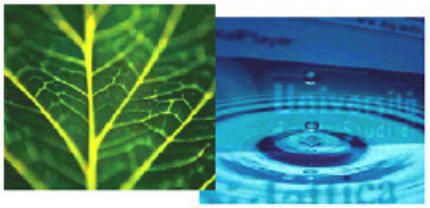
PhD Dissertation



International Doctorate School in Information and Communication Technologies

DISI - University of Trento

Socially aware motion planning of assistive robots in crowded environments

Alessio Colombo

Advisor:

Prof. Luigi Palopoli

Università degli Studi di Trento

April 2015

Abstract

People with impaired physical or mental ability often find it challenging to negotiate crowded or unfamiliar environments, leading to a vicious cycle of deteriorating mobility and sociability. In particular, crowded environments pose a challenge to the comfort and safety of those people. To address this issue we present a novel two-level motion planning framework to be embedded efficiently in portable devices.

At the top level, the long term planner *deals with crowded areas, permanent or temporary anomalies in the environment (e.g., road blocks, wet floors), and hard and soft constraints (e.g., "keep a toilet within reach of 10 meters during the journey", "always avoid stairs"). A priority tailored on the user's needs can also be assigned to the constraints.*

At the bottom level, the short term planner *anticipates undesirable circumstances in real time, by verifying simulation traces of local crowd dynamics against temporal logical formulae. The model takes into account the objectives of the user, preexisting knowledge of the environment and real time sensor data. The algorithm is thus able to suggest a course of action to achieve the user's changing goals, while minimising the probability of problems for the user and other people in the environment.*

An accurate model of human behaviour is crucial when planning motion of a robotic platform in human environments. The Social Force Model (SFM) is such a model, having parameters that control both deterministic and stochastic elements. The short term planner *embeds the SFM in a control loop that determines higher level objectives and reacts to environmental changes. Low level predictive modelling is provided by the SFM fed by sensors; high level logic is provided by statistical model checking. To parametrise and improve the* short term planner*, we have conducted experiments to consider typical human interactions in crowded environments. We have identified a number of behavioural patterns which may be explicitly incorporated in the SFM to enhance its predictive power.*

To validate our hierarchical motion planner we have run simulations and experiments with elderly people within the context of the DALi European project. The performance of our implementation demonstrates that our technology can be successfully embedded in a portable device or robot.

Keywords[Motion planning, human models, assistive robotics, model checking]

Acknowledgements

First of all, I would like to express my gratitude to my advisor, Prof. Dr. Luigi Palopoli, for his patience, guidance and encouragement throughout the whole Ph.D. experience. His fundamental support helped me to grow as a research scientist.

My sincere thanks to Dr. Daniele Fontanelli and Dr. Sean Sedwards for their support and the several interesting and fruitful discussions we had; and to Prof. Dr. Axel Legay for inviting me as a visiting student at INRIA, Rennes.

Many thanks also to all the people I met within the DALi European project, it was a really great experience.

Finally, I would like to thank my girlfriend Belén for her priceless support, I could not have done it without you; my family, to whom this dissertation is dedicated, for their encouragement and patience; and my friends for their endless support and colleagues who I have met in these years.

Alessio Colombo

The research leading to these results has received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° ICT-2011-288917 "DALi - Devices for Assisted Living".

Contents

List of Tables

List of Figures

Chapter 1

Introduction

With unimpaired ability, pedestrians are able to find their way across complex and crowded areas without major problems. With reduced abilities this apparently simple task easily becomes a challenging one [1]. For instance, a person with reduced mobility needs to minimise the travelled distance, while a person with cognitive problems should avoid situations that challenge her sense of direction and confuse her perception of the environment. The difficulty in identifying the most correct path and in making proper reactions to unexpected contingencies may gradually reduce the confidence of the impaired person in using public spaces [2]. The afflicted are most often older adults and the problem worsens quickly if no adequate countermeasure is taken [3, 4]. She can be deprived of essential social relations with a negative impact on her physical condition (reduced exercise), on her psychological wellbeing (reduced social contact) and even on the quality of her nutrition if she reduces the frequency of her visits to supermarkets [5, 6].

A growing body of research [7] suggests that physical activity can have widespread, beneficial effects for older adults and ultimately even decelerate the process of ageing. The application of assistive robotic technologies [8] can be of significant help to amplify these effects. In particular, the type of support that a robotic system with cognitive abilities can offer in the navigation of a complex environment depends on the type of robot assistant and can be adapted to the user's needs. If the robot is simply a guiding vehicle, like a tour-guiding vehicle [9], guidance just consists in following the path and ensuring that the user is trailing behind. If the robot is a robotic walker, it can guide the user by mechanically turning its wheels and acting on the wheel brakes [10] or by providing visual, audio or tactile signals [11, 12]. If the robot is a robotised wheelchair, it can be compared to a robotic vehicle driving in crowded spaces [13].

In the last few years, robotic platforms able to perform autonomous tasks are increasing in complexity and number. Nowadays, the market offers all kinds of devices that can, in some terms, be considered as smart and autonomous: from vacuum cleaners capable

of cleaning the house while the householder is absent, to industrial robots able to move goods in warehouses without human intervention. This trend is visible also in the assistive robotic field and the ability of a robotic platform to navigate crowded spaces in a socially-acceptable manner is becoming increasingly important [14].

Indeed, robots need to understand what is happening around them, so they can choose a course of actions that respects the social rules that govern the environment in which they are moving.

Social rules [15, 16] are non-written rules that manage the cohabitation of people in their environments. If a person is walking and a group of talking bystanders is obstructing her way, she should not pass through them if there are other possible ways around.

If we imagine to substitute this person with a robot, we obtain a *human-robot interaction* problem. A robot respecting the social rules will recognise the bystanders, will realise that they are chatting and will identify them as a single cluster that should not be disturbed. Thus, it will modify its planned path to avoid the people without getting too close. However, a robot ignoring these social rules will see the people as two independent obstacles. Hence, it may decide to pass in between them with the consequence of invading their personal space and interrupting the conversation.

Important guidelines for implementing social rules in a robot can be extracted from literature on human social interaction. One example is the work by Goffman [17] that describes the interplay between two people as "focused" or "unfocused" interaction. A focused interaction occurs when one person deliberately searches for the other person's attention and she responds. An unfocused interaction, instead, takes place when one person finds out information about the other without requiring her attention (e.g., by looking at her behaviour). One interesting theory is proxemic theory by Hall [18] (more details in Section 6.3) that describes the use of personal space and territory, and the relationship between human psychology and non-verbal or iconic communication.

The goal of this dissertation is the design and development of a so called "cognitive engine" for assistive robotic platforms that obeys social rules. Such engine is a motion planning framework that safely steers and controls robots in semi-structured environments populated by human beings while taking into consideration the user's goals and preferences. We introduce it in Section 1.1.

1.1 Cognitive Engine

The cognitive engine is, in our terminology, the set of decision algorithms that suggests the user a trajectory from a source position to a destination position accounting for the state of the environment and the user's personal preferences. The cognitive engine

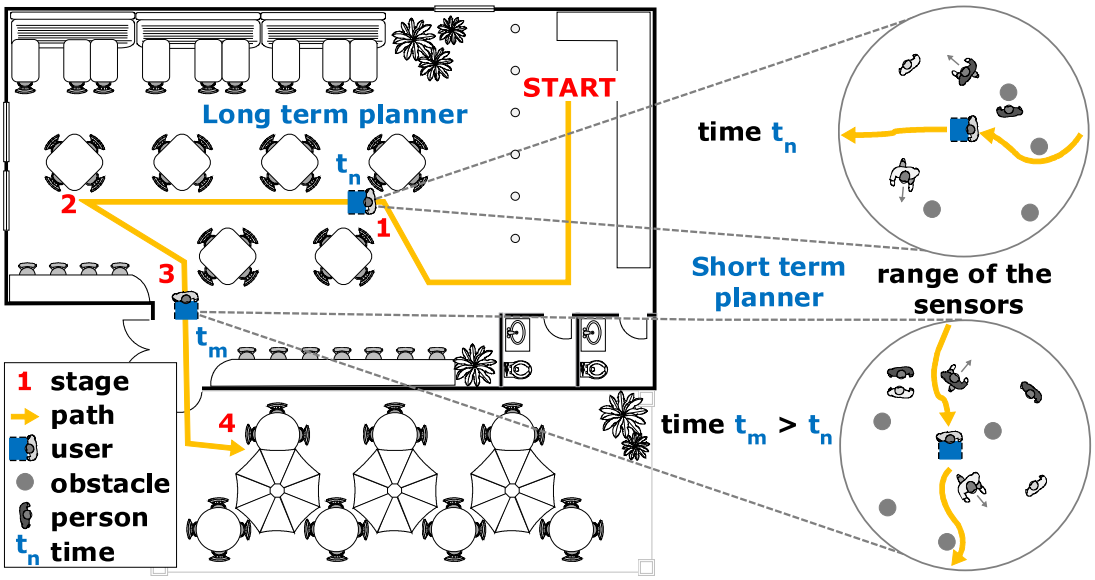


Figure 1.1: High level representation of the motion planner. Initially a coarse long term plan is built on the map by the *long term planner*. Then, the *short term planner* is in charge of modifying the short term path in order to avoid obstacles in the surroundings of the platform.

periodically samples the surrounding environment using the sensing capabilities provided by the robotic platform and by the sensors deployed in the environment.

We propose a two-level hierarchical approach, visible in Figure 1.1. A long term plan is first constructed by the *long term planner* considering both the floor plan and the information gathered from sensors deployed in the environment. During the journey, the *short term planner* adjusts this path and produces a short term plan that avoids dynamic obstacles detected by the sensors. User preferences and goals are taken into consideration during the whole process.

Further details are reported in Chapter 2.

1.2 The *c-Walker*

The *c-Walker*, visible in Figure 1.2, is a kinematically passive haptic device based on a standard mobile robotic platform. Provides physical, cognitive, and emotional support to older adults in public environments such as shopping centres and airports. The user remains always in charge of final decisions and the system does not override her intent. Instead, it operates supportively, offering appropriate recommendations.

The key features of the *c-Walker* are: 1. the ability to construct the path across the space that best supports the user's preferences and goals; 2. the detection and recognition of anomalies along the way; 3. the observation of humans in the area of interest of

Figure 1.2: The *c-Walker* is a novel cognitive walking assistant developed within the DALi project. It safely guides the user through complex indoor environments.

the device and the prediction of their future position; 4. the possibility to locally reshape the path to avoid potential risks and collisions with other humans; 5. a rich set of interfaces that the system can use to recommend a path to the user, which include passive interfaces (visual, acoustic, and haptic) and active interfaces (electromechanical brakes, and motorised turning wheels). These complex functionalities are implemented relying in part on the embedded sensing and intelligence, in part on the ambient intelligence.

The *c-Walker* prototype has been fully integrated and tested in both synthetic and real environments.

This walker has been developed within the DALi[1] (Devices for Assisted Living) European project that targeted a user group consisting of older adults with emerging non-severe cognitive disabilities. Final users have been involved in all phases of the development of the system and the project was acutely sensitive to their needs.

The DALi project ended in October 2014. The ACANTO[2] (A CyberphysicAl social NeTwOrk using robot friends) H2020 European project started in February 2015, and is the follow-up of the DALi project. It will extend the assistive paradigm moving from a

[1] `http://www.ict-dali.eu/`
[2] `http://www.ict-acanto.eu/`

single user to a network of users, connected via a cyber-physical social network.

1.3 Scientific Contributions

This work proposes a hierarchical motion planning algorithm for assistive robotic platforms. An overview of this approach has been presented in [19]. The algorithm is able to cope efficiently with dynamic and partially unknown environments, while remaining reactive to potentially uncooperative behaviour of the user.

Three main contributions can be devised:

- Long term planner: an efficient long term motion planner [20] based on the Dijkstra shortest path algorithm [21]. The construction of the path can be customised with soft and hard constraints with priority, and is reactive to temporal anomalies and crowded areas represented as heat maps.
- Short term planner: a probabilistic and efficient short term motion planning algorithm for highly dynamical crowded environments [22]. User's preferences and goals are defined via Bounded Linear Temporal Logic (BLTL) formulae, and enforced using statistical model checking.
- Identification of human motion models: evaluation of a well-known human motion model through experiments with people in a simulated supermarket [23]. Analysis of its limitations and proposal of an extended model.

1.4 Outline of the Dissertation

The dissertation is divided in chapters, each of which covers different aspects of the work. The reader will be guided through a top-down discovery of the motion planner starting from a overview in Chapter 2. The discussion on related work has been condensed in Chapter 3 in order to give a complete overview in one shot, without the need of going back and forth into the text. We will then describe in details the *long term planner* in Chapter 4, the *short term planner* in Chapter 5 and the identification of human motion models in Chapter 6. The experimental evaluation is presented in Chapter 7. Finally, conclusions and future objectives can be found in Chapter 8.

Chapter 2

Overview of the Approach

In this chapter we will give an overview of the proposed cognitive engine. Figure 2.1 shows the logical blocks that compose the motion planning algorithm.

From a top-down perspective, the first type of assistance is offered before starting the navigation activity and consists of the production of a plan that takes into account long term objectives. This is accomplished by the *long term planner* (Section 2.1), which takes into account the topology of the space, the user's preferences and the possible presence of obstacles or problems along the way, which are revealed by environmental sensors. While the user is moving, she could encounter contingent problems that cannot be anticipated (e.g., a small group of people obstructing the path). In this case, her robot assistant could react by planning a minimal deviation from the path that preserves her safety and wellbeing. In our terminology, this component is called the *short term planner* (see Figure 2.1) and is introduced in Section 2.2. Finally, the guidance system of the robot assistant can guide the user along the planned path.

In our vision, the user is not required to strictly follow the path, and potential conflicts are detected and resolved automatically without any loss of comfort/safety for the user. During the journey, the motion planner refines its strategy in order to be compliant with the decisions of the user, while reducing the number of conflicts. Every user is associated with a profile that describes specific known interests and dislikes that help the motion planner in generating the path. This profile can be explicitly programmed (e.g., by answering some high level questions) by the user or the caregiver.

2.1 Long Term Planner

Consider a person willing to execute a set of activities in a public space. The problem the user faces is to identify the best way to reach her points of interest. From an assistive robot point of view, this decision could potentially be taken using any state-of-the-art algorithms

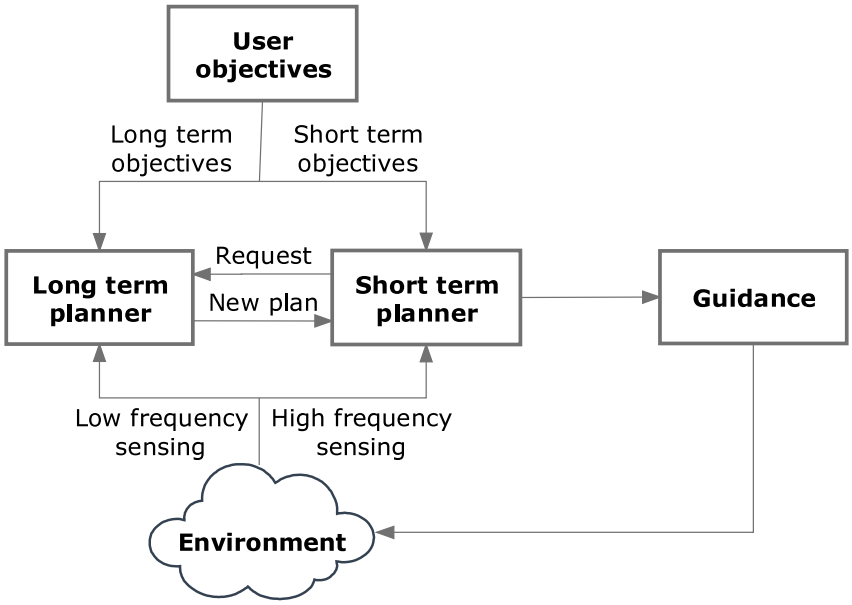


Figure 2.1: Diagrammatic overview of the motion planning framework. The whole process can be divided into three main elements: the *long term planner* that considers the long term objectives, the *short term planner* that optimises the long term plan taking into account the short term objectives and constraints, and the guidance that drives the robot towards the goal.

for motion planning, able to identify the path with minimum length (or requiring minimum time) given the a priori knowledge of the map. A first problem is that while the position of most fixed objects (e.g., buildings, rooms, and points of interest) is known a priori, the algorithm must take account of the possibility of changes, such as temporary obstructions. Standard motion planning algorithms can easily be adapted to consider an up-to-date picture of the state of the environment (e.g., presence of obstructions or over-crowded spaces) as it arrives from environmental sensors. However, a simple modification to a standard planner could be insufficient. First, the detected anomaly could be a temporary one. So, the likelihood of having to deal with the problem during the navigation depends on the time needed to reach the place where the anomaly is located, which in turn depends on the chosen path. What is more, the user (who is typically an older adult) will likely have specific additional requirements. For instance, the user could need a frequent access to the toilet, and if the optimum path offers no easy access to the toilet on the way, it could easily generate discomfort. Whereas, the user could be hyper-vigilant and overly concerned with her personal security. In this case, she might appreciate always being within reach of a policeman or of other staff member that she perceives as a reassuring presence.

In a few words, what we need is an algorithm for motion planning in public spaces that accounts for 1. the topological and metric information about the space, 2. time-varying

environmental information about the space, such as the availability of services (is the shop that the user wants to visit actually open?), the presence of occlusions and overcrowded areas, etc., 3. preferences of the user (e.g., the need to be in easy reach of assistance, toilets, etc.).

The presence of these specific requirements makes the planning algorithms offered by commonplace navigators (such as Google Maps) infeasible. A different approach that carefully considers the strong psychological aspects involved in the selection of a route is needed.

The *long term planner* periodically collects information from environment sensors and from other *c-Walkers* deployed on the ground. This information consists of anomalies, heat maps (i.e., crowded areas), status of points of interest (e.g., queue length for shops) and is merged with prior information on the place (the map). The user checks in a request with a sequence of places to visit, and a profile condensing her preferences is attached to it. The *long term planner* receives the request and produces the optimal path operating as follows: 1. the map is broken down into a grid of discrete cells, 2. a graph is derived from the grid, where each node represents the centre of a cell and each arc is a path joining two cells, 3. the graph is changed by adding relevant semantic information (e.g., associating points of interest with some of the cells), 4. each arc is associated with a cost that accounts for the distance to travel and for the occupancy of the area (people density translates into a longer time to travel), 5. additional manipulations are made to exclude (or to increase the travelling cost of) paths that violate the user preferences, 6. the optimal path is found using the modified Dijkstra algorithm. Figure 2.2 graphically describes this process.

The long term plan is propagated to the system for its execution along with additional constraints related to the user profile that could not be enforced at the level of the *long term planner* (e.g., if the user requires not to be in close contact with any other person, this cannot be enforce before the situation in her proximity is known to the system). Then the execution of the desired motion begins and the *short term planner* takes over.

Details of the *long term planner* are extensively presented in Chapter 4.

2.2 Short Term Planner

Periodically, the *short term planner* acquires the state of the system, comprising the position of static objects and the position and velocity of the user and other people in the environment. The current state is provided by the sensing system that is subject to measurement error and noise. Given the current state, the algorithm hypothesises alternative courses of action using a comprehensive human motion model able to capture

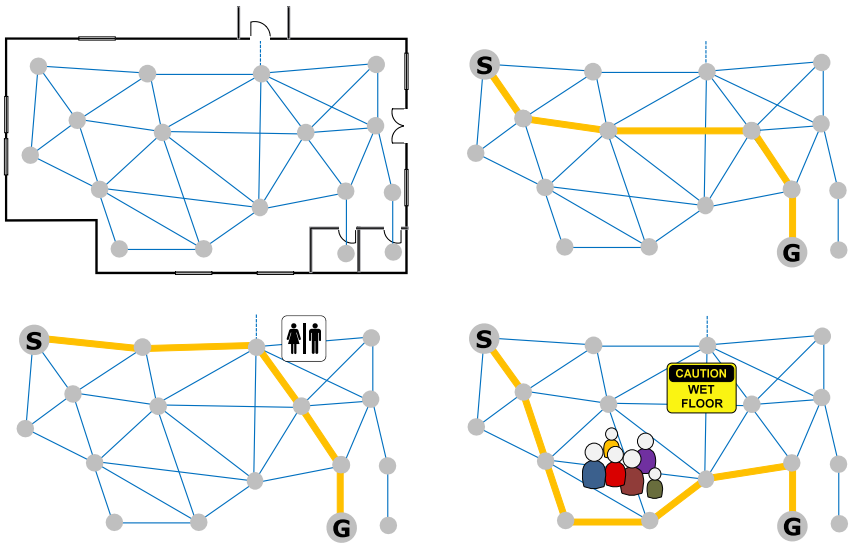


Figure 2.2: Overview of the *long term planner*. It first takes in input the map of the environment and computes the graph that maps the empty space (top left). The optimal yellow-coloured trajectory connecting start **S** to goal **G** can have different shapes according to the objectives: finding the Euclidean shortest path (top right), being closer to the toilets (bottom left), or being reactive to unpredictable anomalies, such as crowded areas or wet floors, (bottom right).

the dynamism of crowd.

The model includes stochasticity to take into account the natural unpredictability of human behaviour, which is exploited by the algorithm to generate multiple independent simulation traces.

Changing the number of simulations can significantly affect performance. The number of simulations can be used as a tuning knob for balancing execution time and statistical confidence. For example, the latter could be temporarily boosted, thus increasing the number of simulations and the time required to compute them, when the system is off-load.

Each of the simulation traces is then formally verified (*model-checked*) against properties that express goals and constraints required for the user trajectory (i.e., where the user wants to go, minimum distance from obstacles or people). This leads to a statistical distribution of potentially successful trajectories. The algorithm uses this distribution to choose an immediate action that maximises the probability of achieving the objectives of the user while minimising the probability of accidents.

For validating the approach we used the Social Force Model described in Section 6.2, well known in the literature for simulating crowd behaviour. However, as anticipated in Section 2.3, experiments revealed that this model does not capture all features of human motion of our interest, pushing for the need of an improved crowd model.

More details can be found in Chapter 5.

2.3 Identification of Human Motion

The ability to understand human behaviour and social interactions is of primary importance. The goal is to identify an accurate two dimensional model for predicting human motion. Our starting point was the Social Force Model (Section 6.2), generally used for simulating people in normal or panic situations. This model is very flexible and broadly used in the literature. However, in some cases it does not produce realistic motions, especially when the trajectories of pedestrians are interrupted by sudden short term pauses and deviations.

We thus started by observing people in a real scenario, running controlled experiments in a simulated supermarket, where several people concurrently had to follow their shopping list. The videos of the experiments were then manually labeled according to the observed behaviours, and we were able to match the observation with proxemic theory. Proxemics (Section 6.3) is an important theory borrowed from the literature on human social interaction, and relates human psychology with spatial behaviour.

These results helped us to identify a number of behavioural patterns that can be incorporated in an extended version of the Social Force Model.

Further details are presented in Chapter 6.

Chapter 3

Related work

The purpose of this chapter is to relate the work presented in this dissertation with the current state of the art. We start with an overview of similar projects for assistive robotics in Section 3.1, then a survey of the literature for the different components that compose the motion planner.

Motion planning in crowded environment is a relevant research problem in robotics that has received a constant attention throughout the past two decades [24, 25, 26]. The approach that we advocate is based on a hierarchical decomposition of the problem between short term and long term planning. Different authors in the literature propose a strategy of this kind [27, 28], but the solution at each of the two levels of the hierarchy differ significantly based on the requirements that each author considers.

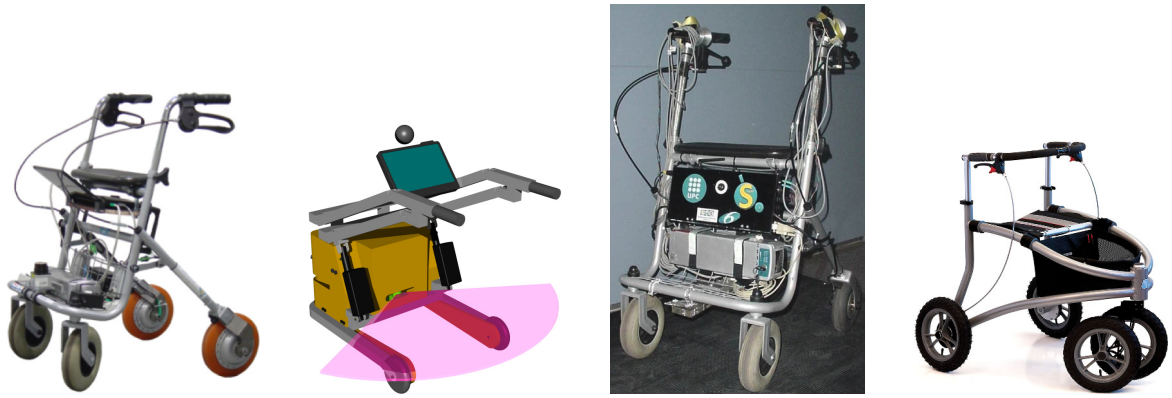
The *long term planner* is discussed in Section 3.2, the *short term planner* in Section 3.3 and the identification of human motion models in Section 3.4.

3.1 Assistive Robotics

Several projects have targeted assistive mobility with the development of a robotic walking assistant and some of them share the basis ideas with the DALi project.

The ASSAM project [29] aims to develop a modular navigation assistants for users with different level of disabilities. They target different mobility platforms, such as tricycle, wheelchair and walker. The latter is called *eWalker* (Figure 3.1(a)) and is an electronic walker providing navigation support by means of an active platform that compensates declining walking capabilities, as well as cognitive disabilities. However, the behaviour of people in the surroundings is not considered.

The DOMEO project [30] developed the *robuWalker* visible in Figure 3.1(b), a walking assistant that monitors the health state of the user. The walker is endowed with devices for tele-presence and intuitive interaction both with tactile and voice recognition. From



(a) ASSAM [29] (b) DOMEO [30] (c) I-DONT-FALL [31] (d) Veloped [32]

Figure 3.1: Assistive walkers developed in other projects.

the proposers' point of view, the older adults should be helped in staying longer and safer at home. This is clearly in contrast with the DALi project that actually encourages the elderly to socialize and move in large public spaces.

The *i-Walker*, showed in Figure 3.1(c), is the assistive device developed within the I-DONT-FALL project [31]. The main goal of the project is to improve quality of life of the elderly through the efficient prevention and detection of falls. The device is part of an ecosystem of healthcare services and is used for physical training and data logging.

The Veloped [32] (Figure 3.1(d)) is a commercial walking support for outdoor exploration with an appealing look. Different versions are available according to the target outdoor environment. Some of them are passive and exploit only the mechanical design for easing the walk. Others, instead, are equipped with motorised wheels for providing aid on critical terrains.

3.2 Long Term Planner

The goal of a *long term planner* is to find a collision-free path from a starting point to some desired destination, given the topological and metric constraints derived from the map. When the map is not entirely known in advance (e.g., due to uncontrollable changes in the environment), a convenient choice can be the adoption of sampling-based algorithms. In this class the Probabilistic RoadMap (PRM) algorithm by Kavraki et al. [33] and the Rapidly Exploring Random Trees (RRT) [34] have gained an undisputed reputation and visibility in the past few years. The idea of this class of algorithm is to generate feasible points by sampling randomly the neighbourhood of known points and connecting them into a data structure (e.g., a tree). When the destination is finally reached an optimal

path can be found by exploring the data structure. The more time that is given to the computation, the more points that can be added and the higher the probability becomes of finding an optimal solution. Such algorithms have recently been revisited by Karaman and Frazzoli [35]. The revised versions, PRM* and RRT*, are probabilistically complete, meaning that if the algorithm is given enough time to explore the space, it eventually identifies the optimal solution with probability 1. An important point of these algorithms is that while the data structure is being created it is possible to enforce a hierarchy of hard and soft constraints penalising (or ruling out) points that would violate them. This is an appealing feature for us because our problem is characterised by a set of constraints. However, the construction on the fly of the path and on the map is not required in our case. Our intended operational scenario is a public space (e.g., a mall or a museum) for which a large amount of a priori information is usually available.

