UNIVERSITÀ DEGLI STUDI
DI TRENTO

Department of Information Engineering and Computer Science
ICT International Doctoral School

Efficient Motion Planning for Wheeled Mobile Robotics

ADVISOR

Prof. Luigi Palopoli
University of Trento

PH.D. CANDIDATE

Dr. Paolo Bevilacqua
University of Trento

CO-ADVISOR

Prof. Daniele Fontanelli
University of Trento

15 July 2019

Abstract

Nowadays, the field of wheeled robotics is undergoing an impressive growth and development. Different hardware and software components are being developed and applied in various contexts, including assistive robotics, industrial robotics, automotive, ...

Motion Planning is a fundamental aspect for the development of autonomous wheeled mobile robots. The capability of planning safe, smooth trajectories, and to locally adjust them in real-time to deal with contingent situations and avoid collisions is an essential requirement to allow robots to work and perform activities in public spaces shared with humans. Moreover, in general, efficiency is a key constraint for this kind of applications, given the limited computational power usually available on robotic platforms. In this thesis, we focus on the development of efficient algorithms to solve different kind of motion planning problems. Specifically, in the first part of the thesis, we propose a complete planning system for an assistive robot supporting the navigation of older users. The developed planner generates paths connecting different locations on the map, that are smooth and specifically tailored to optimize the comfort perceived by the human users. During the navigation, the system applies an efficient model to predict the behaviours of the surrounding pedestrians, and to locally adapt the reference path to minimise the probability of collisions. Finally, the motion planner is integrated with an "high-level" reasoning component, to generate and propose complete activities, like the visit to a museum or a shopping mall, specifically tailored to the preferences, needs and requirements of each user. In the second part of the thesis, we show how the efficient solutions and building blocks developed for the assistive robots, can be adapted and applied also to a completely different context, such as the generation of optimal trajectories for an autonomous racing vehicle.

This research has received funding from the European Unions Horizon 2020 Research and Innovation Programme - Societal Challenge 1 (DG CONNECT/H) under grant agreement n° 643644 "ACANTO - A CyberphysicAl social NeTwOrk using robot friends".

Acknowledgments

First of all, I would like to thank my advisor, Prof. Luigi Palopoli, and my co-advisor, Prof. Daniele Fontanelli, for the continuous help, support and guidance provided during my Ph.D. Their knowledge and insightful advices played a fundamental role in the completion of this work. Then, I would like to thank all the members of the Embedded Electronics and Computing Systems group that I met during these years. Thanks to Marco Frego, Fabiano Zenatti, Stefano Divan, Marco Andreetto, Valerio Magnago, Bernardo Villalba Frías, Cristina Guerrero Flores and to the newcomers Manuel Boldrer and Alessandro Antonucci, for the help, support and friendship provided during these years, and for the lot of funny experiences lived together. I would also like to extend a special thanks to Roberta Guidolin, whose administrative help often extended beyond the call of duty. I want to thank all my friends and the people that supported and helped me. Finally, my family deserves a special thank for always assisting and spurring me, and for their precious advices. Thanks for everything!

Contents

List of Figures

1 Introduction

Nowadays, robotics and automation are undergoing an impressive growth and development. Their application encompasses an endless number of different contexts, from industrial automation, to assistive robotics, to automotive.

All these applications display a number of different challenges and open problems, regarding a large number of different research fields. Among them, a fundamental problem, common to all the various robotic applications, is the capability of the robot to move autonomously from a certain initial configuration, to reach a different, given final configuration. To accomplish this task, the robot is required to automatically determine the sequence of movements, i.e. the trajectory, to follow, considering both environmental and physical constraints. While the formers depend on the obstacles in the environment and on the geometry of the robot, the latter are for example kinematic constraints (e.g. the impossibility to turn on the spot for a car-like vehicle).

The main objective of this thesis is the development of efficient motion planning algorithms, both to synthesise a global path, and to replan local modifications to apply on-the-fly in real-time to avoid collisions with unforeseen static and dynamic obstacles. We propose efficient approaches and solutions, all based on a set of common and characterising basic algorithms and techniques, composing a set of building blocks that are adequately adapted and combined together to solve the different problems at hand.

Specifically, we focus on two different kinds of applications:

- **Assistive robotics**: path planning with a careful consideration and optimization of the comfort and satisfaction perceived by the assisted user. Dynamic adjustment of the original path to avoid collisions and ensure a safe navigation in public spaces shared with other people.
- **Automotive**: synthesis of minimum time trajectories for autonomous racing vehicles.

1.1 Assistive Robotics

During the last decade, Western societies have experienced an increase in life expectancies and a simultaneous collapse of birth rates, that have brought to a rapid ageing of the population (in Figure 1.1 are shown the trend projections of population pyramids for the years 2020 and 2050). This phenomenon poses a fundamental challenge to modern societies: guaranteeing older adults an acceptable level of quality of life, while ensuring the economical sustainability of the National Health Systems. For these reasons, the concept of "active ageing" has been developed [1]:

> *Active ageing is the process of optimizing opportunities for health, participation and security in order to enhance quality of life as people age.*

Indeed, "active" older people represent an important resource, as they provide a fundamental support to their families, peers and communities.

However, often a reduction of physical capabilities lead older adults to remain more and more time at home, causing isolation, depression, reduced fitness and increased mobility problems [1]. These in turn produce several detrimental effects: the lack of physical exercise leads often to the development of chronic diseases, such as diabetes. Moreover, reduced mobility increases the number of falls, that for older people represent an increasing cause of injury, requiring extended and costly periods of rehabilitation, and an higher risk of dying. Isolation, due to both the reduction of mobility, and the lost of family members and friends, leads to a decline of both physical and cognitive capabilities.

For these reasons, a number of research projects have been developed, to apply the latest technologies in the fields of both ICT and robotics to assist elder and

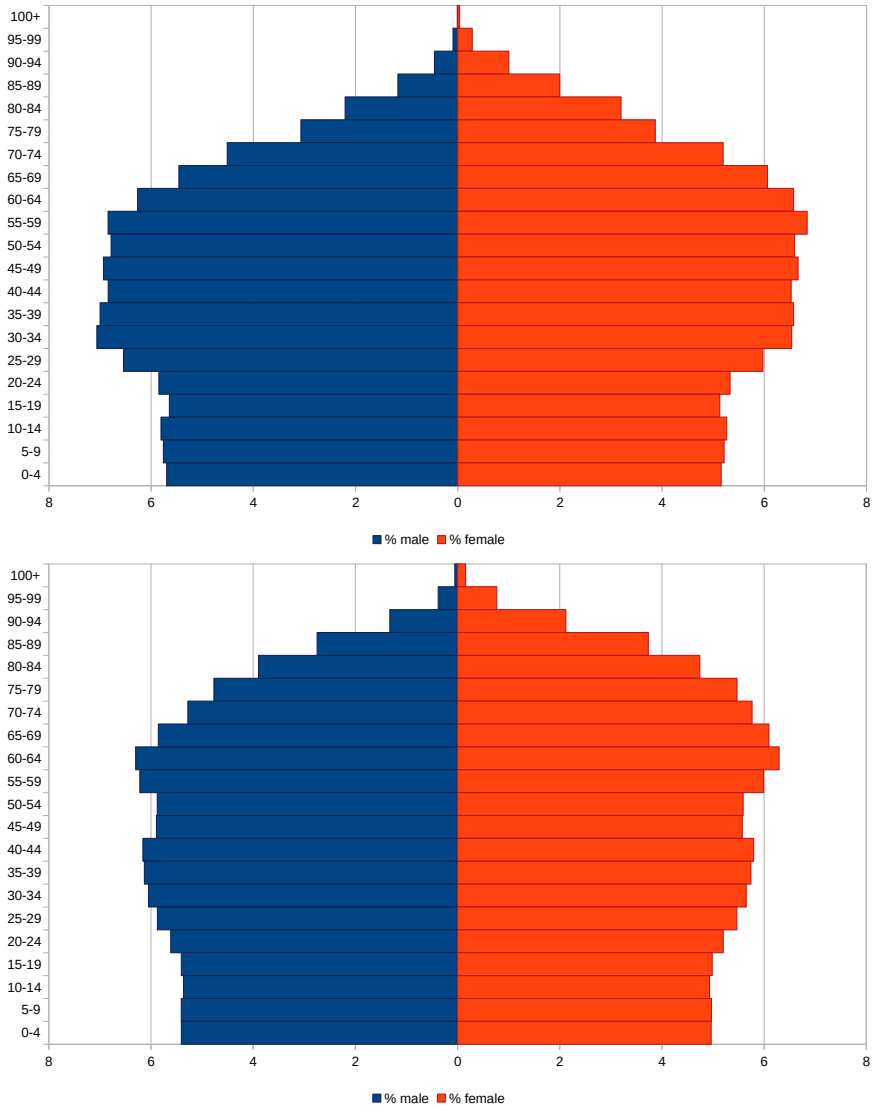


Figure 1.1: Trend distribution of population by age in more developed regions in year 2020 (top) and 2050 (bottom). Data Source: *United Nations, World Population Prospects: The 2017 Revision.*

impaired people, in order to increase their quality of life, and to give them the possibility of living and actively ageing in their own homes. Within the field of assistive robotics, a fundamental role is played by robots assisting the ambulation and allowing older adults to move autonomously. Indeed, these kind of robotic platforms represent an effective tool for older people to maintain an acceptable level of physical and social activity.

1.1.1 ACANTO

ACANTO is an European research project aiming to develop a smart assistive walker, the FriWalk. Its aim is to spur older users to maintain a good life style, and to regularly perform physical and social activities. The FriWalk is a tool providing a complete support to the user, during the execution of social and physical activities, and capable of assisting users with reduced mobility or cognitive capabilities. Its development is based on the adaptation of a standard rollator, with the addition of a sensing system to perceive the surrounding environment, and a set of actuators to provide it with navigation capabilities. In Figure 1.2 is shown a prototype of the smart robotic walker FriWalk developed within the ACANTO project.

In the next sections, we provide an overview of the main planning technologies and algorithms developed within the context of the ACANTO project, that are an important contribution of this thesis.

In addition to the planning system, other fundamental challenges, that are beyond the scope of this thesis, had to be solved for the development of a smart assistive rollator supporting the navigation of older users. In particular, we required a path following controller capable of sharing the responsibility for the guidance of the vehicle with the user [2, 3, 4, 5], and a robust indoor localisation system allowing the robot to know its exact position on the map [6, 7, 8].

Planning System

The planning system is a fundamental software component for a navigation assistive robot. It is responsible for the planning of activities and tasks at different levels:

- at the lowest level of abstraction, to support older users during the visit of a public space and the execution of social activities, it is necessary to integrate the navigation module with a tool capable of generating paths between pairs

Figure 1.2: Example of the FriWalk robotic walker developed within the ACANTO project [9]

of points on the map. The generated paths should be comfortable to follow by older users with the support of the smart rollator. To determine the best path to follow, the planning system has to take into account user specific needs and preferences. To that hand, the ACANTO software infrastructure provides a shared knowledge base, containing all the information and profiles associated with each user. During the autonomous robot navigation, or while carrying an older user along the planned path, it may be necessary to apply local modifications, to overcome unforeseen events, e.g. the presence of pedestrians or obstacles hindering the original path. In case of users with reduced cognitive capabilities, or in situations where the robot is required to move autonomously between different locations on the map, the navigation system has to be able to apply local modifications to the reference trajectory, in order to avoid collisions. Thus, the global motion planner must be integrated with a reactive planner, capable of handling these scenarios.

- on a higher level, the smart rollator requires a planning system capable of

suggesting to the users a complete activity (e.g. visit to a museum, shopping mall, etc.), considering their requirements and preferences. This requires the capability of a "high-level", abstract reasoning. Therefore, together with the low level motion planner, an high level planner is needed to determine the best sequence of activities to propose to the user, that will then be refined into a sequence of paths by the low level motion planning algorithm.

Over the years, many different approaches have been proposed to solve the motion planning problem for a robot. Given an initial and a final configuration, the motion planner finds an "optimal" trajectory connecting them, given the kinematic and dynamic constraints of the mechanical platform, and the map with all the obstacles that must be avoided. Also the problem of planning a sequence of generic, "high–level" actions to reach a specific goal, given a set of constraints and preferences, has been studied within the artificial intelligence community, and interesting solutions have been proposed.

However, for the implementation of a complete activity planning system for an assistive robot, the adoption of traditional motion planners is not sufficient, since they do not allow the modelling of tasks (e.g. which places to visit, how long, in which sequence) and "high–level" user constraints, essential to plan activities adequate for the user. On the other hand, high-level task planners generate abstract plans, composed by sequences of complex actions that cannot be directly used by the controller of the robot. Thus, the output of the task planner has to be refined into an executable plan. The refined plan consists in a path that must respect the same constraints and requirements of the original, abstract plan, but also the physical constraints of the robot.

1.2 Automotive

A large amount of research is focusing on the application of new ICT technologies and tools for automotive applications. Advanced Driving Assistance Systems such as collision avoidance, lane keeping and autonomous driving are becoming more and more popular in modern cars. Some of these features are offered already in relatively low cost markets segments; it is expected that others will soon be. A key component for the implementation of these systems is the ability to plan (or

re-plan) in a short time a trajectory that satisfies all the dynamic constraints of the vehicle.

For all these reasons, different problems regarding different automotive applications have been studied with increasing interest during the last years. One of them, becoming more and more relevant, is the automatic driving of cars at their limits, i.e. for autonomous racing. In Figure 1.3 is shown an example of a fully autonomous racing car, the Roborace Robocar.

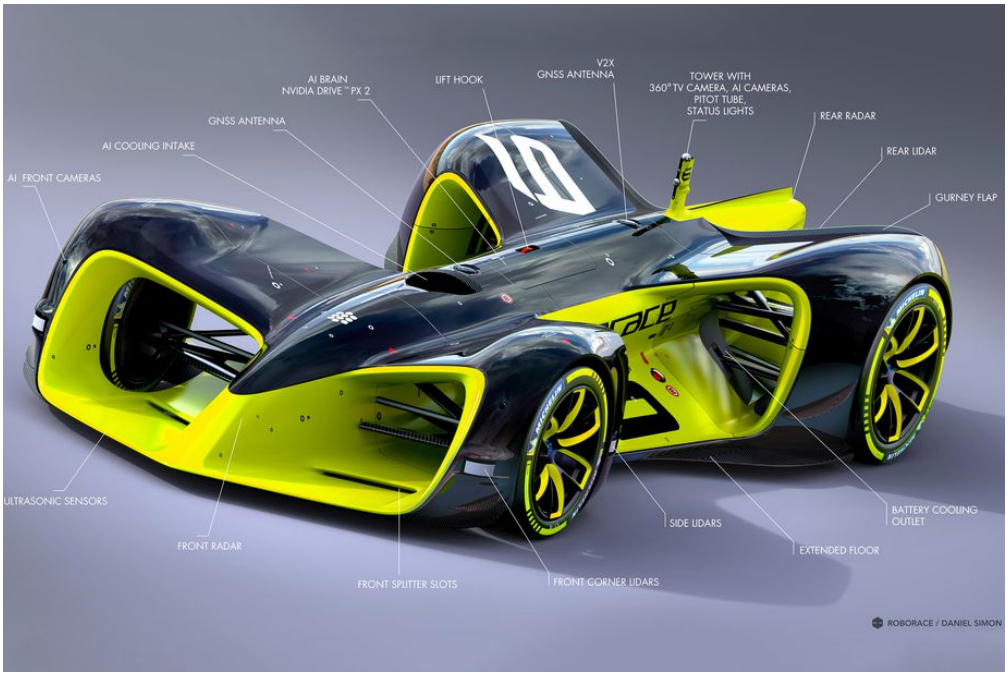


Figure 1.3: The Roborace autonomous racing car: Robocar [10]

In this context, the motion planning problem consists in the determination of the optimal trajectory to follow, minimizing the overall travel time. Within this thesis, we show how it is possible to adapt and reuse the same basic building blocks applied for the solution of planning problems in the field of assistive robotics, and to combine them with other efficient techniques, in order to solve the problem of trajectory optimization for racing cars. We show how to apply the same kind of motion primitives (based on clothoid curves), and to decompose the original problem into a geometric and a trajectory optimization subproblem. Finally, we propose a complete solution for the generation of the minimum time lap on a given racing track. In addition, we develop a solution for the efficient online, real-time

local replanning and modification of the global trajectory to overcome unforeseen obstacles.

1.3 Structure of the Thesis

The thesis is organised as follows. Chapter 2 introduces background knowledge and definitions on motion planning. Then, Chapter 3 presents a detailed literature review on the different kind of planning problems tackled by this thesis. Following, Chapter 4 contains a description of clothoid curves, a family of curves that we adopt as the basic motion primitives composing the paths that are generated by all the motion planning solutions proposed in this thesis. We propose a set of relevant problems involving these curves, together with efficient solutions for each of them. The remainder of the thesis is organised into two different parts.

The first part deals with the development of a complete planning solution for assistive robotics applications. In Chapter 5, we present an efficient solution for the generation of a smooth path specifically tailored to optimize the comfort perceived by the human during the navigation with the support of the assistive rollator. In Chapter 6, we develop a reactive planning solution, capable of locally adjusting the global reference path during the navigation, to avoid collisions with unforeseen static obstacles, and with the surrounding, moving pedestrians. To develop a solution both safe and efficient, we apply and approximate a model of human walking behaviour, the Headed Social Force Model, to determine the best local modification to apply, minimising the probability of collision while trying to minimise the amount of deviation from the original path. In Chapter 7, we show to integrate the global and reactive planning solutions developed in the previous chapters with a task planner providing high-level, abstract reasoning capabilities, to generate and propose to the user complete activities such as the visit to a museum or a shopping mall. These activities are synthesised considering the requirements, needs and preferences specific for each user.

