y., Nakami, Y., and Takamori, T. (1995). On motion planning of mobile robots which coexist and cooperate with human. In *1995 IEEE/RSJ International Conference on Intelligent Robots and Systems 95. 'Human Robot Interaction and Cooperative Robots', Proceedings*, volume 2, pages 518–523 vol.2.

Takayama, L. and Pantofaru, C. (2009). Influences on proxemic behaviors in human-robot interaction. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2009. IROS 2009*, pages 5495–5502. IEEE.

Tay, C. and Laugier, C. (2007). Modelling smooth paths using gaussian processes. In *Proc. of the Int. Conf. on Field and Service Robotics*, Chamonix, France.

Thrun, S., Bennewitz, M., Burgard, W., Cremers, A. B., Dellaert, F., Fox, D., Hahnel, D., Rosenberg, C., Roy, N., and Schulte, J. (1999). MINERVA: a second-generation museum tour-guide robot. In *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, volume 3.

Tipaldi, G. and Arras, K. (2011). I want my coffee hot! learning to find people under spatio-temporal constraints. In *2011 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1217 –1222.

Tranberg Hansen, S., Svenstrup, M., Andersen, H., and Bak, T. (2009). Adaptive human aware navigation based on motion pattern analysis. In *The 18th IEEE International Symposium on Robot and Human Interactive Communication, 2009. RO-MAN 2009*, pages 927–932.

Trautman, P. and Krause, A. (2010). Unfreezing the robot: Navigation in dense, interacting crowds. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 797–803. IEEE.

Treuille, A., Cooper, S., and Popovic, Z. (2006). Continuum crowds. In *ACM SIGGRAPH 2006 Papers*, page 1168.

Tsuchiya, M. and Fujiyoshi, H. (2006). Evaluating feature importance for object classification in visual surveillance. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 2, pages 978–981.

van den Berg, J., Guy, S., Lin, M., and Manocha, D. (2011). Reciprocal n-body collision avoidance. *Robotics Research*, pages 3–19.

van den Berg, J., Lin, M., and Manocha, D. (2008a). Reciprocal velocity obstacles for real-time multi-agent navigation. In *Proc. of IEEE Conference on Robotics and Automation*, pages 1928–1935.

van den Berg, J., Patil, S., Sewall, J., Manocha, D., and Lin, M. (2008b). Interactive navigation of multiple agents in crowded environments. In *Proceedings of the 2008 symposium on Interactive 3D graphics and games*, pages 139–147.

Vasquez, D., Stein, P., Rios-Martinez, J., Escobedo, A., Spalanzani, A., and Laugier, C. (2012). Human aware navigation for assistive robotics. In *International Symposium on Experimental Robotics (ISER)*.

Vasquez Govea, D. A., Fraichard, T., and Laugier, C. (2009). Growing hidden markov models: An incremental tool for learning and predicting human and vehicle motion. *International Journal of Robotics Research*, 28(11-12):1486–1506.

Vaughan, R. (2008). Massively multi-robot simulation in stage. *Swarm Intelligence*, 2(2-4):189–208.

Walters, M. L., Dautenhahn, K., Te Boekhorst, R., Koay, K. L., Syrdal, D. S., and Nehaniv, C. L. (2009). An empirical framework for human-robot proxemics. In *Symposium at the AISB09 Convention,*.

Yu, C.-C. and Wang, C.-C. (2012). Collision- and freezing-free navigation in dynamic environments using learning to search. In *2012 Conference on Technologies and Applications of Artificial Intelligence (TAAI)*, pages 151–156.

Zhu, Q. (1991). Hidden markov model for dynamic obstacle avoidance of mobile robot navigation. *Robotics and Automation, IEEE Transactions on*, 7(3):390–397.

Appendix A

Test Framework for Typical Paths Modeling and Prediction Using Gaussian Process

A Gaussian Process is a generalization of the Gaussian probability distribution. One advantage of this method is that it is possible to incorporate human motion uncertainties into the model, providing a probabilistic framework for the prediction of future paths. Another advantage of this technique is that it allows the representation of pedestrians' trajectories as continuous functions, eliminating the need of a discretization step of the sample paths. This also allows predictions to be made in any desired timestep, without the need of interpolation.

A GP can be fully defined by a mean function $m(x)$ and a covariance function $k(x, x')$

$$f \sim GP(m(x), k(x, x'))$$

these functions have free parameters that are called *hyperparameters* and which are computed during the learning phase. The resulting mean function can be understood as the typical path and the covariance function represents the uncertainty of the model in each point.