Another family of algorithms are based on the definition of potential fields [36, 37] around obstacles and points of interest that can attract or repel the robot. Such approaches are known to be effective for obstacle avoidance, but they are often plagued by local minima (which sometimes delay or deadlock the progress). While encoding all the user's planning requirements, constraints and preferences with a potential function is generally a difficult problem, our approach makes use of the notion of gradients to encode user-defined desirable and undesirable zones. The full details are given in Section 4.2.3

The *long term planner* proposed in this work falls in the class of graph based techniques. In essence, the idea is to decompose the environment into a grid and then generate a graph by associating nodes to elements of the grid and by then connecting with arcs the nodes associated to adjacent cells. Minimum time paths on the graph can be found using the well-known Dijkstra algorithm [21] or its extension A* [38]. The use of a constant size grid is generally discouraged due to the explosion of the configuration space size, hence several more efficient ways to construct the graph have been proposed. Possible approaches include Voronoi diagrams [39] and PRM [33]. We follow Chen et al. [40] and construct a graph using quad tree decomposition of the space, exploring it with an extended version of the Dijkstra algorithm. The generation of our quad tree is specific for its application to structured indoor spaces, with large rooms connected through corridors, doors and passageways and where each room may contain such things as counters, shelves and exhibition paraphernalia that compromise its regularity.

The most important feature of our algorithm is its ability to deal with temporary anomalies (e.g., obstructions or large groups of people hindering the user's motion across some of the areas). In particular, anomalies (i.e., temporary graph obstructions) require the generation of time-dependent paths, a problem that is known to be challenging and is the focus of independent research [41, 42, 43, 44]. None of these papers qualifies

itself as a clear winner. In our particular case, we adopt a conservative assumption, described in Section 4.2.6, that allows us to solve a simplified problem very efficiently. Our requirement analysis reveals that senior users of a navigation tool are very annoyed by a long wait in front of a screen. Therefore, efficiency and quick deliveries of decisions are more important than producing "optimal" decisions (as long as the decisions do not violate any hard constraints and they respect soft constraints to a reasonable extent).

The global constraints (not to be confused with kinodynamic constraints, not considered here) are used for customising the behaviour of the planner and for introducing the notion of "comfort" for the user. Constraints are prioritised and some of them can be violated if their compliance prevents the system from finding any path.

They embed priority and the possibility for one or more constraints to being ignored if a path cannot be found otherwise (namely, conflicting constraints). This is called "planning with partial satisfaction", and is studied in the literature under the notion of *preference-based planning*. In [45] they focus on computation of relaxed plan-based heuristics that guide the planner towards good solutions satisfying the given preferences. In [46] they introduce a method for quantifying the satisfaction of linear temporal logic (LTL) formulae, and propose a planning framework using this method to synthesise robot trajectories with the optimal satisfaction value. However, they do not consider constraints where the cost or priority changes over time. Tumova et al. [47] present an automatic generator for control strategies for a robotic vehicle where constraints are expressed with LTL formulae. The novelty is the possibility of violating a constraint, according to its priority, in order to complete the task (e.g., a road lane should not be crossed, but this is allowed during car parking).

The concept of "comfort" has already appeared in the literature but generally with different meanings: 1) comfort of the user when navigating using a robotic platform [48, 49] and 2) comfort of the humans in the area surrounding an autonomous robot [50]. Our notion of comfort belongs to the first class and it is deeply rooted in the requirement analysis and in the validation activities with senior users that we have been conducting in the context of the DALi and of the ACANTO projects. Our findings are that the user needs to specify zones that she likes or dislikes. As an example, more often than not she would prefer to bypass crowded areas or to always have a toilet or a resting place within easy reach, even if this entails choosing a slightly longer path.

3.3 Short Term Planner

The *short term planner* algorithm is related to sampling methods (e.g., [51, 33, 34]) and to recent methods using temporal logic (e.g., [52, 53, 54]). It is also related to methods

that predict behaviour based on models parametrised with data from sensors (e.g., [55]).

In common with existing sampling methods, our algorithm uses randomisation to cover an intractably large configuration space. In contrast to many existing uses of sampling, however, we do not assume a fixed environment. In our application the environment contains both fixed and dynamic elements, such that a single optimal path cannot be defined a priori. Hence, the problem we solve by sampling is not one of creating an optimal long term plan, but one of finding an optimal short term plan given a changing environment.

Model checking is an automatic process to verify that a system satisfies a property specified in temporal logic. In the present context, temporal logic can express complex dynamical properties such as "the user will visit all the desired locations in a specified sequence, within the specified time" and "the user will never get too close to any other pedestrian". If the notion of an optimal path can be so defined, the principles of model checking can be used to directly synthesise a 'correct' motion planner or to prove that an existing motion planner is correct [52, 53, 54]. The use of probabilistic model checking in combination with the theory of stochastic hybrid automata [56] is particularly appealing for control and robotic applications where a non-zero probability of failing the mission can be tolerated. For example, in [57, 58] for air traffic control or in [59, 60] for industrial robotics. Combining model checking with sampling, algorithms can be constructed which provably converge to optimal schedulers [54]. Standard model checking algorithms are computationally intensive, hence existing applications have used model checking offline. By using *statistical* model checking, we are able to perform online verification. We do not prove correctness, but we find a short term plan that maximises the probability of success.

3.4 Identification of Human Models

Basically two main approaches exist in the literature for modelling human motion. The first one proposes a single model that captures the features of the human motion. The second one relies on different dynamic models that are combined using a switching logic.

One advantage of the first approach is that it is easier from a computational point of view, while with the second one it is easier to highlight the different decision points that compose the human behaviour.

One of the most cited models in the literature falls in the first category and is the Social Force Model (SFM) by Helbing et al. [61], discussed in details in Section 6.2. It is a continuum model that borrows concepts from molecular dynamics. It assumes the human motion respects both the physical laws of motion and social conventions,

and models them as attractive (e.g., friends, points of interest), and repulsive forces (e.g., walls, strangers). Its strengths and weaknesses are well tested and understood [62, 63, 64]. However, the resulting behaviours strictly depend on the input parameters, which are not easy to estimate.

Lämmel et al. [65] presented an interesting comparison between some models for pedestrian dynamics and the real world. Specifically, they investigated three approaches (including the SFM) and their ability to reproduce collision-free movements in dynamic environments.

Kelly et al. [66] proposed a switching model that combines constant acceleration, constant velocity and constant position models with a Kalman filter. The switching logic is based on the statistical properties of the innovation sequences computed by the filter.

Moussaïd et al. [67, 68] focused on social groups and how the self-organisation mechanism of people affects crowd dynamics.

Burstedde et al. [69] proposed to track pedestrian dynamics using a two-dimensional cellular automaton.

Lau et al. [70] proposed a multi-model for group tracking and group size estimation. They have a set of hypotheses (e.g., split, merge and continuation) that they validate on the observed data. However, in their case the microscopic motion of a single person loses importance, in favour of the behaviour of the whole group.

A comprehensive survey on the available mathematical models for pedestrian motion has been presented by Schlake in her master thesis [71].

Chapter 4

Long Term Planner

This chapter goes into details of the *long term planner*. The algorithm and its validation are presented in the following sections, where both qualitative and quantitative results are given.

4.1 Preliminaries

The proposed *long term planner* has been developed bearing in mind a number of requirements. The key point is letting the user personalise her journey while keeping the planner reactive to changes in the environment. For this reason we have implemented three main features.

The first feature gives the user the possibility of adding hard (non-violable) and soft (violable) constraints, according to some customisable priority. It is possible to encode rules like "never get closer than 5 meters to any stair" or "try to keep within 10 meters of a toilet", or "always be within sight of a clerk or of a policeman". Should a soft constraint be in conflict with another one, the issue is resolved by violating the one with lower priority. A hard constraint, instead, cannot be violated.

The second feature reacts to anomalies detected in the environment by the sensing subsystem. An anomaly is a bounded zone in the environment that becomes inaccessible for a limited period of time (e.g., a wet floor or blocked passage). After this period expires, the anomaly is cleared and the zone is accessible again.

The last feature takes into account the crowded spots in the environment. They are represented as heat maps (an example is shown in Figure 4.6) where the apparent "heat"

Part of this chapter was published in
A. Colombo, D. Fontanelli, A. Legay, L. Palopoli and S. Sedwards, "Efficient Customisable Dynamic Motion Planning for Assistive Robots in Complex Human Environments", *Journal of Ambient Intelligence and Smart Environments, IOS Press*, September 2015, [20].

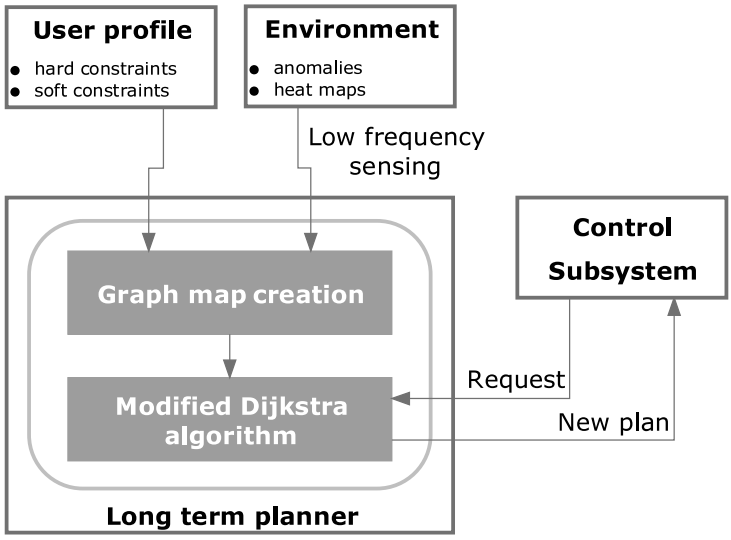


Figure 4.1: Diagrammatic overview of the *long term planner*. Informed by the heat maps and anomaly detectors, the *long term planner* constructs a long term plan according to the user's constraints. The plan is then transferred to the control subsystem.

represents the level of crowdedness. The planner interprets this level as a penalising factor that slows down the user. Some users could also have specific constraints related to avoiding crowded areas.

The work flow of the algorithm begins with the user specifying a list of target locations she wants to visit in the environment. The *long term planner* constructs a plausible path (a long term plan) according to the constraints in the user's profile and to the current conditions in the environment (known anomalies and current crowding represented by heat maps). This data is sampled periodically from remote sensors (e.g., surveillance cameras). If other robots are deployed in the environment, they can use their local sensing system to detect anomalies and share this information through a cloud infrastructure. For instance, if a walker detects a wet floor sign, this information is propagated to the other robots and accounted for in the generation of long term plans. Once the user accepts the plan and starts moving, the control subsystem takes over, allowing the *short term planner* to make limited adjustments depending on the contingencies encountered on the ground. In the event that the user is unable to follow the plan with only such limited modification (e.g., an unforeseen obstacle), the control subsystem has the capability to report the event and can request the construction of a new long term plan.

The *long term planner* produces the optimal path according to the diagram depicted in Figure 4.1 and described as follows: 1. the map is broken down into a grid of discrete cells containing free space, 2. a graph is derived from the grid, where each node is on the border between two cells cell and each arc is a path in free space, 3. the graph is

changed by adding relevant semantic information (e.g., associating points of interest with some of the cells), 4. each arc is associated with a cost that accounts for the distance to travel and for the occupancy of the area (the more people, the longer the time to travel), 5. additional manipulation are made to exclude (or to have a negative reward for) paths that violate the user preferences, 6. the optimal path is found using the modified Dijkstra algorithm.

In the next sections the algorithm is presented in its full details.

4.1.1 Preliminaries

To describe our long term planner, we first define some notation and operations on graphs.

A graph $G = (N, E)$ is a set of nodes $n \in N$ linked by a set of edges $e \in E$. An edge $e = (n, n') \in E$ is defined by its two adjacent nodes $n, n' \in N$.

Given graphs $G_1 = (N_1, E_1)$ and $G_2 = (N_2, E_2)$, $G_1 \subseteq G_2 \implies N_1 \subseteq N_2 \wedge E_1 \subseteq E_2$ means that G_1 is a subgraph of G_2.

Given $G_1 \subseteq G_2$, $G_2 \backslash G_1 = (N_2 \backslash N_1, E_2 \backslash \{e \in E_1 \mid e = (n, n') \wedge (n \in N_1 \vee n' \in N_1)\}$ is the graph that remains after removing G_1 from G_2. We do not consider $G_2 \backslash G_1$ if $G_1 \not\subseteq G_2$

Pairwise graph union is defined by $G_1 \cup G_2 = (N_1 \cup N_2, E_1 \cup E_2)$. The union of a set of graphs $\mathcal{G} = \{G_1, G_2, G_3, G_4, \ldots, G_m\}$ is denoted $\bigcup \mathcal{G}$ and performed pairwise, such that $\bigcup \mathcal{G} = ((\cdots(((G_1 \cup G_2) \cup G_3) \cup G_4) \cup \cdots) \cup G_m)$.

4.2 Planning Algorithm

The *long term planner* proposes feasible paths that efficiently visit the user's specified points of interest, while respecting the user's preferences and accommodating the prevailing conditions in the environment. To achieve this, the *long term planner* abstracts a complex environment, such as a shopping mall, airport, museum, etc., as a weighted directed graph, comprising a set of *nodes* linked by *edges*. The nodes represent places in the environment, while the edges represent direct paths between the places and are weighted by their *effective length*. The a priori length of an edge is the Euclidean distance between its adjacent nodes. The effective length of an edge is generally longer, modelling its undesirability with respect to crowding and the user's preferences.

Nodes are labelled with their physical location (coordinates on the plan of the environment) and their corresponding semantic position (supermarket, toilet, post office, café, bar, bakery, etc.). Each edge in the graph is labelled (weighted) with the effective distance between its adjacent nodes. Then, using efficient graph traversal algorithms, such as the Dijkstra algorithm [21], it is possible to find the shortest paths that link the user's points of interest. Moreover, anomalies and crowding can be included in the same framework

by simply modifying the graph prior to finding the shortest path. In particular, anomalies cause parts of the graph to be (temporarily) removed, while crowding increases the weights of edges in crowded areas (their *effective* length is increased because crowding slows the user's progress). Certain user preferences, such as always being near a toilet, may also be encoded as graph transformations.

4.2.1 Creating graphs from floor plans

To construct a graph that efficiently maps the free space in the environment, we first decompose its floor plan into a 'quad tree' [72], comprising quadrants containing free space (free quadrants) and quadrants occupied by fixed objects (occupied quadrants). A graph is constructed by embedding nodes in only the free quadrants and linking them with appropriate edges. The quad trees typically have substantially fewer cells than a uniform grid with the same level of minimum granularity, with the density of cells generally following the density of features [73]. An example is shown in Figure 4.4.

Given a quad tree decomposition of the free space, the corresponding graph is constructed as follows. For all pairs of adjacent free quadrants, a node is embedded at the mid point of the border of the smaller of the quadrants. By definition, a free quadrant is a convex shape containing only free space. Hence, any node on the border of a free quadrant has a "line of sight" to all other nodes on the borders of the same quadrant. We therefore join such nodes with a complete graph. Since nodes are shared between adjacent quadrants, this is sufficient to link all the free space in the environment.

To guarantee that the robotic platform may occupy any point in free space represented by a node, or travel the line represented by any edge, prior to building the quad tree the fixed objects are enlarged in all directions by a distance greater than the radius of the robotic platform. In this way no point in the *effective* free space is ever too close to a fixed object and all paths in the graph correspond to plausible paths in the environment.

4.2.2 Creating a long term plan

To represent the a priori knowledge about the environment we define a "graphmap" data structure $\mathcal{M} = (G, W, C, L)$. $G = (N, E)$ is a graph of the environment derived from a quad tree, as described in Section 4.2.1. Function $W : E \to (0, +\infty]$ assigns a length (the Euclidean distance between the points denoted by adjacent nodes) to all the edges of the graph. Function $C : N \to (\mathbb{Q}, \mathbb{Q})$ labels each node with its spatial coordinates in the environment. Function $L : N \to P \cup \{\textit{uninteresting}\}$ labels each node with its semantic location, where $P = \{\textit{supermarket}, \textit{bakery}, \textit{café}, \text{etc.}\}$ is a set of points of interest.

To generate a long term plan we also define a "working copy" of the graph map (the

working graphmap), modified according to the user's constraints, the current crowding and the known anomalies. We denote the working graphmap $\mathcal{M}' = (G' = (N', E') \subseteq G, W', C, L)$. In general, the graph G' excludes any inaccessible subgraphs arising from anomalies or the user's constraints. The weighting function W' assigns an *effective* length to all edges, which includes the effects of crowding and the user's constraints. The construction of G' and W' are described in Sections 4.2.3, 4.2.4 and 4.2.5.

Given a working graphmap $\mathcal{M}'$ and a (possibly ordered) set of user-specified points of interest, the *long term planner* proposes a path that visits the points of interest while respecting the user's global constraints. Formally, given a user-specified set of points of interest $\{p_j \in P\}_{j=1}^{m}$, the planner suggests a path $\{n_i \in N'\}_{i=1}^{k}$ s.t. $\forall p_j \in \{p_1, \ldots, p_m\}\ \exists n_i \in \{n_1, \ldots, n_k\} \wedge L(n_i) = p_j$. If the path must respect the order of the specified points of interest, then additionally $\forall p_s, p_t \in \{p_1, \ldots, p_m\}, \nexists n_i, n_j \in \{n_1, \ldots, n_k\}$ s.t. $s > t \wedge i < j \wedge L(n_i) = p_s \wedge L(n_j) = p_t$ holds true.

Finding the minimum length path that visits a set of unordered points of interest is an instance of the well known NP-hard 'travelling salesman problem' [74]. Moreover, given that the overall excursion (including stops at the points of interest) may take considerable time, an overall plan optimised for the current level of crowding may eventually be significantly sub-optimal if the crowds dissipate. Our approach is therefore to optimise each leg of the journey separately, using the most up-to-date information about anomalies and crowding.

In general and in simple terms, long term planning works in the following way. The planner first identifies the node $n_0 \in G'$ that is closest to the user's current coordinates (x_0, y_0). This is given by $n_0 = \arg\min_{n \in G'} \parallel C(n) - (x_0, y_0) \parallel$. If the user's points of interest have been specified in order, the planner uses Dijkstra's algorithm to find the shortest path between n_0 and the next unvisited point of interest specified by the user. If the user has not specified an order, the planner uses a modification of Dijkstra's algorithm to find the shortest path between n_0 and the closest unvisited point of interest. Given the trajectory and the user's coordinates, n_0 may not be the optimum first node in the path (it may be effectively behind the user on the path). The planner therefore sets the first node of the path to be the node by which the user will leave the current quadrant.

The inclusion of time-dependent anomalies makes the actual long term planning algorithm slightly more complex. Handling such anomalies is described in Section 4.2.5.

4.2.3 Global constraints

The user may specify constraints that affect the long term plan (e.g., always remain within 50 metres of a toilet). We call these global constraints to distinguish them from, for example, local constraints that might be implemented by the short term planner (e.g.,

don't get too close to other pedestrians). Global constraints may be hard or soft. Hard constraints exclude parts of the environment that the user does not wish to visit under any circumstances. They are implemented by removing subgraphs from G. The set of hard constraints is denoted $x \in X, x \subseteq G$, hence $G' = G \backslash \bigcup X$. Removing parts of the graph may significantly lengthen the planned journey or make it impossible, hence the final plan (or lack of it) is presented to the user for approval.

Soft constraints make parts of the environment desirable or undesirable to the *long term planner*, causing the planned path to deviate towards or away from them, respectively. They are implemented by defining a function $K : E \rightarrow [1, +\infty]$ that modifies the weights of edges to and from desirable and undesirable nodes. The function K is applied according to (4.1), introduced in Section 4.2.4. If no constraint applies to the nodes adjacent to edge e then $K(e) = 1$. In general, given two nodes n and n' connected by edges $e = (n, n')$ and $e' = (n', n)$, for a single constraint $K(e) > K(e') \iff$ n is more desirable than n'. In the case of multiple constraints applying to the same edge e, the value of $K(e)$ is the maximum considering all constraints.

In our implementation, soft constraints are specified using sets of triples (*location*, *radius*, *intensity*), which respectively define the semantic position, the radius of influence and the intensity of the constraint. In general, a constraint creates a gradient of weights that increase towards undesirable zones and vice versa for desirable zones.

We define a function $\tilde{K}_i : [0, radius] \rightarrow [1, intensity]$ that maps distance from the border of location i to the weight of the gradient. This function should be monotonic non-increasing in case of undesired locations, and monotonic non-decreasing in case of desired locations. In both cases its integral should be finite (i.e., the *radius* of influence should be finite). Function $\tilde{K}_i$ is later used by $K(e)$ for associating the weight to each edge. It is worth noting that there is high flexibility in the choice of $\tilde{K}_i$, which improves the expressiveness of global constraints, allowing per-user customisations (e.g., the profile of attraction to toilets might be different across users) as well as location based personalisation (e.g., the profile of repulsion of an open window is different from the one of a stair).

More formally, we assume the existence of a set of constraints $s \in S$. The *location* of each constraint defines a corresponding set of either desirable or undesirable nodes $N_s \subseteq N$ that are not necessarily disjoint. Let $d(i, j)$ denote the minimum Euclidean path distance from node i to node j, then for any edge $e = (n, n') \in E$, the value of $K(e)$ is given by

$$K(e) = \max_{\forall s \in S} \left(K^*_{location_s} \left[\min_{n'' \in N_s} \left(d(n'', n') \right) \right] \right)$$

where $K^*_{location_s}$ is defined as:

$$K^*_{location_s}(r) = \begin{cases} \tilde{K}_{location_s}(r) & \text{if } r \in [0, radius] \\ 1 & \text{otherwise} \end{cases}$$

4.2.4 Heat maps

Cameras in the environment monitor pedestrian traffic and construct "heat maps" that estimate average occupancy of the free space over useful time periods (e.g., the last five minutes or a long-term average for a particular day and time). Each point in the free space is thus assigned a value in the interval $[0, 1]$, denoting its time-averaged occupancy density. A point with average density 1 is effectively impassable. In practice, not all areas are monitored and monitored areas will be divided into an array of square cells of uniform local density. Unmonitored areas are assumed to have zero density. Areas occupied by fixed objects have density 1.

An edge represents a straight line path between the points in free space represented by its adjacent nodes. The average occupancy in the area surrounding the line affects the time taken to travel from one end to the other. The free space that the *short term planner* will allow the user to explore can be approximated by an ellipse whose vertices ("ends") coincide with the ends of the line. The area of the ellipse represents the capacity of the edge, while the heat within the ellipse represents the amount of capacity that is being used by others. To calculate the average occupancy of an edge, we integrate the occupancy density over its corresponding ellipse. The size and shape of the ellipse is a function of the edge. For simplicity we define an occupancy function $H : E \to [0, 1]$ that implicitly includes knowledge of the current heat map and performs this integration. The *effective* length of an edge is then given by the function $W' : E \to (0, +\infty]$, defined

$$W'(e) = \frac{K(e)W(e)}{1 - H(e)} \qquad \forall e \in E. \tag{4.1}$$

The intuition behind (4.1) is that the effective length of an edge e is proportional to the desirability $K(e)$ of the destination node and inversely proportional to the occupancy $H(e)$. When there is zero occupancy, $H(e) = 0$ and the effective length is only related to the desirability $K(e)$ and the Euclidean distance $W(e)$. With full occupancy, $H(e) = 1$, the effective length is infinite and (4.1) correctly models the fact that the edge is impassable.

4.2.5 Anomalies

During the course of a journey the user may encounter *anomalies* (semi-permanent obstructions, such as a wet floor, locked exit, dense crowd, etc.) that prevent the *short term planner* from making progress along the long term plan. An anomaly is represented by a data structure $(g \subset G, t \in (0, +\infty])$, where $g \subset G$ represents the inaccessible region of the environment and t is the estimated remaining time that the anomaly will last. The set of active anomalies (those with remaining time > 0) is denoted $a \in A$. Anomalies are removed from A when their remaining time reaches 0.

Anomalies exclude parts of the environment, but their effect is not permanent and is dependent on the chosen path. When a new anomaly (g, t) is detected by the short term planner, it is added to the set of active anomalies and its subgraph is immediately removed from the working graphmap. Symbolically, $A \leftarrow A \cup (g, t)$ and $G' \leftarrow G' \backslash g$. The shortest path to the next point of interest is calculated according to the procedure described in Section 4.2.2. The approximate time of reaching every node in the proposed path is calculated according to the average speed of the user.

The new trajectory definitely excludes the recently detected anomaly, but may include one or more anomalies in A. Hence, the proposed plan is compared to the subgraphs in the set of active anomalies, to find if there is any intersection. If there is no intersection the proposed plan is valid. If the proposed trajectory intersects the subgraph of an anomaly, the time of reaching the anomaly is compared to its remaining time. If the anomaly will not exist by the time the user reaches it, it is ignored. If no anomalies exist by the time the user reaches them, the proposed plan is valid. If, on the other hand, one or more anomalies remain valid by the time the user reaches them, their subgraphs are removed from the working graphmap and the above procedure is repeated until a valid path is found.

4.2.6 Time-dependent shortest paths

Our *long term planner* intelligently avoids looping paths by regularly updating heat maps and assigning persistence times to anomalies. In this way the planner never returns to permanent obstacles, but may take advantage of crowding and obstacles that clear. Our current approach with heat maps assumes that crowding averaged over a period of time in the immediate past is a good indicator of average crowding for the same time period in the immediate future. This is reliable for short term predictions, but is less so over the longer term because long term averages may mask large peaks of crowding. With regard to anomalies, our planning algorithm takes a cautious approach, assuming that an active anomaly encountered in one proposed path should not be considered in future plans to

the same point of interest.

Algorithm 1 describes the basis of our shortest path algorithm that considers timed anomalies, heat maps and user constraints. The algorithm finds the shortest path between the user's current position and the closest point of interest. If points of interest are required to be visited in a specific order, it is assumed that the set *Targets* contains only those nodes corresponding to the next point of interest to visit.

The algorithm makes use of several functions. $K(e)$, $W(e)$ and $H(e)$ are as in (4.1). Function Edges(n) returns the set of outgoing edges of node n. Function Dest(e) returns the destination node of edge e. Function Anomalytime(e) returns the absolute time at which edge e will be available. This function returns 0 for all edges that are not part of an anomaly. Two functions are updated during the planning process. Function Dist(n) returns the currently known shortest distance to node n. This is initially ∞ for all nodes except the initial node, for which the function returns 0. Function Time(n) returns the estimated time to reach node n given the user's average speed (denoted *speed*). The function initially returns ∞ for all nodes except the initial node, for which it returns 0.

In trying to satisfy the conflicting constraints of dynamic motion planning in complex human environments, we have considered many alternatives and refinements to our algorithms. There is no off-the-shelf perfect solution, given the inherent uncertainties and variability of the problem. In particular, finding time-dependent shortest paths is known to be hard and is itself the subject of active research [41, 42, 43, 44]. Our present approach is a satisfactory compromise between efficiency and efficacy. We can imagine circumstances under which it might be challenged, but we propose to allow further development to be led by problems encountered in real applications.