The second part of the thesis, on the other hand, is focused on the application of the same efficient algorithms developed and applied in the first part of the thesis, and founded on the use of clothoid curves as the basic motion primitives, to generate time optimal trajectories for a racing vehicle on a track. In Chapter 8, we illustrate efficient solutions to compute the optimal manoeuvre for a given path composed

of clothoid curves, explicitly considering a model of the vehicle with non–linear dynamics, aerodynamic drag effects, bounds on the control and constraints on the acceleration and speed. In Chapters 9 and 10, we show how to efficiently combine the solutions to the geometric and dynamic sub-problems, i.e. efficient generation of a path composed of clothoid arcs connecting two configurations, and efficient computation of the optimal manoeuvre, to generate the optimal trajectory on a given racing track, and to locally re-plan the reference trajectory in the presence of unforeseen obstacles along the way in real-time.

1.4 Scientific Contributions and List of Publications

This thesis proposes several contributions to the state-of-the-art in the field of motion planning, regarding different scopes and applications. In this section, we provide an overview of the main contributions of the thesis, and the list of peer-reviewed publications relevant for each of these contributions.

1. The development of an efficient motion planning solution for assistive robots, mainly focused on the optimization of the comfort perceived by the human users. This work has been presented and published at the IEEE Conference on Control Applications (CCA), 2016 [11].

2. The development of an efficient solution to reactively adjust the reference path during the navigation of an environment shared with other humans, through an efficient modelling of candidate human trajectories and an analytical computation of the probabilities of collision. This work has been presented at the IEEE International Conference on Robotics and Automation (ICRA), 2018, and has been published on the IEEE Robotics and Automation Letters (RA-L) [12].

3. The development of an activity planner, focused on the close collaboration of the high-level reasoning component with the low-level global and reactive planning modules, to generate optimised activities tailored on the specific constraints, preferences and requirements of the user. The low-level planning components are then used to refine the high-level sequence of tasks composing an activity, to a sequence of paths that the user can follow with the support of

the assistive rollator. This work has been submitted to the IEEE Transactions on Industrial Informatics (TII), and is still undergoing the review process [13].

4. The extension and adaptation of the efficient building blocks developed for the motion planning for assistive robotics, and its application to determine the optimal trajectory for a racing vehicle on a track. Part of this work has been presented and published at the 55th IEEE Conference on Decision and Control (CDC), 2016 [14].

5. The development of an efficient solution for the local replanning of the optimal trajectory for a racing vehicle moving on a track, to avoid new, unforeseen, obstacles. This work has been presented and published at the IEEE European Control Conference (ECC), 2018 [15].

2
Background

In this chapter we will provide all the definitions and background knowledge relevant for this thesis.

2.1 Motion Planning Definitions

World The representation of the space where the robot and the obstacles lie. Usually, the world is either 2D ($\mathbb{R}^2$) or 3D ($\mathbb{R}^3$). The world is denoted by the symbol $\mathcal{W}$.

Robot From the perspective of motion planning, a robot is seen as a physical body modelled geometrically, and that can be controlled via the motion plan. The pose of each joint composing the robot can be expressed using a set of parameters. The set of all the parameters determining the global state of the robot is known as its configuration. In general, a robot has associated some constraints limiting the set of feasible configurations. Notice that with this definition, a robot in terms of motion planning may not be an actual robot, but it could model any collection of moving bodies, and could be applied also for different applications, e.g. simulation of humans in virtual reality, molecules, etc. The rigid body associated with a robot is usually denoted by $\mathcal{A}$.

Obstacles Obstacles represent physical bodies that are occupying some portions of the world, therefore rendering some robot configurations unfeasible. For example,

obstacles may models walls, furniture, etc. Usually, the set of rigid bodies associated with the obstacles are denoted by $\mathcal{B}_1, \ldots, \mathcal{B}_n$.

Configuration Space In general, motion planners do not search a path directly in the world space $\mathcal{W}$, but they work on the configuration space, that is the space of all the possible parameters for the different configurations of a robot. This space is usually denoted as $\mathcal{C}$. Moreover, let $\mathcal{W} = \mathbb{R}^m$ represents the world space, $\mathcal{O} \subset \mathcal{W}$ the obstacle region, $\mathcal{A}(q) \subset \mathcal{W}$ the region of the world occupied by the robot while in configuration $q \in \mathcal{C}$, then we define:

$$\mathcal{C}_{\text{free}} = \{q \in \mathcal{C} \mid \mathcal{A}(q) \cap \mathcal{O} = \emptyset\}$$

and

$$\mathcal{C}_{\text{obs}} = \mathcal{C} \setminus \mathcal{C}_{\text{free}}$$

Path A path in the configuration space is a continuous function τ such that:

$$\tau : [0, 1] \to \mathcal{C}$$

In addition, if τ lies entirely in the free configuration space, i.e.

$$\tau : [0, 1] \to \mathcal{C}_{\text{free}}$$

then τ is a collision-free path

Motion Planning Problem Given the initial configuration of the robot $q_I \in \mathcal{C}_{\text{free}}$, the goal configuration of the robot $q_G \in \mathcal{C}_{\text{free}}$, the free and obstacle configurations spaces $\mathcal{C}_{\text{free}}$ and $\mathcal{C}_{\text{obs}}$, the motion planning problem requires to find a collision-free path τ such that $\tau(0) = q_I$ and $\tau(1) = q_G$ (see Figure 2.1).

Trajectory Let's define a time parametrisation of a path as a strictly increasing function

$$s : [0, T] \to [0, 1]$$

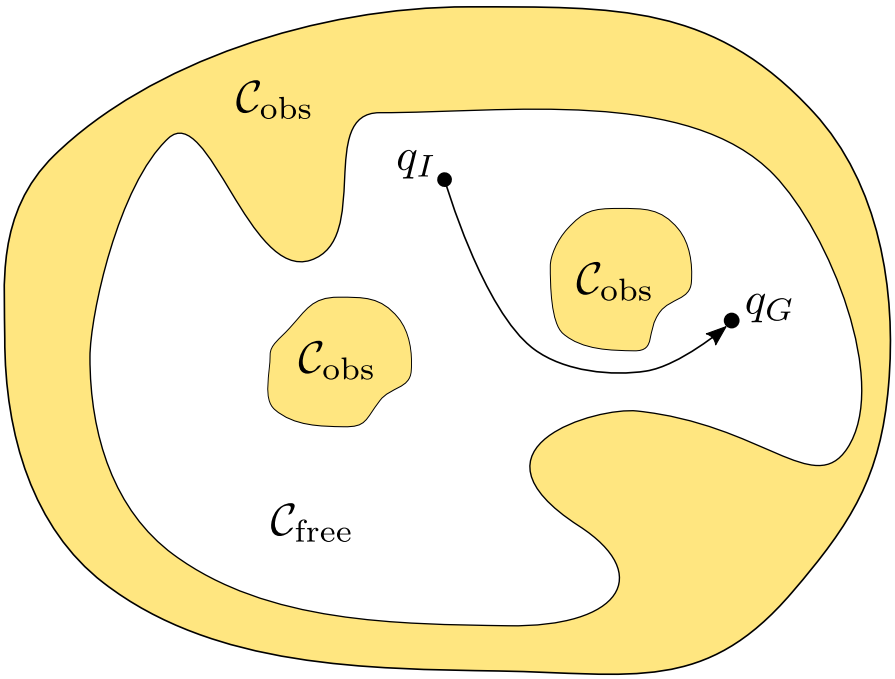


Figure 2.1: The motion planning problem consists in finding a path connecting the start with the goal within the $\mathcal{C}_{\text{free}}$ space.

with $s(0) = 0$ and $s(T) = 1$, that gives the position on the path for each time instant $t \in [0, T]$.
A trajectory Π is a path τ endowed with a time parametrisation s

$$\Pi : [0, T] \to \mathcal{C}$$

where $\Pi(t) = \tau(s(t))$.

2.2 Motion Planning Algorithms

In [16], it has been shown that the motion planning problem is PSPACE-hard. In the literature, there exist several solutions that are correct and complete to solve the motion planning problem. Unfortunately, for all of these algorithms, the computational complexity is exponential with respect to the dimension of the configuration space [17, 18]. Over the years, various different algorithmic approaches have been proposed to solve real-life problems in reasonable amounts of time. Different notions of completeness can be applied to different kind of motion

planning algorithms. A complete algorithm guarantees that it will be able to find a solution to the problem, if a feasible solution exists. Weaker notions of completeness are resolution completeness, i.e. the guarantee to find a feasible solution, when one exists, up to the resolution of the discretization, and probabilistic completeness, i.e. the guarantee that if a problem is feasible, a solution will be found with probability one when the running time tends to infinity.

In general, motion planning algorithms are classified into two main different families, distinguished by the way in which the configuration space is modelled:

- **combinatorial**: explicit representation of the configuration space.
- **sampling based**: implicit representation of the configuration space.

Combinatorial Motion Planning

Combinatorial Motion Planning algorithms are based on an explicit representation of the free configuration space $\mathcal{C}_{\text{free}}$. These kind of algorithms seeks to identify and model the topology of $\mathcal{C}_{\text{free}}$ and $\mathcal{C}_{\text{obs}}$. Usually, the configuration space is represented as a graph, named "roadmap", where nodes represent regions connected regions of $\mathcal{C}_{\text{free}}$, and edges model the connectivity between adjacent regions. The explicit representation of $\mathcal{C}_{\text{free}}$ renders combinatorial motion planning algorithms complete, i.e. if a problem is feasible, in the end the algorithm will produce a valid solution. However, this kind of algorithms are computationally extremely expensive (due to the inherently high complexity of motion planning), and are therefore applicable only to small, low dimensional problems. Among this class of algorithms, common solutions inspired from computational geometry are the vertical cell decomposition approach, the visibility graph approach and the maximum clearance approach (based on the construction of a Voronoi diagram).

For practical applications, to reduce the computational times, some approximated solutions have been developed. These approaches are based on a discretised representation of $\mathcal{C}_{\text{obs}}$, with some given resolution. Among them, quite popular is the quad-tree decomposition approach (for 2D worlds, in 3D worlds oct-tree are used instead). The discretization of the configuration space renders this family of planners resolution complete, meaning that they are guaranteed to find a solution if the problem is feasible and does not require the path to pass closer to any obstacle than the chosen resolution.

Sampling Based Motion Planning

Sampling Based Motion Planning algorithms, on the other hand, are based on an implicit representation of the configuration space. In general, sampling based algorithms incrementally explore the configuration space by sampling configurations within $\mathcal{C}_{\text{free}}$, and by connecting them to build graphs or trees of feasible paths. In general, since sampling based algorithms do not consider explicitly the configuration space, the task of determining whether a given configuration or path connecting two configurations is collision-free is delegated to a black-box collision detection module. For complex configuration spaces, the collision detection is in general one of the main bottlenecks for this family of algorithms. For this reason, usually extremely efficient data structures are adopted, and approximated, conservative approaches are applied to produce solutions with acceptable computational times.

Sampling based algorithms can be classified into two different families:

- Single-query: the algorithm needs to compute just a single path between a start configuration and a goal configuration. Therefore, this family of algorithms are mainly focused on exploring the regions of the configuration space that are of interest for a particular instance of the problem. Among single-query algorithms, popular solutions are based on the Rapidly-Exploring Random Tree (RRT) approach [19], that is probabilistically complete. An important extension of RRT, named RRT* [20], in addition to probabilistic completeness, guarantees also probabilistic optimality, meaning that, as the running time tends to infinity, the produced solution will converge to the optimum with probability one. To achieve this result, differently from the classical RRT algorithm, RRT* dynamically updates and "rewires" some subtrees of the current search tree, whenever shortest paths are found.

- Multi-queries: the algorithm is required to solve many queries between different pairs of configurations on the same configuration space. Thus, to improve the efficiency when answering queries, an expensive preprocessing phase is applied, to explore the whole configuration space and to generate a roadmap of feasible paths, trying to approximate the connectivity of $\mathcal{C}_{\text{free}}$. Among multi-query algorithms, the most populares are based on probabilistic roadmaps (PRM) [21, 22].

3
Related Work

In this chapter, we present state-of-the-art research related with the different kinds of planning problems that we are required to solve to produce the planning system for an assistive robot, and for the generation of time-optimal trajectories for autonomous racing cars. The various problems solved within the thesis are quite heterogeneous, and are therefore related with different research communities and different relevant literature. The remainder of this chapter, therefore, is organised in four different sections, covering the different families of problems tackled in the thesis. More in details, the first section is dedicated to the global motion planning, and focuses on existing algorithms to produce smooth and comfortable paths between pairs of given robot configurations. The second focuses on recently emerging trends in the field of "human-aware" motion planning, i.e. effective strategies to navigate a robot in environments populated by humans. The third section focuses on high-level reasoning techniques, dealing with the selection of an optimal subset of point of interest to visit on an abstract representation of the environment, given some specific constraints. Finally, the fourth section deals with the generation and dynamic adjustment of optimal time manoeuvres for racing vehicles.

3.1 Motion planning in static environments

The motion planner on board of the FriWalk is required to generate a collision-free, smooth path that is optimised and tailored to be comfortably followed by an human during the assisted navigation of a public space. Therefore, the generated path

should be short and smooth at the same time (i.e. avoid abrupt changes of direction and minimize lateral accelerations).

In literature, there are various solutions to generate smooth paths for nonholonomic, wheeled robotic vehicles but, in general, they are not focused on assistive robots, and are not modelling comfort indexes related to human users. These solutions can be roughly classified into two different categories.

On one hand, a large number of approaches are based on the decomposition of the problem into two distinct sub-problems, i.e. the generation of an initial "guess" solution ignoring the nonholonomic constraints, by using simple motion primitives (e.g. straight line segments) to connect the starting position with the goal, and on the successive refinement and smoothing of the path by means of more complex, smoother curves, such as Bezier splines, to achieve solutions with continuous tangent (G^1-splines) or even curvature (G^2-splines). The first problem, that is the generation of a simple path (i.e. a sequence of waypoints) connecting the starting position with the goal, can be solved using standard combinatorial based or sampling based motion planning algorithms (see Chapter 2). Regarding the second problem, i.e. the generation of a smooth path starting from a sequence of waypoints, common approaches are based on their interpolation or fitting using Bezier curves or polynomial splines [23, 24], or on the simulation of a vehicle moving along the sequence of waypoints to obtain a smooth solution [25].

On the other hand, alternative approaches are based on the direct generation of smooth paths during the exploration of the configuration space and the search for a feasible route connecting the start location with the goal. A possible solution is based on the direct application of sampling based motion planning algorithms such as RRT and RRT* [26, 27, 28, 29, 30], that allow the use of complex motion primitives during the construction of the search tree. While these approaches allow the direct generation of a smooth, feasible solution, the direct application of complex motion primitives that are computationally costly to generate, renders these kind of approaches applicable only for small problems and simple motion primitives, or for problems where higher computational times are acceptable. To reduce the computational cost deriving from a direct application of complex motion primitives, another widespread approach is based on the discretisation of the search space, to produce a lattice of configurations that are connected by a discrete set of precomputed motion primitives [31, 32, 33, 34].

While all of these solutions are capable of producing smooth G^1 or even G^2 continuous paths, that can be followed by a nonholonomic robot, they do not explicitly take into account the comfort perceived by a human user following the path that, as shown in Chapter 5, is related with the overall jerk. Moreover, solutions relying on the direct application of complex motion primitives, while being more robust and guaranteeing the feasibility of the generated path, often require elevated computational times, that are not applicable for our purposes. Indeed, for our motion planning system to be applicable on a robotic assistant for an elder user performing a social activity within a public space, the generation of a valid solution should take a few seconds at worst.

3.2 Socially acceptable navigation in human populated environments

Given the importance of being able to avoid unforeseen static and moving obstacles to develop robots capable of navigating autonomously in a public environment, over the years an increasing number of solutions have been developed, to tackle this problem in different scenarios and under different assumptions. Since we are dealing with assistive robotics applications, our robot is moving mainly in public spaces, so usually dynamic obstacles correspond to other pedestrians moving in the same area. In general, through the sensing system available on board of the robot, it is possible to determine the position and velocity of surrounding humans quite accurately. Unfortunately, it is not possible to know in advance their intentions and goals. Therefore, usually, the predicted future state for each nearby pedestrian is modelled as a random variable, associated with a certain probability distribution. We discuss now some solutions that are relevant and related to this problem. In [35], the reactive avoidance problem is formalised as a chance constrained optimisation, and an efficient, approximated solution is proposed. However, this approximation is valid for "small" robots and Gaussian distributions of the uncertainties on the obstacle future positions. In [36], the set of velocities causing a collision with an obstacle before a given time is determined. Other solutions are based on the construction of a probabilistic occupancy map, and on the search of a safe path using Probabilistic Roadmaps [37]. While these solutions try to model in probabilistic terms the future states of surrounding obstacles, none of the mentioned techniques

take explicitly into consideration the fact that these moving obstacles are actually humans, and therefore that it would be possible to apply existing human motion models to obtain more accurate predictions of their behaviours.