Test Setup On the tests conducted, a group of trajectories were generated using a small-scale robot (10 cm long), shown in Figure A.1. It is interesting to use a real setup because, in this way, the real constraints of the system could be taken into account. A small scale setup also has the advantages that complex paths and situations could be experimented in a small indoor area. Figure A.2 shows the schema of the setup that was built in the laboratory.

As the robot has no internal sensors to determine its position on the environment, an external system tracks and computes the position and orientation of the robot. The position and orientation of the robot in the environment are obtained by an overhanging camera, using the *ARToolKit* augmented reality system (Kato et al., 2000), together with a Kalman Filter

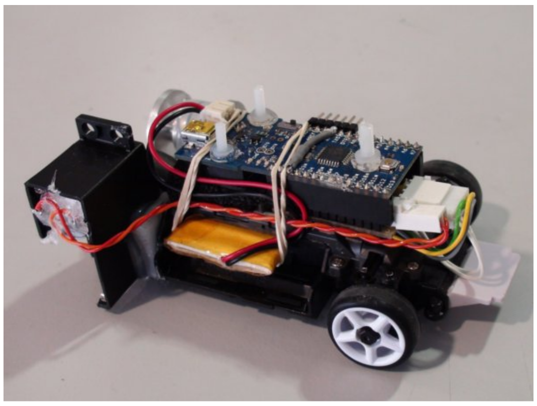


Figure A.1: Small scale robot used to generate real trajectories for the typical paths modelling with GP.

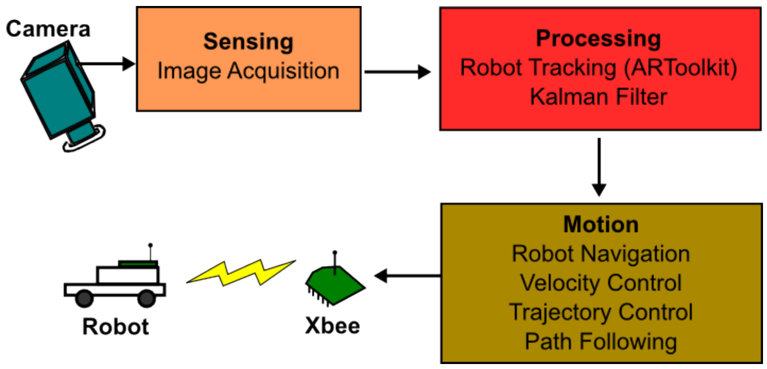


Figure A.2: Schema of the implemented setup. The image of the arena is acquired and passed to a module that performs detection and tracking, based on the robot's position and orientation, control commands are computed and sent via a wireless link.

to cope with noise and temporary failures in the tracking. This approach allows inexpensive robots to have accurate information about their surroundings and about other robots, what would only be possible with high grade sensors if this setup was not used (a similar setup has been developed by Snape et al. (2009)).

The path that the robot should follow was represented by a series of virtual checkpoints. Finally, a control loop computed the desired speed and steer, based on the robot's position, in order to pass through the intended checkpoints and the computed actuators commands are sent through a wireless channel to the robot, using a Xbee®Radio Modem.

The resulting trajectories of a test (Figure A.3) can be clustered according to some criteria, as common initial and final points, for example, as illustrated by Figure A.4. Each group of trajectories can then be used as the training dataset for the supervised learning of the hyperparameters of a GP, which results in a particular mean and covariance function, as

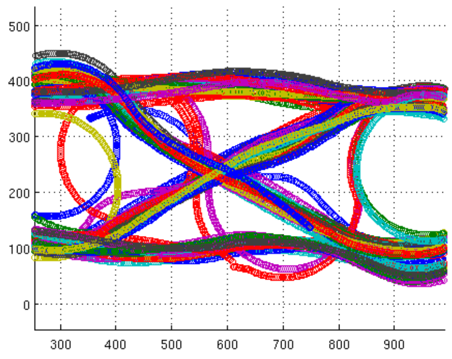


Figure A.3: Resulting trajectories of an experiment.