4.3 Implementation Aspects

The algorithm presented in this work has been designed keeping flexibility in mind. We devised an API that abstracts the low level structures and exposes a simple but efficient interface. It is divided into a number of layers visible in Figure 4.2. The bottom layer is represented by the *long term planner* itself, which is linked with the top level (the API) via three main blocks.

The first block is denoted "Environment and map" and, as the name suggests, allows external services to access and update information about the environment. Such data includes the map of static obstacles and walls, the heat maps and the anomalies. The latter can be grouped into categories, two being available by default: "wet floor" and "destination out of order". New categories can be added at runtime upon request by the third-party services.

Algorithm 1 Shortest path considering anomalies, heat and constraints

The initial node is the closest node to the user
Targets: a set of target nodes corresponding to the user's points of interest
Visited: a set of visited nodes, initially containing the initial node
Unvisited: a set of unvisited nodes, initially containing all nodes except the initial node
current $\leftarrow$ initial node
while *current* $\notin$ *Targets* **do**
  for all $e \in \text{Edges}(current)$ **do**
    if $\text{Dest}(e) \in Visited$ **then**
      continue
    end if
    if $\text{Anomalytime}(e) > \text{Time}(current)$ **then**
      continue
    end if
    $d \leftarrow \text{Dist}(current)K(e)W(e)/(1 - H(e))$
    if $\text{Dest}(e) \in Unvisited$ **then**
      if $d > \text{Dist}(\text{Dest}(e))$ **then**
        continue
      else
        $Unvisited \leftarrow Unvisited \backslash \{\text{Dest}(e)\}$
      end if
    end if
    $\text{Time}(\text{Dist}(e)) \leftarrow \text{Time}(current)$
      $+W(e)/(1 - H(e))/speed$
    $\text{Dist}(\text{Dest}(e)) \leftarrow d$
    $Visited \leftarrow Visited \cup \{\text{Dest}(e)\}$
  end for
  $Visited \leftarrow Visited \cup \{current\}$
  $Unvisited \leftarrow Unvisited \backslash \{current\}$
  if $|Unvisited| > 0$ **then**
    $current \leftarrow n \in Unvisited :$
      $\text{Dist}(n) \leq \text{Dist}(n'), \forall n' \in Unvisited$
  else
    report no possible path and quit
  end if
end while

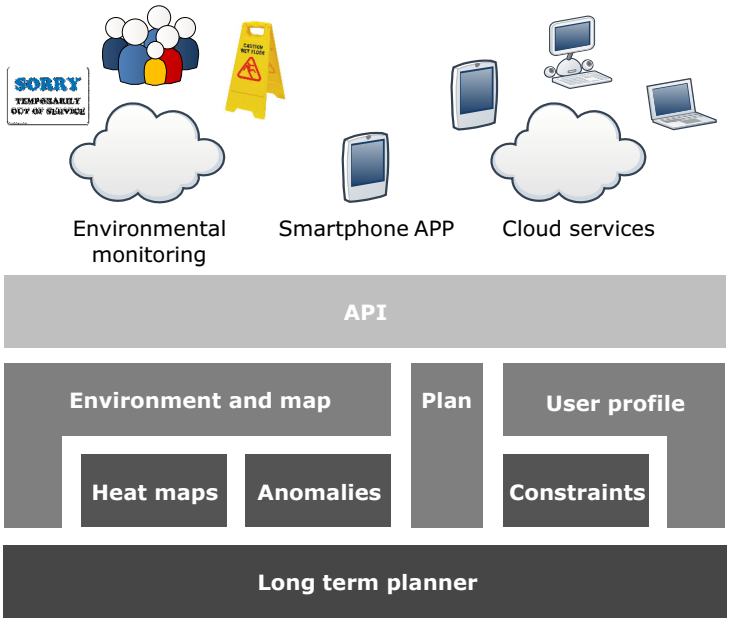


Figure 4.2: Structure of the API. The layers are of increasing abstraction, where the public interface is flexible and extensible at runtime by the third-party services. The overall low complexity enables a broad choice of implementations, from a service in the cloud to a standalone smartphone app.

The second block, "Plan", exposes the planning capabilities. Given the starting position, it is possible to query for the construction of the optimal path directed to one or more goals. The planner automatically considers the current status of the environment and biases the resulting trajectory according to the user preferences. Moreover, alternative sub-optimal paths can be generated upon request, for example when the chosen path is blocked by an unforeseen obstacle detected by the short term planner.

The last block is the "User profile" and encapsulates the interface for accessing the global constraints and other user information, such as her location and the tuning parameters for dealing with anomalies and crowded areas.

This API can be installed and accessed practically anywhere, thanks to the low computational burden highlighted in Section 4.5.3. For example, it can be packaged in a standalone mobile application for providing the users an interactive map of a shopping mall, or implemented as a cloud service, as described in the next section.

4.3.1 Implementation of a Cloud Service

For the experiments presented in details in Chapter 7, we implemented the planner as a service in the cloud. The interface was written in C++, while efficient Java was used for the planning part. The standard Java Native Interface (JNI) provides the link between

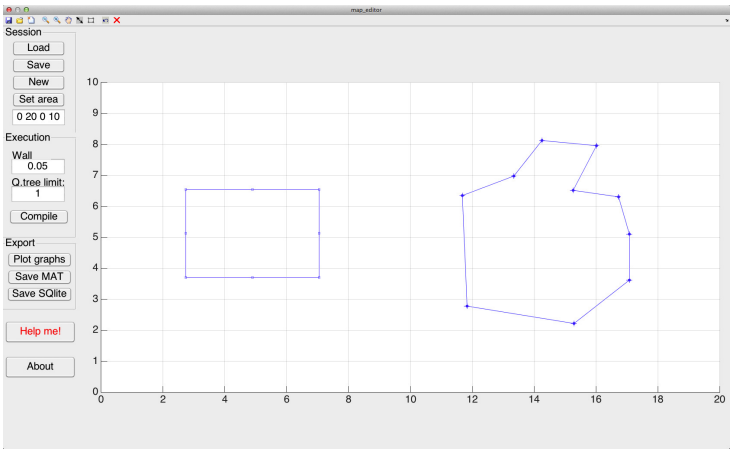


Figure 4.3: Screenshot of the map designer tool showing an example floor plan. Enables the user to create maps and generate the associated graph, compliant with the *long term planner*.

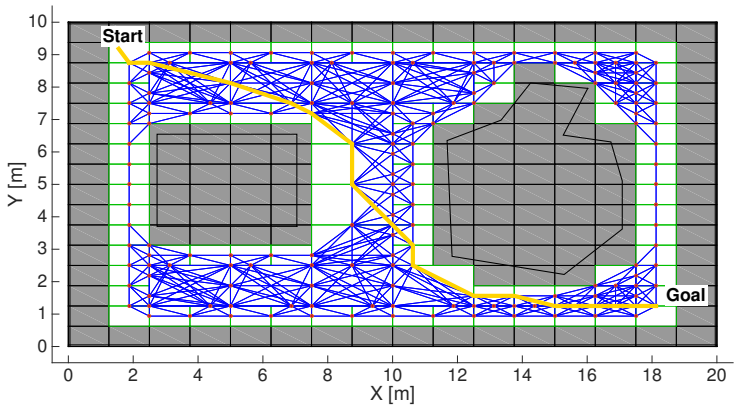


Figure 4.4: Graph and a sample path generated from the map depicted in Figure 4.3.

these elements.

The map of the environment is stored using SpatiaLite[1], a lightweight serverless spatial database that allows performing queries in the geometric space. A quad tree decomposition is then performed on the map and the resulting graph is used by the planner.

The communication with the remote clients takes place through exchange of JSON messages over a TCP link, in a request-reply mechanism, where the planner acts as a server.

4.4 Qualitative Analysis

We have implemented two tools, a map designer and a simulator. The map designer is written in MATLAB and enables the user to draw, load and save floor plans, as well as

[1] `http://www.gaia-gis.it/gaia-sins/`

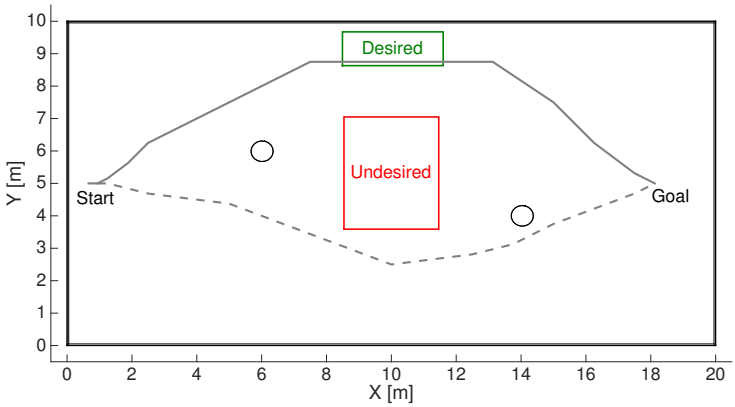


Figure 4.5: Simulation with constraints. The picture shows two independent simulations of how the planner deals with desirable and undesirable zones. The continuous line is the result of the constraint "stay close to the desirable zone" and "stay away from the undesirable zone", while the dashed line addresses only the latter.

performing quad tree decomposition and graph construction. The user is provided with a GUI to freely draw geometric shapes (Figure 4.3) and generate the corresponding graph (Figure 4.4) to be used in the simulator.

The simulator is written in MATLAB and Java and allows the user to visually configure global constraints, heat maps, anomalies and all parameters required by the *long term planner*. For performance reasons, the planning algorithm has been developed in Java and communicates with MATLAB through the integrated Java interface.

The chosen floor plan for the validation is a large room of approximately 200 m^2 with two non-aligned central columns. The starting point is set at the left hand side of the map in the midway along the shortest wall. The goal is set at opposite side of the room, such that the shortest path connecting the starting point to the goal is a straight line.

In the remainder of this section we will go through each feature separately and, finally, show a more complex simulation combining different features.

4.4.1 Global constraints

We show how the planner is able to deal with the user preferences when computing the plan. In the first simulation we put an undesirable zone in the middle of the room, overlapping the shortest path. In the second simulation, instead, we identified a desirable zone (e.g., a restroom) close to the top wall of the map without interfering with the shortest path. In both simulations the minimum distance to the zones is set equal to 1.5 m and the intensity is set to 2.

We ran these two simulation separately and the results can be seen in Figure 4.5. The planner correctly takes the constraints into account by properly bending and extending

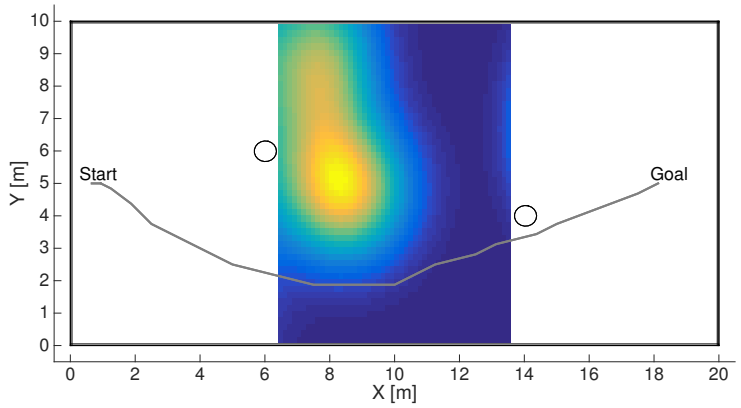


Figure 4.6: Simulation with a heat map. The path computed by the planner is represented by the continuous line and bypasses the crowded spot in the middle of the map (i.e., yellow zones). The preference is for cold zones (i.e., blue-coloured areas).

the original shortest path.

Should the undesirable zone be the only possible access point for reaching the goal, the planner can violate the constraints as long as the intensity is not $-\infty$ (i.e., *never* touch the undesirable zone).

4.4.2 Heat maps

We placed one rectangular shaped heat map in the centre of the room, covering the whole space between the two columns and the walls at the top and bottom of the figure. The planner is thus forced to go through the area covered by the heat map to reach the goal. We ran 50 simulations with different heat distribution generated by a sum of bivariate Normal probability density functions (normalised between $[0, 1]$) with random parameters. In all cases the planner correctly took into account the presence of the heat map.

The outcome of one particular simulation can be seen in Figure 4.6, where the planner properly avoids hot (yellow-coloured) zones.

4.4.3 Anomalies

In order to test the time based anomalies we set the average user speed to 0.5 m/s and we placed a rectangular anomaly in the middle of the room. Figure 4.7 depicts two paths constructed by the *long term planner* during two independent simulations with different durations of the anomaly. In this way we are able to show how the planner manages the disappearance of an anomaly. In the first run (dashed line) we configured the expiration of the anomaly in such a way that it expires when the user has covered approximately half of the path. In the second run (continuous line) the anomaly disappears after the

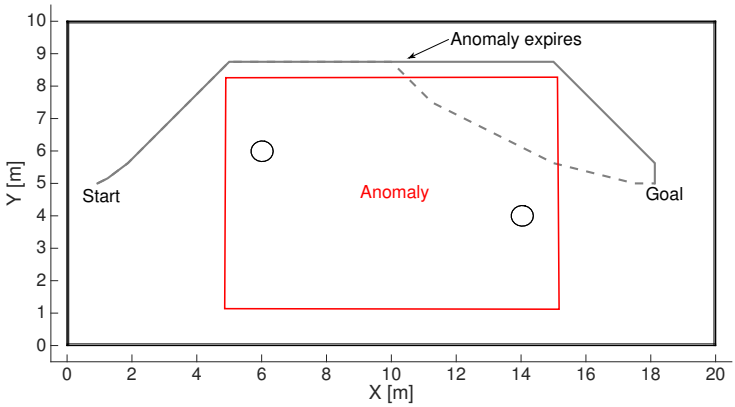


Figure 4.7: Simulation with anomalies: two independent simulations are shown. The dashed line represents the path generated when the anomaly is set to expire half way to the goal. The continuous line, instead, shows the resulting path when the anomaly does not expire.

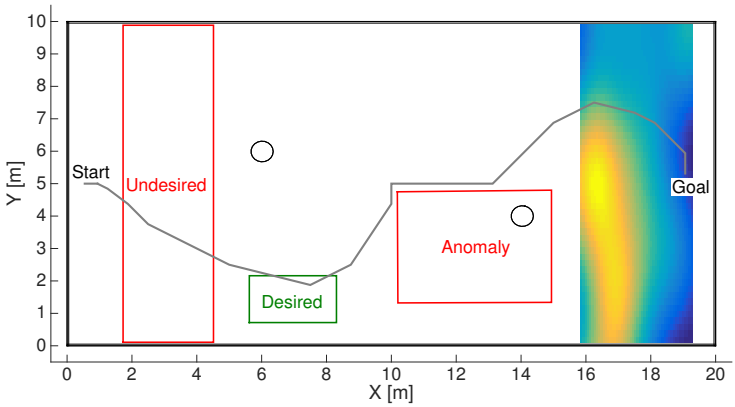


Figure 4.8: Simulation with multiple features. The planner satisfies all the user requests, but is forced to ignore the constraint for the undesired zone, as it is the only way for reaching the goal.

user reaches the goal position. It is clearly visible that, in the first case, as soon as the anomaly expires the planner re-routes the user towards the shortest path, overlapping what was the area occupied by the anomaly.

4.4.4 Combination of features

The last validation test considers a combination of multiple features in one simulation. We placed one undesired zone, one desired zone, one anomaly and one heat map as shown in Figure 4.8. In particular, the undesired zone completely blocks the passage for reaching the goal. However, as visible in the previous figure, the *long term planner* is able to ignore the unfeasible constraint. The path then bends towards the desired zone, bypasses the unexpired anomaly and, finally, avoids the crowded region represented by the heat map.

4.5 Quantitative Analysis

We now go through the results of some simulations providing a quantitative analysis of the performance of the *long term planner*. The goal is to show that the benefits of using the *long term planner* are evident not only from a qualitative point of view, as shown in Section 6.4, but also from a tangible set of performance metrics.

4.5.1 Global constraints

The simulations presented in this section show how the planner interprets the *intensity* parameter of an undesired or a desired global constraint. The environment and the position of the desired/undesired locations are the same as those considered in Section 4.4.1 and illustrated in Figure 4.5. We identified this particular scenario because it is a worst case situation: the desirable *location* is at the farthest possible distance from the shortest path and the undesirable *location* conflicts with the shortest path.

For simplicity and without any loss of generality, in these simulations we define $\tilde{K}$ as a linear function that is monotonically increasing for desired constraints, and monotonically decreasing for the undesired ones.

To measure the characteristics of a constraint for a desirable zone, we set the *location* as far as possible from the Euclidean shortest path and we fixed the *radius* to a large value. We then iteratively executed the planner with increasing *intensity* and we computed the minimum direct Euclidean distance of the path from the *location* (i.e., not considering the graph). The results are reported in Figure 4.9. As expected, the minimum Euclidean distance between the path and the desirable zone decreases as *intensity* increases. The steps in the plot are due to the quantisation of the free space imposed by the underlying graph.

A similar procedure was carried out using a constraint for an undesirable zone. We set the location of the constraint midway along the Euclidean shortest path and fixed the *radius* of the constraint to be the largest possible value (in the simulations the limit is the distance from farthest wall). The *intensity* of the constraint was then iteratively increased and we computed the minimum direct Euclidean distance of the resulting path from the *location*. The results are shown in Figure 4.10. We observe that as the *intensity* grows, the planner "pushes" away the constructed path until the minimum Euclidean distance is close to the *radius*. Again, the steps in the plot are due to quantisation of the free space.

The relations highlighted in these paragraphs are strictly dependent on the considered environment. Different locations, position of obstacles or constraints lead to different relations. An open problem, to be addressed in future work, is how to generalise the relationship between these parameters.

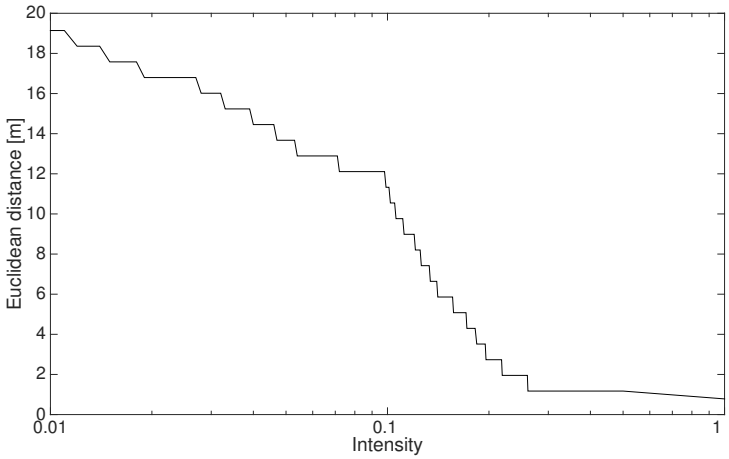


Figure 4.9: The effect of *intensity* on the minimum Euclidean distance from a path to a desirable zone on a particular simulation run. As *intensity* increases the path is attracted towards the desirable location: the Euclidean distance decreases.

4.5.2 Heat maps

We demonstrate that the *long term planner* is able to provide better (i.e., quicker) trajectories when it is aware of the crowdedness in the environment.

We set up a simulation similar to the one in Section 4.4.2, where the heat map covers the environment as in Figure 4.6. We then iteratively increase the heat surface, simulating an expanding crowd, starting from no crowd (0% crowdedness) up to a completely crowded area (100% crowdedness). At each iteration we call the *long term planner* and we compute both the optimal path (e.g., considering the heat encoded in the effective distance) and the Euclidean shortest path (e.g., a straight line directed to the goal that passes through the crowded area).

The Euclidean shortest path $N_e = \{n_i \in N'\}_{i=i}^{k}$ is constructed by assuming $H(e) = 0$ in (4.1). The true cost W_e of N_e is then computed by removing the $H(e) = 0$ assumption, thus

$$W_e = \sum W'(e), \forall e = (n, n') \in N_e$$

The results are shown in Figure 4.11. The very slow growth of the effective length of the path considering heat (thick line) is clearly visible. The planner diverts the path to avoid the hot areas until this becomes impossible (i.e., when crowdedness reaches 100%). In contrast, the effective length of the Euclidean shortest path explodes exponentially (thin line), making the planner unable to find a path when the average crowdedness level is greater than 5%.

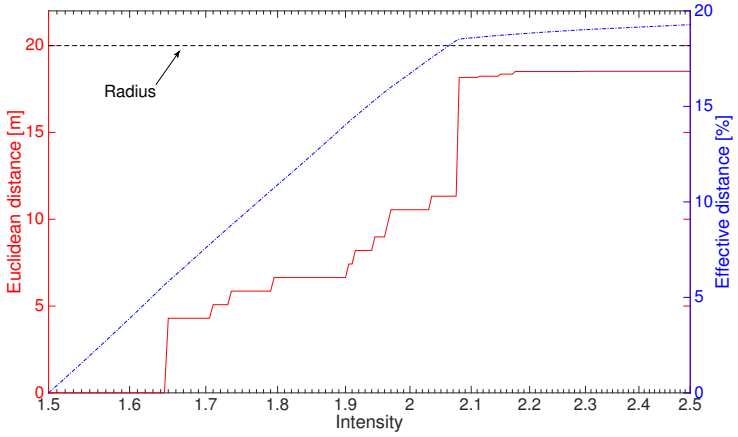


Figure 4.10: The effect of *intensity* on the minimum Euclidean distance from a path to an undesirable zone on a particular simulation run. As the *intensity* increases the path is pushed away from the undesirable location (the Euclidean distance increases) until it approximates the specified *radius* (20 m, dashed line). The constraint is actually implemented with respect to the effective length of the path, which is shown for comparison.

4.5.3 Computing time

We tested the performance of the *long term planner* on the BeagleBoard xM[2], an affordable embedded board equipped with an ARM processor running at 1 GHz and 512 MB LPDDR RAM. The operating system is Ubuntu 12.04 and the Oracle Java Virtual Machine 1.8.0_u6 is installed.

Our goal was to verify the feasibility of an online implementation in a realistic scenario and the scalability of the performance with increasing dimensions of the graph. We thus designed a map of a large shopping mall (500 m x 250 m) and performed quad trees decomposition (Section 4.2.1) with different minimum resolutions of the quadrant, varying from 4 m to 0.8 m. This way the resulting graphs had different sizes, from 1686 nodes and 13832 edges, to 23016 nodes 264026 edges.

For each graph we prepared a benchmark script that sets up the Java planning algorithm and queries 20 times for a path between the same two points at the opposite sides of the shopping mall. We then timed both the setup phase (e.g., loading the graph structure in the planning algorithm) and each of the planning queries. Finally, we computed the mean (μ) and standard deviation (σ) of the timings.

The results are reported in Table 4.1. The worst case, as expected, occurs with the largest graph. In this case we measured $\mu = 1983$ ms and $\sigma = 239$ ms for the setup phase, and $\mu = 812$ ms and $\sigma = 139$ ms for the query phase. These results are encouraging and show that the current implementation, that can easily be further improved, is already

[2] `http://beagleboard.org`

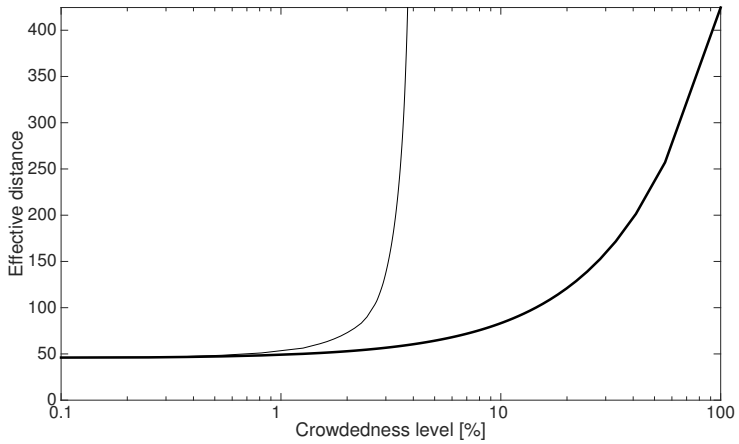


Figure 4.11: Relation between level of crowdedness and effective length of the path. When the planner is aware of the heat maps in the environment, the *long term planner* is able to avoid the heat and the effective distance increases slowly with increasing crowdedness (thick line). Without this information, the *long term planner* just chooses the shortest Euclidean path, whose effective length increases exponentially.

reasonably fast for an online execution. As a final note, it should be noted that in real scenarios the setup phase needs to be executed only when the graph structure (i.e., the floor plan) changes permanently.

Table 4.1: Performance of the *long term planner* on a BeagleBoard xM with different graph dimensions for both setup and query phase. Mean (μ) and standard deviation (σ) are reported for each phase.

Resolution of	Graph size		Setup [ms]		Query [ms]	
the quad tree [m]	Nodes	Edges	μ	σ	μ	σ
4.0	1686	13832	78	35	56	61
2.0	4404	43720	282	141	163	117
1.0	10133	108818	829	354	361	176
0.8	23016	264026	1983	239	812	139

Chapter 5

Short Term Planner

In this chapter we present the *short term planner*, the local motion planning algorithm that drives the user in a dynamic environment.

5.1 Preliminaries

Figure 5.1 gives a high level overview of the algorithm. At each iterative step the algorithm acquires the state of the system, comprising the position of static objects and the position and velocity of the user and of other people in the environment.

Given the current state, the algorithm hypothesises alternative courses of action using a human motion model. Each hypothesised trajectory is formally verified (model-checked) against properties that express goals and constraints required for the user's trajectory (i.e., where the user wants to go, obeying the appropriate social rules). This leads to a statistical distribution of potentially successful trajectories. The algorithm uses this distribution to choose an immediate action that maximises the probability of achieving the user's objectives and minimises the probability of problems. In this probabilistic context, the measurement noise is considered as an additional source of stochasticity.

The human motion model is an external module that generates plausible trajectories of people and remains transparent to the *short term planner*. Obvious requirements for this module are: 1) efficient prediction of the interactions that take places between pedestrians, 2) explicit inclusion of the user's objectives, and 3) possibility to capture the stochastic variations coming from the imponderable decisions of the different humans in the scene.

Part of this chapter was published in
A. Colombo, Fontanelli, A. Legay, L. Palopoli and S. Sedwards, "Motion planning in crowds using statistical model checking to enhance the social force model", *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*, 10-13 December 2013, [22].

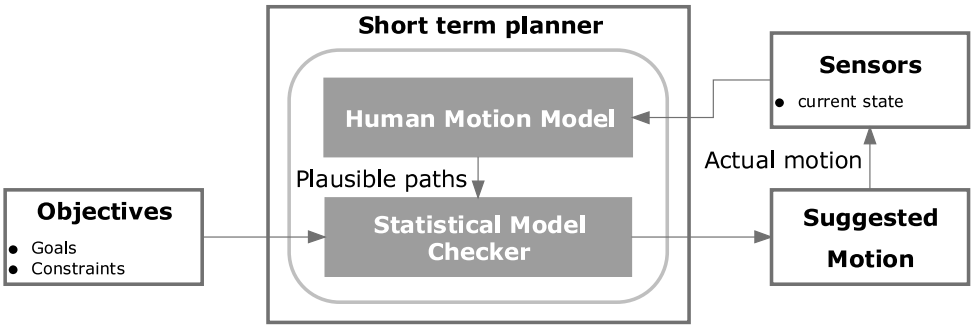


Figure 5.1: Diagrammatic overview of the *short term planner*. The sensors detect the current state of objects in the environment. This state is used by the human motion model to generate plausible future paths of the user and other pedestrians. The distribution of paths is verified against the objectives of the user in order to suggest an optimal course.

We use the stochasticity to generate a random sample of possible futures and choose the course of action that maximises the probability of success. Such trajectories respect the basic social and physical laws of pedestrian interactions and include the possibility of unpredicted behaviour. Their *distribution* allows the algorithm to choose a course of action that maximises the probability of success.