More directly related to our approach, based on an explicit consideration of human motion, are solutions that model the trajectory followed by pedestrians in a specific environment as Gaussian Processes [37], interactive Gaussian processes [38] or Hidden Markov Models [39]. The probabilistic information on the future position of the obstacles is used in different ways to find probabilistically safe trajectories. In this line of work, a large number of prior observations in a specific environment is used to fit the model parameters. For example, Risk-RRT [40] is based on the modelling of possible trajectories using a mixture of Gaussian Processes, and on the search of the solution using an adaptation of the RRT path planning algorithm [19]. The disadvantage of this kind of solutions is that they are applicable only to the specific scenarios for which the probabilistic model has been constructed (e.g. a corridor, a room, etc.). Whenever a different scenario has to be tackled, a new dataset of observations has to be constructed and used to produce a new, effective model. Our efforts are toward solutions that are more generally applicable and not bound to some specific scenario.

More accurate models describing human motion are considered in a different line of papers [41, 42]. The idea is in these cases to use Monte Carlo simulations to predict the future positions of the obstacles. This solution comes at the cost of a non-negligible computation time, which could potentially compromise the possibility of a real-time execution.

Besides, none of these solutions take into explicit consideration the smoothness of the path followed by the robot. Since in our context the synthesised path has to be followed not only by the robot, but also by the assisted human, an explicit consideration of the comfort perceived by the user is required while applying local adjustments to the global, reference path.

Furthermore, a correct understanding of how people move in a crowded environment is key to the reactive planner presented in this thesis. The widely known Social Force Model (SFM) [43] assumes that a person is supposed to be able to move freely in any direction at any time, acting like a mass particle subjected to external forces. On the contrary, empirical evidence shows that, most of the time, pedestrians tend to move forward, i.e. their velocity vector is most often

aligned with their heading, due to the biomechanics of humans. This phenomenon has been observed by several studies [44, 45, 46], which come to the conclusion that a nonholonomic model, e.g. unicycle-like or car-like models, may be more appropriate to describe human motion. The adoption of such models gives a nice interpretation of the mechanism underlying the formation of human trajectories, i.e. the minimisation of the derivative of the path curvature, the jerk [45]. In [47] the Headed Social Force Model (HSFM) is proposed to enhance the traditional SFM by explicitly accounting for the pedestrians' heading and thus retrieving the smoothness of the trajectories.

3.3 Activity Planning: linking high and low-level planning

The objective of the activity planner installed on the FriWalk is to determine, given the constraints, preferences and requirements of the user, a sequence of goals to visit, to maximize the value of the proposed activity. A strict interaction with the motion planner is fundamental for a correct characterisation of the properties and physical parameters associated with each action composing an activity.

Therefore, our activity planning problem bears important similarities with a family of optimisation problems known as Multi-Goal Path and Motion Planning. The objective of these problems is to determine the optimal path visiting a given set of goal locations. Existing solutions are based on the reduction of the problem to a Travelling Salesman Problem, solved exactly for small problems, or approximatively. For example, in [48], a Minimum Spanning Tree greedy approach is employed, and the edges of the tree are iteratively refined by lower-level motion planners until they become feasible. Other solutions presenting a strong synergy between high-level logical reasoning and low-level dynamic motion planning are applied in the literature for various applications, e.g. in [49] a multi-layered approach is adopted for the planning for ground and flying robots with consideration for the vehicles' dynamics.

Another family of planning and optimisation problems related to the activity planner are the Partial Satisfaction Planning (PSP) [50], the Orienteering Problem (OP) [51], and in particular the Tourist Trip Design Problem (TTDP), whose objective is to determine a subset of the available points of interest to visit according to the preferences of the tourists, in order to maximise their satisfaction, given an

upper bound on the overall duration of the activity. Most state-of-the-art techniques to solve the Orienteering-Problem and its variants are based on the application of metaheuristic optimisation algorithms [52]. A rather complete survey on variants of the TTDP and possible solutions can be found in [53]. Unfortunately, to the best of our knowledge, there exist no solutions in literature capable of handling the kind of optimisation problems required by the activity planning system that we seek to develop, involving multiple probabilistic constraints. Indeed, the TSP is focused on the visit of all the nodes of the graph minimising the overall distance, whereas we are interested in maximising the total score achieved by visiting only a subset of the nodes that meet the user preferences, given deterministic and stochastic constraints. Most of the existing TTDP solutions handle exclusively deterministic constraints, others consider just a single stochastic bound. This represents a large limitation of these metaheuristic approaches, rendering them unsuitable to tackle the activity planning problem that we are required to solve, involving the effective handling of a number of probabilistic constraints.

Up to now, we have presented main research trends and relevant literature concerning the different problems that we have to face in order to develop a complete planning framework for assistive robotics. The remainder of this chapter, instead, deals with existing state-of-the-art research in the field of time-optimal trajectory planning for autonomous racing cars, that is the main focus of the second part of the thesis.

3.4 Time-optimal trajectory planning for racing cars

In his famous work Dubins [54] set the stage for the solution of path planning problems for vehicles moving at constant speed, with limited curvature radii and described by a kinematic model. More recently, Sanfelice et al. [55] applied the Pontryagin Maximum Principle to generalise Dubins results. Despite the recognised importance of these contributions, their applicability is limited in all cases in which dynamic effects, acceleration bounds and slipping constraints cannot be neglected (e.g., high speed vehicles or aggressive manoeuvres).

Several different techniques have been developed capable of effectively taking into account dynamic effects while optimising the global trajectory on a given

racing track. These techniques can be roughly classified into two different categories. On one hand, many solutions are based on the application of complex nonlinear optimisation tools and on the formalisation of the problem as an optimal control or as a geometric optimisation problem. On the other hand, we find solutions based on a direct, incremental exploration of the configuration space, inspired and adapted from the classical path planning algorithms presented in Chapter 2.

When modelling the minimum lap time problem as an optimal control (OCP), dynamic equations governing the motion of the system are translated into constraints to satisfy. The solution of the obtained OCP can be based both on indirect or on direct methods. Indirect methods are based on the Pontryagin Maximum Principle [56], giving the necessary conditions for optimality, that are translated into a set of differential equations with boundary conditions. The resulting problem is therefore a two-points boundary value problem (TPBVP), associated with a cost functional to optimise, and can be solved using several numerical techniques. Solutions based on this approach can be found for example in [57, 58, 59, 60]. On the other hand, direct methods are based on the direct translation of the OCP into a discrete, constrained nonlinear programming problem (NLP), solved by standard numerical algorithms. Solutions based on a direct method are presented in [61, 62], while [63] reports a detailed comparison between the two methods. The major drawbacks of this kind of tools is the difficulty in the set-up, in the selection of the different parameters, and the lack of any convergence guarantee. In addition, this approaches lack the flexibility to easily model more complex scenarios, for example the presence of an obstacle along the way, that introduces a non-convex, binary choice in the optimisation problem, i.e. on whether to pass on the left or right side. Therefore, over the years, also different and alternative approaches have been developed, based on the fast generation and selection of feasible kinematic trajectories with a direct search over the configuration space, using and adapting classical path planning search algorithms, such as RRT or RRT* [64, 26, 20], and by incorporating information on the dynamic properties and constraints [65, 66, 67, 68, 69]. The main weaknesses of these approaches are the elevated computational times required to yield a solution of high quality, and the necessity of sampling and efficiently determining nearest sets of configurations in non-metric and non-smooth spaces.

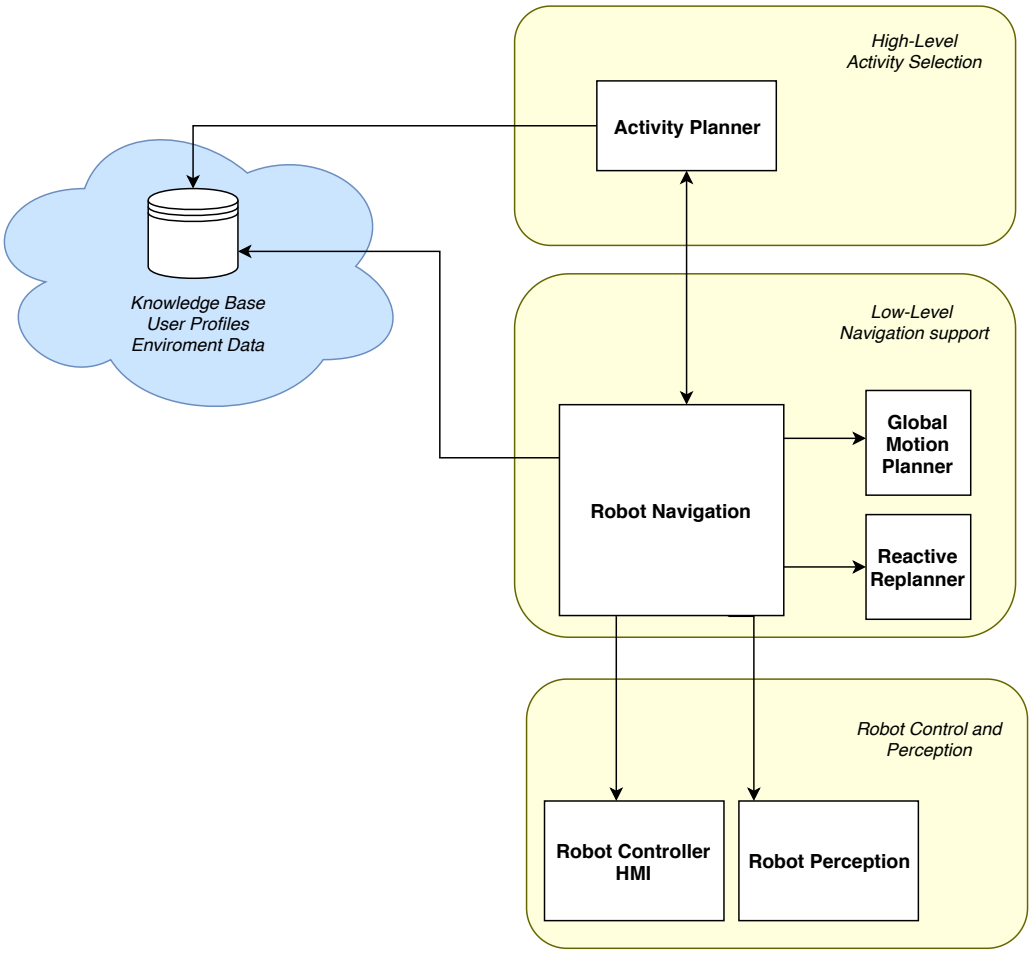
Regarding the real-time update of the reference trajectory in response to

unexpected mutations of the environment (e.g. the presence of an unforeseen obstacle), in general, a hierarchical two-level scheme is applied. On the higher level, a path planner determines the adjustments to the reference path and, then, a low level tracking controller is employed to follow this path[70, 71, 72]. The main drawbacks of these kind of approaches is that either they apply a simple vehicle model to reduce the computational time or, otherwise, they require too high computational resources to be computed on board of the vehicle in real-time, and are therefore not well suited for autonomous racing. Other approaches, where a one-level scheme is applied, and obstacle avoidance and control are solved together, are for example [70, 73], where the problem is modelled as an NLP, but the solution requires prohibitively long solve times, or [74], that achieves low compuational times and is therefore applicable for real-time applications, but it assumes to know in advance on which side the obstacle should be overtaken, that is quite a strong assumption. In addition, a hierarchical Model Predictive Control (MPC) solution has been applied to control miniature racing cars and avoid obstacles lying on the lane [75].

Part I

Assistive Robotics Applications

The first part of the thesis focuses on the different kind of low and high-level planning problems that we are required to solve for the development of a complete planning system for an assistive robot. The following diagram illustrates the different interacting components that are part of the software infrastructure deployed on the robotic rollator to provide it with navigation assistance capabilities.



Low Level

From a bottom-up perspective, at the lower level, the robotic vehicle has been equipped with a sensing and actuation system, to allow it to localise itself within a

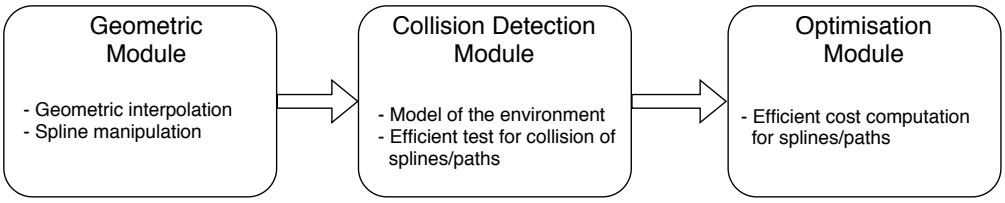
known environment, and to navigate autonomously or in cooperation with the user. While this part of the research is outside the scope of this thesis, we will provide a brief description of the general principles and choices that have been made during the development of the assisitive robot. The localisation of the vehicle during the navigation is performed by means of a camera mounted in front of the vehicle, and a set of visual markers (QR codes) placed on the ground, whose absolute coordinates are known and used, together with odometry data obtained from optical encoders mounted on the wheels, to yield highly accurate self-positioning information. Regarding the actual robot navigation, different guidance solutions have been developed, with main focus on the effective sharing of authority between the assistive walker and the elder, that, depending on his cognitive capabilities, can be given more or less authority and freedom during the execution of the navigation task.

At the intermediate level of the software infrastructure we find the low-level planning components, i.e. the motion planning and the reactive planning modules. The motion planning takes as input the known map of the environment, the initial position of the robot and the desired goal destination, and generates a collision free, safe path specifically tailored to be comfortably followed by an human. The aim of the reactive planner, on the other hand, is to locally and dynamically adjust the global path generated by the motion planner on real-time during the navigation, to cope with unforeseen external factors that render the execution of the original plan infeasible. The state of the surrounding environment is provided by the low-level sensing system, and contains information on the position of static obstacles and position and velocity of approaching pedestrians. Through this information, the algorithm is able to monitor the risk of collision with the surrounding obstacles while moving along the reference path. Whenever the probability of an accident exceeds a certain threshold value, a local replanning is applied, to adjust the reference trajectory and render it safe again.

Common building blocks

All the different kind of motion planning problems that we are required to solve, share some common core building blocks, depicted in the following diagram, that are integrated and combined together in different ways, depending on the specific

problem at hand.



Firstly, we need an effective way to generate, manipulate and adjust (portions of) a geometric path to be followed by the robot. Secondly, once a geometric path has been computed, the planning algorithm needs to check whether the path is feasible, i.e. whether there are no collisions with any of the known obstacles. Thirdly, to produce an "optimal" solution, once a path has been synthesised, we need some efficient way to evaluate its quality/cost, depending on the properties of the path that are of interest for the specific problem under consideration.

For the purposes of this thesis, we employ a well known family of curves as the basic motion primitive composing all the paths synthesised and manipulated by the different developed algorithms, i.e. clothoid curves. The main functionalities exposed by the geometric module are the capability of smoothly interpolating pairs of robot poses with G^1 or G^2 continuity. Concerning the collision detection module, it has to provide an effective way to represent the boundaries of the environment and the obstacles within, and, when queried with a geometric path, to determine whether it lies entirely within the boundaries of the map and is not in collision with any obstacle. Since this operation may be performed many times while the motion planning algorithm explores the environment and produces a large number of candidate paths, the computational efficiency of the developed solution is of fundamental importance and plays a fundamental role in the performance of the planning system. The third module, responsible for the evaluation of the path score, is instead dependent on the specific problem at hand: for the generation of the global path, the score depends on the comfort perceived by the user, that is a function of both the length and the overall smoothness (related with the jerk) of the path. On the other hand, within the reactive replanning module, that has to determine the best adjustment to apply to the reference path, the cost assigned to each alternative path depends both on the probability of collision and on the amount of deviation from the reference.

High Level

At the top level of the depicted software architecture lies the activity planner. It works on an abstraction of the environment, that is represented as a graph, with nodes corresponding to known attractions and points of interest, and edges corresponding to connections between this locations. Therefore, each edge maps to a specific navigation action between a pair of points of interest, and is thus associated with some random variables modelling in probability physical properties associated with that specific action (e.g. travel length, travel time, ...), that are characterized by means of a large number of simulations run on the map of the environment (and adjustable with real data collected during the execution of these actions).

The next chapter of the thesis provides a detailed explanation of the different algorithms and strategies developed to build the geometric and collision detection modules, that are the main building blocks on which the low-level global and reactive planning algorithm are based. Then, the successive three chapters of the thesis present an extensive description and formalisation of the main components characterising the assistive robotics planning architecture (i.e. the motion planner, the reactive replanner and the activity planner), and a discussion of the developed solutions and algorithms, supported by an extensive experimental validation.

4

Clothoid Curves

This chapter is devoted to review the definition and properties of clothoid curves, that are the basic motion primitives adopted for all the developed global and local motion planning algorithms. Based on the recent developments and new state-of-the-art algorithms to efficiently solve different kind of problems involving the interpolation of clothoids with different constraints, appropriate strategies and solutions have been applied to integrate and apply these curves to solve different kinds of motion planning problems.

A clothoid, known also under the names of Euler spiral and Cornu spiral, is a curve in the plane with the property that the curvature varies linearly with respect to the arc length.