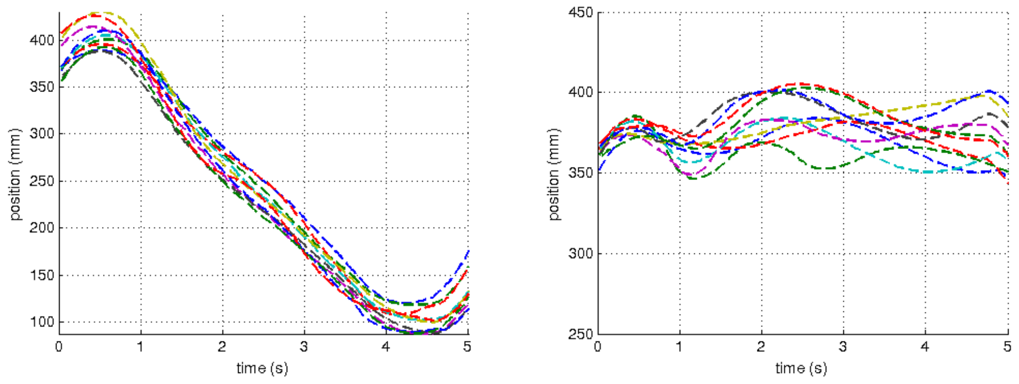


Figure A.4: Two clusters of recorded trajectories used for the GP training, grouped according to common start and finish regions.

illustrated in Figure A.5.

After this phase, several GPs would represent each of the trajectories clusters. For the prediction phase, the partial observations of the motion of an object are fed to each one of the available GPs models, resulting in predictions of the mean and covariance of future states. According to the matching degree between the observations and the outputs of each GPs models, more confidence will be given to a certain GP model (Figure A.6). In this way, the GP model that best represents the acquired observations will be used for the prediction of the path likely to be traveled by the observed person, together with its uncertainty, represented by the covariance function. For more detail on the modeling of typical using GPs, the reader is referred to Tay and Laugier (2007), Ellis et al. (2009), Fulgenzi et al. (2010).

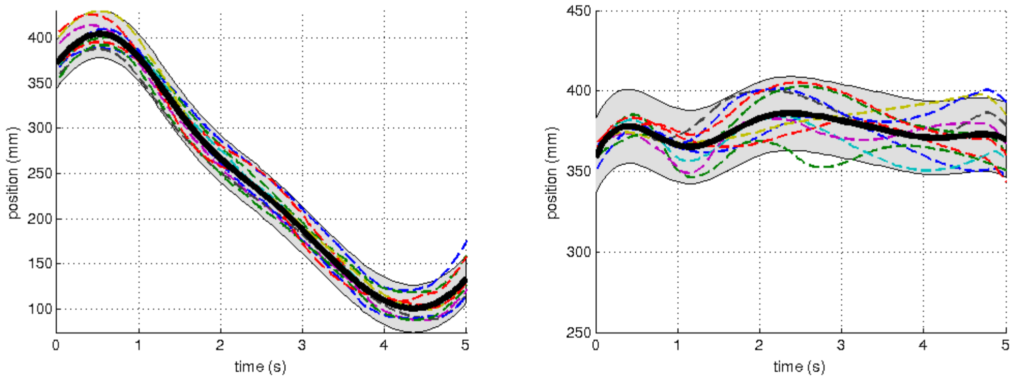


Figure A.5: The resulting GP for each of the clusters. The black solid line represents the mean function of the GP, which can be seen as the resulting typical path, while the gray area is the GP's covariance function, drawn with 95% of confidence.

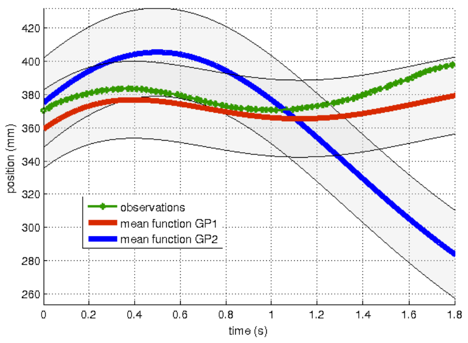


Figure A.6: Given a trajectory observation, represented in dark green, two GP models are evaluated to find the best match. In this case the GP number 1 is the winner and can be used for the prediction of the future path likely to be traveled by the observed object.

Appendix B

Comparison of Leader Evaluation using AdaBoost and Artificial Neural Networks

In this thesis, the evaluation of the performance of the AdaBoost classifier was subject to the quality of the ground truth used. As this reference was labeled using inputs from persons, it is feasible to assume that, in some tests, the algorithms could perform better classifications than the ones performed by humans. However, these would produce bad results, due to the discrepancies between bad ground truth and good classification results.

Therefore, for a better assessment of the performance of the developed classifier, it was compared with another machine learning algorithm, the Artificial Neural Network (ANN).

B.1 Comparison of Leader Classification Results

Three ANNs were trained, using the data sets that included the *od* and *nm* situations. The same number of samples and groups of features of the AdaBoost algorithms created in chapter 5 were used here. The composition of the dataset used, regarding the leader situations (*complete classifier*) and the balance of bad leaders samples over the total samples are shown in Table B.1.