The stochasticity, while realistic, places an upper bound on the predictive accuracy of the model. Moreover, the model alone cannot account for the overall "mission" of the user. The predictive model needs to be managed *reactively*. Fortunately, the field of statistical model checking (SMC) encapsulates the technologies that we require to do this. SMC provides efficient algorithms to verify hypothesised trajectories against the user's constraints and objectives expressed in temporal logic. SMC can estimate the probability of success and bound the error of the estimation.

The key elements of our approach are (*i*) the human motion model to hypothesise trajectories that respect low level social and physical "forces"; (*ii*) temporal logic to express the high level goals of the user and (*iii*) a statistical model checker to verify the traces with respect to the goals.

5.2 Statistical and Probabilistic Model Checking

Model checking is an automatic technique to verify that a system satisfies a property [75]. Typically, the system has discrete states and the property is specified in temporal logic. The output of standard model checking algorithms is either *true* or *false*, with the possibility to give corresponding examples or counter-examples. The logics used are desired to be expressive, but must be decidable and tractable. Typical logics for standard model checking include LTL and CTL [76]. To give a result with certainty, the algorithms effec-

tively perform an exhaustive exploration of the state space of the system. The number of states scales exponentially with the number of interacting components in the system, leading to a 'state explosion problem' [75] that can make standard model checking slow or intractable.

Probabilistic model checking extends the standard notion to include probabilistic or stochastic transitions. These can express the uncertainties of modelling and reality. The output of probabilistic model checking algorithms is the probability that an arbitrary execution of the system will satisfy a given property. Such properties are specified in probabilistic or stochastic logics, such as PCTL and CSL [76]. The probabilistic model checking problem is solved with *numerical* model checking algorithms. These calculate the notionally exact probability by considering all the states. As such, they suffer the same state explosion problem as standard model checking algorithms.

Statistical model checking (SMC) is a type of probabilistic model checking that avoids the state explosion problem by estimating the probability of a property ϕ from executions (simulations) of the system. Given N independent simulation traces $\omega_i, i \in \{1 \dots N\}$, and a model checking function $\mathbb{1}(\omega_i \models \phi) \in \{0, 1\}$ that indicates whether $\omega_i \models \phi$ (read "ω_i *satisfies* ϕ"), the probability γ that an arbitrary execution satisfies ϕ is estimated using $\gamma \approx 1/N \sum_{i=1}^{N} \mathbb{1}(\omega_i \models \phi)$. This reduces the probabilistic model checking problem to estimating the parameter of a Bernoulli random variable, hence the confidence of the estimate can be guaranteed by standard statistical bounds, described in Section 5.3. In general, the confidence of the estimate increases with increasing N. In comparison to standard and numerical model checking, SMC does not require decidable logics nor a finite state space, making it particularly suitable for the present application that considers continuous time and space.

5.2.1 Bounded Linear Temporal Logic

Our model checking engine is based on the PLASMA-lab library [77]. PLASMA-lab implements the function $\mathbb{1}(\omega_i \models \phi)$ using bounded linear temporal logic (BLTL [78]) to express the property ϕ:

$$\phi = \phi \vee \phi \mid \phi \wedge \phi \mid \neg\phi \mid \mathrm{F}_{\leq t}\phi \mid \mathrm{G}_{\leq t}\phi \mid \phi\,\mathrm{U}_{\leq t}\phi \mid \mathrm{X}\phi \mid \alpha$$

$\vee, \wedge$ and $\neg$ are the standard logical connectives and α is a Boolean constant or an atomic proposition constructed from numerical constants, state variables and relational operators. X is the *next* temporal operator: $\mathrm{X}\phi$ means that ϕ will be true on the next step. F, G and U are temporal operators bounded by time interval $[0, t]$, relative to the time interval of any enclosing formula. We refer to this as a *relative interval*. F is the *finally* or *eventually*

operator: $\mathrm{F}_{\leq t}\phi$ means that ϕ will be true at least once in the relative interval $[0, t]$. G is the *globally* or *always* operator: $\mathrm{G}_{\leq t}\phi$ means that ϕ will be true at all times in the relative interval $[0, t]$. U is the *until* operator: $\psi \mathrm{U}_{\leq t}\phi$ means that in the relative interval $[0, t]$, either ϕ is initially true or ψ will be true until ϕ is true. Combining these temporal operators creates complex properties with interleaved notions of *eventually* (F), *always* (G) and *one thing after another* (U). A detailed description of the semantics of BLTL is given in [78].

5.3 Statistical Confidence

The statistical model checker deals with N independent simulations ω_i that can either satisfy ($\omega_i \vDash \phi$) or not satisfy ($\omega_i \nvDash \phi$) a given logical property. The goal is to produce an approximation interval $[p - \delta, p + \delta]$ for $p = Pr(\omega_i \models \phi)$, with a confidence $(1 - \alpha)$ with $\alpha \in [0, 1]$.

Hence, the problem is reduced to the estimation of the parameter of a Bernoulli random variable X_i where

$$X_i = \begin{cases} 1 & \text{if } \omega_i \vDash \phi \\ 0 & \text{if } \omega_i \nvDash \phi \end{cases}$$

One possibility to guarantee the confidence of the estimate is using the Chernoff-Hoeffding bound [79] for sum of i.i.d. binary random variables $X = \sum_1^N X_i$. In particular, *Theorem 1* states that

$$Pr(X - \mathrm{E}[X] \geq \delta) \leq e^{-2N\delta^2}$$

and symmetrically,

$$Pr(-X + \mathrm{E}[X] \geq \delta) \leq e^{-2N\delta^2}.$$

Thus, we can sum up these two inequalities and obtain

$$Pr(|X - \mathrm{E}[X]| \geq \delta) \leq 2e^{-2N\delta^2}$$

that can be also interpreted as the probability that X falls outside the interval of confidence of size 2δ around $E[X]$, namely the level of statistical confidence α

$$\alpha = Pr(X \notin [\mathrm{E}[X] - \delta, \mathrm{E}[X] + \delta]) \leq 2e^{-2N\delta^2}.$$

If we solve this equation for N we get

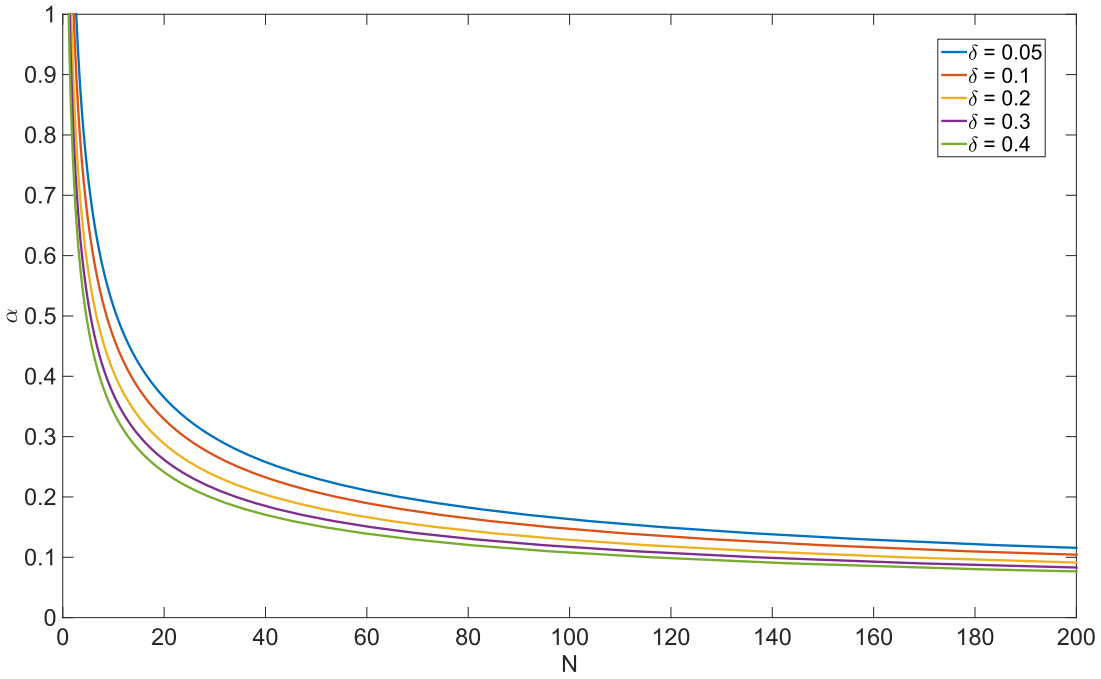


Figure 5.2: Relation between number of simulations N and statistical confidence α according to the Chernoff-Hoeffding bound in (5.1).

$$N \geq -\frac{ln(\alpha/2)}{2\delta^2} \tag{5.1}$$

meaning that we require at least this number of simulations in order to have $(1-\alpha)$ confidence interval $\mathrm{E}[X] \pm \delta$. Figure 5.2 shows how changing the number of simulations N affects the confidence of being within the interval α according to the (5.1).

A very good source for going into more details has been written by Mitzenmacher and Upfal in their book *Probability and computing* [80].

5.4 SMC–based Motion Planner

Our motion planner is based on the Algorithm 2. The planner assumes the existence of a pre-calculated long term plan that visits the user's objectives in an a priori optimal way, that is, considering all things known in advance. Typically, as discussed extensively in Chapter 4, the long term plan is computed with respect to a map of the static objects in the environment, the user's objectives and predicted anomalies (e.g., known crowded areas). Any contradiction of the a priori assumptions (e.g., an unforeseen blockage) triggers a recalculation of the long term plan.

The sensor board provides the current state of the local environment, located with

Algorithm 2 The planning algorithm

```
1: function FindLocalPath(state_user, state_ped1, state_ped2, ..., Map, GlobalPlan, Formula, N)
2:     Real P_curr, d_curr, P_best, d_best;
3:     [P_best, d_best]=[0,∞];
4:     for α_curr ∈ {0, ±25, ±50, ±75, ±90} do
5:         [P_curr, d_curr] = SMC(N, Formula);
6:         if is_better([P_curr, d_curr], [P_best, d_best]) then
7:             α_best = α_curr;
8:             [P_best, d_best] = [P_curr, d_curr];
9:         end if
10:    end for
11:    if P_best == 0 then
12:        return STOP;
13:    else
14:        return α_best;
15:    end if
16: end function
```

respect to the long term plan: the position and velocity of the user (state_{user}); the positions and velocities of other pedestrians ($\text{state}_{ped_1}, \text{state}_{ped_2}, \dots$); the position of static objects (Map). The algorithm calculates a local way point $\mathbf{w}$, which is the user's point of greatest straight line progress along the long term plan within the sensor range. $\mathbf{w}$ is used to calculate the user's driving velocity $\mathbf{v}^0$, assuming a constant desired speed. The driving velocities of the other pedestrians are estimated from their current velocities.

The algorithm uses the above information to parametrise the human motion model of the local environment. For the purpose of validation we have used the Social Force Model (Section 6.2). It is a good tradeoff between complexity and flexibility, as well as being one of the best known models literature.

Some of the parameters (e.g., τ_i) of other pedestrians are unknown to the algorithm, so it assumes the default values given in [81]. In the current implementation we construct the noise term $\boldsymbol{\xi}_i$ from two normal distributions; one for the magnitude and one for the direction.

The motion planner assumes the user will follow the long term plan, but need to temporarily deviate to avoid collisions. The output of the algorithm is a suggested deviation, α_{best}, in the range ± 90 degrees relative to the user's direct path to $\mathbf{w}$. To find α_{best}, the algorithm constructs models for each hypothesised deviation in the set $\{0, \pm 25, \pm 50, \pm 75, \pm 90\}$. These values are chosen to span ± 75 degrees using a tractable number of different values, with ± 90 included in case the user needs to sidestep an obstacle (see Chapter 6). Each model is then investigated using statistical model checking.

The algorithm sets $\alpha_{curr} \in \{0, \pm 25, \pm 50, \pm 75, \pm 90\}$ and calls function SMC with arguments N and Formula. SMC estimates the probability of success P_{curr} for a particlar deviation α_{curr} by the proportion of N simulation traces that satisfy the BLTL property Formula. The value of α_{curr} is used as the *intial* deviation: the user's driving velocity is initially rotated by α_{curr}, but at each successive step of the simulation the deviation from a direct path to $\mathbf{w}$ is reduced to zero. This ensures that the user will eventually be close to the long term plan.

BLTL is expressive enough to define complex sequences of high and low level requirements. For the results presented here, Formula merely expresses the basic constraints of the user:

$$(\mathrm{G}_{[0,T_{horizon}]} \bigwedge_{i \neq u} \|\mathbf{x}_u - \mathbf{x}_i\| > 0.5) \wedge (\mathrm{F}_{[0,T_{horizon}]} \|\mathbf{x}_u - \mathbf{w}\| < 0.2) \tag{5.2}$$

$\mathbf{x}_u$ denotes the position of the user and $\| \cdot \|$ denotes Euclidean distance. Intuitively, (5.2) means that "in the next $T_{horizon}$ time units the user will get no closer than 0.5m to any other pedestrian and will eventually be less than 0.2m from the long term plan".

$T_{horizon}$ is chosen to be the expected time for the user to walk a distance equivalent to the range of the sensors. Using a higher value might produce impossible trajectories that pass through unseen fixed objects; using a lower value might exclude possible collisions. In our implementation we use $T_{horizon} = 4$s.

For each hypothesised deviation α_{curr}, the SMC function returns the probabillity of success P_{curr} and the expected distance from the long term plan, d_{curr}. These are used by function *is_better* to decide α_{best}. *is_better* chooses the smallest $\mid \alpha_{curr} \mid$ which maximises P_{curr}. Ties are resolved by choosing the $alpha_{curr}$ with smallest d_{curr} or randomly if d_{curr} also ties. If $P_{best} == 0$ the user is required to stop (the long term plan will be recalculated).

$T_{decision}$ is the actual time the algorithm takes to make its predictions and must be less than the time period it is predicting, i.e., $T_{horizon}$. In practice $T_{decision}$ is bounded below by the performance of the hardware, the complexity of the environment (fixed and moving objects) and the confidence required (controlled by the number of simulations, N). In our implementation, $T_{decision} \approx 1s$.

At each decision point α_{best} is suggested to the user. The user may ignore this suggestion and move in a different direction, but the operation of the algorithm in the next decision period remains the same: α_{best} is calculated according to the long term plan and the actual positions and velocities of the user and other pedestrians. Since the user specifies the long term plan, when generating hypothesised traces the algorithm assumes that the user is compliant, however $\boldsymbol{\xi}_{user}$ may be used to model a lack of compliance.

Given an accurate stochastic model of the behaviour of pedestrians, the Chernoff bound [82] predicts that with $N = 10$ simulation runs the estimate of the probability of success has a maximum error of ± 0.3 with probability 0.7. With $N = 50$ the probability of success has a maximum error of ± 0.2 with probability 0.90. In general, the statistical confidence of the estimate increases with increasing N, but this only increases the probability of choosing the correct α_{best}. The predictive power of the model is bounded by its stochasticty. Thus, given finite computational power, we choose a value of N that balances the reactive and predictive aspects of the algorithm. That is, we choose a value of N that allows us to make $T_{decision}$ sufficiently small.

The algorithm solves (6.1) using a standard ODE solver [83], which produces traces comprising a sequence of states at discrete time points. Since the model given in Section 6.2 is based on continuous time and space, to guarantee properties that rely on the distance between objects it is necessary to choose time points that are sufficiently close. This is achieved by the ODE solver using adaptive time steps.

Simulating the traces accounts for most of the computational cost of the algorithm. We have found our chosen ODE solver to be efficient and presume its performance scales in a standard way with respect to the number of visible moving agents M and the complexity of their interactions. Since the forces in the model are dependent on the distances between agents, there is an additional $\mathcal{O}(M^2)$ cost, however M is bounded by the range of the sensors.

5.5 Quantitative Analysis

In this section we demonstrate the algorithm by means of computer simulations. We have implemented the algorithm in C++ and we use PLASMA-lab [77] as the statistical model checking library. To test the algorithm we have created a virtual environment that evolves according to the Social Force Model and contains fixed objects and other pedestrians that react to the user's presence.

The pedestrians are assigned individual long term trajectories to simulate their objectives and individual parameters that reflect the variation seen in reality. The values of the parameters are based on the ones estimated in [81] and two different but correlated sets, one for the planner and one for the virtual environment, have been defined in order to increase the sense of reality. The noise term ξ has been differentiated as well, the standard deviation of the two normal distributions in the planner has been set as the double of the one in the virtual environment.

In this way we simulate pedestrians that are reactive to the user and each other, with behaviour that is realistically unpredictable. Moreover, the simulated device has limited

omnidirectional sensing range, we suppose it is able to detect agents moving within a radius of 4 meters with respect to the current position of the user. In the final application, a sensor board connected to the single board computer will provide the real (estimated) positions and velocities of the user and nearby pedestrians.

We compared three different strategies:

- SMC with the Social Force Model ($SMC + SFM$): our novel approach, where Algorithm 2 computes the short term plan. When detected, an agent is supposed to evolve according to the Social Force Model.
- SMC with a linear motion model ($SMC + LIN$): similar to $SMC + SFM$ but agents evolve according to a different and simpler model. When detected, an agent is suposed to keep moving with same speed and same direction.
- Social Force Model only (SFM): we analyze the evolution of the environment without any decision points ($T_{decision} = \infty$).

For $SMC + SFM$ and $SMC + LIN$ we use the temporal logic formula defined by Eq. (5.2). We also used the following parameters: $T_{horizon} = \{1, 2, 4, 6, 8\}$, $T_{decision} = 1$ and $N = 50$. We performed 500 independent runs for SFM and 500 for every combination of $T_{horizon}$ for $SMC+SFM$ and $SMC+LIN$. Our objective was to demonstrate that 1) the higher complexity of our approach leads to valuable payoff in terms of performance and 2) it can be implemented online on an embedded device with limited computing power.

5.5.1 Algorithm Performance

We have devised two scenarios that challenge our algorithm and highlight significant features of its performance. In the first one (namely, *scenario 1*, depicted in Figure 5.3(a)) the user moves on a straight line close to a fixed obstacle, while two agents are moving towards her following a straight line as well. In the second one (namely, *scenario 2*, showed in Figure 5.3(b)) the user attempts to visit a market stall at the end of the market while some pedestrians (Agent 1-6) block the user's progress by entering the scenario and moving from one market stall to another. The user's long term plan is a straight line from the left to the right of the market. Figure 5.4 depicts the distances over time with respect to the user, respectively, for one particular run of *scenario 2*.

We defined 4 indicators to measure performance: 1) the time needed for the user to reach the right side of the scenario (T_{exit}), 2) the measured probability of respecting the minimum safety distance to agents (P_{safe}), 3) the average deviation in position from the long term plan (ϵ_x) and 4) the average deviation of the orientation of the user with

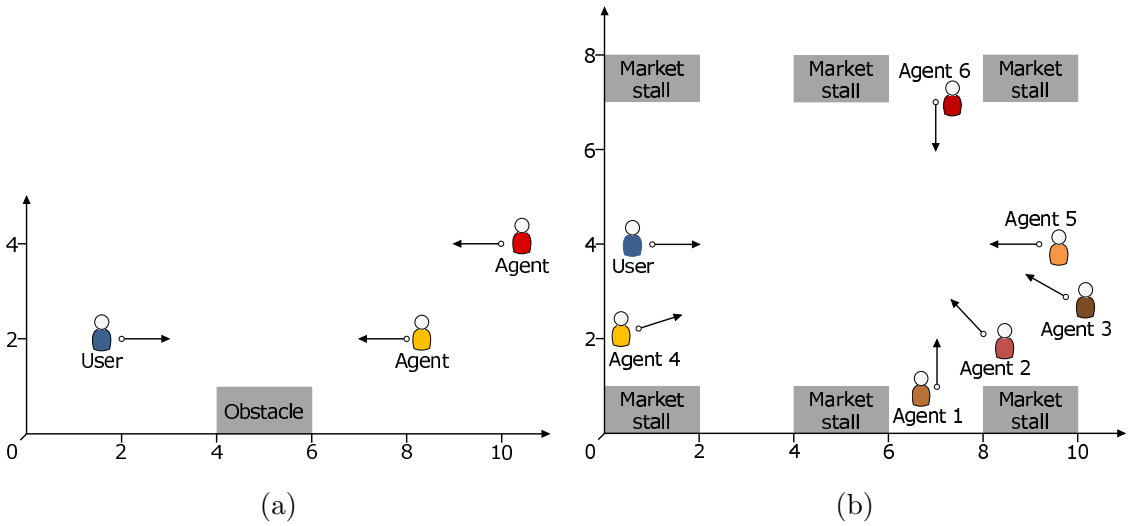


Figure 5.3: Scenarios used to test the algorithm, *scenario 1* (a) and, *scenario 2* (b).

respect to the ideal orientation of a user perfectly following the long term plan (ϵ_θ). These indicators are formally defined as follows.

Let $x(t)$ represent the cartesian coordinates of the position of the user after the planning for each time t, $\theta(t)$ represent its orientation with respect to a fixed frame, $\tilde{x}(t)$ the long term plan and $\tilde{\theta}(t)$ the orientation decided according to the long term plan. The integral error of the difference between the corrected plan and the long term plan can be defined as:

$$\epsilon_x = E\left\{\sqrt{\frac{1}{T}\int_0^T |x(t) - \tilde{x}(t)|^2\, dt}\right\}$$

A similar performance indicator ϵ_θ is defined for the orientation $\theta(t)$:

$$\epsilon_\theta = E\left\{\sqrt{\frac{1}{T}\int_0^T \left|\theta(t) - \tilde{\theta}(t)\right|^2 dt}\right\}$$

Indicators ϵ_θ and P_{safe} can be used to quantify the "comfort" of the user. Indeed, frequent changes in the direction reduce the user experience, especially if elderly, and so does the probability of accidents. Table 5.1 and Table 5.2 reports the performance we obtained for *scenario 1* and *scenario 2*, respectively, using different values for $T_{horizon}$.

Scenario 1 is the most problematic for SFM due to the limitations of this model we discussed in Chapter 6. The SMC-based strategies exhibit a higher P_{safe} and a lower ϵ_θ

Table 5.1: Scenario 1: performance for $SMC + SFM$, $SMC + LIN$ and SFM strategies. 500 simulations each were conducted.

	Unit	$SMC + SFM$					SFM
$T_{horizon}$	[s]	1	2	4	6	8	-
T_{exit}	[s]	23.08	23.38	22.72	21.68	21.11	23.56
P_{safe}	-	0.7444	0.8923	0.9933	0.9981	0.9985	0.7386
ϵ_x	[m]	0.3504	0.9914	1.4377	1.6131	1.7386	0.3062
ϵ_θ	[DEG]	53.19	37.22	13.93	10.84	9.36	36.86

	Unit	$SMC + LIN$				
$T_{horizon}$	[s]	1	2	4	6	8
T_{exit}	[s]	23.17	24.63	24.55	24.42	24.19
P_{safe}	-	0.7511	0.8709	0.9565	0.9989	0.9925
ϵ_x	[m]	0.3322	0.9832	1.4761	1.9007	2.0384
ϵ_θ	[DEG]	55.89	48.03	40.11	24.66	22.47

when $T_{horizon} \geq 6$. $SMC + SFM$, in turn, outperform $SMC + LIN$ on all indicators.

In *scenario 2*, from the safety and comfort point of view of the user, $SMC + SFM$ approach obtains a higher P_{safe} and a lower ϵ_θ with respect to SFM and $SMC + LIN$, when $T_{horizon} \leq 6$. Nonetheless, P_{safe} decreases and ϵ_θ increases when $T_{horizon} > 6$. This is motivated by the fact that the tested temporal logic formula is less likely to be satisfied over a large horizon in a crowded environment. As a consequence, the planning algorithm suggests the user to stop and/or to change direction in order to avoid the unfeasible path, thus raising ϵ_θ.

The SFM strategy exhibits the lower ϵ_x because it tends to keep the user closer to the long term plan. However, this reflects negatively on the "comfort" of the user, especially on P_{safe}, because the model doesn't have an explicit notion of *minimum safety distance to agents*. This behaviour is more evident in *scenario 1*.

5.5.2 Computing time

In order to show the performance of our algorithm in a real scenario, we ran the planning algorithm on a off-the-shelf low power embedded system, the Beagleboard xM[1]. It is a portable device that may run from battery power and provides performance comparable

[1] http://www.beagleboard.org

Table 5.2: Scenario 2: performance for $SMC + SFM$, $SMC + LIN$ and SFM strategies. 500 simulations each were conducted.

	Unit	$SMC + SFM$					SFM
$T_{horizon}$	[s]	1	2	4	6	8	-
T_{exit}	[s]	26.82	23.89	24.08	21.12	20.27	23.16
P_{safe}	-	0.9908	0.9998	0.9993	0.9977	0.9316	0.9665
ϵ_x	[m]	0.7677	0.7927	0.6497	0.5701	0.5430	0.3825
ϵ_θ	[DEG]	9.97	5.50	9.20	10.59	19.97	13.67

	Unit	$SMC + LIN$				
$T_{horizon}$	[s]	1	2	4	6	8
T_{exit}	[s]	27.33	31.48	36.03	29.98	23.71
P_{safe}	-	0.9882	0.9965	0.9977	0.9925	0.9486
ϵ_x	[m]	0.7902	1.4279	1.2607	0.8282	0.6760
ϵ_θ	[DEG]	10.62	14.25	20.33	21.56	21.52

to a small computer. We measured the time needed by the Beagleboard xM to execute the scenarios presented in the previous section. We ran 500 simulations each and we timed the execution of every single decision step for the $SMC + SFM$ strategy, that is, the time needed for a single run of Algorithm 2 using the Social Force Model as the model for the agents. We also set $N = 50$. We computed both the average $\mu_1 = 228.9$ ms and $\mu_2 = 2026.1$ ms and standard deviations $\sigma_1 = 392.1$ ms and $\sigma_2 = 2432.1$ ms of the timings for *scenario 1* and *scenario 2*, respectively. If we allow a maximum 1000 ms latency to compute the decision step, the current implementation is able to satisfy it in 93.4% of the cases for *scenario 1* and in 40.9% of the cases for *scenario 2*.

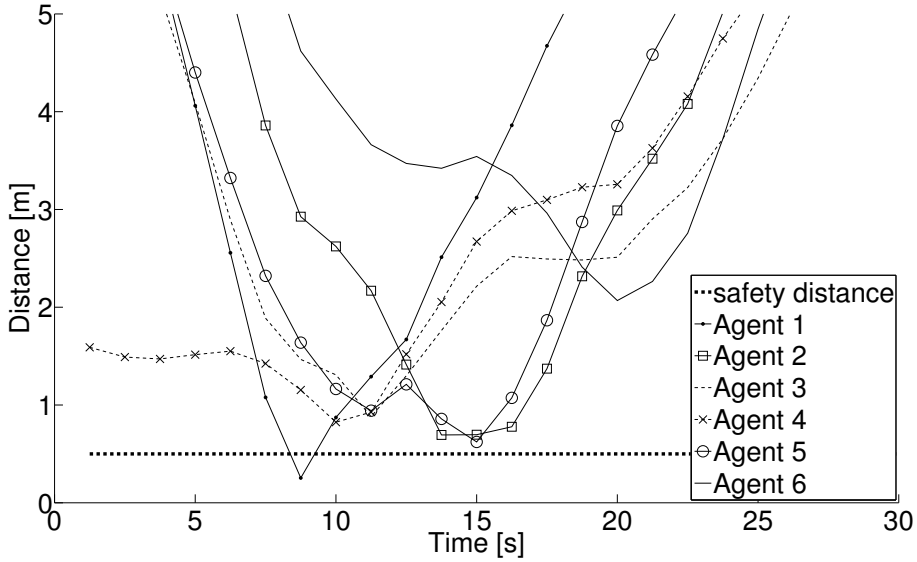


Figure 5.4: Scenario 2. Distances of the agents with respect to the user during one particular run of *scenario 2*. In this case the safety distance has been set to 0.5 m (dashed line) and has been violated once.