A clothoid arc can be represented as a parametric function of the arc-length s via the Fresnel Integrals [76], and is characterised by six real parameters: (x_0, y_0), the initial point, θ_0, the initial angle, κ, κ' the curvatures and the length L. The space coordinates of a point on the clothoid at arc-length s, the corresponding angle and curvature are given by:

$$x(s) = x_0 + \int_0^s \cos\left(\frac{1}{2}\kappa' t^2 + \kappa t + \theta_0\right) \mathrm{d}t \tag{4.1}$$

$$y(s) = y_0 + \int_0^s \sin\left(\frac{1}{2}\kappa' t^2 + \kappa t + \theta_0\right) \mathrm{d}t \tag{4.2}$$

$$\theta(s) = \frac{1}{2}\kappa' s^2 + \kappa s + \theta_0, \tag{4.3}$$

$$k(s) = \kappa' s + \kappa. \tag{4.4}$$

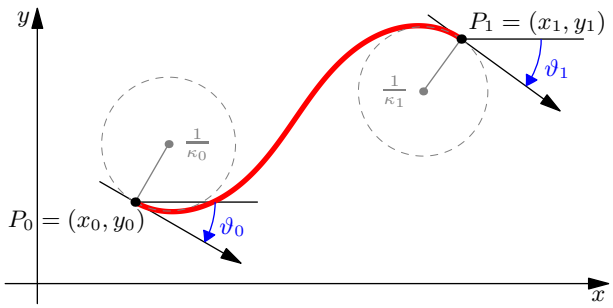


Figure 4.1: An example of a clothoid arc and its parameters.

By using the Fresnel Generalised Integrals, defined as:

$$\begin{aligned} X_j(a,b,c) &= \int_0^1 \tau^j \cos\left(\frac{a}{2}\tau^2 + b\tau + c\right) \mathrm{d}\tau, \\ Y_j(a,b,c) &= \int_0^1 \tau^j \sin\left(\frac{a}{2}\tau^2 + b\tau + c\right) \mathrm{d}\tau, \end{aligned}$$

and given the following identities, obtained by a simple change of variables:

$$\begin{aligned} \int_0^s \tau^j \cos\left(\frac{a}{2}\tau^2 + b\tau + c\right) \mathrm{d}\tau &= s^{1+j} X_j(as^2, bs, c), \\ \int_0^s \tau^j \sin\left(\frac{a}{2}\tau^2 + b\tau + c\right) \mathrm{d}\tau &= s^{1+j} Y_j(as^2, bs, c), \end{aligned}$$

equations (4.1) and (4.2) can be rewritten as:

$$\begin{aligned} x(s) &= x_0 + s\, X_0(s^2\kappa', s\kappa, \theta_i) \\ y(s) &= x_0 + s\, Y_0(s^2\kappa', s\kappa, \theta_i). \end{aligned}$$

In Figure 4.1 is shown an example of a clothoid arc and its parameters.

4.1 Computational Issues

The dependence on the Fresnel Generalized Integrals for the definition of the parametric equations of a clothoid arc in equations (4.1) and (4.2) renders it difficult to efficiently solve a number of interesting problems involving this family of curves. For these reason, a number of different works has been developed to find computationally efficient and numerically robust solutions to these problems. The

remainder of this section is devoted to the presentation of some of these problems involving clothoid curves that are relevant for motion planning applications, and that are used in different ways to solve the various planning problems presented in the next chapters of this thesis. An open source library implementing all the algorithms and efficient solutions discussed in this chapter is available online, at [77].

4.1.1 G^1 and G^2 Hermite interpolation problems

G^1 Hermite interpolation problem The G^1 Hermite Interpolation Problem requires to find the clothoid arc interpolating a given initial position (x_0, y_0) and orientation θ_0 with a given final position (x_1, y_1) and orientation θ_1. More formally, we need to solve the following system:

$$\begin{aligned} x'(s) &= \cos\theta(s), & x(0) &= x_0, & x(L) &= x_1, \\ y'(s) &= \sin\theta(s), & y(0) &= y_0, & y(L) &= y_1, \\ \theta'(s) &= k(s), & \theta(0) &= \theta_0, & \theta(L) &= \theta_1, \end{aligned}$$

where $L > 0$ is the total length of the curve. An efficient, numerically robust solution to this problem is discussed in [76]. The paper shows how this problem can be solved with a single clothoid curve. The problem is thus to find the three unknown parameters κ, κ' and L of this curve, and is solved by reducing the original nonlinear system of three equations and three unknowns to a single nonlinear equation. A root of this nonlinear equation is found using the Newton-Rhapson technique; with the good initial guess proposed in the paper, convergence is achieved in at most four iterations (with a tolerance of 10^{-10}). The found solutions can finally be used to compute the three unknown parameters.

G^2 Hermite interpolation problem The G^2 Hermite Interpolation Problem requires to find the clothoid arc interpolating a given initial position (x_0, y_0), orientation θ_0 and curvature κ_0 with a given final position (x_1, y_1), orientation θ_1

and curvature κ_1. More formally, we need to solve the following system:

$$\begin{aligned}
x'(s) &= \cos\theta(s), & x(0) &= x_0, & x(L) &= x_1,\\
y'(s) &= \sin\theta(s), & y(0) &= y_0, & y(L) &= y_1,\\
\theta'(s) &= k(s), & \theta(0) &= \theta_0, & \theta(L) &= \theta_1,\\
k'(s) &= u(s), & k(0) &= \kappa_0, & k(L) &= \kappa_1,
\end{aligned}$$

where $L > 0$ is the total length of the curve and $u(s)$ is a piecewise constant function to be determined that can be interpreted as a control variable, representing the κ' parameter associated with each clothoid curve composing the solution. Indeed, differently from the G^1 interpolation problem, the G^2 cannot generally be solved with a single arc, but instead requires up to three arcs. In [78] a numerically robust and efficient solution to this problem is proposed. The original nonlinear system is composed by eight equations and ten unknowns, but for the computation of a solution it is reduced to a system of two equations with four unknowns, solved with just a few iterations of Newton-Rhapson. The system is underconstrained, but the feasibility of the problem is strictly dependent on the correct choice of the two free parameters. Thus, a relevant part of the paper is dedicated to the analysis of the correct selection of the values for these parameters.

4.1.2 Intersection of clothoid arcs

Another fundamental issue to adopt clothoid curves as the basic motion primitives for motion planning applications, is the problem of efficiently determining the intersection between pairs of clothoid arcs, and between clothoid arcs and other geometric primitives, such as line segments or polygons. The computation of intersections involving clothoid curves is a challenging task. Indeed, clothoids are spiral curves revolving around the point at infinity. For this reason, the direct application of the Newton-Rhapson method to determine intersections may fail; for example, depending on the initial guess, it may converge to a wrong solution, or, in the presence of multiple solutions, some of them may get undetected. Moreover, the repeated application of Newton methods from a large number of different initial points along the curve is computationally heavy. For these reasons, an efficient solution to this problem is based on a first decomposition of the curve into smaller subcurves with constant curvature signs; each of them is then inscribed into a set

of triangles. The original problem is thus simplified to an intersection between triangles: if a triangle of the first curve overlaps a triangle of the second curve, then the Newton method is applied only on the corresponding subarcs. Often we are required to test for intersection pairs of clothoid splines, i.e. sequences of clothoid arcs joining with G^1 or G^2 continuity. In these circumstances, the number of subarcs resulting from the decomposition may be quite large, so, to speed up the computation, it is convenient to organise the resulting set of triangles into a hierarchical spatial partitioning data structure.

Triangle construction

To inscribe a clothoid arc into a triangle, first it must be split into subarcs with constant curvature sign, so that each of them is convex or concave. Notice that, each clothoid arc can have at most one change of sign in the curvature, for $s = -\kappa/\kappa'$. If $s \in [0, L]$, i.e. the inflection point lies within the arc, a split is applied at this point to obtain two clothoid subarcs.

Each of the clothoid subsegments satisfying the constraint on the curvature sign has to be furtherly split into subarcs such that the travelled angle is less than a chosen parameter α_{max}. To enclose a clothoid subarc inside a triangle, α_{max} has to be smaller than π. However, in practice, to improve the quality of the solution, and determine a set of triangles with a good fitting of the original curve, much lower values of α_{max} are chosen (e.g., we usually adopt the values $\pi/8$ or $\pi/16$).

Given each clothoid subsegment originating from (x_0, y_0) with initial orientation θ_0 and terminating in (x_1, y_1) with final orientation θ_1, the circumscribing triangle is constructed by selecting the two endpoints of the arc as two vertices, i.e. $P_0 \triangleq (x_0,\ y_0)$, $P_1 \triangleq (x_1,\ y_1)$. The third vertex $P_2 \triangleq (x_2,\ y_2)$ of the triangle $P_0P_1P_2$ is determined as the intersection of the tangent lines in P_0 and in P_1. The tangent at P_0 is $y = m_0x + q_0$, where $m_0 = \tan(\theta_0)$ and $q_0 = y_0 - m_0x_0$. In the same way, the tangent at P_1 is $y = m_1x + q_1$, where $m_1 = \tan(\theta_1)$ and $q_1 = y_1 - m_1x_1$. Henceforth, the point P_2 has coordinates:

$$P_2 = \left(-\frac{q_0 - q_1}{m_0 - m_1},\ \frac{m_0q_1 - m_1q_0}{m_0 - m_1} \right), \tag{4.5}$$

Efficient spatial partitioning: AABB trees

In the previous section we illustrated how it is possible to cover a clothoid arc with a set of triangles. However, for large clothoid splines, the number of triangles grows quickly. The determination of all the intersections between two splines requires a test for overlap between all pairs of triangles of the two splines. The computational time is thus quadratic in the number of triangles (i.e. in the size of the splines), and becomes prohibitive for various real-time applications. For this reason, it convenient to organise the sets of triangles into efficient data structures. In particular, we use Bounding Volume Hierarchies, with Axis Aligned Bounding Boxes (AABB) as Bounding Volume Primitives, [79]. Bounding volume primitives are simple volumes encapsulating more complex objects. The idea is that, by using simple bounding shapes, the test for intersection is computationally cheap, and a first, broad-phase, collision detection test can be performed, to filter out and reduce the number of more complex, accurate and costly intersections.

The construction is done by determining the minimum and maximum coordinates of the triangle $P_0P_1P_m$. The intersection can be determined with at most 4 comparisons, by checking whether one of the two AABBs lies completely on one side of the other. Let p_1 and p_2 be two AABB bounding primitives, where p_1 lies between $(x_1^{\min}, y_1^{\min})$ and $(x_1^{\max}, y_1^{\max})$, and analogously for p_2. There is no overlapping if

$$\left(x_1^{\max} < x_2^{\min}\right) \vee \left(x_2^{\max} < x_1^{\min}\right) \vee \left(y_1^{\max} < y_2^{\min}\right) \vee \left(y_2^{\max} < y_1^{\min}\right).$$

Once the AABBs are constructed around each triangle, we organize them in a hierarchical tree structure, called AABB tree, where each node represents a bounding box. The children of a node are contained inside the parent. A leaf is associated to each of the original triangles, see Figure 4.2. The construction of the tree is performed with a top-down approach: first the AABB enclosing the set of all boxes is determined. The successive splitting is based on spatial considerations: the splitting axis is chosen as the longest side of the parent box dimensions. The mean coordinate along this axis is selected as the threshold value to partition the current set of boxes into two subsets. We compare the position of the centroid of each box with the splitting axis coordinate found at the previous step to classify a box into one of the two new subsets. The procedure is recursively applied to each

of these two subsets to generate two subtrees, corresponding to the two children of the current root node. The recursion terminates when the current set of boxes is empty or contains a single element, that corresponds to a leaf node, associated with a single triangle. The pseudo-code is shown in Algorithm 1. Given two AABB trees associated with two clothoid splines, the determination of the pairs of intersecting triangles can be performed by a recursive traversal of the two trees. First the bounding boxes associated with the root nodes are tested for intersection. If the test fails, the two curves are not intersecting, and the algorithm terminates. Otherwise, the test for intersection is repeated recursively on all the pairs of children nodes of the two trees. If the test for intersection succeeds, the traversal of the trees continues until pairs of leaf nodes are reached. At this point, the more accurate (but more expensive) triangle-triangle intersection test is performed [80]. For all pairs of intersecting triangles, the actual clothoid-clothoid intersection test is performed on the corresponding clothoid segments. The pseudo-code is shown in Algorithm 1. The complexity of building an AABB tree is $\mathcal{O}(n \log n)$ on average and $\mathcal{O}(n^2)$ in the worst case [81], whereas the complexity of a collision detection or an intersection test is on average $\mathcal{O}(m \log n)$, where n, m are the number of primitives of the two trees. In the worst case, the cost increases to $\mathcal{O}(nm)$. Without this algorithm the complexity of an intersection test would be always $\mathcal{O}(nm)$, because each primitive of the first tree must be compared with all the primitives of the second tree.

In Table 4.1 are reported the computational times required for finding the intersection of two clothoid splines (with n and m triangles, respectively) with and without the adoption of the hierarchical partitioning strategy. The results are obtained on a 2.9GHz Intel Core i7 with 16Gb ram. It is clear how the hierarchical approach is much more efficient, especially for large problems involving many triangles.

Size (n, m)	$(54, 53)$	$(265, 344)$	$(340, 333)$
AABB tree	0.2 ms	2 ms	2 ms
plain	3.5 ms	170 ms	132 ms

Table 4.1: Comparison of the computational times to determine intersections between pairs of clothoid splines, with and without the spatial partitioning strategy, for different problem sizes.

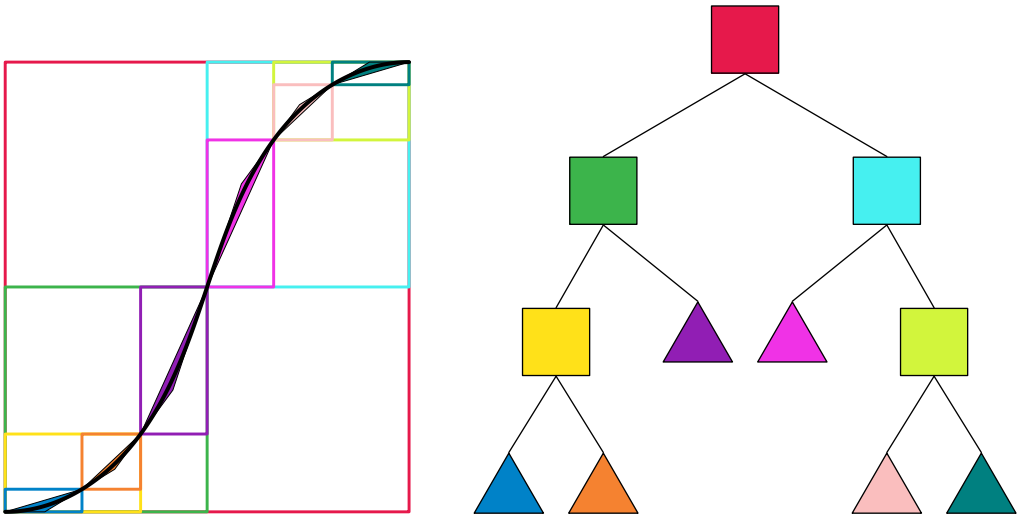


Figure 4.2: Bounding boxes constructed over the clothoid spline (left) organised as an AABB tree (right). The different colours match the boxes and the triangles of the left with the corresponding node at the right.

Algorithm 1: AABB tree construction and intersection

```
1  BuildTree (Boxes);
2  begin
3  |  if Boxes = ∅ then return nil;
4  |  if #Boxes = 1 then return (Boxes_1, nil, nil);
5  |  root ← Bbox(Boxes); /* finds global Bbox                          */
6  |  w ← Width(root); h ← Height(root) ;
7  |  if w > h then
8  |  |  split ← mean_x(Boxes); /* splits vertically                   */
9  |  |  B_1 ← {b ∈ Boxes s.t. centroid_x(b) < split};
10 |  |  B_2 ← {b ∈ Boxes s.t. centroid_x(b) ≥ split};
11 |  |  return (root, BuildTree (B1), BuildTree (B2));
12 |  else
   |  |  /* same as above but horizontally                           */
13 |  end
14 end
15 Intersect (T_1,T_2 );
16 begin
17 |  if (T_1 = nil) or (T_2 = nil) then return ∅;
18 |  if isLeaf (T_1) and isLeaf (T_2) then
19 |  |  if TriInt (getT (T_1)),getT (T_2)) then return (getT(T_1), getT(T_2));
20 |  |  return ∅;
21 |  end
22 |  res ← ∅; /* N.B: Children(leaf) returns leaf itself              */
23 |  foreach c_1 ∈ Children(T_1) do
24 |  |  foreach c_2 ∈ Children(T_2) do  res ← res ∪ Intersect(c_1, c_2) ;
25 |  end
26 |  return res; /* returns pairs of overlapping triangles           */
27 end
```

5 Motion Planning

5.1 General Overview

In this chapter, we introduce the problem of motion planning for assistive robotics. The development of a motion planner tailored for robots employed to support and assist elder humans during the navigation of complex environments is a challenging task, that requires the consideration of different aspects. Like in the case of a classical motion planning problem, the generated path has to connect the initial and the final configurations, to lie entirely in the free configuration space and to be feasible, given the physical constraints of the robot. In addition, this specific kind of motion planning problems require an explicit and careful consideration of the geometry and smoothness of the produced path, that has to be followed by human users with the support of the robotic assistant. Therefore, additional constraints and a more complex objective function need to be considered, in order to maximize their perceived comfort.