Table B.1: Train dataset composition, with the number of datasets of each class used, the number of samples and the proportion of bad leader labels.

	gd	as	fr	st	nm	od	samples	bad leader label %
train	7	4	3	6	4	7	8504	34.11
validation	0	1	1	1	1	1	1692	47.16
test	2	2	1	2	2	2	2715	39.30

The ANN used is a feedforward, pattern recognition network. Its topology has a number of input neurons that match the feature set used: one hidden layer with 12 neurons and 2 output neurons that represent the two classes used in this work, good or bad leader. The activation function of the neurons is the sigmoid function. All the trained ANN stopped due to a condition of 100 iterations without improvement in the error, using the validation dataset. For information about the features used in each training, refer to section 5.1.1.

B.1.1 Performance comparison using 21 Features

The first ANN trained used the 21 original features extracted from laser measurements. The performance of the training is shown in Figure B.1.

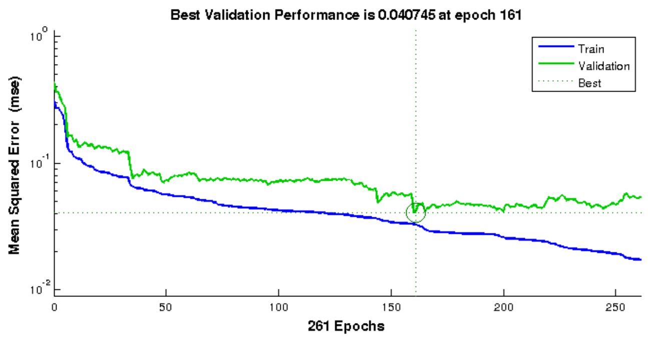


Figure B.1: Performance of an ANN trained with 21 features.

The three types of classification error, false good, false bad and false total are shown in Figure B.2, compared with the errors of the AdaBoost classifier trained with the same dataset.

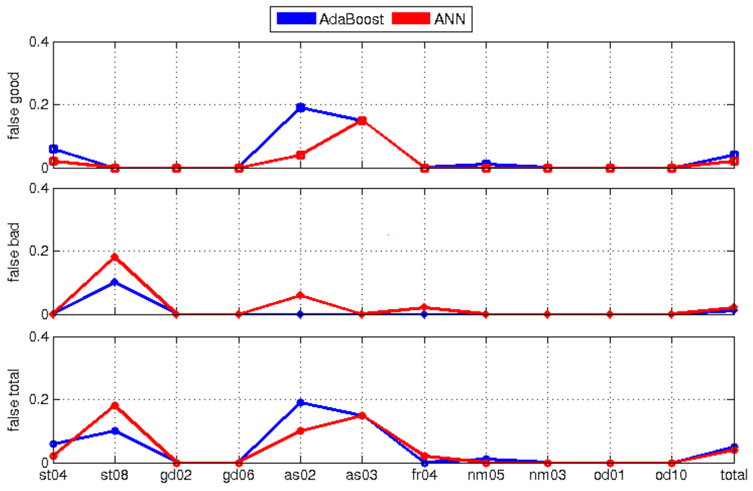


Figure B.2: Comparison between AdaBoost and ANN classification error, using 21 features.

This comparison show very similar results, with the larger differences in two bad leader situations, the leader stopped (st08) and leader moved aside (as02).

B.1.2 Performance comparison using 13 Features

The second train used a new features set, reduced to only 13 features, where the features removed are all the derivative group and the standard deviation of the relative velocity y. The performance of the ANN training using 13 features is shown in Figure B.3.

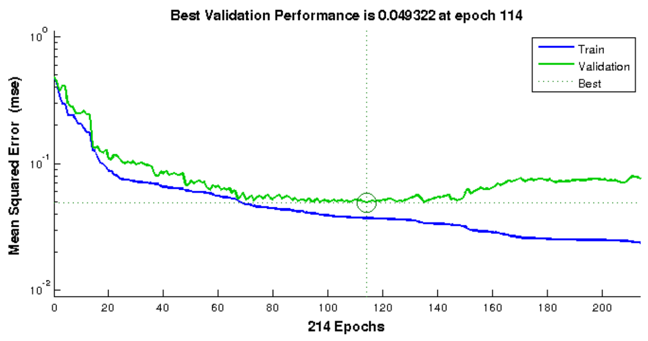


Figure B.3: Performance of an ANN trained with 13 features.

The comparison in the classification error between the resulting ANN and the equivalent AdaBoost classifier is shown in Figure B.4.