Chapter 6

Identification of Human Motion Models

In this chapter we discuss the Social Force Model, the model for human motion used by the *short term planner* in Chapter 5. We highlight its advantages and limitations, and we propose to improve its performance by exploiting the proxemic theory discussed in Section 6.3.

6.1 Preliminaries

The Social Force Model (SFM), Section 6.2, is a model of continuous interaction (mutual reaction), driven by the desired trajectories of moving agents (denoted $\mathbf{v}^0(t)$). When using the SFM in a generative context (e.g., [84]), $\mathbf{v}^0(t)$ may be specified in advance. The desired trajectories of real pedestrians, however, are a function of their objectives and the instantaneous positions and velocities of other pedestrians. In our motion planning application we are able to measure such positions and velocities, but only the objectives of the user are known with any certainty. The intentions of other pedestrians must be inferred from their trajectories. Noting that humans tend to walk in straight lines, we approximate the desired trajectories of other pedestrians piecewise, using short term linear extrapolations of their most recently detected motion. This is sufficient when the motion is smooth and the prediction timescale is short. In reality, the relatively smooth long and medium term trajectories of pedestrians are often interrupted by sudden short term pauses and deviations.

Part of this chapter was published in
A. Colombo, Fontanelli, D., D. Gandhi, A. De Angeli, L. Palopoli, S. Sedwards and A. Legay, "Behavioural templates improve robot motion planning with social force model in human environments", *Emerging Technologies & Factory Automation (ETFA), 2013 IEEE 18th Conference on*, 10-13 September 2013, [23].

To improve the performance of the SFM, we have therefore investigated in detail the behaviour of human participants in a simulated shopping environment. We have defined an experimental procedure comprising a concurrent verbal protocol (to identify the motivation of the choices made by the participants), video recording and motion tracking using a RGB-D camera. The reconstructed trajectories of the participants were manually annotated with corresponding motivations, thus identifying a set of behavioural patterns. The data corresponding to each pattern, with the help of the proxemic theory, was then used to parametrise the SFM using standard algorithms.

The results of our investigation show that, for some patterns, parametrisation of our existing model is sufficient to produce a good reconstruction of observed behaviours. In other cases, we find that it will be necessary to incorporate the patterns as explicit modifications to the SFM. Since the patterns have recognisable signatures, we conclude that we can construct an improved SFM using behavioural templates.

6.2 The Social Force Model

Following [81], our model is constructed in two dimensions, with human agents represented by circular discs and fixed objects represented by lines. In what follows we denote vectors in bold type. Thus, agent i has mass m_i centered at position $\mathbf{x}_i \in \mathbb{R}^2$ in the environment, radius r_i and velocity $\mathbf{v}_i \in \mathbb{R}^2$. The SFM is described by a system of linear differential equations

$$\begin{cases} \dot{\mathbf{x}}_i = \mathbf{v}_i \\ \dot{\mathbf{v}}_i = \frac{\mathbf{v}_i^0 - \mathbf{v}_i}{\tau_i} + \frac{\mathbf{f}_i + \boldsymbol{\xi}_i}{m_i} \end{cases} \tag{6.1}$$

$\mathbf{v}_i^0$ is the *driving (desired) velocity* of agent i, represented by a product of speed v_i^0 and normalised direction $\mathbf{e}_i^0$. In our algorithm $\mathbf{e}_i^0$ is given by the line joining the current position and the next via point. Importantly, since v_i^0 is by default set to the user's preferred walking speed, $\mathbf{v}_i^0$ is time invariant between via points. τ_i is the time taken to react to the difference between desired and actual velocity, while $\boldsymbol{\xi}_i$ is a noise term modelling fluctuations not accounted for by the deterministic part of the model. The noise term can also serve to avoid deadlocks and hypothesise alternative trajectories. In our implementation we assume $\boldsymbol{\xi}_i$ is normally distributed. In the absence of the exogenous inputs $\mathbf{f}_i$ and $\boldsymbol{\xi}_i$, the agent's trajectory simply converges to the driving velocity with time constant τ_i. $\mathbf{f}_i$ is the overall force acting on agent i resulting from other objects in the environment and is given by

$$\mathbf{f}_i = \sum_{j \neq i} [\mathbf{f}_{ij}^{\text{soc}} + \mathbf{f}_{ij}^{\text{att}} + \mathbf{f}_{ij}^{\text{ph}}] + \sum_b [\mathbf{f}_{ib}^{\text{soc}} + \mathbf{f}_{ib}^{\text{ph}}] + \sum_c \mathbf{f}_{ic}^{\text{att}} \tag{6.2}$$

The first term on the right-hand side of (6.2) includes all the forces on agent i resulting from interactions with other agents: $\mathbf{f}_{ij}^{\text{soc}}$ is the repulsive social force that inhibits strangers from getting too close, $\mathbf{f}_{ij}^{\text{att}}$ is the attractive social force that, e.g., brings friends together, $\mathbf{f}_{ij}^{\text{ph}}$ is the physical force that exists when two people come into contact. The second term includes the forces acting on agent i as a result of fixed environmental obstacles (e.g., walls): $\mathbf{f}_{ib}^{\text{soc}}$ is the social force that inhibits agent i from getting too close to the boundaries, $\mathbf{f}_{ib}^{\text{ph}}$ is the physical force that exists when agent i touches the boundary b. Finally, $\mathbf{f}_{ic}^{\text{att}}$ is the attractive social force that draws agent i towards fixed objects of incidental interest (shops, cafés, toilets, etc.).

$\mathbf{f}$ is principally a function of the distance between an agent and the other objects in the model. d_{ib} is the minimum distance between the circumference of agent i and fixed object b. d_{ij} is the distance between the centres of mass of agents i and j, i.e., the centres of the discs, while $r_{ij} = r_i + r_j$ is the "touching distance". To aid modelling the different force regimes that exist when agents are not in contact and when they touch (i.e. agents i and j touch if $r_{ij} - d_{ij} \leq 0$) we adopt the function $\Theta(r_{ij}, d_{ij}) = \max(0, r_{ij} - d_{ij})$.

Using these notions, the various repulsive social and physical forces of (6.2)) are defined as follows:

$$\mathbf{f}_{ij}^{\text{soc}} = \{A_i \exp[(r_{ij} - d_{ij})/B_i]\}\, \mathbf{n}_{ij} \Lambda(\lambda_i, \varphi_{ij}) \tag{6.3}$$

$$\mathbf{f}_{ij}^{\text{ph}} = k_1 \Theta(r_{ij} - d_{ij}) \mathbf{n}_{ij} + k_2 \Theta(r_{ij} - d_{ij}) \Delta v_{ji}^t \mathbf{t}_{ij} \tag{6.4}$$

$$\mathbf{f}_{ib}^{\text{soc}} = \{A_i \exp[(r_i - d_{ib})/B_i] + k_1 \Theta(r_i - d_{ib})\}\, \mathbf{n}_{ib} \tag{6.5}$$

$$\mathbf{f}_{ib}^{\text{ph}} = -k_2 \Theta(r_i - d_{ib})(\mathbf{v}_i \cdot \mathbf{t}_{ib}) \mathbf{t}_{ib} \tag{6.6}$$

$\mathbf{n}_{ij}$ ($\mathbf{n}_{ib}$) is a normalised vector pointing from agent j (fixed object b) to agent i, i.e., the direction of the repulsive force. $\mathbf{t}_{ij}$ ($\mathbf{t}_{ib}$) is a normalised vector tangential to the relative movement of agent i and agent j (fixed obstacle b), i.e., the motion tangential direction. $\Delta v_{ji}^t = (\mathbf{v}_j - \mathbf{v}_i) \cdot \mathbf{t}_{ij}$ is the tangential velocity difference. The social forces (6.3) and (6.5) increase exponentially with reducing distance between objects, with a scale defined by constants A_i and B_i. In particular, A_i is the force acting on agent i at the touching distance; B_i is loosely the distance at which the force takes effect.

$\Lambda : \mathbb{R}^2 \mapsto [0, 1]$ is a function that gives greater weight to the social force (6.3) arising from the agents in front of (notionally, *seen* by) an agent. λ_i is a parameter that regulates the effect of Λ on agent i, while φ_{ij} is the angle between the directions $\mathbf{e}_i^0$ and $-\mathbf{n}_{ij}$, i.e., the field of view of the agent. The physical force (6.4) between agents comprises a repulsive body compression force (first term) that acts in direction $\mathbf{n}_{ij}$, plus a frictional force (second term) that acts in direction $\mathbf{t}_{ij}$ to impede the relative tangential movement of two agents in contact. k_1 and k_2 are constants that define the scale of the physical

forces. The physical force (6.6) between an agent and a fixed object is solely described by a frictional term.

6.3 Proxemic Theory

Proxemic theory [18] has been developed by Edward T. Hall in 1962 and relates human psychology with non verbal communication, iconic communication and the use of personal space and territory.

According to Hall's theory there are four different territories:

1. "Body territory" is the personal space, represented as a bubble carried around the person.

2. "Primary territory" is the living space, such as one's home or a car.

3. "Secondary territory" refers to structured places where access is exclusive to some individuals and certain rules are expected, for example an office or a school.

4. "Public territory" is related to areas that anyone can access freely, such as shopping malls.

Basically, territories are a way for protecting their owners' comfort from undesirable people. Moreover, territories can overlap depending on the individuals. For example, some friends might organize a dinner at home. For the homeowner the home is a primary territory, while for the others it is a secondary territory.

In this work we focus on the body territory. The bubble around the individual's body is divided into a number of concentric circles where the nearest areas are reserved for trusted people.

Hall identifies four important zones visible in Figure 6.1:

1. "Intimate" for family and close friends. Varies from touching to 0.5 meters.

2. "Personal" for conversations with friends. Varies from 0.5 meters to 1.2 meters.

3. "Social" for formal conversations. Varies from 1.2 meters to 3.7 meters.

4. "Public" for addressing groups of people. Varies from 3.7 meters to 7.6 meters

The actual size of the zones is strictly dependent on the cultural and personal aspects of the person. For example, the "personal" zone in Middle East is generally much more closer than in Europe. As opposed to Japan, where usually distances are larger.

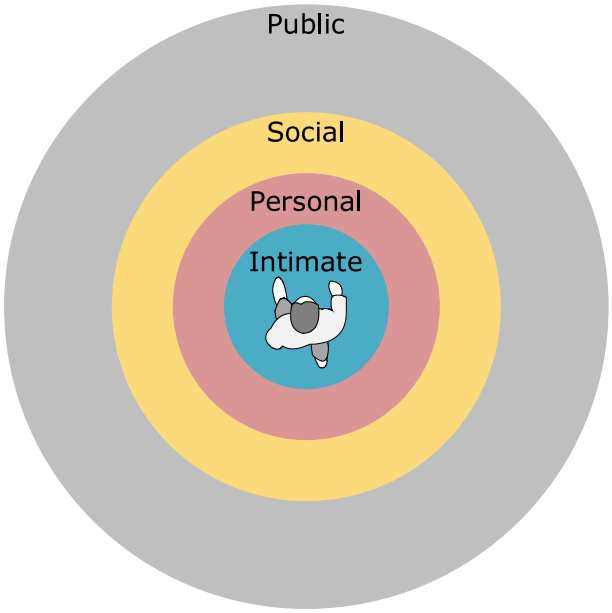
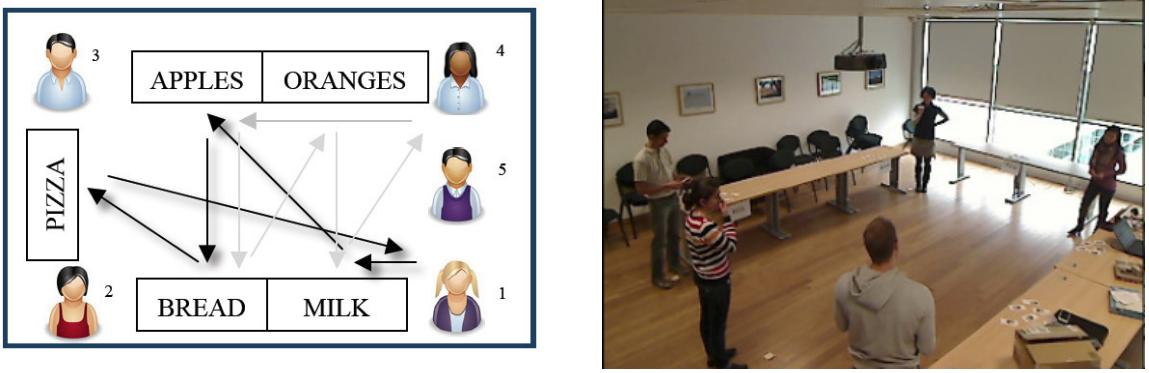


Figure 6.1: Proxemics zones of the body territory defined by Hall's theory. The dimension of each zone strongly depends on cultural and personal aspects of the individual but they usually range from 0.5 meters for the intimate space to the 7.6 meters of the public space.

6.4 Qualitative studies

A qualitative user study was conducted to collect information about how people behave in crowded environments. We considered three main themes: (i) how people interact in confined environments; (ii) how people negotiate shared space and (iii) how people behave with other people in shared space. A simplified shopping context was constructed in a laboratory, as illustrated in Figure 6.2. Our experimental procedure involved the use of a concurrent protocol, video-recorded observation and questionnaire administration. These techniques have been found to be particularly effective when conducting experimental investigations [85, 86]. The concurrent or talk-aloud protocol - a narration of thought and action during action - was chosen as literature suggests the alternative retrospective protocol (where participants return to view and comment upon their recorded experience) may not accurately reveal participants actual task performance experience. Concurrent protocol participants have been found to go into greater detail and provide more in-depth evaluations [87].



(a) Diagrammatic scheme of participants and shops showing potential trajectories.

(b) Video grab of laboratory set-up.

Figure 6.2: Experimental set-up.

6.4.1 Procedure

The participant sample consisted of 25 University students. Five participants at a time were involved in each experiment, and assigned a different list of four shopping items to collect. They were asked to collect all items in the shortest time possible, verbally explaining all their actions. Participants were specifically arranged around the shopping environment and each of the five shopping lists were unique, to maximise shared space interaction and limit the possibility of processional behaviour. The potential route-behaviour of participants 1 and 4 is illustrated in Figure 6.2(a). Participants were asked to collect the items on their list and return to their starting points, while simultaneously verbalising their shopping experience; describing what they saw, where they went, and what they were thinking and doing. The voice-recording functionality of five HUAWEI U8650 Android mobile phones was utilised to capture the speech of the participants. Three Logitech Quickcam Pro 9000 webcams were used to observe and record shared space interaction within the simulated shopping environment. The cameras were coordinated to provide a more encompassing view of the interaction space (Figure 6.2(b)). On completion of the experiment, the participants were asked to complete an open questionnaire.

6.4.2 Data analysis

The transcribed concurrent protocols, once synchronized with the video footage and in conjunction with the development of relevant coding schemes, facilitated assessment of a number of variables, including Task Completion Time, Number of Steps taken, and Number of Critical Instances - instances where physical, visual and/or auditory reference

was made within the verbal protocols to agent-agent interaction. Detailed video analysis was conducted via the Elan software tool [88].

6.4.3 Results

Using the Critical Instances markers from the transcribed protocols synchronised to the video footage, we documented and studied the behaviour exhibited by participants when involved in shared space interaction. Critical instances were defined as physical, visual and/or auditory references within the verbal protocols to agent-agent awareness or interaction. Analysis of the identified critical instances revealed a number of common themes and behaviours. The themes that emerged indicated that the behaviour itself was usually employed to either negotiate shared space interaction or avoid collisions within the shared space, and it was possible to categorise the main types of behaviours in two groups: Active and Reactive behaviours.

Active behaviours, in this instance, were considered to be behaviours employed to understand the environment and determine goal strategies toward task completion, environmental awareness and negotiation, mainly focussing around the visual modality:

(A) Eye-to-eye negotiation of immediate shared space;
(B) Use of peripheral vision in assessment;
(C) Visual scanning of environment;
(D) Verbal interaction.

Reactive behaviours were considered as the reactions of agents in the environment to other agents; the physical reactive movements made to accommodate other agents and successful interaction:

(E) Waiting for space/desired location to become clear;
(F) Stepping backwards to allow others more room;
(G) Moving forwards to allow others more room;
(H) Stationary agent yielding to moving agent;
(I) Move left;
(J) Move right.

Table 6.1 provides an example of how the critical instances were interpreted in terms of the taxonomy of behaviours observed during shared space interaction. By examining critical instances within the complete video footage, it was possible to observe the interactional behaviour occurring during such periods according to the taxonomy, and to determine the frequency with which these behaviours occurred (Figure 6.3). As can be observed in Figure 6.3, participants utilised the visual modality to engage in either scanning

Crit. Inst.	*Observed Behaviour of Participant (Px)*
1	Px aware of other agent in desired space, waits until agent completes task (behaviours (E), (H), (C))
2	Backward glance locates agent directly behind Px and helps avoid collision (behaviour (B))
3	Px aware of an agent in front and another approaching agent (behaviours (B), (C)) Px watches the actions of a moving agent and remains still until the agent passes (behaviours (E), (H))

Table 6.1: Critical instances interpreted in terms of the taxonomy of behaviours.

of the overall environment (usually to identify product locations within it), or peripheral vision to monitor the orientational movement of other agents in the environment. As proximity to others in the environment increased, eye-to-eye contact or negotiation between participants was observed to occur. Similarly, under these circumstances, verbal negotiation was also evident in a small number of cases, although this may have been reduced due to the method of recording the concurrent protocols. The most commonly occurring reactive behaviours according to the study were waiting for free space to become available, stationary agents giving priority to moving agents, moving to the left and right, and moving forwards and backwards to create free space, based upon the interactions occurring during the themes defined.

From the analysis of the videos it was evident that people planned their movement based on their physical distance from others, broadly in line with the assumptions of the SFM. However, we observed modes of behaviour that are not explicitly modelled by the SFM. When somebody came too close to another person, avoidance behaviours were manifested, such as waiting (for the space to be free) or stepping back. These behaviours were clustered in 5 main rules, as reported in Table 6.2. It is also interesting to note that when two active agents meet, the slower one typically gives way to the faster.

Our study highlights the fundamental role of proxemics in human motion. A literature exists concerning psychological aspects of spatial behaviour, taking into consideration concepts such as proximity, body orientation, motion in a physical setting, territorial behaviour and privacy. These concepts can provide a theoretical framework to extend the SFM, informing our understanding of people's behaviour in public spaces.

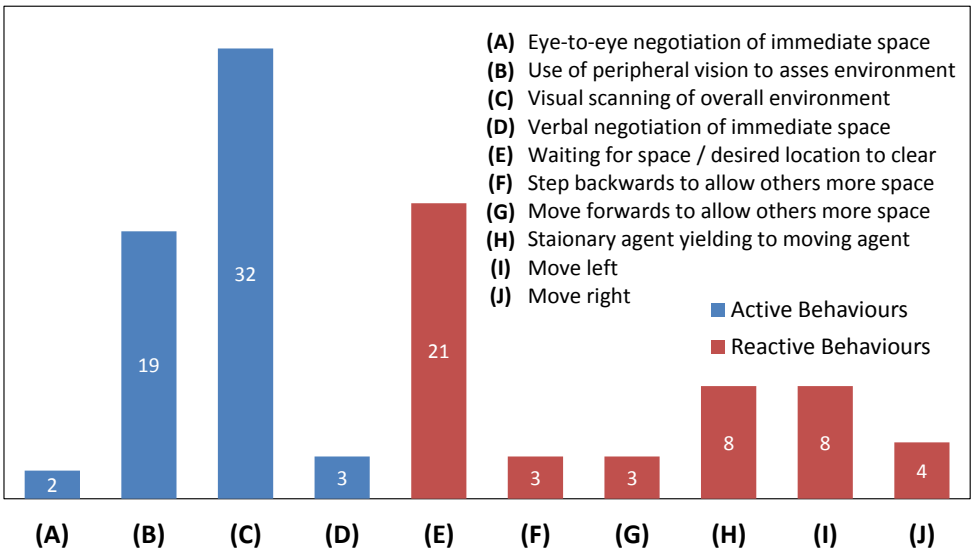


Figure 6.3: Distribution of behaviours.

Rule	*Agent 1*	*Agent 2*	*Behaviour*
1	Active	Passive	Agent 2 **steps back** and allows agent 1
2	Active	Passive	Agent 1 **moves** and shares space
3	Active	Passive	Agent 1 **waits** for empty space
4	Active	Active	Agent 1 **moves** left or agent 2 **moves** right
5	Active	Active	Agent 1 **waits** and gives way to agent 2

Table 6.2: Identified behavioural rules.

6.5 Parametrising the SFM

We conducted a number of motion tracking experiments using the simulated shopping environment described in Section 6.4. We performed 20 experiments considering two people in the environment and a further 20 experiments using four people. Participants were arranged in specific places around the shopping environment and asked to move according to a shopping list provided in advance. Each shopping list contained a set of places to be visited (via points) in a predefined order, to prompt interactions between participants and thus generate interesting social behaviours. Depending on the number of participants, shopping lists were specifically designed to maximise shared space interaction and limit the possibility of processional behaviour. We recorded video and 3D information of the participants' trajectories using a RGB-D camera.

6.5.1 Parameter Estimation

Once the experimental data were collected, we decomposed the participants' trajectories according to the SFM, considering situations where there were at least two agents that interacted, having simultaneously crossing trajectories. Since each participant had been asked to travel through a sequence of via points and to pause in each starting and ending position, it was possible to identify the relevant segments of their trajectories in the experimental data. By assuming that the driving velocity of agents is constant along a line joining the initial and desired positions (see Section 6.2) we were able to infer $\mathbf{v}^0$.

Due to the constraint imposed by the initial and final configurations, for the purposes of parameter estimation we construct a modified version of the SFM described in Section 6.2. We first consider the original model (6.1) and define $\mathbf{x}_i^0 = [x_i^0,\, y_i^0]^T$ and $\mathbf{v}_i^0 = [v_{x_i}^0,\, v_{y_i}^0]^T = [0,\, 0]^T$ to be the desired final position and velocity, respectively, of agent i. We thus define the Cartesian position error variables $\tilde{\mathbf{x}}_i = \mathbf{x}_i^0 - \mathbf{x}_i$, whose dynamics are given by $\dot{\tilde{\mathbf{x}}}_\mathbf{i} = -\mathbf{v}_\mathbf{i}$. Furthermore, we define the polar coordinates

$$\rho_i = \sqrt{\tilde{\mathbf{x}}_i^T \tilde{\mathbf{x}}_i}, \quad \alpha_i = \arctan\left(\frac{\tilde{y}_i}{\tilde{x}_i}\right), \tag{6.7}$$

which are respectively the distance and the orientation from the current to the desired position of agent i. We then find that the normalised direction $\mathbf{e}_i^0$ turns out to be

$$\mathbf{e}_i^0 = \begin{bmatrix} e_{x_i}^0 \\ e_{y_i}^0 \end{bmatrix} = \begin{bmatrix} \dfrac{\tilde{x}_i}{\sqrt{\tilde{\mathbf{x}}_i^T \tilde{\mathbf{x}}_i}} \\ \dfrac{\tilde{y}_i}{\sqrt{\tilde{\mathbf{x}}_i^T \tilde{\mathbf{x}}_i}} \end{bmatrix} = \begin{bmatrix} c_{\alpha_i} \\ s_{\alpha_i} \end{bmatrix}, \tag{6.8}$$

where we adopt the convention $c_{\alpha_i} = \cos(\alpha_i)$ and $s_{\alpha_i} = \sin(\alpha_i)$. Hence, the modified SFM is finally given by the dynamical system

$$\begin{bmatrix} \dot{\rho}_i \\ \dot{\alpha}_i \\ \dot{v}_{x_i} \\ \dot{v}_{y_i} \end{bmatrix} = \begin{bmatrix} -\left(c_{\alpha_i} v_{x_i} + s_{\alpha_i} v_{y_i}\right) \\ \dfrac{s_{\alpha_i} v_{x_i} - c_{\alpha_i} v_{y_i}}{\rho_i} \\ \dfrac{v_i^0 c_{\alpha_i} - v_{x_i}}{\tau_i} + \dfrac{f_{x_i} + \xi_{x_i}}{m_i} \\ \dfrac{v_i^0 s_{\alpha_i} - v_{y_i}}{\tau_i} + \dfrac{f_{y_i} + \xi_{y_i}}{m_i} \end{bmatrix}. \tag{6.9}$$

Using (6.7), (6.8) and (6.9) it is then possible to have an approximate description of

the participant accelerations, i.e.,

$$\frac{\mathbf{v}_i(t_{k+1}) - \mathbf{v}_i(t_k)}{t_{k+1} - t_k} \approx \frac{\mathbf{g}_i(t_k) + \mathbf{g}_i(t_{k+1})}{2} \tag{6.10}$$

where

$$\mathbf{g}_i(t_k) = \frac{v_i^0 \mathbf{e}_i^0(t_k) - \mathbf{v}_i(t_k)}{\tau_i} + \frac{\mathbf{f}_i(t_k) + \boldsymbol{\xi}_i(t_k)}{m_i}.$$

Note that calculating the mean of successive values of $\mathbf{g}_i(\cdot)$ gives a constant mean acceleration in the sampling period $t_{k+1} - t_k$.

Having set the desired speed of each agent to $v_i^0 = \max\{\|\mathbf{v}_i(t_k)\|\}$, we estimated values for parameters A_i and B_i in (6.3) and (6.5), and for parameter τ_i in (6.1). We note here here that in our experiments the agents never touch, so the constants k_1 and k_2 are not used and are not estimated. Moreover, the $\Lambda(\cdot)$ function is for the moment assumed to be equal to 1, without loss of generality, since the trajectories do not take into account interactions from behind. Finally, the mass of the participants m_i plays only the role of a weighting factor for the generated forces, hence it is of no relevance to the problem at hand and can be considered as known in advance and removed form the estimation process.

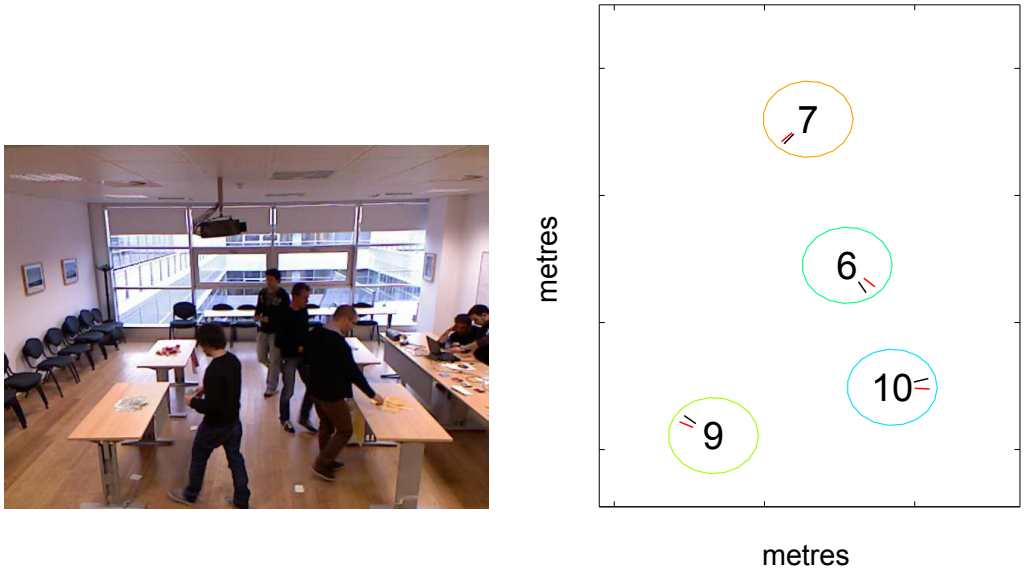
6.5.2 Estimation Algorithm

According to the description of the forces and the assumption that the model noise term $\boldsymbol{\xi}_i$ is normally distributed, we adopt an iterative Weighted Least Squares (WLS) algorithm to identify the parameters of the dynamical system. Such a choice is justified by the limited number of parameters involved in the estimation and by the relatively small amount of noise in determining the positions of the participants. This method is alternative to [89], in which a genetic algorithm based solely on video tracking data is used to estimate parameters. In our case, we make use of the high quality output of the RGB-D camera.