Specifically, a number of studies has been conducted over the years to analyse the (lack of) comfort of humans moving on ground vehicles, such as cars or trains. It has been recognised how the discomfort increases with body accelerations and jerk. These terms, in the context of a mobile robot following a given reference trajectory, are in turn related with the curvature of the path and its discontinuities. Therefore, for the generation of a path optimizing the user comfort, both its length and jerk should be minimized.

Another important requirement for a motion planning module deployed on a robot providing assistance to an elder user, is the limited amount of time available to

generate a solution. Indeed, in order to render the system trustable and compelling from the point of view of the user, the generation of a plan should take no longer than a few seconds.

To develop an efficient solution to this challenging task, it is convenient to isolate two different sub-problems: 1) generation of a collision free sequence of waypoints to reach the destination, and 2) synthesis of a path joining the waypoints, respecting the physical constraints, and minimizing the cost function associated with the problem. These two sub-problems can be solved separately during different phases of execution of an iterative algorithm, discussed in the next sections.

5.2 Problem Statement

This section is devoted to present the model of the robot, the cost function combining a set of discomfort indices to be minimised, and from these a formalization of the motion planning problem at hand.

5.2.1 Platform Model

The large majority of assistive robots can be modelled from a control perspective as unicycle–like or car–like vehicles. With reference to Figure 5.1, let $\langle W \rangle = \{O_w, X_w, Y_w, Z_w\}$ be a fixed right-handed reference frame, whose plane $\Pi = X_w \times Y_w$ is the plane of motion of the vehicle, Z_w pointing outwards the plane Π and let O_w be the origin of the reference frame. Let $\mathbf{x} = [x, y, \theta]^T \in \mathbb{R}^2 \times S$ be the kinematic configuration of the platform, where (x, y) are the coordinates of the mid–point of the rear wheels axle in Π and θ is the orientation of the vehicle w.r.t. the X_w axis (see Figure 5.1). Assume that the kinematic model of the mechanical platform can be assimilated to a unicycle–like vehicle, described by

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v\cos(\theta) \\ v\sin(\theta) \\ \omega \end{bmatrix} \tag{5.1}$$

where v and ω are the forward and the angular velocities.

As previously mentioned, a fundamental aspect of a motion planner tailored for assistive robots, is the explicit consideration and optimization of the comfort

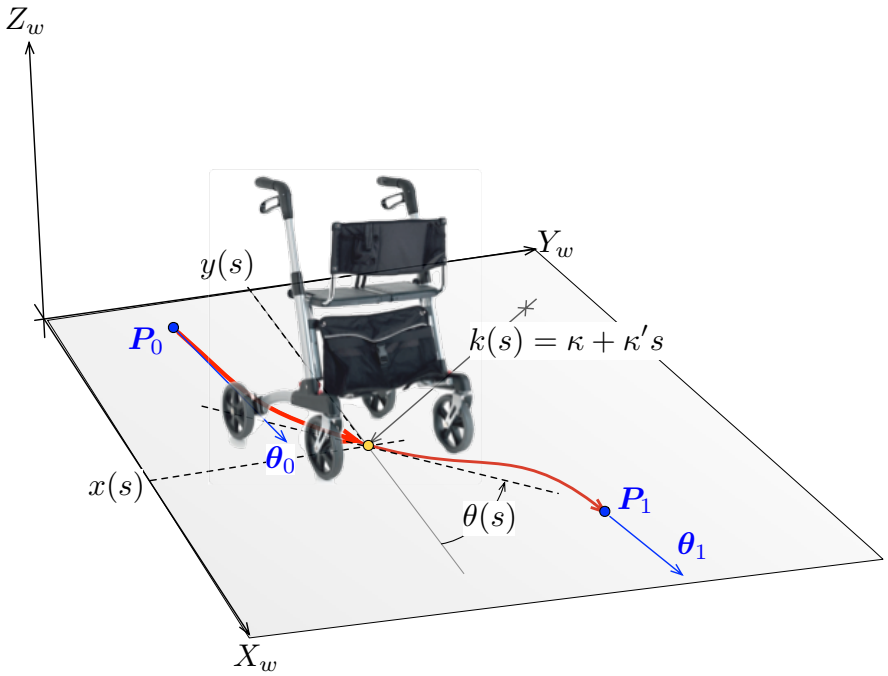


Figure 5.1: The reference system for the trajectory planning with clothoidal elements. [Published in [11]]

experienced by the user. This, in turn, translates to a minimization of the accelerations and jerk, determined by the geometry of the generated path, that should be continuous at least up to the curvature [82]. Since the unicycle model in (5.1) does not satisfy this requirement, we propose a dynamic extension of the unicycle model, whose kinematic ODEs in the cartesian xy-coordinates are given by:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\bar{\delta}} \end{bmatrix} = \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \\ \bar{\delta} \\ 0 \end{bmatrix} v + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \bar{\omega}, \tag{5.2}$$

where $\bar{\delta}$ can be interpreted as the steering angle and $\bar{\omega}$ its velocity.

The same model can also be applied if our vehicle is natively a car–like vehicle

with rear traction, described by

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\delta} \end{bmatrix} = \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \\ \tan(\delta)/l \\ 0 \end{bmatrix} v + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \omega_\delta, \tag{5.3}$$

where δ is the actual steering angle, ω_δ is the normalised angular velocity of the steering wheel and $l > 0$ is the wheelbase. In this case, the kinematic model (5.3) can be easily reduced to the model (5.2) by using the auxiliary control input

$$\bar{\omega} = \left(\delta^2 + 1\right) \omega_\delta,$$

and assuming that $\tan(\delta) \approx \bar{\delta}$ (see [83, 84]).

In [45], a large number of real trajectories followed by groups of people moving in a real environment is analysed. The results show how these trajectories are well approximated by the optimal solution of model (5.2).

5.2.2 Comfort

During the last years, a relevant amount of research has been dedicated to an accurate modelling and a deep understanding of the trajectories followed by walking humans. An important research direction regards the determination of the functional that a person optimises while walking, and of a suitable dynamic system to describe it. While convincing results on the latter problem are available in the form of a system of differential equations that governs the human locomotion, the optimised functional is much more difficult to understand and model accurately. One possible formulation, based on model (5.2), has been developed in [45], where it is shown that an human tends to move with a constant longitudinal velocity along its reference trajectory. In addition, this trajectory is determined by optimising a functional representing the minimum square of the jerk, that is defined as the derivative of the acceleration. An extended experimental validation has shown the effectiveness of the proposed human model, matching the ground truth measurements taken from real pedestrians very satisfactorily.

The expression of the jerk in terms of the velocity v and the curvature k is

$j = v^2\dot{k}$. For the optimal curve resulting from the human motion model, i.e., the clothoid, the curvature is a linear function of the arc-length s (see chapter 4). Thus we specialise the expression of the jerk combining the information of the curvature, which yields $j = v^2\kappa'$. By assuming the velocity to be constant, according to [45], we can consider as a measure of the trajectory jerk the square of κ'. As a consequence, a possible cost index that can be considered for the human comfort is the minimisation of the jerk, i.e.

$$T_1 = f_1(\kappa'^2),$$

where $f_1(\cdot)$ is a suitable function to be defined. While the jerk is obviously relevant, other comfort indices could be considered as well. From the previous analysis, a minimisation of the overall curvature can also be a suitable cost index to be minimised to increase the comfort, which can be denoted as:

$$T_2 = f_2(\kappa, \kappa').$$

Finally, according to [82], the minimisation of the path length could be considered as a relevant comfort index, i.e.

$$T_3 = f_3(\kappa, \kappa'),$$

where T_3 is equal to the time optimal path, the velocity v being assumed constant.

5.2.3 Problem Formulation

This subsection is dedicated to the formal definition of the motion planning problem that we are required to solve.

First of all, the cost index to optimize contemplates different aspects:

- the overall length of the path
- the time required to walk to the destination
- the comfort along the path

Indeed, to maximize the satisfaction of the user, the overall length of the path, and also the expected travel time, should be minimized, while preserving

the smoothness and perceived comfort. Mathematically, the objective of this optimization problem can be stated as the weighted sum of the three terms:

$$\min J = w_1 \int_\gamma \mathrm{d}s + w_2 \int_\gamma \mathrm{d}t + w_3 \int_\gamma c(s)\,\mathrm{d}s \tag{5.4}$$

where the first term represents the overall path length, the second the travel time and the third the discomfort index.

The problem that we are solving, is thus to determine a collision-free path connecting two given positions with a spline of clothoid curves, optimizing the cost function (5.4).

5.3 Proposed Solution

As mentioned in the introduction of this chapter, given the particular kind of application, a fundamental aspect that we need to consider is the computational time required to determine a solution, that should be limited to at most a few seconds. For this reason, our solution is based on the identification and separate solution of two different sub-problems:

- Generation of a reference route, modelled as a spline of line segments, determining a collision-free "corridor" within the known map of the environment
- Selection of a sequence of waypoints along the reference route, and interpolation with a spline of clothoids characterised by G^2 continuity and minimizing the target functional

5.3.1 Route planning

The generation of the reference collision-free route, i.e. a spline of line segments connecting the start and the goal location, while avoiding collisions with the obstacles, can be implemented using standard motion planning algorithms, both combinatorial and sampling based. These route represent a sort of reference "corridor", along which the complete path will be sought. The use of simple motion primitives, based on straight line segments, to compose a collision-free non-smooth corridor from the start to the goal location on the map, allows this first phase of the algorithm to be completed very quickly. This point is fundamental to render

the whole approach applicable within the context of assistive robotics, that, as previously mentioned, requires the computation of a solution in a limited amount of time.

In particular, we have applied a well-known sampling based motion planning algorithm, that is a variant of the RRT*, called Informed-RRT* (I-RRT*) [85]. Initially, while a feasible solution has not been found, this algorithm behaves identically to RRT*. A new point x_{new} is sampled within the free configuration space during each iteration of the algorithm, and is connected to its closest node $x_{nearest}$ among the subset of nodes of the tree that are directly reachable from x_{new} (i.e., the subpath associated with the new edge of the tree must lie in the free configuration space). When a node is inserted, the path and the cost to reach neighbour nodes already in the tree by passing through this new node are computed. For all the neighbours for which this new path is collision free, and its cost is lower than the current cost, a "rewiring" operation is performed, i.e., the new node is set as their parent, and the costs of the associated subtrees are updated accordingly. Differently from traditional RRT*, when a solution has been found I-RRT* does not sample all the possible regions of the free space anymore, but it samples possible candidates only within an elliptical region of the configuration space for which an heuristically calculated total-cost is lower than the current optimal cost.

The pseudocode illustrating the main steps of this algorithm is shown in Algorithm 2. In Figure 5.2, on the top-left is shown (in blue) the tree constructed after a few iterations of the I-RRT* algorithm.

5.3.2 Path generation

An heuristic algorithm is then applied to this spline of line segments, to extract a set of waypoints, that are interpolated with a spline of clothoids optimizing the comfort index. If some portions of the generated spline are in collision with some obstacles, the number of waypoints placed in those zones of the map is increased, to force the interpolated path to stay closer to the reference, collision free route path. This process is repeated iteratively until a valid solution is found, or the number of attempts exceeds the maximum allowed value. As shown in Algorithm 3, the sequence of waypoints is generated initially from the sequence of points composing the segment spline. A filtering is applied to remove superfluous,

Algorithm 2: Generation of a route (spline of segments) from a given starting position to a given goal, using I-RRT*.

```
   Data: map, start, goal
   Result: Route from start to goal
 1 function GenerateRoute
 2     tree ← initTree(start, goal)
 3     ellipse ← initEllipse(start, goal)
 4     termConditions ← initTermConditions()
 5     while (not termConditions) do
 6         samplePos ← sampleNextPosEllipse(map, ellipse)
 7         parentFound ← findParent(samplePos, map, tree, parent, minCost)
 8         if parentFound then
 9             newNode ← insertNewNode(samplePos, tree, parent, minCost)
10             rewireNeighbours(newNode, map, tree)
11             if goalInRange(samplePos, goal, map) then
12                 connectGoal(node, goalNode, tree)
13             end
14             updateEllipse(ellipse)
15         end
16         termConditions ← updateTermConditions(termConditions)
17     end
18     route ← extractRoute(tree)
19     return route
20 end
```

aligned intermediate points (line 2). Then, a certain number of waypoints is inserted near to the location where there is a change of direction in the segment spline, to force the interpolated path to stay close to the reference route (line 13). The distance between consecutive waypoints is a parameter of the algorithm. If the clothoid connecting two waypoints was in collision during the previous iteration, the distance between consecutive waypoints is reduced, depending on the current number of attempts (line 11). Finally, the sequence of waypoints is interpolated. If the resulting clothoid spline is not colliding with any obstacle, the algorithm terminates, otherwise, the indexes of the colliding segments are stored and used to perform a new iteration of the algorithm.

Figure 5.2 shows the outcome of the different stages of the algorithm, from the selection of the initial route by means of the I-RRT* algorithm, to its filtering and waypoint selection, and finally to the G^2 interpolation by means of a clothoid

spline.

Algorithm 3: Generate a collision-free spline of clothoids interpolating a segment spline

```
Data: map: map of environment, points: spline of line segments
Result: Smooth path interpolating the given waypoints
function GeneratePath
    points ← Filter(points)
    collisionSegments ← ∅
    for i ∈ 1 ... attempts do
        waypoints ← ∅
        for j ∈ 2 ... Length(points) do
            p1 ← points[j − 1]
            p2 ← points[j]
            step ← STEP_SIZE
            if (j − 1) ∈ collisionSegments then
                step ← (2/3)^(i−1) STEP_SIZE_COLLISION
            end
            waypoints ← waypoints ∪ GenIntermPts(p1, p2, stepLen)
        end
        spline ← InterpolateSpline(waypoints)
        collisionSegments ← Collision(spline, map)
        if Empty(collisionSegments) then
            return spline
        end
        points ← waypoints
    end
    return ∅
end
```

G^2 Spline Interpolation

As discussed in the previous section, once a sequence of waypoints has been generated, we interpolate it using a spline of clothoids with G^2 continuity. This is a sequence of n clothoid arcs, where the curvature is a continuous piecewise linear function of the arc-length. Such a curve $\mathcal{C}$ is given by a finite collection of real parameters $0 = s_0 < s_1 < \ldots < s_n$ and $(\boldsymbol{p}_i, \theta_i, \kappa_i, \kappa_i')$ with $\boldsymbol{p}_i = (x_i, y_i) \in \mathbb{R}^2$ and $i = 0, 1, \ldots, n-1$. For each i, $\mathcal{C}$ is a clothoid over $[s_i, s_{i+1}]$. For a G^2 clothoid spline, each pair of consecutive segments i and $i+1$ satisfies the following continuity

conditions:

$$\begin{aligned} H_{i,0} = &\quad x_i + L_i X_0\big(\kappa_i' L_i^2, \kappa_i L_i, \theta_i\big) - x_{i+1} = 0, \\ H_{i,1} = &\quad y_i + L_i Y_0\big(\kappa_i' L_i^2, \kappa_i L_i, \theta_i\big) - y_{i+1} = 0, \\ H_{i,2} = &\quad \frac{1}{2}\kappa_i' L_i^2 + \kappa_i L_i + \theta_i - \theta_{i+1} = 0, \\ H_{i,3} = &\quad \kappa_i' L_i + \kappa_i - \kappa_{i+1} = 0, \end{aligned} \tag{5.5}$$

where $L_i = s_{i+1} - s_i > 0$ and $i = 0, 1, \ldots, n-1$.

The first two equations enforce that the coordinates of the final point on segment i coincide with the coordinates of the first point on segment $i+1$. The third and the fourth equations, instead, enforce that the final tangent and curvature of segment i is equal to the initial tangent and curvature of segment $i+1$. All the constraints (5.5) can be expressed in vector form, by introducing the function $\boldsymbol{H}_i(\theta_i, \kappa_i, \kappa_i', L_i, \theta_{i+1}, \kappa_{i+1}) = \mathbf{0} \in \mathbb{R}^4$ for $i = 0, 1, \ldots, n-1$. For each clothoid composing the spline, only two of the six parameters are fixed, i.e. the initial coordinates x_0 and y_0, while the other four parameters are free, and need to be determined. To obtain a G^2 interpolating spline, all the constraints (5.5) need to be satisfied. Therefore, the optimization problem to be solved contains $4N + 2$ free parameters, where N is the number of waypoints. Indeed, there are $N + 1$ angles θ_i and curvatures k_i, and N curvature slopes k_i' and lengths L_i. On the other hand, there are $4N$ constraints (the functions $\boldsymbol{H}_i$). As a consequence, the nonlinear system of constraints (5.5) is not completely determined; this provides us with the possibility to optimize a cost functional related to the human comfort during the solution of the interpolation problem. The algorithm to solve the G^1 Hermite Interpolation Problem with clothoids illustrated in chapter 4 allows an efficient solution to the first three equations of (5.5). Since the sequence of waypoints is taken as input, the value of all the coordinates x_i and y_i is fixed and known in advance. Thus, the only free parameters that need to be determined to construct the interpolating spline are the orientations θ_i of the clothoid spline at each of the waypoints. In addition, the original nonlinear system can be reduced to a single equation per clothoid segment, to enforce the continuity of the curvature.