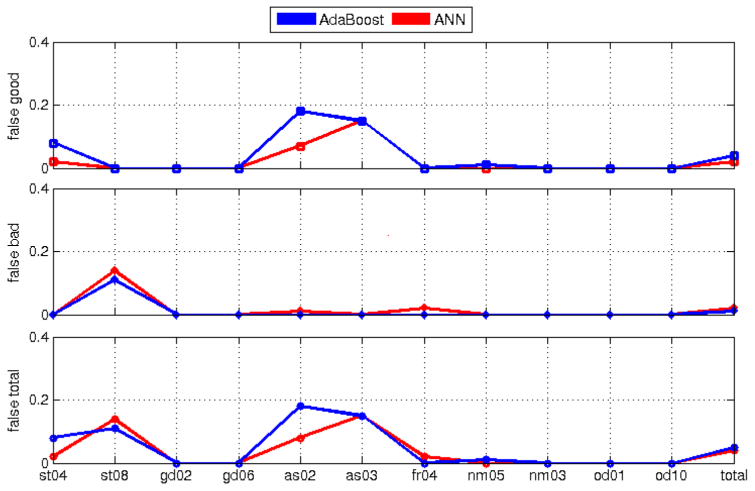


Figure B.4: Comparison between AdaBoost and ANN classification error, using 13 features.

In this experiment, the differences on the performance between the two classifiers is even smaller, with larger differences in the bad leader situation st04 and, again, in a situation where the leader moved aside (as02).

B.1.3 Performance comparison using 9 Features

In the final training, the features set was removed once again, removing the standard deviation group, except for the standard deviation of the distance and of the target velocity. The resulting training performance is shown in Figure B.5, while the comparison of the classification error is shown in Figure B.6.

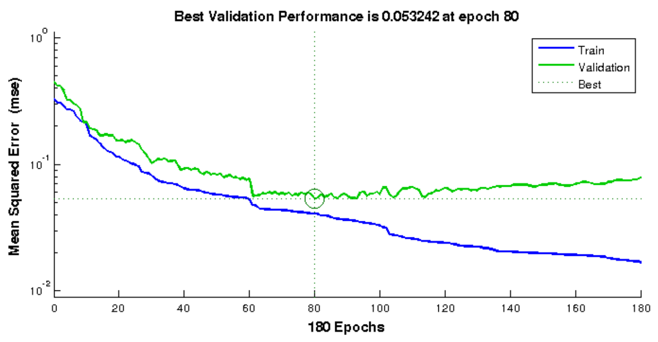


Figure B.5: Performance of an ANN trained with 9 features.

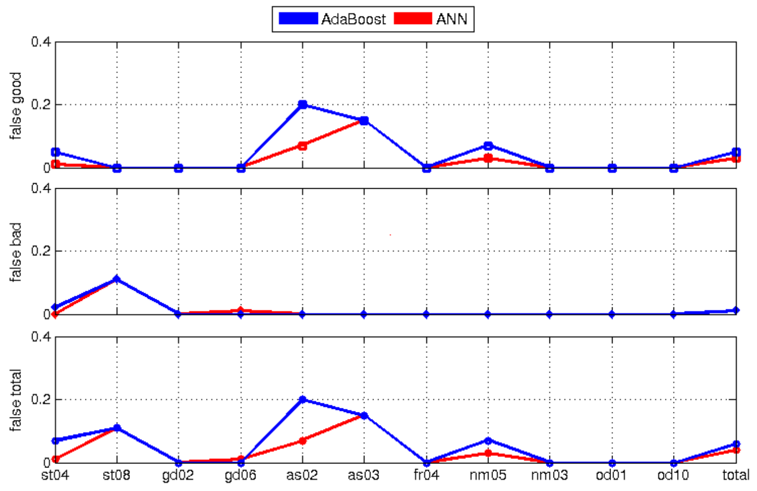


Figure B.6: Comparison between AdaBoost and ANN classification error, using 9 features.

Once again, there was no significant difference in the classification errors comparing the ANN and the AdaBoost classifiers. Only a residual error was present on one of the leader stopped situation (st04), and a minimal increase in the error of the not moving situation (nm05). Still, the largest difference was, once more, at the same leader moved aside situation (as02).

B.2 Analysis

Results showed that both the classifiers had virtually the same performance, with minimal differences, that were evident mostly in the leader moved aside situation (as02). In overall, the ANN had better results, although the total error difference was so small (about 1%), that it cannot be said that the ANN was superior to the AdaBoost classifier.

These results, where two different learning methods had virtually the same performance, show that limitations in improving the performance of the classification algorithms are likely to be in the types and number of features used, and similar errors are probably a result of suboptimal ground truth datasets.