6.5.3 Results

Figure 6.4 illustrates typical instantaneous output from an experiment: Figure 6.4(a) shows a grabbed image of the simulated shopping environment; Figure 6.4(b) shows a plan view of the corresponding tracked participants. Each participant is represented by a disk of radius r_i – the same value used in (6.3), (6.4), (6.5) and (6.6) – and identified by a unique number for tracking purposes.

Figure 6.5 illustrates typical global trajectories, reconstructed from the tracked positions of the participants. The global trajectories were divided into local trajectories



(a) Video grab. (b) Tracked participants.

Figure 6.4: Instantaneous output of experiment.

describing the motion between the pre-defined via points. Even without time information, the locations of the via points are clear in Figure 6.5. The local trajectories were then used to estimate parameters of the SFM, to better "predict" the actual trajectories.

In Figure 6.6(a) the local trajectories of two participants going in opposite directions are represented by solid lines. The dashed lines represent the trajectories generated by the parametrised SFM. The figure demonstrates the good agreement of the SFM with reality in situations where only relatively small corrections are necessary and there is little conflict.

In Figure 6.6(b) the interaction is more conflictual and the negotiation of shared space increases in complexity. In this case our simple linear extrapolation to infer the participants' desired trajectories is not adequate and the SFM's prediction is poor. Interestingly, the point at which the actual and predicted trajectories diverge (identified by a circle in the figure) is the point at which one agent stops to give way to the other. We have found such active behaviour, i.e., moving to facilitate the motion of other agents or waiting for free space, to be most frequent in our experimental context (see Figure 6.3).

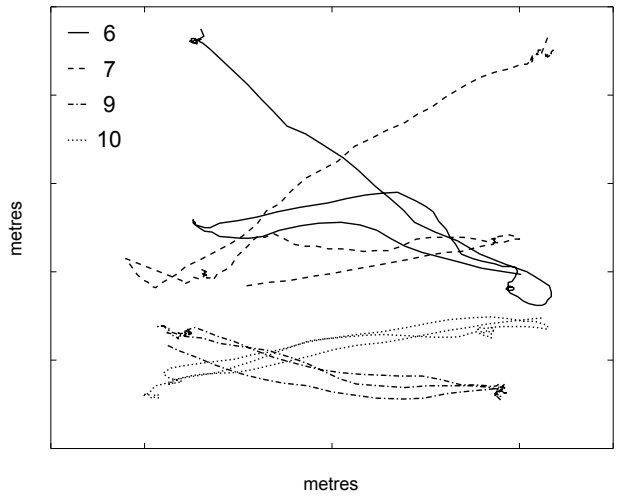
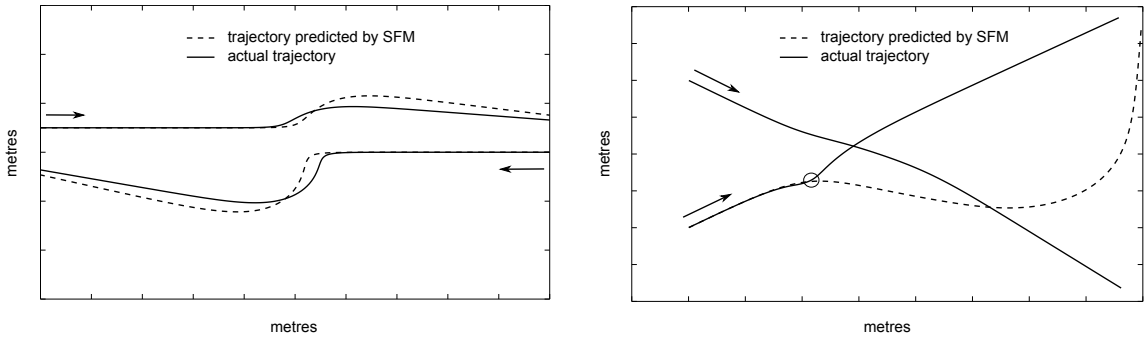


Figure 6.5: Trajectories corresponding to experiment shown in Figure 6.4.



(a) Good prediction of smooth flow in opposite directions

(b) Poor prediction of crossing paths

Figure 6.6: Performance of parametrised SFM.

Chapter 7

Experimental Evaluation

This chapter presents the evaluation of the proposed motion planner. A set of experiments have been carried out at our facilities, for both the *long term planner* (Chapter 4) and the *short term planner* (Chapter 5). The purpose of this chapter is to give a closer look to the practical implementation that complements the computer simulations presented in the previous sections. The *c-Walker* developed within the DALi project has been used and its technical aspects are briefly described in Section 7.1.

The people tracking algorithm was also of critical importance during the tests for the *short term planner* and it is described in Section 7.1.2. A Kalman filter has been designed to reduce the noise of its measurements, and is presented in Section 7.2.

Finally, the results are presented in Section 7.3 for the *long term planner*, and Section 7.4 for the *short term planner*.

7.1 Technical Aspects of the *c-Walker*

The *c-Walker* is a standard assistive walker instrumented with several sensors and actuators, as shown in Figure 7.1. It provides great flexibility and intelligence on-board. In particular, the embedded computer is used for running the both motion planning algorithm and the people tracker, while the touch-screen display is the human-machine interface, and the front RGB-D camera is the "eye" of the people tracker.

The embedded computer is an Intel NUC DC53427HYE, endowed with an Intel Core i5-3427U CPU and 8 GB DDR3 RAM and running a standard Linux Ubuntu 14.04.1. It is powered by an external rechargeable LiPo battery.

The front RGB-D camera is an Asus XtionPRO Live, self-powered via the USB interface.

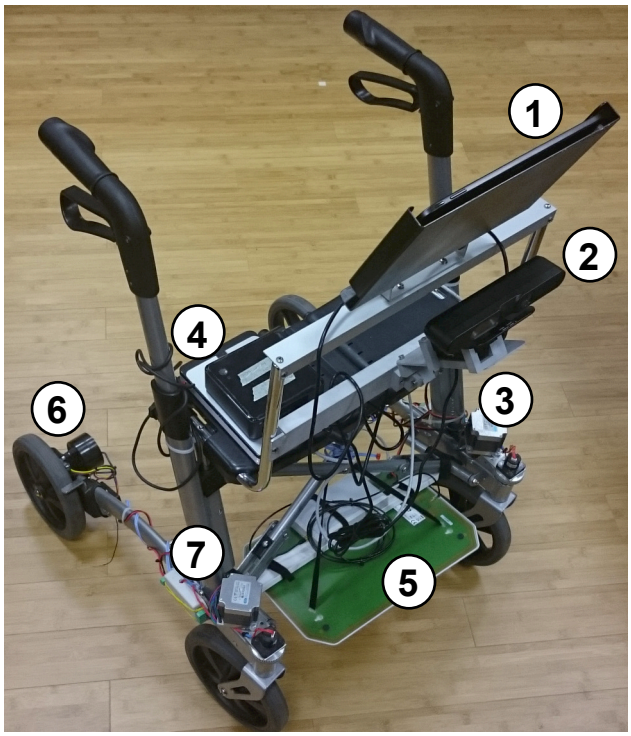


Figure 7.1: The *c-Walker* and its components. (1) touch-screen display, (2) front RGB-D camera for people tracking, (3) front camera for QR detection, (4) embedded computer and battery pack, (5) RFID reader and antenna, (6) electromechanical brake, and (7) stepper motor to turn front wheels.

7.1.1 Mechanical Guidance

The role of the guidance is to take a path generated by the cognitive engine and guide the user through its execution. The *c-Walker* integrates different types of guidance [90], such as passive (haptic or visual), and active (by brake or mechanical). In the experiments below we have used the mechanical guidance [91].

The mechanical guidance acts on the front wheel and actuates the stepper motors visible in Figure 7.1. The control algorithm implements the notion of virtual corridor. In pratice, the user is left free to navigate inside this corridor and the control action, that keeps the user in the middle of the corridor, becomes more and more authoritative as soon as the user gets closer to its border.

This particular guidance system has been chosen for the experiments of the *short term planner*, as discussed in Section 7.4. For the purpose of this validation, we have used a very narrow corridor of $\pm 10cm$ from the center of the path, which corresponds to steadily control the user over the planned path.

7.1.2 People Tracker

The approach for detecting and tracking humans in the surrounding of the *c-Walker*, presented by Panteleris et al. [92], is based on segmenting and tracking objects that move independently in the field of view of a moving RGB-D camera. The camera is allowed to move with 6 degrees of freedom (DOFs), while moving objects in the environment are assumed to move on a planar floor. This is the only a-priori information about the environment. Motion is estimated with respect to a coordinate system related to the static environment. In order to segment the static background from the moving foreground, the algorithm first selects a small number of points of interest whose 3D positions are estimated directly from the sensory information. The camera motion is computed by fitting those points to a progressively built model of the environment. A 3D point may not match the current version of the map either because it is a noise-contaminated observation, or because it belongs to a moving object, or because it belongs to a structure attached to the static environment that is observed for the first time. A classification mechanism is used to perform this disambiguation. Based on its output, noise is filtered, points on independently moving objects are grouped to form moving object hypotheses and static points are integrated to the evolving map of the environment. Sample results obtained from the execution of the algorithm are shown in Figure 7.2.

Several experimental results demonstrate that their proposed method is able to track moving objects correctly. Interestingly, the performance of egomotion estimation and map construction practically remains unaffected by the presence of independently moving objects. From a computational point of view, the method works at a frame rate of 50 fps on a laptop with an Intel Core i7 CPU without the use of GPU acceleration, and can perform at near real-time speeds on ARM-based embedded platforms.

7.2 Filtering of the Tracked Trajectories

We implemented a Kalman filter (KF) [93] for compensating the noise of the people tracker. It estimates the position of tracked people by assuming they are moving with constant velocity in the environment. This is not a strong assumption because the sampling frequency of the people tracker is usually in the order of 20 - 30 Hz, at these frequencies the human motion can be assumed to be a piecewise-linear function [94]. The tracker is assumed to measure the 2D position $\bar{\mathbf{x}}_i(k) = [\bar{x}_i^x(k), \bar{x}_i^y(k)]^T$ of each person i within the range of the sensors. The sampling time at step k is computed as the time difference between the timestamps of messages k and $k-1$, that is $t_k - t_{k-1}$. The diagram is depicted in Figure 7.3.

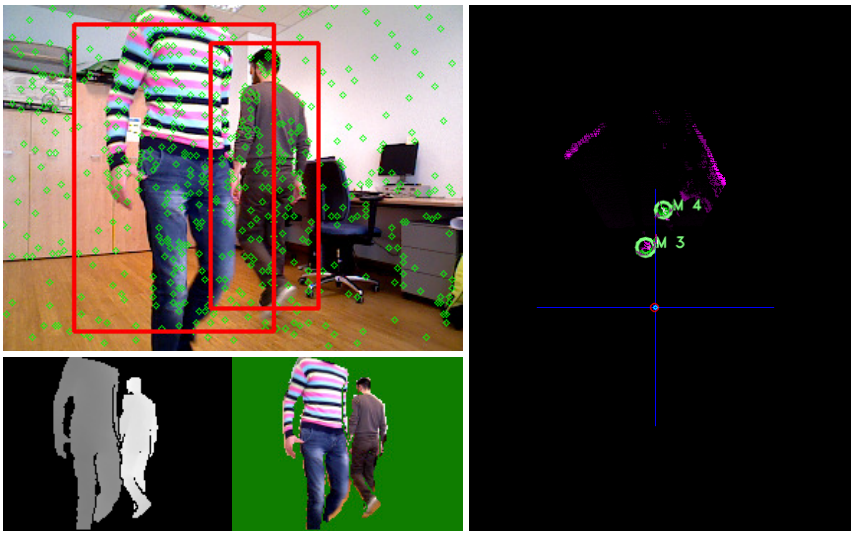


Figure 7.2: Snapshot of an execution of the people tracking algorithm. Bottom left: the depth and RGB images relative to independently moving people. Right: the top view of the local environment map showing the motion hypotheses for two people. Top left: the bounding boxes that identify the two moving people.

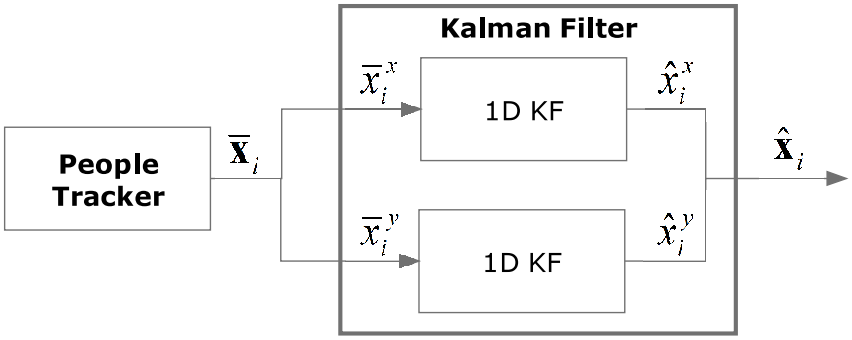


Figure 7.3: Diagram of the Kalman filter for position estimation.

7.2.1 Overview of the Kalman Filter

The standard linear Kalman filter assumes that the true state of the system $\mathbf{x}$ evolves from step k to step $k+1$ according to

$$\mathbf{x}(k+1) = \mathbf{F}(k)\mathbf{x}(k) + \mathbf{G}(k)\mathbf{u}(k) + \mathbf{w}(k) \tag{7.1}$$

where $\mathbf{F}(k)$ is the state transition matrix applied to the previous state $\mathbf{x}(k)$, $\mathbf{G}(k)$ is the control input matrix applied to the control vector $\mathbf{u}(k)$ and $\mathbf{w}(k)$ is the process noise assumed to be Gaussian distributed with zero mean and covariance $\mathbf{Q}(k)$. At some point

in time a measurement $\mathbf{z}(k+1)$ of the true state $\mathbf{x}(k+1)$ is observed according to

$$\mathbf{z}(k+1) = \mathbf{H}(k+1)\mathbf{x}(k+1) + \mathbf{v}(k+1) \tag{7.2}$$

where $\mathbf{H}(k+1)$ is the measurement matrix mapping the state space into the measurement space and $\mathbf{v}(k+1)$ is the measurement noise, assumed to be white Gaussian distributed with zero mean and covariance $\mathbf{R}(k+1)$.

The Kalman filter is an iterative observer. By convenience the equations have been split into "predict" and "update" phases that, commonly, are executed in succession. The "predict" phase is used to propagate the state, while the "update" phase enters into play when a measurement of the state is available. The "predict" equations are defined as follows

$$\begin{aligned} \hat{\mathbf{x}}^{+}(k+1) &= \mathbf{F}(k)\hat{\mathbf{x}}(k) + \mathbf{G}(k)\mathbf{u}(k) \\ \mathbf{P}^{+}(k+1) &= \mathbf{F}(k)\mathbf{P}(k)\mathbf{F}(k)^{\mathrm{T}} + \mathbf{Q}(k) \end{aligned} \tag{7.3}$$

where $\hat{\mathbf{x}}^{+}(k+1)$ is the *a priori* state estimate, $\hat{\mathbf{x}}(k)$ is the *a posteriori* state estimate of the previous step and $\mathbf{P}^{+}(k+1)$ is the *a priori* error covariance matrix that measures the accuracy of the state estimate.

The "update" equations are defined as

$$\begin{aligned} \tilde{\mathbf{y}}(k+1) &= \mathbf{z}(k+1) - \mathbf{H}(k+1)\hat{\mathbf{x}}^{+}(k+1) \\ \mathbf{S}(k+1) &= \mathbf{H}(k+1)\mathbf{P}^{+}(k+1)\mathbf{H}(k+1)^{T} + \mathbf{R}(k+1) \\ \mathbf{K}(k+1) &= \mathbf{P}^{+}(k+1)\mathbf{H}(k+1)^{T}\mathbf{S}(k+1)^{-1} \\ \hat{\mathbf{x}}(k+1) &= \hat{\mathbf{x}}^{+}(k+1) + \mathbf{K}(k+1)\tilde{\mathbf{y}}(k+1) \\ \mathbf{P}(k+1) &= [I - \mathbf{K}(k+1)\mathbf{H}(k+1)]\mathbf{P}^{+}(k+1) \end{aligned} \tag{7.4}$$

where is $\tilde{\mathbf{y}}(k+1)$ the innovation, $\mathbf{S}(k+1)$ is the innovation covariance, $\mathbf{K}(k+1)$ is the Kalman gain, $\hat{\mathbf{x}}(k+1)$ is the updated state estimate and $\mathbf{P}(k+1)$ is the updated error covariance matrix.

7.2.2 Kalman Filter for Position Estimation

The motion model for person i is a standard model with constant velocity, where the movements along the two axes is supposed to be independent with each other

$$\begin{cases} \dot{\mathbf{x}}_i = \mathbf{v}_i \\ \dot{\mathbf{v}}_i = \tilde{\mathbf{w}} \end{cases}$$

where $\tilde{\mathbf{w}} = [\tilde{\mathrm{w}}^x, \tilde{\mathrm{w}}^y]^T$ is an exogenous input that models the independent time variation of the velocity. Since we do not have any knowledge about the time evolution of the velocity, the input is modeled as a bivariate random variable with zero mean and constant variances, hence the velocity is modeled as a random walk. The associated covariance matrix is $\mathbf{W} = \mathrm{diag}([\sigma^2_{\tilde{\mathrm{w}}^x}, \sigma^2_{\tilde{\mathrm{w}}^y}])$, where diag(.) creates a diagonal matrix from its vector argument, and $\sigma^2_{\tilde{\mathrm{w}}^x}$ and $\sigma^2_{\tilde{\mathrm{w}}^y}$ are the variances of $\tilde{\mathrm{w}}^x$ and $\tilde{\mathrm{w}}^y$ respectively.

The discretised model is defined by Equations (7.1) and (7.2) and the system matrices are defined below. The KF estimates $\hat{\mathbf{x}}_i(k) = [\hat{x}^x_i(k), \hat{x}^y_i(k)]^T$, that is the 2D position of person i. In the following, we describe the KF for estimating $\hat{x}^x_i(k)$, the one for $\hat{x}^y_i(k)$ is identical.

The KF equations for predict and update are defined in (7.3) and (7.4) respectively, where

$$\mathbf{F}(k) = \begin{bmatrix} 1 & (t_{k+1} - t_k) \\ 0 & 1 \end{bmatrix},$$

$$\mathbf{H}(k+1) = \mathbf{H} = \begin{bmatrix} 1 & 0 \end{bmatrix},$$

$$\mathbf{z}(k+1) = \bar{x}^x_i(k+1),$$

and finally matrix $\mathbf{G}$ is not needed because there are no inputs $\mathbf{u}$ in this model. Recall from (7.2) that $\mathbf{v}(k+1)$ is the measurement noise associated to $\mathbf{z}(k+1)$, and is assumed to be zero mean Gaussian white noise with covariance matrix $\mathbf{R}(k+1)$ that is obtained experimentally from the people tracker.

The discrete-time process noise $\mathrm{w}^x(k)$ relates to the continuous-time noise $\tilde{\mathrm{w}}^x(t)$ as

$$\mathrm{w}^x(t_k) = \int_{t_k}^{t_{k+1}} e^{(t_{k+1}-\tau)\mathbf{A}} \mathbf{B} \tilde{\mathrm{w}}^x(\tau) d\tau \equiv \mathrm{w}^x(k)$$

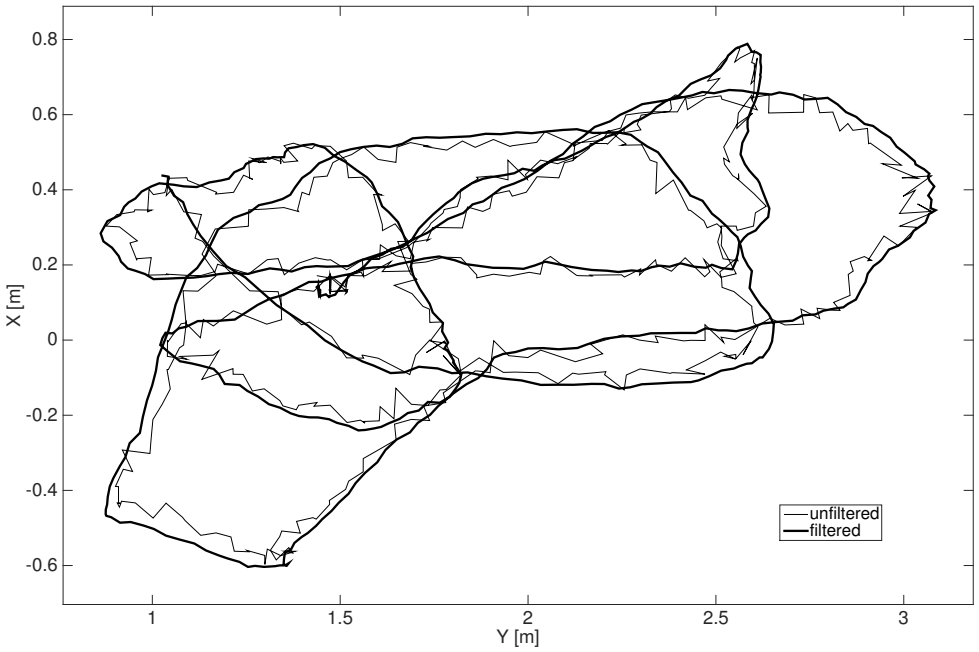


Figure 7.4: Comparison between unfiltered (thin line) and filtered position (thick line), output of one experiment.

where $\mathbf{B} = [0, 1]^T$ and $\mathbf{A}$ is the system matrix of the continuous model, indeed

$$e^{(t_{k+1}-t_k)\mathbf{A}} \equiv \mathbf{F}(k)$$

Since the process noise is assumed to be Gaussian distributed with zero mean and white, it follows that

$$\begin{aligned} E[\mathrm{w}^x(k)] &= 0 \\ E[\mathrm{w}^x(k)\mathrm{w}^x(j)] &= Q(k)\delta_{kj} \end{aligned}$$

where δ_{kj} is the Kronecker delta function. The variance $Q(k)$ is thus given by

$$Q(k) = \int_{t_k}^{t_{k+1}} e^{(t_{k+1}-\tau)\mathbf{A}}\mathbf{B}\sigma^2_{\tilde{\mathrm{w}}^x}(\tau)\mathbf{B}^T e^{(t_{k+1}-\tau)\mathbf{A}^T} d\tau$$

An example comparison between filtered and unfiltered position for one experiment is depicted in Figure 7.4. A similar comparison but showing the low-pass filtering effect of the Kalman algorithm on the speed of the same experiments is visible in Figure 7.5.

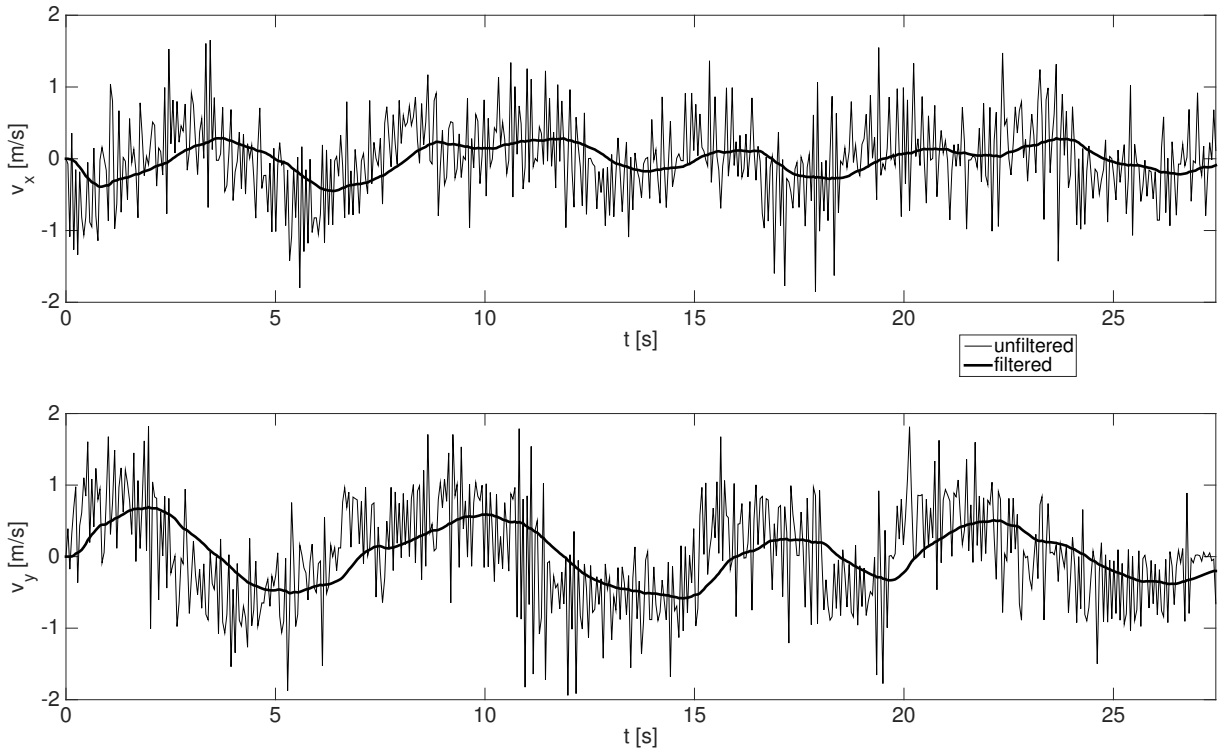


Figure 7.5: Comparison between unfiltered (thin line) and filtered (thick line) speed along x and y axis, output of one experiment.

7.3 Long Term Planner

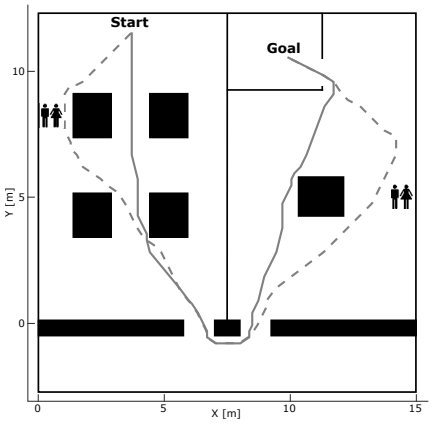
In October 2014 we ran an experimental campaign that involved several elderly people at our facilities. The goal was to test the functionalities of the walker as well as of the *long term planner*. To this end, we created a simulated shopping mall environment and recruited a cohort of 12 senior users.

We asked each participant to choose a destination in the environment (Figure 7.7(a)) and then follow the suggestions of the guidance of the walker.

At the end of each test we collected results on the participant's performance and asked her to answer some questions about the quality of the guidance, suggestions and her personal satisfaction.