In order to optimize the human comfort, we need to define and add to the non-linear problem a comfort index to be optimised, as explained in Section 5.2.2. Specifically, we consider three different functionals $T_j(\theta_0, \theta_1, \ldots, \theta_n)$, $j = 1, 2, 3$, depending on the free variables of the problem, i.e., the unknown angles θ_i:

- Minimise the jerk, which is, in this case, equivalent to the minimisation of the variation of the curvature:

$$T_1 = \sum_{i=0}^{n-1} \kappa_i'(\theta_i, \theta_{i+1})^2; \tag{5.6}$$

- Minimise the integral of the curvature squared,

$$T_2 = \sum_{i=0}^{n-1} \int_0^{L_i} (\kappa_i(\theta_i, \theta_{i+1}) + s\kappa_i'(\theta_i, \theta_{i+1}))^2 \, \mathrm{d}s \tag{5.7}$$

- Minimise the total length of the curve:

$$T_3 = \sum_{i=0}^{n-1} L_i(\theta_i, \theta_{i+1}). \tag{5.8}$$

Each functional (5.6), (5.7), (5.8) with the constraints (5.5) defines an optimal problem

$$\begin{aligned} \text{Minimise} \quad & T_j(\theta_0, \theta_1, \ldots, \theta_N) \\ \text{Subject to} \quad & \boldsymbol{H}_i(\theta_i, \kappa_i, \kappa_i', L_i, \theta_{i+1}, \kappa_{i+1}) = \mathbf{0}. \end{aligned}$$

where $j = 1, 2, 3$ and $i = 0, \ldots, n-1$, that can be solved using Non-Linear Programming (NLP).

We solve the NLP problem using the IPOPT software [86]. To improve the efficiency of the algorithm, it is possible to explicitly provide the derivatives of the target and the constraints (i.e., gradient and Jacobian). This has been done by analytically differentiating the equations of the G^1 algorithm, i.e., the derivative with respect to the angles θ_i and θ_{i+1} [87].

5.3.3 Path feasibility

Once the clothoid spline trajectory passing through the set of planned points is synthesised, its feasibility, i.e., the fact that is confined in the free space, has to be carefully considered. To reduce the computational times, and simplify the test for collision, we approximate the shape of the rollator by inscribing it into a circle. In

this way, it is sufficient to "buffer" the obstacles, that is, to increase their size by the radius of the robot. This process can be performed only once, at the beginning of the execution of the algorithm. After this processing, collision detection can be achieved by just checking whether the clothoid spline corresponding to the path of the robot collides with any of the enlarged obstacles. The clothoid spline is split and approximated with a sequence of triangles, that are organized into an AABB tree. Also the boundaries of the obstacles, represented as line segments, are organized into an AABB tree, so that tests for intersection can be performed extremely efficiently, as discussed in Chapter 4.

5.4 Experimental Validation

In this section we propose paths synthesis on the map of the Department of Information Engineering and Computer Science of the University of Trento. The map of the building is represented by a set of polygons corresponding to obstacles and walls.

To compare the proposed solution with existing approaches, we have implemented different algorithms to generate the initial sequence of waypoints, and to smooth and interpolate them into a feasible path, and tested them by running a large number of experiments on different pairs of start and goal locations randomly chosen on the map.

The sequence of waypoints joining source and destination has been generated using two different algorithms. The first is based on the decomposition of the map into the well known quad-tree cells [88]. Cells on the boundary between the free space and the obstacles are recursively subdivided into four subcells, until their size is below a fixed resolution parameter. A graph representing free adjacent cells is built, and the shortest path between two nodes is found using the Dijkstra's algorithm [89]. The second algorithm is the Informed RRT* presented in Section 5.3.1, that we have selected as the route planning algorithm to integrate in our solution.

The sequence of points constituting the plan obtained with any of the two is then further processed by removing redundant points aligned on straight lines, and inserting some points before and after each curve, in order to add degrees of freedom for the clothoid spline in the close proximity of a change of direction, as

Table 5.1: Comparison of different discomfort indices and planning algorithms on the two different paths of Figure 5.3. [Published in [11]]

		I-RRT*			Dijkstra		
		Target	Time [s]	Length [m]	Target	Time [s]	Length [m]
Path_1	T_1	0.002	0.90	67.75	4.52	2.39	69.24
	T_2	1.00	1.19	67.66	5.42	2.27	68.46
	T_3	67.66	1.10	67.66	68.44	2.16	68.44
Path_2	T_1	0.19	2.87	116.54	77.79	8.78	121.35
	T_2	1.57	2.89	116.44	25.22	8.20	121.33
	T_3	116.43	2.76	116.43	121.33	8.78	121.33

explained in Section 5.3.2.

Figure 5.3 reports, for a visual comparison, two paths (namely Path_1 and Path_2) generated using the Dijkstra's algorithm and the I-RRT* minimising the T_1 discomfort index (minimum jerk trajectories). The solution with Dijkstra's algorithm looks less natural with respect to the paths generated using I-RRT*. This is due both to the discrete set of configurations available and to the particular structure of quad-trees, having a higher number of cells along the boundaries of the obstacles. For a quantitative comparison between the different solutions, Table 5.1 reports the discomfort index target value, the computation time and the path length for all of the three discomfort indices reported applied to the two sample paths reported in Figure 5.3. The solution based on I-RRT* is the one with the best performance for all cases.

Figure 5.4 shows the curvature of the paths interpolated with the three different cost functionals. The curvatures are overlapping almost everywhere over the path, implying that the constraints imposed by the curvature continuity dominates the path synthesis, except for the initial and the final part of the path. Indeed, the initial and the final curvature are 0 when the adopted functional is T_2 (overall curvature minimisation). Independently from the chosen functional, the range of values of the curvature is much higher for paths generated using Dijkstra, which provides an insight for the results reported in Table 5.1.

The method here proposed has been further compared with a quite popular solution in the literature, which is based on the application of the *elastic bands* (El.B.) [25] to smooth the path, make it more compact and to remove redundant points. This approach is based on the concept of bubbles of free space around discrete configurations composing the path (provided by either Dijkstra's algorithm

Table 5.2: Comparison of the results with and without elastic bands for path 1 and the minimisation of the jerk T_1, using both clothoid splines with curvature continuity and cubic splines.

	I-RRT*			Dijkstra		
	Target	Time [s]	Length [m]	Target	Time [s]	Length [m]
No El.B.-T_1	0.002	0.90	67.75	4.52	2.39	69.24
El.B.-cubics	–	0.21	68.54	–	0.305	69.43
El.B.-T_1	3.30	12.67	65.48	10.26	21.31	65.37

or I-RRT*), which are moved away from obstacles and toward each other (similarly to an elastic band), by means of virtual forces. The obtained points are then usually interpolated with standard cubic splines. Table 5.2 reports a comparison between the previous clothoid trajectories (No El.B.-T_1) with El.B. with cubics for Path 1 of Figure 5.3 minimising the trajectory jerk (T_1 discomfort index). It can be seen that the solution with El.B. interpolated with cubics is faster than No El.B.-T_1, but it results into higher lengths and into a much higher discomfort. Figure 5.5 reports a comparison of the curvature for the El.B. with cubics and with clothoid splines for both the Dijkstra's algorithm and the I-RRT*. It can be seen how El.B.-T_1 produces lower discomfort with respect to the cubic interpolation, that is however higher than the original solution (see Table 5.2). Interestingly, El.B.-T_1 produces shorter paths than No El.B.-T_1.

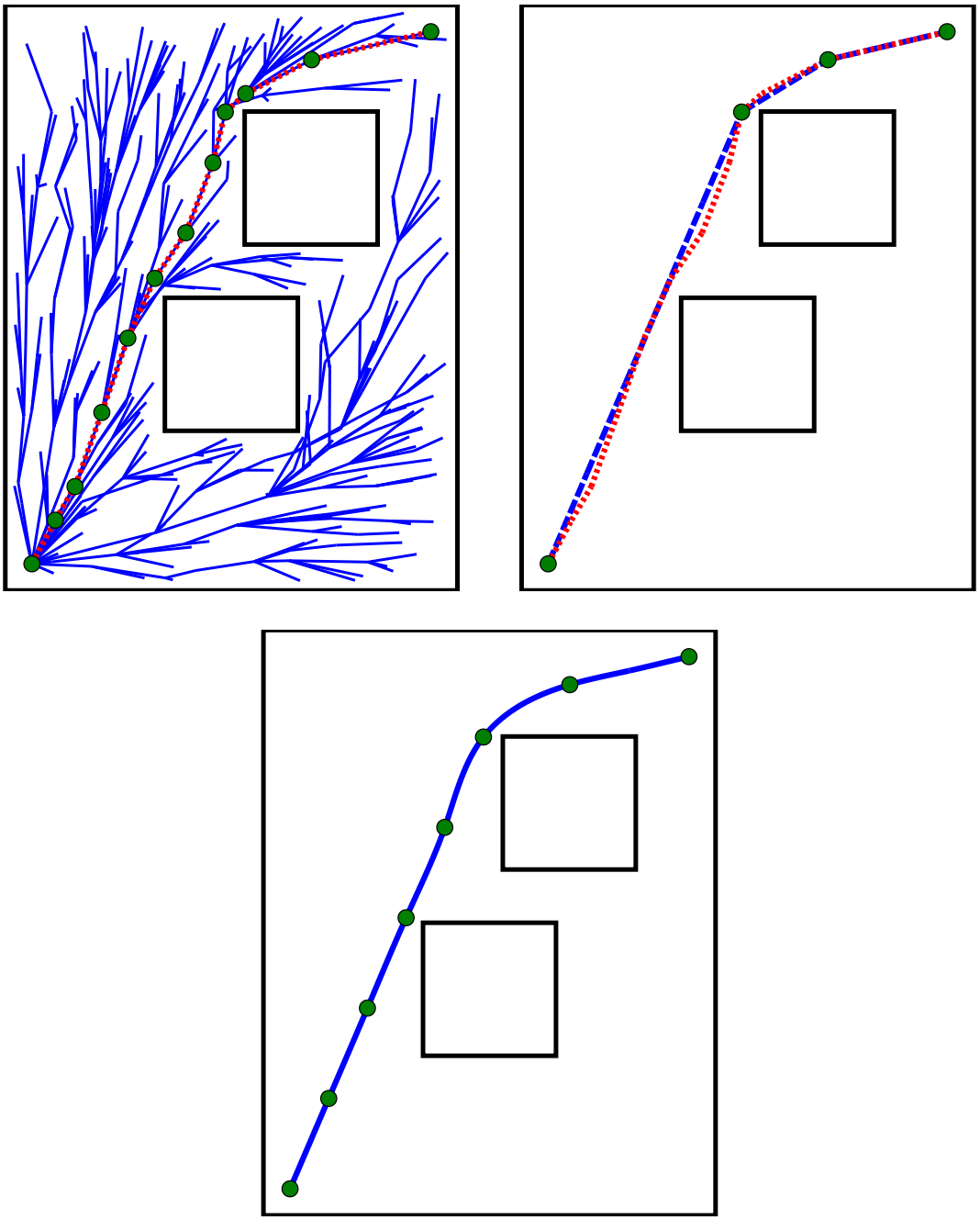


Figure 5.2: [Top-left] The tree constructed after a few iterations of I-RRT* (in blue), and the optimal route found so far (dotted red). [Top-Right] The filtered I-RRT* route (dashed blue). [Bottom] The uniformly distributed waypoints along the filtered route, and the resulting interpolated smooth clothoid spline.

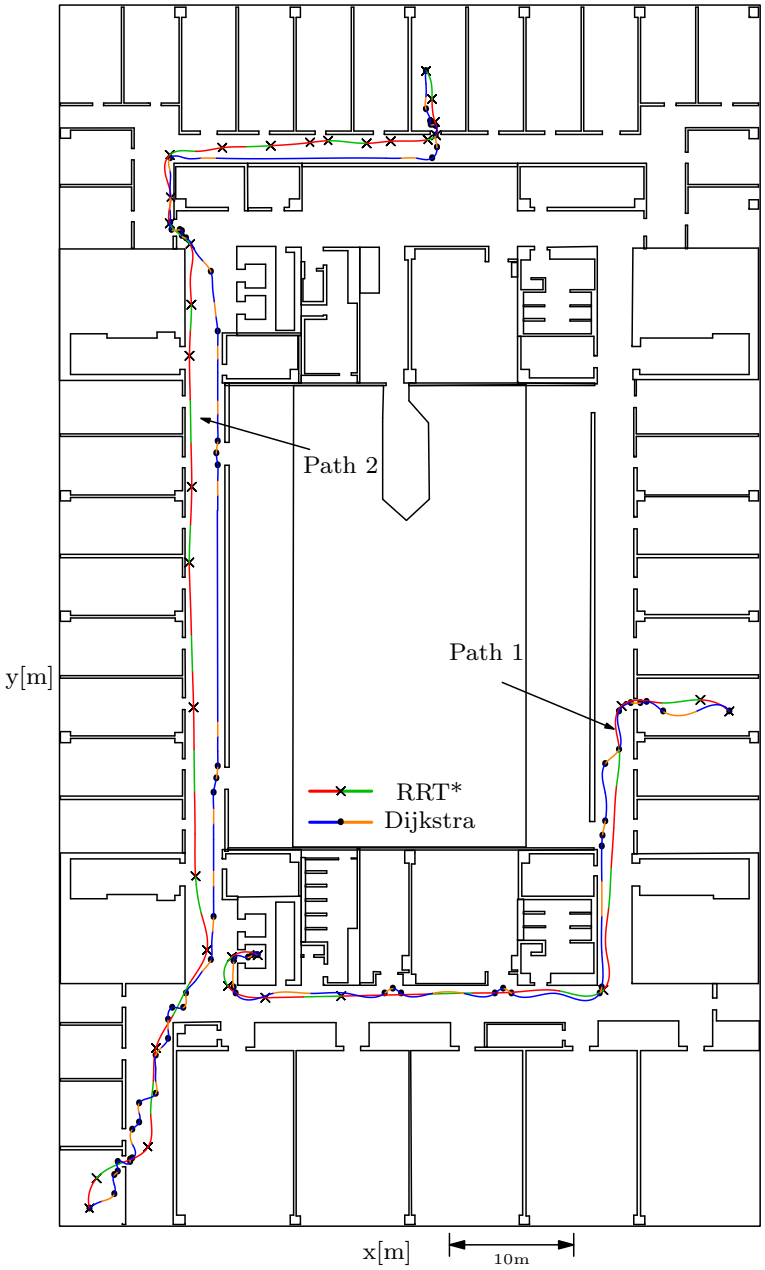


Figure 5.3: Comparison of two pairs of clothoid paths generated with I-RRT* and Dijkstra's algorithm on the map of the Department of Information Engineering and Computer Science of the University of Trento. [Published in [11]]

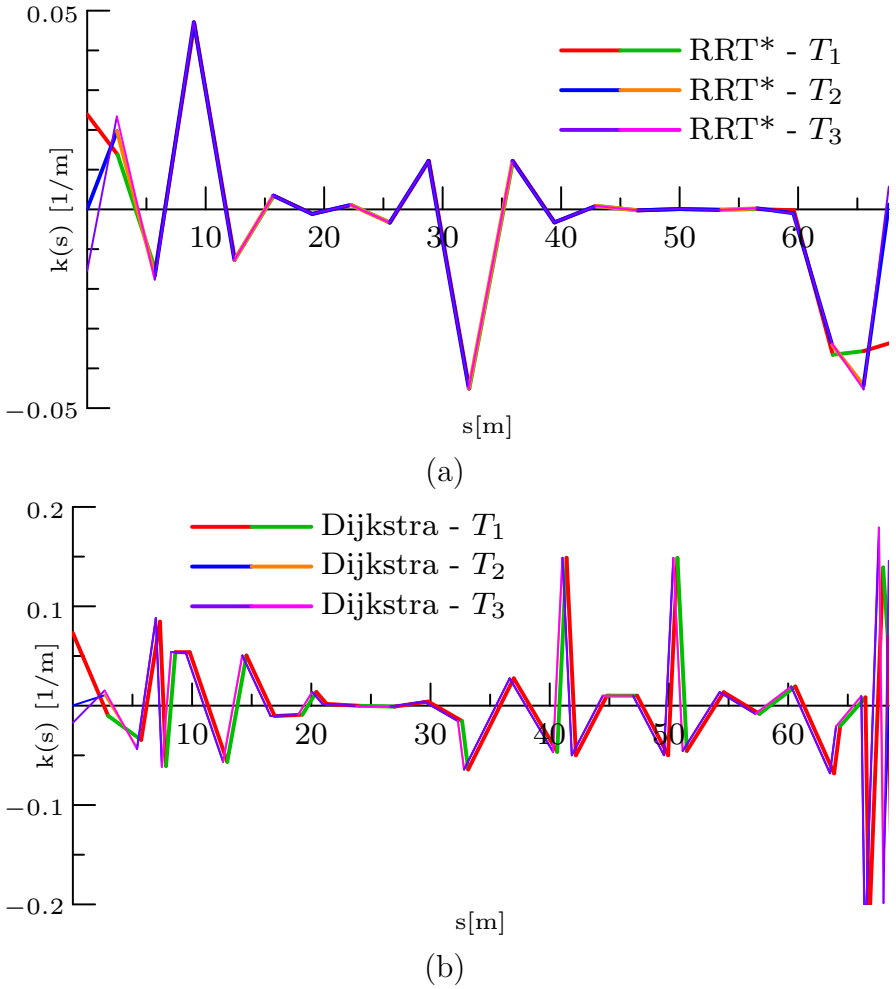
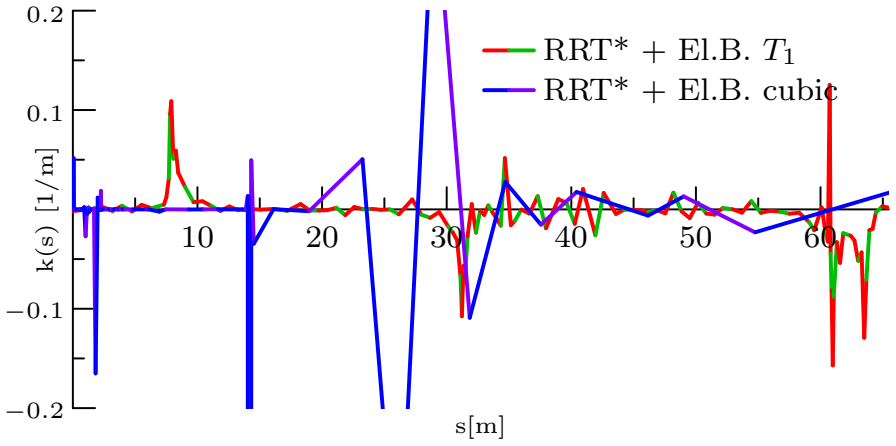
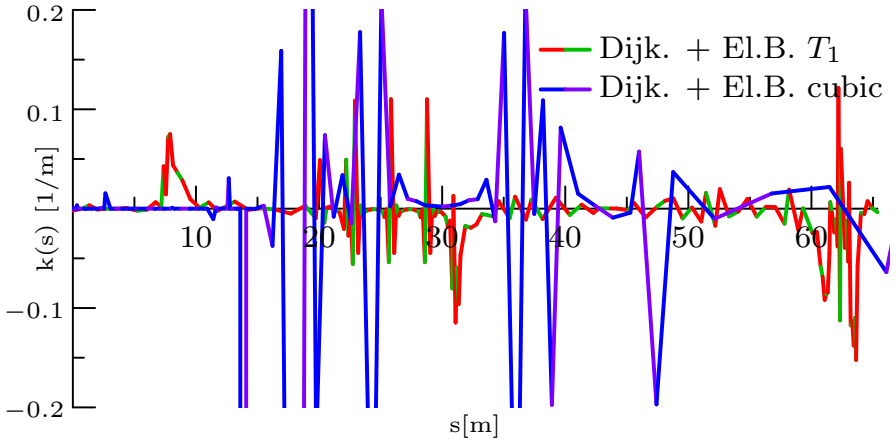


Figure 5.4: (a): Plot of the curvatures for Path 1 of Figure 5.3 generated with I-RRT* for the different discomfort indices. (b): Plot of the curvatures for Path 1 of Figure 5.3 generated with the Dijkstra's algorithm for the different discomfort indices. [Published in [11]]



(a)



(b)

Figure 5.5: (a): Path 1 with I-RRT* and elastic bands smoothing for clothoid trajectories (target T_1) or cubic splines. (b): Path 1 with the Dijkstra's algorithm and elastic bands smoothing for clothoid trajectories (target T_1) or cubic splines. [Published in [11]]

6
Reactive Planning

6.1 General Overview

In the previous chapters we have presented a motion planning algorithm specifically tailored to the generation of paths optimizing the comfort perceived by the user. This algorithm can be used to generate a complete path connecting a starting position on the map with a goal position, and avoiding collisions with all the obstacles, under the assumption that the layout of the map (walls, corridors, ...) is fully known. During the navigation of the global reference path, an unforeseen obstacle may be encountered, rendering the planned trajectory unfeasible. Indeed, additional geometric and dynamic constraints are introduced by the occasional presence of static or dynamic obstacles, such as other human beings standing by or walking in the neighbourhood of the robot. In such circumstances, a specific module, the reactive planner, intervenes to determine a local modification that can be applied to the reference trajectory to render it feasible again. The reactive planner should seek a solution minimising the deviation, and joining the global reference shortly after the obstacle. Moreover, the requirements of smoothness and comfort, of fundamental importance during the generation of the global plan, need to be accounted for also by the reactive planner.

A few important assumptions underlie the design of the reactive planner. First, we assume the prior knowledge of an optimal global plan that avoids static obstacles. Therefore, the reactive planner always has an available solution: stay on the reference path, slow down or stop when dynamic obstacles come across and wait until they pass by. If an obstruction that cannot be overcome by the

Figure 6.1: The *FriWalk* with the sensing system and embedded hardware. The vision system, consisting of a RGB-D sensor and a standard webcam, are visible within the orange circle.

reactive planner becomes permanent, the problem is escalated to the global motion planner, which modifies the reference path leveraging its long range information (e.g., deriving from fixed surveillance cameras), to generate a new, feasible route. Second, the vehicle is equipped with a sensing system able to reveal obstacles and anomalies in the surroundings (primarily, people walking in the proximity of the path, as in [90]). As an example, the *FriWalk* uses an an RGB-D camera [91] (see Figure 6.1).

There are several aspects that the developed reactive planner needs to consider. First of all, the plan has to be collision free to a reasonable extent (low speed collisions are not dangerous but are annoying and undermine the user's trust in the system). Secondly, it has to be socially acceptable and comfortable to follow. Moreover, the trajectory re-planning has to be computed in real-time and using the lean hardware available on a mobile robot.

6.2 Proposed Approach

To satisfy all the given requirements, we decided to base our reactive planner on the use of a human motion model, the Headed Social Force Model (HSFM), to

predict the behaviours of surrounding pedestrians. Since the possible consequences of accidents are not critical, we adopt a probabilistic formulation: the reactive planner seeks local modifications to the original plan that avoid all the obstacles with an high probability, while minimising the deviation from the reference path.

Contrary to previous works [41, 42], the proposed approach is not based on run-time simulations. Indeed, we propose a solution to analytically compute the probability of collisions, that drastically reduces the required amount of computation. The analytical model used by the proposed algorithm is derived from the HSFM. The HFSM tuning parameters are identified analysing a large amount of data, which are generated from simulations or observations on the field. The proposed analysis is based on the observation that each trajectory generated by the HFSM can be closely approximated using clothoid curves.

When the system detects a pedestrian, it generates a number of possible trajectories, each one associated with a possible destination and with a possible velocity profile. In addition, to each of these trajectories is associated a probability, that derives from the likelihood that the user will (approximately) follow that specific trajectory. The planning algorithm considers alternative paths for the vehicle, and for each computes the intersections with the possible curves taken by the pedestrian. By considering the probability associated with each of these curves, the planner computes the total probability of a collision; this information is used to identify a trajectory for the robot that has a small probability of collision (below a given threshold), that minimises the deviation from the main reference.

The use of the HSFM produces realistic predictions of the behaviour of the nearby pedestrians, enabling the robot to avoid them. The proposed analytic computation allows us to compute the solution efficiently and in real-time. In our experiments, we report a computation time for the planner in the order of a few milliseconds on a standard embedded PC.

6.3 Pedestrian Modelling

The HSFM [47] is a recently introduced model to describe the motion of pedestrians in a social environment. In this model, the i-th individual is represented by a particle with mass m_i, whose position expressed in the world reference frame $\langle W \rangle$ is denoted by $\mathbf{r}_i = [x_i,\ y_i]^\top$. In order to model the pedestrians' heading,

it is convenient to attach a body frame $\langle B \rangle$ to each individual, i.e. a reference frame centred at the pedestrian's position and whose x-axis is aligned with the pedestrian's forward direction of motion. Let $\mathbf{q}_i = [\theta_i,\ \omega_i]^\top$ be the vector containing the i-th pedestrian's heading and angular velocity are θ_i (angle between the x-axis of the body frame and that of the global reference frame) and $\omega_i = \dot{\theta}_i$, respectively. Denote by $\mathbf{v}_i^B = [v_i^f,\ v_i^o]^\top$ the velocity vector expressed in the body frame. The components v_i^f and v_i^o of vector $\mathbf{v}_i^B$ correspond to the projection of the velocity vector $\mathbf{v}_i$ along the forward direction and the orthogonal direction, respectively. Then, similarly to [92], the human locomotion model becomes

$$\dot{\mathbf{r}}_i = \mathbf{R}_o(\theta_i)\mathbf{v}_i^B,\ \ \dot{\mathbf{v}}_i^B = \frac{1}{m_i}\mathbf{u}_i^B,\ \text{ and }\ \dot{\mathbf{q}}_i = \mathbf{A}\mathbf{q}_i + \mathbf{b}_i u_i^\theta, \tag{6.1}$$

where $\mathbf{R}_o(\theta_i)$ is the 2D rotation matrix of angle θ_i,

$$\mathbf{A} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \quad \mathbf{b}_i = \begin{bmatrix} 0 \\ \frac{1}{I_i} \end{bmatrix}, \tag{6.2}$$

and I_i denotes the moment of inertia of pedestrian i. In the model (6.1), the inputs are $\mathbf{u}_i^B = [u_i^f,\ u_i^o]^\top$, whose entries are the forces acting along the forward direction and the orthogonal direction, respectively, as well as the torque u_i^θ about the vertical axis. In this model, if we set $v_i^o(0) = 0$ and $u_i^o(t) = 0$, for all t, the dynamic unicycle model is recovered, hence the model features a nonholonomic behaviour.

As in the SFM [43], the HSFM model inputs u_i^f, u_i^o and u_i^θ are computed on the basis of external forces (see Figure 6.2). Two terms are considered: the first is $\mathbf{f}_i^0$ and accounts for the pedestrian's desire to move with a given velocity vector $\boldsymbol{v}0i$, i.e.

$$\mathbf{f}_i^0 = m_i \frac{\boldsymbol{v}0i - \mathbf{v}_i}{\tau_i} \tag{6.3}$$

where $\tau_i > 0$ is a parameter determining the rate of change of the velocity vector. The second term $\mathbf{f}_i^e$ is the sum of the forces generated by the environment, e.g. fixed obstacles, walls, furnitures, etc., and other pedestrians in the environments. Then, u_i^f is given by the projection of $\mathbf{f}_i^0 + \mathbf{f}_i^e$ along the pedestrian forward direction, u_i^o has the damped dynamic $u_i^o = k^o(\mathbf{f}_i^e)^\top \mathbf{r}_i^o - k^d v_i^o$ depending on the external forces f_i^e projected onto the orthogonal of the pedestrian's direction $\mathbf{r}_i^o$ and scaled by a

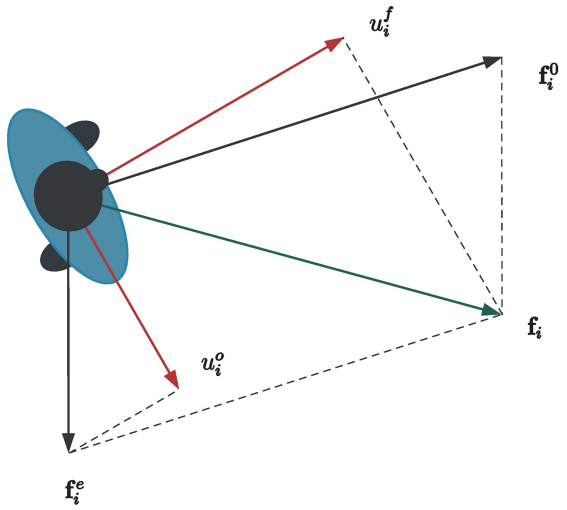


Figure 6.2: Force decomposition in the Headed Social Force Model. [47]

gain parameter $k^o > 0$. The term $-k^d v_i^o$, with $k^d > 0$, drives to zero the sideward velocity v_i^o when the sideward force is zero. Denoting with f_i^0 and θ_i^0 the magnitude and the phase in the global reference frame of $\mathbf{f}_i^0$, the input u_i^θ is computed as $u_i^\theta = -k^\theta(\theta_i - \theta_i^0) - k^\omega \omega_i$, where k^θ and k^ω are designed in order to achieve suitable dynamics of the heading. In [47], those tuning constants are set as $k^\theta = I_i k^\lambda f_i^0$ and $k^\omega = I_i(1+\alpha)\sqrt{\frac{k^\lambda f_i^0}{\alpha}}$, where $k^\lambda > 0$ is used to tune the rate of convergence of θ_i towards θ_i^0, whereas $\alpha > 1$ is the ratio of the desired two poles of that dynamic. The underlying idea is that the higher is the convergence rate, the faster the change in the pedestrian's heading.

6.3.1 Approximation of the HSFM trajectories with clothoids

The complexity of the HSFM does not allow us to directly use it in our reactive planner, due to the elevated computational cost required to solve a set of differential equations on–line. Moreover, the uncertainty about the model parameters of the actual person encountered, together with the fact that we can only detect a moving obstacle within a range of about 3 meters (due to limited sensing capabilities), justify a model simplification. In such a short space, we have observed that the trajectories produced by the HSFM can be well approximated by the concatenation of only two clothoid arcs. As noted above, this is in perfect accordance with the findings of Laumond et al. [45].

In Figure 6.3 are shown a set of HSFM trajectories starting from the origin, and reaching different goals at a distance of 3 meters (solid blue arrows), and the corresponding clothoid approximations (dotted red arrows). The original trajectories present an initial curved path, during which the pedestrian aligns his heading with the goal, and a successive straight motion to the goal. The curvature of an HSFM trajectory, therefore, presents an initial phase, during which the value monotonically decreases, and a second phase where the curvature is 0 (i.e. motion on a straight line). For this reason, a good approximation of each candidate HSFM trajectory in the considered scenario can be achieved by using just two clothoid segments: the first to model the turning part with monotonically decreasing curvature, and the second to model the straight motion (modelled by a degenerate clothoid where both κ and κ' are 0). In Figure 6.4 is shown a comparison of the curvature profiles for an HSFM trajectory (solid blue), and the corresponding approximation with two clothoid segments (dashed red). While it would be possible to use more than two clothoid segments to achieve a better fitting, the application of only two clothoid segments is enough to capture the shape and profile on the considered HSFM trajectories and readily obtain a good approximation. Moreover, the construction of the approximation with just two segments allows an extremely efficient computation, that is a fundamental requirement to satisfy the real-time constraints posed by the problem, given that, due to the uncertainty in the surrounding pedestrians behaviour, for each of them tens of alternative possible trajectories have to be generated, in order to avoid possible collisions.

Therefore, the construction of this approximated trajectory requires us to find the parameters of two clothoids, starting in P_0 and ending in P_2, that meet in the middle at point P_1 with curvature continuity, such that the second segment is a straight line (see Figure 6.5 for reference). Our method seeks the path that joins the two positions P_0 and P_2, i.e. the current position of the sensed pedestrian and the hypothesised final position. At P_0 we measure the xy-position, that is, $P_0 = (x_0, y_0)$, and an initial angle β_0. The second segment must be a straight line that meets the given final point $P_2 = (x_2, y_2)$, hence the curvatures κ' and κ must be zero. To model the different shapes of the resulting spline as in Figure 6.5, we use a tuning parameter $p \in (0, 1)$ and define the length L_2 of the straight line as a percentage of the Euclidean distance between P_0 and P_2. In other words, we