In addition, we selected a group of caregivers working in protected residences and proposed to each of them a tour through the functionalities of the system, where each of them could define hard and soft constraints and test the system. During each test we randomly triggered anomalies (Figures 7.7(b) and 7.7(c)) and heat maps 7.7(d) to show the reactions of the system to such conditions. At the end of the field test, we collected informal opinions and suggestions.

(a) A senior user interacting with the *c-Walker* during a trial



(b) Map of the mall and two example of shortest paths, with and without "toilet" constraint

Figure 7.6: The simulated shopping mall created for the DALi experiments.

The impression we derived from reading the questionnaires collected from the users and from talking to the care givers was of a general interest and appreciation toward the system and its functionalities (including the *long term planner*). Most users are keen on being actively engaged with future development activities. This motivates us in pursuing this line of research in the upcoming years.

7.4 Short Term Planner

We performed a set of qualitative experiments in our laboratories for testing the algorithm on a real robotic platform. We used the *c-Walker* described in Section 7.1, and the *short term planner* has been linked to the mechanical guidance (Section 7.1.1) following the diagram reported in Figure 2.1.

The *short term planner* algorithm has been parametrised as follows. The replanning period $T_{decision} = 0.5$, the prediction horizon $T_{horizon} = 8$, and the number of simulations $N = 50$ for each alternative direction α_{curr}. These parameters are explained in Section 5.4.

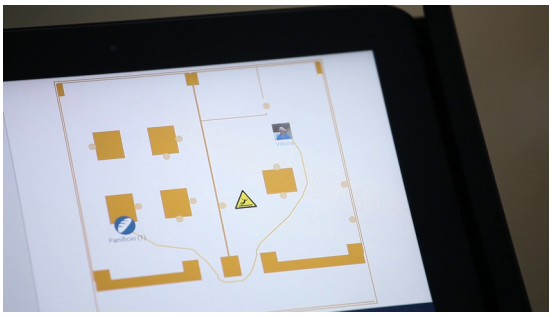
The proposed scenarios are described in the following paragraphs and involve interaction with one person (Scenarios 1, 2 and 3) and two people (Scenarios 4 and 5).

Scenario 1. The user's long term plan is a straight line (blue-coloured line in Figure 7.8(e)) and a person is walking towards the *c-Walker* (Figure 7.8(a)). As soon as she is detected by the people tracker (Figure 7.8(d)), the *short term planner* reacts and suggests the user a path that avoids the obstacle (Figure 7.8(f)).

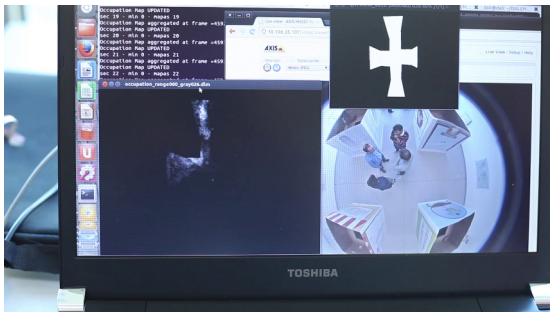
The trajectories followed by the user and by the agents are shown in Figure 7.8(g).

(a) Tablet interface for the *long term planner*

(b) Wet floor sign



(c) The *long term planner* reacts to the wet floor sign



(d) Heat map

Figure 7.7: Various pictures from the experimental campaign.

Scenario 2. The user is following the long term plan represented by the blue-coloured line in Figure 7.9(e)) and a person is crossing his trajectory (Figure 7.9(a)). Once he is recognised by the people tracker (Figure 7.9(d)), the *short term planner* detects that the agent won't interfere with the user's trajectory and it does not change the short term trajectory (Figure 7.9(f)).

The paths followed by the user and by the agent are shown in Figure 7.9(g).

Scenario 3. The goal of the user is to follow the long term plan represented by the blue-coloured line in Figure 7.10(e)) while an agent overtakes him on his left (Figure 7.10(a)). As soon as he is recognised by the people tracker (Figure 7.10(d)), the *short term planner* detects that the agent won't hinder the user (Figure 7.10(f)).

The routes followed by the user and by the agent are shown in Figure 7.10(g).

Scenario 4. The long term plan that the user is following is shown in Figure 7.11(e)) and is represented by the blue-coloured line. In the meanwhile, two agents are walking towards the user (Figure 7.11(a)). Once the agents are detected by the people tracker (Figure 7.11(d)), the *short term planner* reacts and suggests the user a safer short term plan that bypasses the obstacles (Figure 7.11(f)).

The trajectories followed by the user and by the agents are shown in Figure 7.11(g).

Scenario 5. The user's long term plan is a straight line (blue-coloured line in Figure 7.12(f)). Two agents are walking towards the *c-Walker* (Figure 7.12(a)) and, as soon as they are detected by the people tracker (Figure 7.12(d)), the *short term planner* recognises that there is enough room for passing in between them (Figure 7.12(f)).

The paths followed by the user and by the agent are shown in Figure 7.12(g).

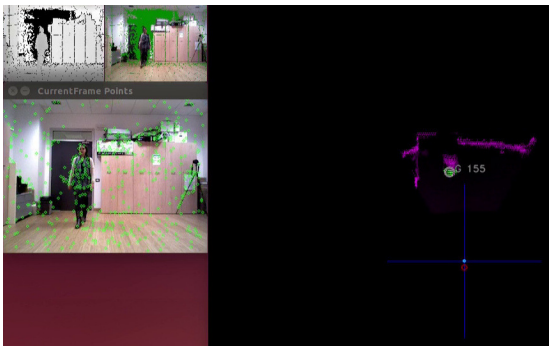
7.4.1 Considerations

The experiments reported in this section demonstrate on a realistic scenarios the effectiveness of the *short term planner*. It is clear that the selected scenarios are just a small subset of the several situations that may arise in reality. However, when the behaviour of people in the scene can be approximated to the chosen human motion model (i.e., the Social Force Model, Section 6.2), the proposed planning algorithm is able to enforce the requested probabilistic guarantees.

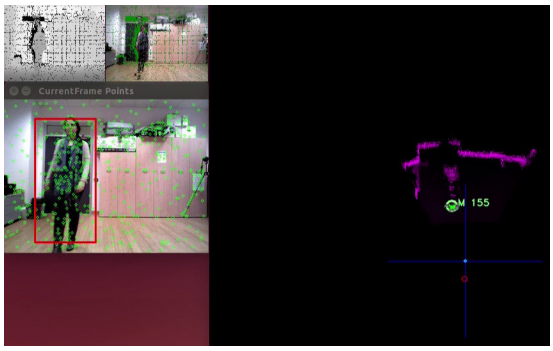
Real world performance of this approach are bounded by the sensing capabilities of the selected RGB-D camera that, in turn, limit the area covered by the people tracker to few square meters. This is evident from the short pieces of the surrounding people trajectories depicted in Figures 7.8 - 7.12, which lasts for an average of 1.5-2 seconds in view to the tracker. Nevertheless, a similar perception behaviour is used by humans beings [95], where closer obstacles have more importance than distant ones. In some respect, we can state that the perception subsystem, upon which the *short term planner* relies, mimic the human behaviour.

(a) Overall scenario at time $t = 2$

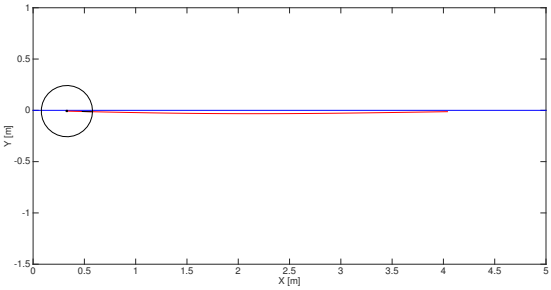
(b) Overall scenario at time $t = 4$



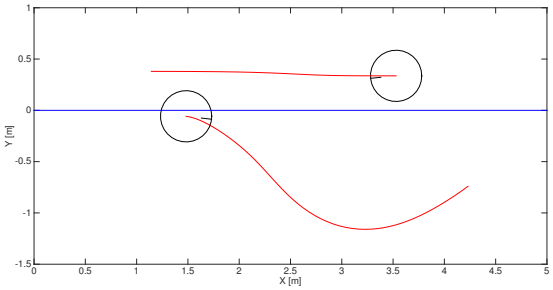
(c) People tracker at time $t = 2$



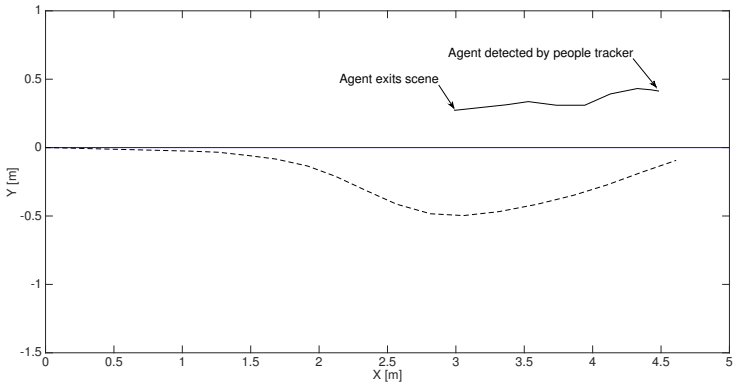
(d) People tracker at time $t = 4$



(e) The path suggested by *short term planner* at time $t = 2$

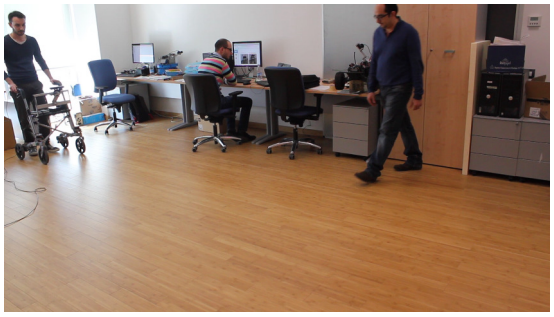


(f) The path suggested by *short term planner* at time $t = 4$



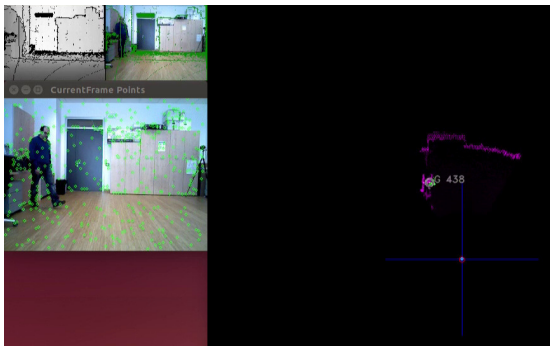
(g) The path followed by the user (dashed line) and by the agent (solid line)

Figure 7.8: Pictures from a run of scenario 1. Time in seconds.

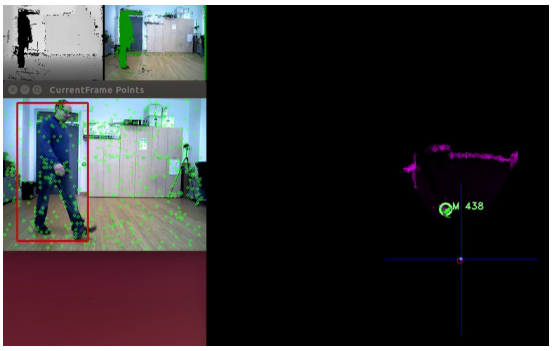


(a) Overall scenario at time $t = 0$

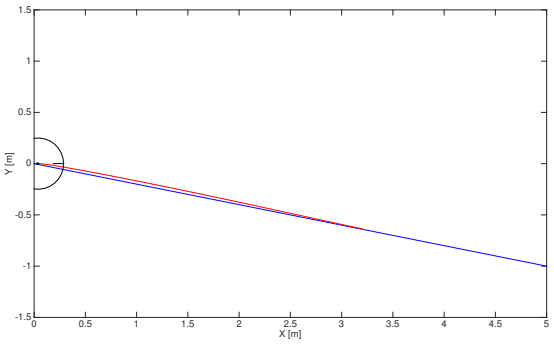
(b) Overall scenario at time $t = 2$



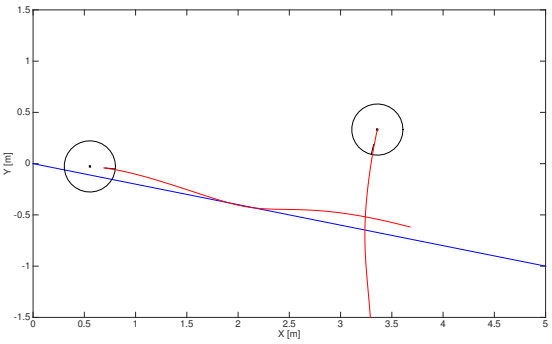
(c) People tracker at time $t = 0$



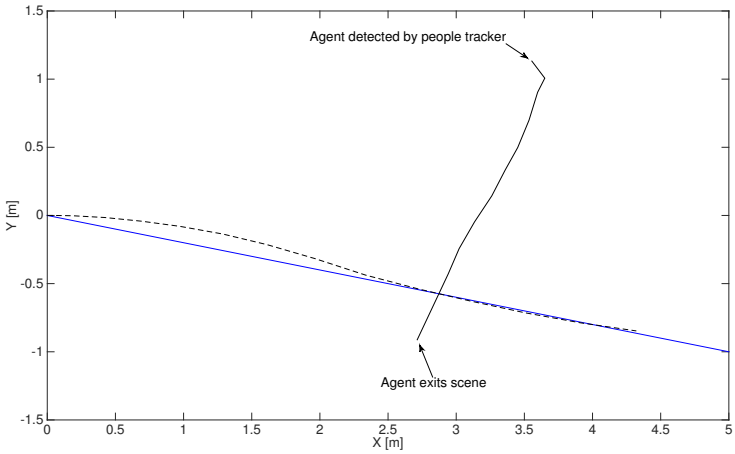
(d) People tracker at time $t = 2$



(e) The path suggested by *short term planner* at time $t = 0$



(f) The path suggested by *short term planner* at time $t = 2$

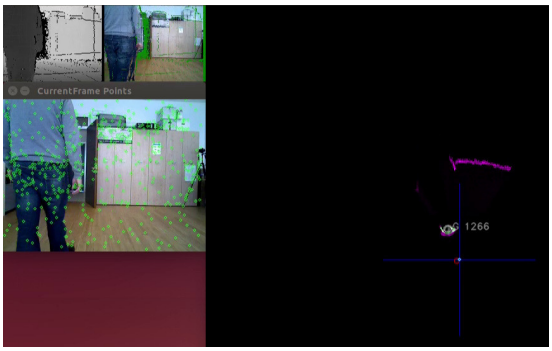


(g) The path followed by the user (dashed line) and by the agent (solid line)

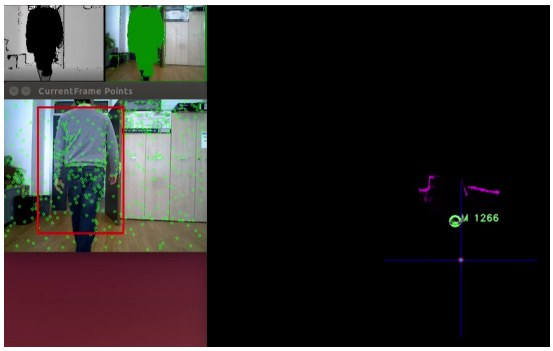
Figure 7.9: Pictures from a run of scenario 2. Time in seconds.

(a) Overall scenario at time $t = 1$

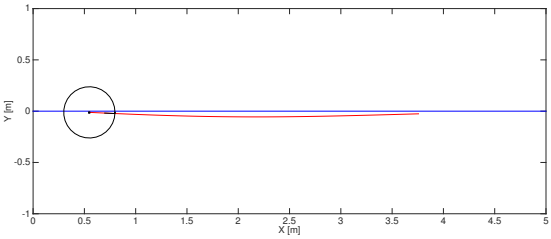
(b) Overall scenario at time $t = 3$



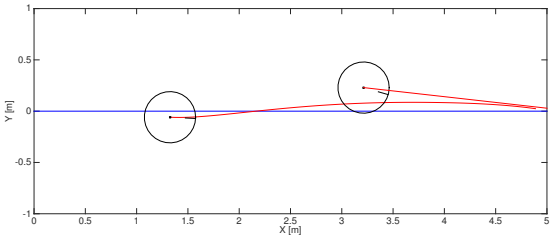
(c) People tracker at time $t = 1$



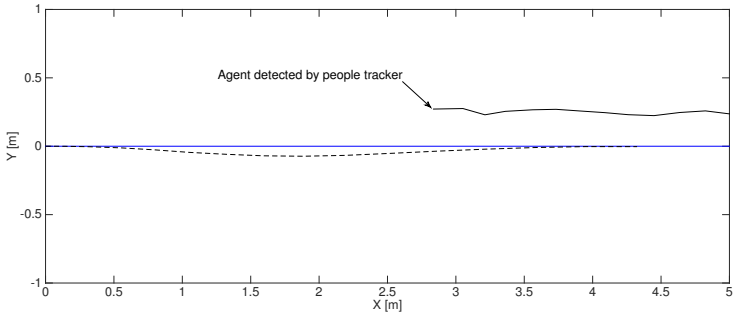
(d) People tracker at time $t = 3$



(e) The path suggested by *short term planner* at time $t = 1$

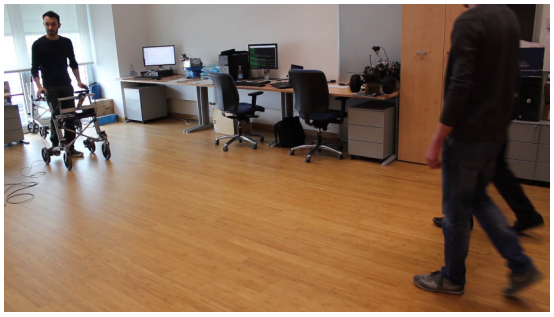


(f) The path suggested by *short term planner* at time $t = 3$



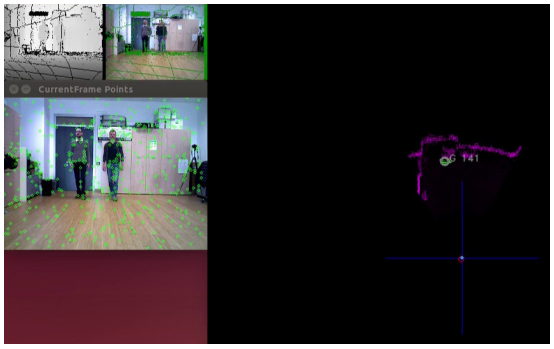
(g) The path followed by the user (dashed line) and by the agent (solid line)

Figure 7.10: Pictures from a run of scenario 3. Time in seconds.

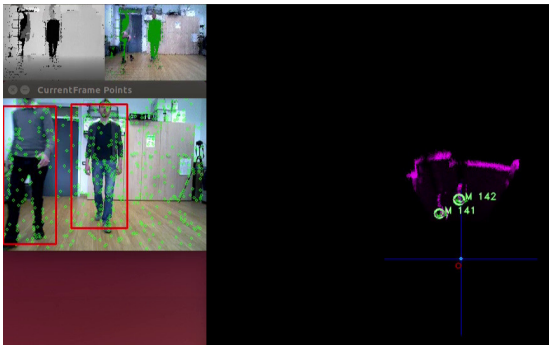


(a) Overall scenario at time $t = 0$

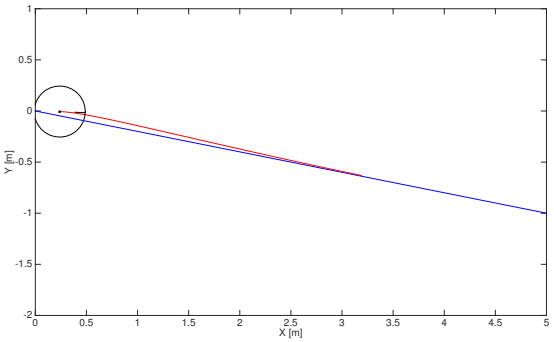
(b) Overall scenario at time $t = 2$



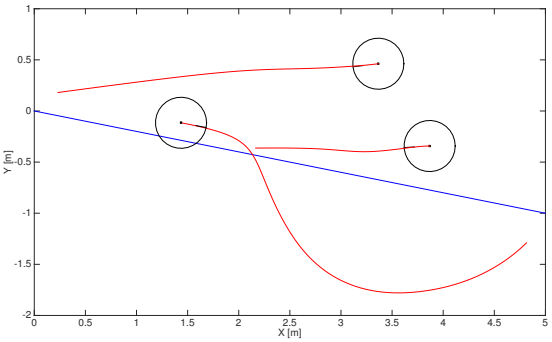
(c) People tracker at time $t = 0$



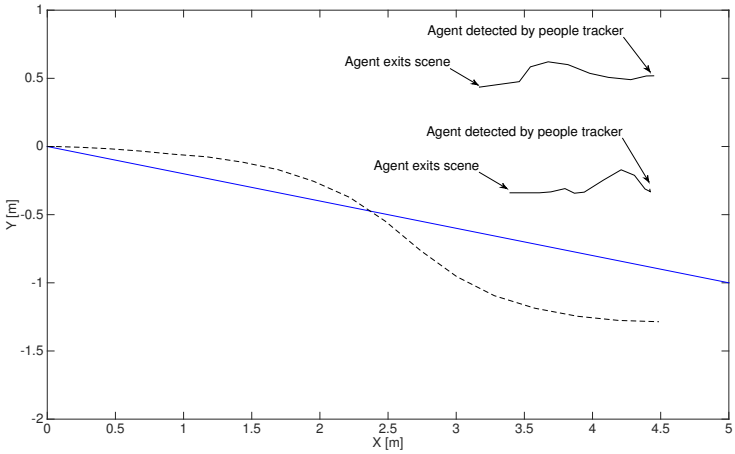
(d) People tracker at time $t = 2$



(e) The path suggested by *short term planner* at time $t = 0$

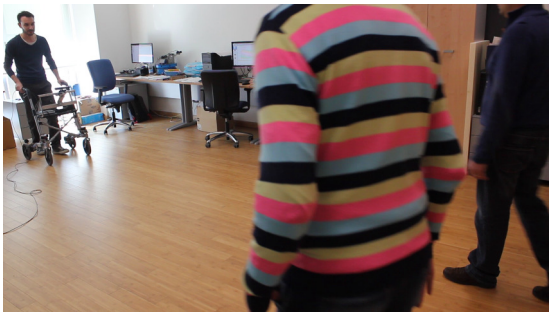


(f) The path suggested by *short term planner* at time $t = 2$



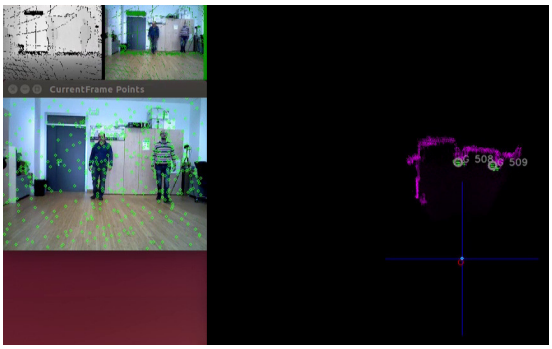
(g) The path followed by the user (dashed line) and by the agent (solid line)

Figure 7.11: Pictures from a run of scenario 4. Time in seconds.

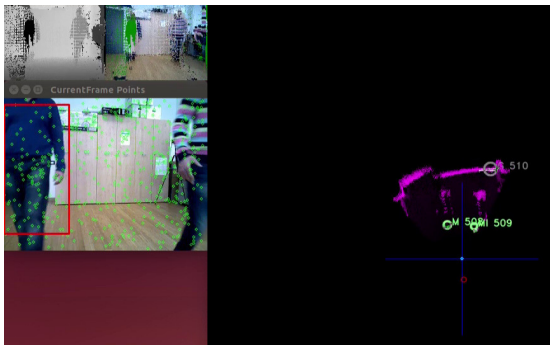


(a) Overall scenario at time $t = 2$

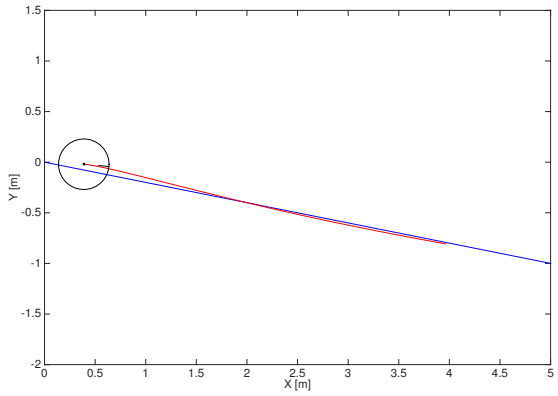
(b) Overall scenario at time $t = 5$



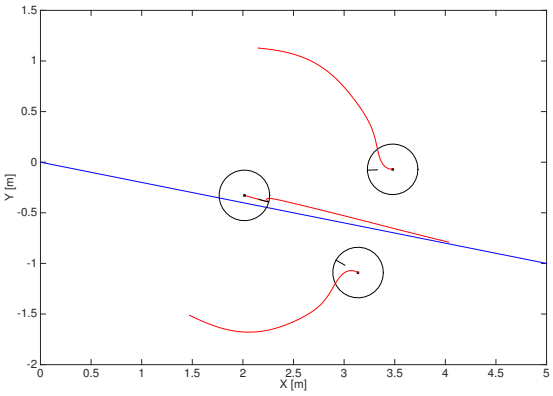
(c) People tracker at time $t = 2$



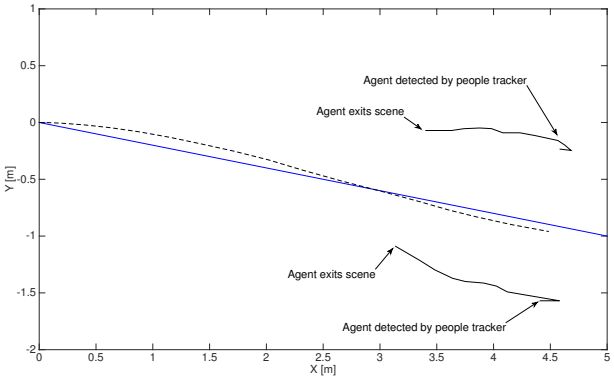
(d) People tracker at time $t = 5$



(e) The path suggested by *short term planner* at time $t = 2$



(f) The path suggested by *short term planner* at time $t = 5$



(g) The path followed by the user (dashed line) and by the agents (solid line)

Figure 7.12: Pictures from a run of scenario 5. Time in seconds.

Chapter 8

Conclusions and Future Work

In this chapter we draw the conclusions and the future directions of the presented work. We first discuss each chapter separately, and finally in Section 8.4 we report the overall conclusions.

8.1 Long Term Planner

We have presented an algorithm for long term motion planning in crowded public spaces. The algorithm applies to robotic platforms assisting the navigation of senior users in large and complex spaces. The key features of the algorithm are: 1. the ability to encode preferences in the user's profile regarding areas that should be avoided during the navigation and others that should be travelled across, 2. the consideration of time-dependent anomalies during the construction of the path, 3. the inclusion of crowdedness as a key parameter to consider when estimating the time to complete a path. Our idea is to use quad trees to generate a graph structure describing the place and encoding user preferences, anomalies and heat maps in the weight of the arcs. We propose a modified version of the Dijkstra algorithm to identify the optimal path that takes into account the time dependencies of the graph.

Our algorithm has been implemented as a cloud service that operates alongside a module for reactive (short term) planning and motion control, which are typically hosted on the robotic platform. Thanks to its flexible API and its low computational burden, the algorithm can be easily implemented in different ways, giving to the system integrators plenty of possibilities.

The different functionalities of the system have been validated in two ways. On one hand, we have tested it through simulated scenarios. On the other hand, we prepared a mockup simulating a realistic case study that we showcased to a group of users and caregivers.

This case study helped us to identify some borderline scenarios that require further analysis, especially when dealing with combinations of constraints. For example, when the user requires a "timed" constraint (e.g., "keep a toilet within 5 minutes walking distance"), when several constraints for desirable and undesirable zones appear to be placed one after the other, or when two or more constraints for desirable zones are placed at the opposite locations of an environment. Simulations have shown that combinations of contrasting requests can be managed efficiently, even though an extensive analysis of this behaviour has not been carried out on the field. Nonetheless, the simplicity and the robustness of the proposed solution is very promising for an efficient handling of such complex situations.

8.1.1 Future Work

Our future plans tackle several aspects. From the analysis of the borderline scenarios highlighted in the previous section, to the support of constraints and anomalies in a probabilistic framework, with the benefit of a higher personalisation level for the user.

Moreover, support of anomalies can be improved and made smarter by adding intermediate stops when constructing the plan (e.g, suppose that an anomaly expires 1 second after the user reaches the blockage, it may be worth waiting 1 second instead of replanning).

Finally, the ACANTO project extended the paradigm of assistance from a single user to a social dimension. The algorithm will be scaled accordingly, with motion plans organised for groups of people supported by a robotic platform.

8.2 Short Term Planner

We presented an efficient online motion planner for crowded environments. The position of most fixed objects (e.g., buildings and rooms) are known a priori, but the algorithm considers the possibility of changes, such as temporary obstructions. The environment contains moving objects (i.e., other pedestrians), whose positions and velocities cannot be known before they are encountered.

The overall goal is to allow the user to visit pre-defined locations in the environment, while avoiding collisions, crowding and delays. The output of the algorithm is a *suggested* trajectory, so the algorithm must be reactive to the potentially uncooperative response of the user.

The algorithm exploits a human motion model, parametrised in real time with data from sensors, for hypothesising future trajectories. The model's stochasticity allows us to consider a distribution of possible future evolutions.

The reactive part of the algorithm is provided by statistical model checking technology. The algorithm verifies the hypothesised trajectories against the user's goals and constraints expressed in temporal logic. This way the algorithm finds the immediate course of action that maximises the user's probability of success.

We validated the planner by means of simulations where the Social Force Model (Section 6.2) has been chosen as the human motion model. Practically, the algorithm has been implemented in a low power embedded computing device and is efficient enough to make course corrections in a time of the order of seconds. This time scale is dictated by the typical velocities of pedestrians and by the fact that frequent readings help to reduce the random errors produced by sensors.

Results showed that the algorithm is able to enforce the user's requirements as well as improve her comfort during navigation.

8.2.1 Future Work

There are three main aspects of our work that we want to improve: 1) human perception, 2) performance of the implementation, and 3) reuse of simulation traces.

Human perception. The apparently random behaviour of pedestrians is often the result of deterministic choices on their part. We want to improve the performance of our algorithm by recognising these choices and replacing some of the stochasticity. To this end, in Chapter 6 we have identified behavioural templates that may be incorporated into an improved human motion model.

Moreover, we propose to include advanced sensor techniques to recognise known interesting or hostile people (e.g., using facial recognition [96]). This information can be used later for generally avoiding people exhibiting hostile behaviour. Such information can be included in the user's goals and constraints, by encoding them into an extended temporal logic formula.

Performance of the implementation. A significant part of the challenge of our motion planning application is the performance of its implementation. Current hardware performance forces us to accept the necessity of multiple boards to handle the overall computational burden, but there is a clear advantage if a portable device can be made to work on a single board. The embedded computing boards we have chosen for our implementation include high performance graphical processor units (GPUs) that can be used for general purpose parallel computing. Since statistical model checking requires multiple independent simulation runs, we propose to exploit the GPU to gain a significant increase in performance.

Reuse of simulation traces. The replanning period $T_{decision}$ is necessarily less than the prediction time horizon $T_{horizon}$, hence the algorithm predicts traces in time periods that overlap from one iteration to the next. While the predictions of older simulations are likely to be less accurate compared to the current reality, data from the previous iterations may be employed, suitably weighted, to build a probabilistic map of the good and bad locations in the local environment. This map can be used to avoid simulations that explore directions that are unlikely to be successful and to provide feedback if the user chooses to diverge from the proposed path.

8.3 Identification of Human Motion Models

We tested and analysed the Social Force Model (SFM) in real scenarios. The unmodified model is accurate under conditions of continuous flow and when there is little competition for space. Under other circumstances, common in the social environments and that we are targeting, human motion is punctuated by frequent stops and starts, and short term changes of direction. Our experiments have demonstrated that these discrete events are not well captured by a simple SFM. We have nevertheless identified several distinguishable behavioural patterns that may be incorporated as templates into an enhanced SFM. In such a model, these patterns would be recognised and predicted, modifying the desired trajectories of modelled pedestrians. We have shown that this increases the predictive accuracy of the SFM.

We note that the patterns we have identified might be explained by a more general theory of human interaction. We therefore propose to explore proxemic theory (Section 6.3) and consider a larger range of social situations.

8.3.1 Future Work

We plan to resort to the broad field of machine learning to infer high level behaviors and usual motion patterns. One possibility is to use Interacting Multiple Model (IMM) [97], or HMM (Hidden Markov Model) inference, which is one of the most known approaches in literature for modeling uncertain and stochastic systems [98].

For instance, using HMM, the problem can be split in two sub-problems. At first, we identify the HMM ruling the behaviours of the humans in shared spaces, comparing the learned rules with the experimental evidence and proxemic theory. In the second stage, we identify the parameters governing the dynamic motion of the user in each learned behavioural pattern. This way, the dynamic is no longer given by the SFM per se but instead can be expressed with other solutions (e.g., a set of different Linear

Dynamic Systems, one for each learned behaviour, for which Kalman–based solutions can be applied [93]).

The key to our ongoing research will be an extensive collection of experimental data that can improve our theory and provide evidence of its applicability and robustness.

8.4 Overall Conclusions

The work presented in this dissertation proposes an efficient hierarchical motion planner for assistive robots in crowded environments. It is composed of a *long term planner* that considers long term objectives, and a *short term planner* that takes into account social rules, short term goals, and reacts to people detected along the way, motivating the need for an accurate human motion model.

We presented formally and validated each component separately. Finally, we performed qualitative experiments with the robotic platform developed within the DALi project. A Kalman filter has been designed for reducing the measurement noise of the the people tracking algorithm.

Results showed great interest by users and motivate us to continue improving this work. To this end, the purpose of the ACANTO project (the follow-up of the DALi project) is to start right from where this work finished, confirming the fact that robotic assistive platforms can definitely improve our elderly's quality of life and wellbeing.

8.4.1 Future Work

Our future plans include some aspects of this work. The Kalman filter for the people tracker (Section 7.2) can be improved by exploiting techniques similar to the ones anticipated in Section 8.3.1. The basic idea is to let evolve several simple linear models in parallel, and then pick the one that best fits the estimated trajectory.

The interaction between *long term planner* and *short term planner* can be also improved. For example, by iteratively smoothing the long term plan (i.e., similar to what a spline does to a piecewise-linear function) we could enforce both short term preferences and basic social rules (at least the ones related to fixed obstacles) before the user actually starts to walk.

Bibliography

[1] J. Van Cauwenberg, V. Van Holle, D. Simons, R. Deridder, P. Clarys, L. Goubert, J. Nasar, J. Salmon, I. De Bourdeaudhuij, B. Deforche, *et al.*, "Environmental factors influencing older adults' walking for transportation: a study using walk-along interviews," *Int J Behav Nutr Phys Act*, vol. 9, no. 1, p. 85, 2012.

[2] A. Kollmuss and J. Agyeman, "Mind the gap: why do people act environmentally and what are the barriers to pro-environmental behavior?," *Environmental education research*, vol. 8, no. 3, pp. 239–260, 2002.

[3] J. T. Cacioppo and L. C. Hawkley, "Perceived social isolation and cognition," *Trends in Cognitive Sciences*, vol. 13, no. 10, pp. 447 – 454, 2009.

[4] H. A. Yeom, C. Keller, and J. Fleury, "Interventions for promoting mobility in community-dwelling older adults," *Journal of the American Academy of Nurse Practitioners*, vol. 21, no. 2, pp. 95–100, 2009.

[5] J. Van Cauwenberg, V. Van Holle, D. Simons, R. Deridder, P. Clarys, L. Goubert, J. Nasar, J. Salmon, I. De Bourdeaudhuij, and B. Deforche, "Environmental factors influencing older adults' walking for transportation: a study using walk-along interviews," *International Journal of Behavioral Nutrition and Physical Activity*, vol. 9, no. 1, p. 85, 2012.

[6] S. J. W. A. L. Baker Philip RA, Francis Daniel P and F. Charles, "Community wide interventions for increasing physical activity," *Cochrane Database of Systematic Reviews*, vol. 9, no. 1, 2015.

[7] D. E. Warburton, C. W. Nicol, and S. S. Bredin, "Health benefits of physical activity: the evidence," *Canadian medical association journal*, vol. 174, no. 6, pp. 801–809, 2006.

[8] D. Feil-Seifer and M. Mataric, "Defining socially assistive robotics," in *Rehabilitation Robotics, 2005. ICORR 2005. 9th International Conference on*, pp. 465–468, June 2005.

[9] S. Thrun, M. Bennewitz, W. Burgard, A. B. Cremers, F. Dellaert, D. Fox, D. Hahnel, C. Rosenberg, N. Roy, J. Schulte, *et al.*, "MINERVA: A second-generation museum tour-guide robot," in *Proc. of IEEE Intl. Conf. on Robotics and Automation*, vol. 3, IEEE, 1999.

[10] D. Fontanelli, A. Giannitrapani, L. Palopoli, and D. Prattichizzo, "Unicycle steering by brakes: A passive guidance support for an assistive cart," pp. 2275–2280, 2013.

[11] L. Rizzon and R. Passerone, "Embedded soundscape rendering for the visually impaired," in *2013 8th IEEE Intl. Symposium on Industrial Embedded Systems*, pp. 101–104, IEEE, 2013.

[12] S. Scheggi, M. Aggravi, F. Morbidi, and D. Prattichizzo, "Cooperative human-robot haptic navigation," in *Proc. of IEEE Intl. Conf. on Robotics and Automation*, pp. 2693–2698, IEEE, 2014.

[13] T. Carlson and Y. Demiris, "Collaborative control for a robotic wheelchair: evaluation of performance, attention, and workload," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 42, no. 3, pp. 876–888, 2012.

[14] M. Vasic and A. Billard, "Safety issues in human-robot interactions," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pp. 197–204, May 2013.

[15] C. Bartneck and J. Forlizzi, "A design-centred framework for social human-robot interaction," in *Robot and Human Interactive Communication, 2004. ROMAN 2004. 13th IEEE International Workshop on*, pp. 591 – 594, September 2004.

[16] D. Feil-Seifer and M. J. Matarić, "Human-robot interaction," *Encyclopedia of Complexity and System Science*, vol. Springer Tracks in Robotics, no. 54, 2009.

[17] E. Goffman, *Behavior in Public Places: Notes on the Social Organization of Gatherings*. Free Press, September 1966.

[18] E. T. Hall, "Proxemics – the study of man's spatial relations," in *Man's image in medicine and anthropology*, International Universities Press, 1962.

[19] L. Palopoli, A. Argyros, J. Birchbauer, A. Colombo, D. Fontanelli, A. Legay, A. Garulli, A. Giannitrapani, D. Macii, F. Moro, P. Nazemzadeh, P. Padeleris, R. Passerone, G. Poier, D. Prattichizzo, T. Rizano, L. Rizzon, S. Scheggi, and S. Sedwards, "Navigation assistance and guidance of older adults across complex public spaces: the dali approach," *Intelligent Service Robotics*, April 2015. to appear.

[20] A. Colombo, D. Fontanelli, A. Legay, L. Palopoli, and S. Sedwards, "Efficient customisable dynamic motion planning for assistive robots in complex human environments," *Journal of Ambient Intelligence and Smart Environments*, vol. 7, pp. 617–634, 2015.

[21] E. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[22] A. Colombo, D. Fontanelli, A. Legay, L. Palopoli, and S. Sedwards, "Motion planning in crowds using statistical model checking to enhance the social force model," in *IEEE 52nd Annual Conference on Decision and Control (CDC2013)*, pp. 3602–3608, December 2013.

[23] A. Colombo, D. Fontanelli, D. Gandhi, A. De Angeli, L. Palopoli, S. Sedwards, and A. Legay, "Behavioural templates improve robot motion planning with social force model in human environments," in *IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*, 2013.

[24] S. LaValle and R. Sharma, "Robot motion planning in a changing, partially predictable environment," in *Proceedings of the 1994 IEEE International Symposium on Intelligent Control*, pp. 261–266, August 1994.

[25] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, May 2006.

[26] Y. Lasovsky and L. Joskowicz, "Motion planning in crowded planar environments," *Robotica*, vol. null, pp. 365–371, July 1999.

[27] J. Miura and Y. Shirai, "Hierarchical vision-motion planning with uncertainty: Local path planning and global route selection," in *Proceedings of the 1992 lEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3, pp. 1847–1854, July 1992.

[28] D. Guo, C. Wang, and X. Wanf, "A hierarchical pedestrians motion planning model for heterogeneous crowds simulation," in *International Conferenceon Information and Automation (ICIA '09)*, pp. 1363–1367, June 2009.

[29] Assistant for Safe Mobility. `http://assam.nmshost.de`, June 2012.

[30] Domestic Robot for Elderly Assistance. `http://www.aal-domeo.org`, July 2009.

[31] Integrated prevention and Detection sOlutioNs Tailored to the population and Risk Factors associated with FALLs. `http://www.idontfall.eu`, April 2012.

[32] Veloped - the walker for active people. `http://www.trionic.us`, April 2012.

[33] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, pp. 566–580, August 1996.

[34] S. M. Lavalle, "Rapidly-exploring random trees: A new tool for path planning," Tech. Rep. TR 98-11, Iowa State university, October 1998.

[35] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[36] C. W. Warren, M. Engineering, and T. Uiiiversi, "Global Path Planning Using Artificial Potential Fields," pp. 316–321, 1989.

[37] F. Arambula Cosío and M. Padilla Castañeda, "Autonomous robot navigation using adaptive potential fields," 2004.

[38] P. Hart, N. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, 1968.

[39] P. Bhattacharya and M. L. Gavrilova, "Voronoi diagram in optimal path planning," *4th International Symposium on Voronoi Diagrams in Science and Engineering (ISVD 2007)*, vol. 1, pp. 38–47, July 2007.

[40] D. Chen, R. Szczerba, and J. Uhran, "A framed-quadtree approach for determining Euclidean shortest paths in a 2-D environment," *IEEE Transactions on Robotics and Automation*, vol. 13, no. 5, pp. 668–681, 1997.

[41] B. Ding, J. X. Yu, and L. Qin, "Finding time-dependent shortest paths over large graphs," in *Proceedings of the 11th International Conference on Extending Database Technology: Advances in Database Technology*, EDBT '08, (New York, NY, USA), pp. 205–216, ACM, 2008.

[42] D. Delling and D. Wagner, "Time-dependent route planning," in *Robust and Online Large-Scale Optimization* (R. Ahuja, R. Möhring, and C. Zaroliagis, eds.), vol. 5868 of *Lecture Notes in Computer Science*, pp. 207–230, Springer Berlin Heidelberg, 2009.

[43] U. Demiryurek, F. Banaei-Kashani, and C. Shahabi, "A case for time-dependent shortest path computation in spatial networks," in *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '10, (New York, NY, USA), pp. 474–477, ACM, 2010.

[44] L. Foschini, J. Hershberger, and S. Suri, "On the complexity of time-dependent shortest paths," *Algorithmica*, vol. 68, no. 4, pp. 1075–1097, 2014.

[45] A. J. Coles and A. Coles, "LPRPG-P: relaxed plan heuristics for planning with preferences," in *Proceedings of the 21st International Conference on Automated Planning and Scheduling, ICAPS 2011, Freiburg, Germany June 11-16, 2011*, 2011.

[46] M. Lahijanian, S. Almagor, D. Fried, L. E. Kavraki, and M. Y. Vardi, "This time the robot settles for a cost: A quantitative approach to temporal logic planning with partial satisfaction," in *The Twenty-Ninth AAAI Conference (AAAI-15)*, (Austin, TX), AAAI, AAAI, January 2015. to appear.

[47] J. Tumova, L. I. R. Castro, S. Karaman, E. Frazzoli, and D. Rus, "Minimum-violation LTL planning with conflicting specifications," *2013 American Control Conference*, pp. 200–205, June 2013.

[48] S. Gulati, C. Jhurani, B. Kuipers, and R. Longoria, "A framework for planning comfortable and customizable motion of an assistive mobile robot," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2009*, pp. 4253–4260, 2009.

[49] Y. Morales, N. Kallakuri, K. Shinozawa, T. Miyashita, and N. Hagita, "Human-comfortable navigation for an autonomous robotic wheelchair," *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2737–2743, November 2013.

[50] T. Kruse, A. K. Pandey, R. Alami, and A. Kirsch, "Human-aware robot navigation: A survey," *Robotics and Autonomous Systems*, vol. 61, pp. 1726–1743, 2013.

[51] J. Barraquand and J.-C. Latombe, "A Monte-Carlo algorithm for path planning with many degrees of freedom," in *Proc. IEEE International Conference on Robotics and Automation*, vol. 3, pp. 1712–1717, 1990.

[52] S. Loizou and K. Kyriakopoulos, "Automatic synthesis of multi-agent motion tasks based on ltl specifications," in *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, vol. 1, pp. 153–158, Dec.

[53] H. Kress-Gazit, G. Fainekos, and G. Pappas, "Where's waldo? sensor-based temporal logic motion planning," in *Robotics and Automation, 2007 IEEE International Conference on*, pp. 3116–3121, IEEE, 2007.

[54] S. Karaman and E. Frazzoli, "Sampling-based motion planning with deterministic μ-calculus specifications," in *IEEE Conference on Decision and Control (CDC)*, (Shanghai, China), December 2009.

[55] Y. Kuwata, J. Teo, S. Karaman, G. Fiore, E. Frazzoli, and J. How, "Motion planning in complex environments using closed-loop prediction," *Proc. AIAA Guidance, Navigation, and Control Conf. and Exhibit*, 2008.

[56] J. Hu, J. Lygeros, and S. Sastry, "Towars a theory of stochastic hybrid systems," in *HSCC* (N. A. Lynch and B. H. Krogh, eds.), vol. 1790 of *Lecture Notes in Computer Science*, pp. 160–173, Springer, 2000.

[57] J. Hu, M. Prandini, and S. Sastry, "Aircraft conflict prediction in the presence of a spatially correlated wind field," *IEEE Transactions on Intelligent Transportation Systems*, vol. 6, no. 3, pp. 326–340, 2005.

[58] M. Prandini, J. Lygeros, A. Nilim, and S. Sastry, "A probabilistic framework for aircraft conflict detection," *IEEE Transactions on Intelligent Transportation Systems*, vol. 1, no. 4, pp. 199–220, 2000.

[59] R. Asaula, D. Fontanelli, and L. Palopoli, "Safety provisions for human/robot interactions using stochastic discrete abstractions," in *Proc. of 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2010)*, October 2010.

[60] R. Asaula, D. Fontanelli, and L. Palopoli, "A probabilistic methodology for predicting injuries to human operators in automated production lines," in *Proc. of the 14th IEEE Conference on Emerging Technologies and Factory Automation (ETFA2009)*, (Mallorca, Spain), September 2009.

[61] D. Helbing and P. Molnár, "Social force model for pedestrian dynamics," *Phys. Rev. E*, vol. 51, pp. 4282–4286, 1995.

[62] T. I. Lakoba, D. J. Kaup, and N. M. Finkelstein, "Modifications of the Helbing-Molnár-farkas-vicsek social force model for pedestrian evolution," *Simulation*, vol. 81, pp. 339–352, May 2005.

[63] D. R. Parisi, M. Gilman, and H. Moldovan, "A modification of the social force model can reproduce experimental data of pedestrian flows in normal conditions," *Physica A: Statistical Mechanics and its Applications*, vol. 388, no. 17, pp. 3600–3608, 2009.

[64] P. Ratsamee, Y. Mae, K. Ohara, T. Takubo, and T. Arai, "Modified social force model with face pose for human collision avoidance," *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction - HRI '12*, p. 215, 2012.

[65] G. Lämmel and M. Plaue, "Getting out of the way: Collision-avoiding pedestrian models compared to the realworld," in *Pedestrian and Evacuation Dynamics 2012* (U. Weidmann, U. Kirsch, and M. Schreckenberg, eds.), pp. 1275–1289, Springer International Publishing, 2014.

[66] D. Kelly and F. Boland, "Motion model selection in tracking humans," in *Irish Signals and Systems Conference, 2006. IET*, pp. 363–368, June 2006.

[67] M. Moussaïd, D. Helbing, S. Garnier, A. Johansson, M. Combe, and G. Theraulaz, "Experimental study of the behavioural mechanisms underlying self-organization in human crowds," *Proceedings of the Royal Society of London B: Biological Sciences*, vol. 276, no. 1668, pp. 2755–2762, 2009.

[68] M. Moussaïd, N. Perozo, S. Garnier, D. Helbing, and G. Theraulaz, "The walking behaviour of pedestrian social groups and its impact on crowd dynamics," *PLoS ONE*, vol. 5, p. e10047, 04 2010.

[69] C. Burstedde, K. Klauck, a. Schadschneider, and J. Zittartz, "Simulation of pedestrian dynamics using a two-dimensional cellular automaton," *Physica A: Statistical Mechanics and its Applications*, vol. 295, pp. 507–525, June 2001.

[70] B. Lau, K. Arras, and W. Burgard, "Multi-model hypothesis group tracking and group size estimation," *International Journal of Social Robotics*, vol. 2, pp. 19–30, December 2010.

[71] B. A. Schlake, "Mathematical models for pedestrian motion," Master's thesis, Westfälische Wilhelms - Universität Münster Mathematical, Münster, Germany, April 2008.

[72] R. A. Finkel and J. L. Bentley, "Quad trees: A data structure for retrieval on composite keys," *Acta Informatica*, vol. 4, no. 1, pp. 1–9, 1974.

[73] C. R. Dyer, "The space efficiency of quadtrees," *Computer Graphics and Image Processing*, vol. 19, no. 4, pp. 335–348, 1982.

[74] C. H. Papadimitriou, "The Euclidean travelling salesman problem is NP-complete," *Theoretical Computer Science*, vol. 4, no. 3, pp. 237–244, 1977.

[75] E. Clarke, E. A. Emerson, and J. Sifakis, "Model checking: algorithmic verification and debugging," *Commun. ACM*, vol. 52, pp. 74–84, November 2009.

[76] C. Baier and J.-P. Katoen, *Principles of Model Checking*. MIT Press, May 2008.

[77] B. Boyer, K. Corre, A. Legay, and S. Sedwards, "Plasma-lab: A flexible, distributable statistical model checking library," in *Proceedings of the 10th International Conference on Quantitative Evaluation of Systems*, QEST'13, (Berlin, Heidelberg), pp. 160–164, Springer-Verlag, 2013.

[78] S. K. Jha, E. M. Clarke, C. Langmead, A. Legay, A. Platzer, and P. Zuliani, "A bayesian approach to model checking biological systems," in *Computational Methods in Systems Biology* (P. Degano and R. Gorrieri, eds.), vol. 5688 of *Lecture Notes in Computer Science*, pp. 218–234, Springer Berlin Heidelberg, 2009.

[79] W. Hoeffding, "Probability inequalities for sums of bounded random variables," *Journal of the American Statistical Association*, vol. 58, pp. 13–30, March 1963.

[80] M. Mitzenmacher and E. Upfal, *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. New York, NY, USA: Cambridge University Press, 2005.

[81] D. Helbing, I. Farkas, and T. Vicsek, "Simulating dynamical features of escape panic," *Nature*, vol. 407, pp. 487–490, September 2000.

[82] H. Chernoff, "A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations," *Ann. Math. Statist.*, vol. 23, no. 4, pp. 493–507, 1952.

[83] K. Ahnert and M. Mulansky, "Odeint – solving ordinary differential equations in c++," *AIP Conference Proceedings*, vol. 1389, no. 1, pp. 1586–1589, 2011.

[84] D. Helbing, I. J. Farkas, and T. Vicsek, "Freezing by heating in a driven mesoscopic system," *Phys. Rev. Lett.*, vol. 84, pp. 1240–1243, 2000.

[85] M. Jarke, X. T. Bui, and J. M. Carroll, "Scenario management: An interdisciplinary approach," *Requirements Engineering*, vol. 3, no. 3-4, pp. 155–173, 1998.

[86] W. B. Rouse and N. M. Morris, "On looking into the black box: Prospects and limits in the search for mental models.," *Psychological Bulletin*, vol. 100, no. 3, p. 349, 1986.

[87] R. Teague, K. De Jesus, and M. N. Ueno, "Concurrent vs. post-task usability test ratings," in *Proceedings of Computer-Human Interaction (CHI'01)*, pp. 289–290, ACM Press, 2001.

[88] P. Wittenburg, H. Brugman, A. Russel, A. Klassmann, and H. Sloetjes, "Elan: a professional framework for multimodality research," in *Proceedings of LREC*, vol. 2006, 2006.

[89] A. Johansson, D. Helbing, and P. K. Shukla, "Specification of the social force pedestrian model by evolutionary adjustment to video tracking data," *Advances in Complex Systems*, vol. 10, no. supp02, pp. 271–288, 2007.

[90] M. Aggravi, A. Colombo, D. Fontanelli, A. Giannitrapani, D. Macii, F. Moro, P. Nazemzadeh, L. Palopoli, R. Passerone, D. Prattichizzo, T. Rizano, L. Rizzon, and S. Scheggi, "Dali: A smart walking assistant for safe navigation in complex indoor environments," 2014.

[91] F. Moro, A. De Angeli, D. Fontanelli, R. Passerone, D. Prattichizzo, L. Rizzon, S. Scheggi, S. Targher, and L. Palopoli, "Follow, listen, feel and go," *Autonomous Robots*, April 2015. to appear.

[92] P. Panteleris and A. A. Argyros, "Vision-based SLAM and moving objects tracking for the perceptual support of a smart walker platform," in *Workshop on Assistive Computer Vision and Robotics (ACVR 2014), in conjunction with ECCV 2014*, 2014.

[93] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Transactions of the ASME - Journal of Basic Engineering*, no. 82 (Series D), pp. 35–45, 1960.

[94] J.-H. Kim, S. Okuma, Y.-W. Kim, D.-H. Hwang, M.-H. Kim, and D.-H. Kim, "Modeling of human driving behavior based on piecewise linear model," in *Industrial Electronics, 2005. ISIE 2005. Proceedings of the IEEE International Symposium on*, vol. 1, pp. 25–30, June 2005.

[95] J. M. Henderson, "Human gaze control during real-world scene perception," *Trends in Cognitive Sciences*, vol. 7, no. 11, pp. 498 – 504, 2003.

[96] R. Valenti, N. Sebe, and T. Gevers, "Facial expression recognition: A fully integrated approach," in *Image Analysis and Processing Workshops, 2007. ICIAPW 2007. 14th International Conference on*, pp. 125–130, Sept 2007.

[97] E. Mazor, A. Averbuch, Y. Bar-Shalom, and J. Dayan, "Interacting multiple model methods in target tracking: a survey," *Aerospace and Electronic Systems, IEEE Transactions on*, vol. 34, pp. 103–123, Jan 1998.

[98] L. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, pp. 257–286, February 1989.