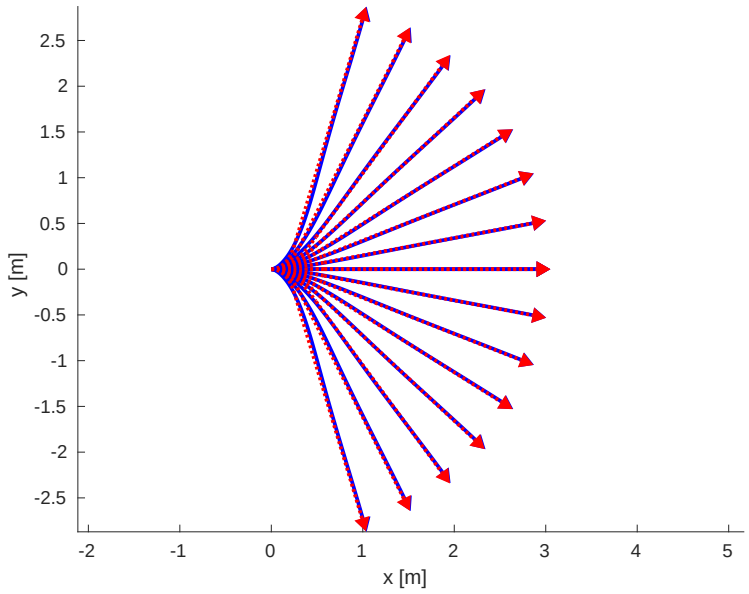


Figure 6.3

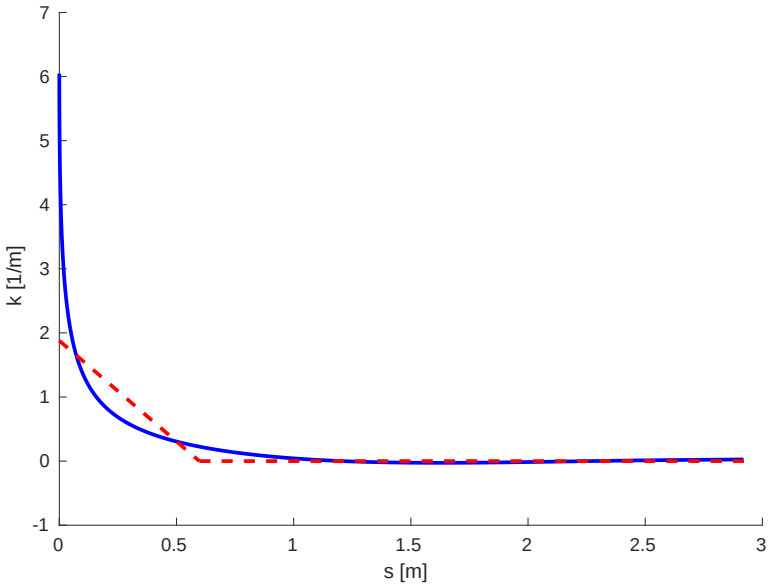


Figure 6.4

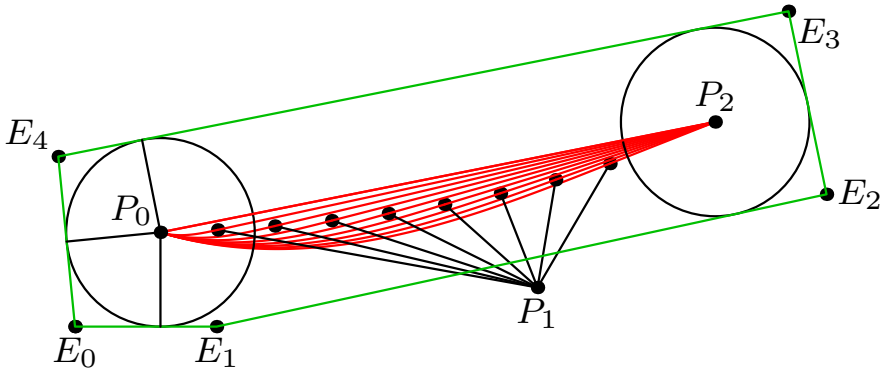


Figure 6.5: Ten different two-segments clothoid splines that join with curvature continuity a starting point P_0 and reach a final (target) point P_2. The points labelled with P_1 are the connection points between first and second segment, respectively proper clothoid and straight line. The splines are modelled with ten different values of the percentage p ranging from 0% to 100% with step 10%. In green it is shown a bounding polygon $E_0E_1 \dots E_4$ that contains all the possible splines with a constant offset. [Published in [12]]

set $L_2 = p \cdot \text{dist}(P_0, P_2)$. The tuning coefficient p is assigned on the basis of the physical characteristics of the modelled pedestrian (more on this in the rest of this section). At the joining point P_1, which is unknown, we require G^2 geometric continuity (up to the second derivative), which means that at the junction points the curvature is continuous. As a consequence, an additional constraint between the segments connecting P_i to P_{i+1} is defined and it requires that

$$H_{i,0} := x_i + L_i X_0\big(\kappa_i' L_i^2, \kappa_i L_i, \beta_i\big) - x_{i+1}, \tag{6.4}$$

$$H_{i,1} := y_i + L_i Y_0\big(\kappa_i' L_i^2, \kappa_i L_i, \beta_i\big) - y_{i+1}, \tag{6.5}$$

$$H_{i,2} := \frac{1}{2}\kappa_i' L_i^2 + \kappa_i L_i + \beta_i - \beta_{i+1}, \tag{6.6}$$

$$H_{i,3} := \kappa_i' L_i + \kappa_i - \kappa_{i+1}, \tag{6.7}$$

must be equal to zero. The subscript i for $i = 0, 1$ refers to a condition relative to the point P_i, while the lengths of the two arcs are modelled with L_1 and L_2 respectively. Equations (6.4) and (6.5) ensure point-wise continuity, whereas (6.6) and (6.7) stand for the angle and curvature, and X_0 and Y_0 are the Fresnel Generalized Integrals defined in Chapter 4. By imposing the angles and curvatures in P_1 and P_2 to be equal (since P_1P_2 corresponds to a line segment), we can simplify the original

system, yielding to the single non-linear equation $H_{1,3}$, and to the unknowns L_1, κ' and κ_0 as a function of β_1. This equation can be viewed as a function of the unknown parameter β_1 resulting from the G^1 solution: we can write it as

$$h(\beta_1) = \kappa'(\beta_1)L_1(\beta_1) + \kappa_0(\beta_1) = 0. \tag{6.8}$$

In other words, with the G^1 solution, we can simplify the nonlinear system to a function of one variable β_1, as in (6.8), and solve it using a few iterations of Newton method. In summary, the solution strategy to synthesise a path between P_0 and P_2 with two clothoids calls first the G^1 algorithm, then finds, via the Newton method, the value of β_1 that solves (6.8). Experimental evidence shows that the problem requires few iterations of the Newton method to converge in most of the cases.

6.3.2 Fitting the clothoid spline to the HSFM parameters

With the technique discussed above, we can approximate a short-horizon trajectory of the HSMF model with a simple two-segments spline of clothoids. The different shapes of this spline are created varying the percentage parameter p previously mentioned, which models the behaviour of the pedestrian. The choice of p depends on the parameters of the HSFM, which in turn model the pedestrian behaviour. In order to find this relation, we have simulated many trajectories between different pair of points P_0 and P_2 by changing all the HSFM parameters [47]. The result of this analysis is that the HSFM parameters that most affect the shape of the trajectory (and hence the choice of p) are: the reaction time of the pedestrian τ, which is slower for young people and higher for more aged people; the parameter k^o that models the orthogonal force, i.e. the step-aside trajectory; the values of the pedestrian heading dynamic behaviour k^λ and α, which determine if the curvature profile of the generated trajectory is loose or sharp. The metric adopted to compare the HSFM trajectory with the clothoid splines is the Root Mean Square Error (RMSE) based on the Euclidean distance. The adoption of this metric allowed us to construct the experimental map that associates to the N different vectors $(\tau, k^o, k^\lambda, \alpha, \beta_2)$ the optimal percentage p yielding the minimum RMSE. This process yields a look-up table that maps each configuration of the HSFM parameters to the optimal value of p. A possible way to synthesise this table is to perform a polynomial fitting in the sense of the least squares. The required function φ, is

a map $\varphi : \mathbb{R}^5 \mapsto [0, 1]$ such that $p = \varphi(\tau, k^o, k^\lambda, \alpha, \beta_2)$. It is worthwhile to note that β_2 is not a parameter of the HSFM but represents the final heading of the pedestrian in position P_2 and hence affects her/his trajectory.

Since the problem is not ill conditioned, we can use a low degree d for the polynomial. By choosing $d = 0$, we found for a large number of trajectories an approximate map φ equal to a constant $p_{\text{opt}} = 88\%$ within a tolerance of 0.5 m, which is a conservative assumption, considering the hindrance of a person and the presence of social constraints [93, 94]. In plain words, this means that we can approximate a large number of pedestrian behaviours by a sequence of two clothoids with a fixed parameter p. We obtained this result by executing random simulations for $N = 160\,000$ sets of parameters, and for 10 possible final destinations P_2 chosen at a distance of 3 metres from P_0 on an arc of circle for an angle between 0 and $\pi/2$, and extended by symmetry arguments on the whole range $(-\pi/2, \pi/2)$. In all these cases the deviation of the HSFM trajectory with exact parameters from the curves with constant p was below 0.5 [m] (using the RMSE metric), which is the approximate hindrance (the diameter) of the pedestrian. An advantage of using a constant p is that we do not need to identify the physical HSFM parameters of the obstacle and do not need to store a look up table or to evaluate the map ϕ at the parameters $\tau, k^o, k^\lambda, \alpha, \beta_2$. This value of p corresponds to a good representative of average human behaviours, and the possible deviations of actual human types can be accounted for by increasing the volume of the obstacle when detecting a collision. We deem this approximation acceptable in the face of the drastic simplification in the computation time.

6.4 Formalisation of the Re-Planning

The result of the previous section can be summarised in the following terms: the short term motion of a pedestrian can be represented by a sequence of two clothoids characterised by a parameter p. Assuming that the final destination of the pedestrian is known, it is possible to generate a specific curve along which she/he is likely to move in a near future. In the discussion below we will make this assumption, while at the end of the section we will discuss how it can be removed.

Even when the path is known, in order to plan a trajectory, we also need to know how the pedestrian will move along the curve over time. Our assumption is

that the pedestrian H moves with a constant velocity $v_{0\gamma}^{(h)}$ according to the linear ODE

$$\dot{h}(t) = v_{0\gamma}^{(h)}, \quad h(0) = h_0, \quad \Rightarrow \quad h(t) = h_0 + v_{0\gamma}^{(h)} t, \tag{6.9}$$

where $h(t)$ is the curvilinear abscissa of the obstacle over the generated clothoid path. The constant velocity $v_{0\gamma}^{(h)}$ is chosen randomly inside the interval $[v_{0\,\min}^{(h)},\ v_{0\,\max}^{(h)}]$ with known density $p_\gamma(v_{0\gamma}^{(h)})$ (which derives from consecutive pedestrian measurements performed by the on-board sensor). Notice that the constant velocity assumption, quite popular in the field [95, 39, 96], is effective, since the interaction between the vehicle and the pedestrian acts in a short amount of time. The constant velocity cumulative distribution function is $\mathbb{P}(v_{0\gamma}^{(h)} < v) = \int_{v_{0\,\min}^{(h)}}^{v} p_\gamma(v_{0\gamma}^{(h)}) \mathrm{d}v_{0\gamma}^{(h)}$. Because of the pedestrian hindrance, the space around its centre of mass is described with a conservative offset $\pm\delta_h$ around the abscissa $h(t)$, hence the obstacle occupies the interval $[h(t) - \delta_h, h(t) + \delta_h]$ at time t. We model the walker vehicle W in the same manner of the obstacle, with hindrance specified by δ_w (as observed above, a conservative estimate of the hindrance can include the possible uncertainty introduced by a constant choice of the parameter p). Its curvilinear abscissa is identified with the variable w (see Figure 6.6). Each of the two agents W and H, moves on a sequence of smoothly joined clothoids. The curve followed by the pedestrian is a spline made up of two segments (see Section 6.3), while for the vehicle it is the candidate path. To handle the two paths and speed-up the computation of the intersections, we decompose each of them into a sequence of triangles organised with the AABB tree structure discussed in Chapter 4. To model the physical hindrance given by δ_w (respectively δ_h for the pedestrian), the spline has two parallel curves at the left and at the right (see Figure 6.6). We call a *clothoid tunnel* the clothoid spline and its two offset curves.

As a final remark, we are assuming, without loss of generality, that the velocity of the vehicle W is constant along the planned path, and that it is a decision variable.

6.4.1 The velocity diagram

We now discuss a tool (the velocity diagram), which can be used to detect collisions and to select the optimal velocity of the vehicle W. In a standard intersection event, the routine to find the collision considering the encumbrance will return a sequence

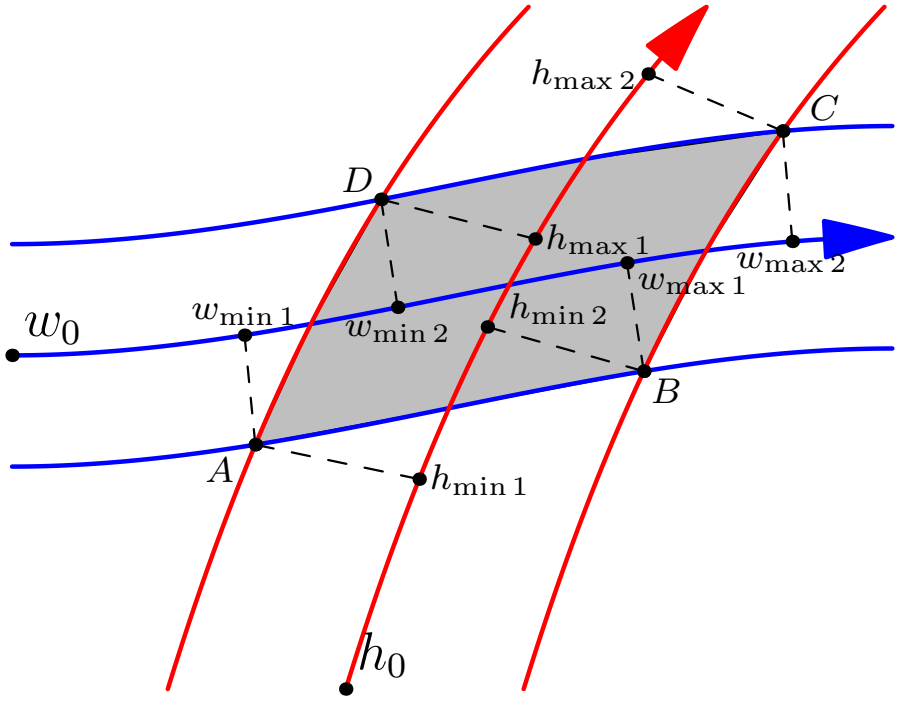


Figure 6.6: Projection scheme of the geometric intersection points into the curvilinear abscissas of the walker and of the pedestrian. Those abscissas are then mapped to the corresponding time instants. [Published in [12]]

of 4 points $\{A, B, C, D\}$ that are the vertexes of a generalised quadrilateral, as depicted in Figure 6.6. It is possible to find situation in which the collision is not described by 4 points. For instance, when trajectories are almost tangent, a smaller number of points points will be found. However, these are simply degenerate cases in which some of the points of the generalised quadrilateral coincide. For each of the four geometric intersection points, we are interested in the corresponding curvilinear abscissas, corresponding to the entry and exit points of the collision zone. They are called $w_{\min 1}$ and $w_{\max 2}$ for the walker and $h_{\min 1}$, $h_{\max 2}$ for the pedestrian (see Figure 6.6). Due to the agents hindrance, it is not enough to simply consider these coordinates. Therefore we define $w_{\min} = w_{\min 1} - \delta_w$, $w_{\max} = w_{\max 2} + \delta_w$ and $h_{\min} = h_{\min 1} - \delta_h$, $h_{\max} = h_{\max 2} + \delta_h$ for the walker and the pedestrian, respectively. Given the curvilinear abscissas, computing the travelling time is an easy step, since by hypothesis both velocities are constant. These time instants are called accordingly to the name of the abscissas, i.e. respectively $t_{w\min}$, $t_{w\max}$ for the walker, and $t_{h\min}$, $t_{h\max}$ for the pedestrian. It is convenient to name the time

intervals as $\Delta T_h := [t_{h\,\min}, t_{h\,\max}]$ and $\Delta T_w := [t_{w\,\min}, t_{w\,\max}]$. A collision happens if and only if the intersection $\Delta T_h \cap \Delta T_w \neq \emptyset$, i.e. the vehicle and the pedestrian are in the same region at the same time. Those quantities lead naturally to a space-time representation, dubbed velocity diagram, in which the horizontal axis represents the time and the vertical axis the curvilinear abscissa $w(t)$ of the walker for a specified path. The velocities, being constant, are thus straight lines from the origin. More interestingly, the collision zone in space and time can be approximated with an octagon, as in Figure 6.7. The function $t_s = \Xi(v_{0\gamma}^{(h)}, v_0^{(w)})$ returns the minimum

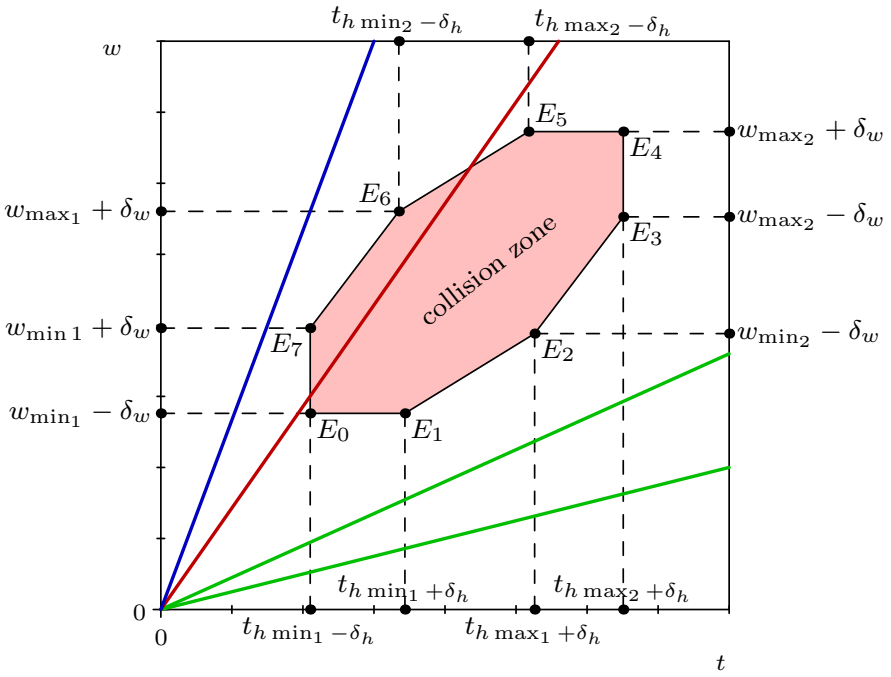


Figure 6.7: The velocity diagram for a single intersection of trajectories: the red area represents the collision zone in terms of space and time. Green lines are walker's velocities that follow the pedestrian, red lines are walker's velocities that will cause a collision, and blue lines are velocity that allow the walker to overtake the obstacle. [Published in [12]]

time required to wait to avoid the collision with fixed $v_0^{(w)}$. This is clearly zero if $\Delta T_h \cap \Delta T_w = \emptyset$. After the amount of time t_s, the vehicle can move with velocity $v_0^{(w)}$ and be sure to avoid the pedestrian. It has to be noted that the function $\Xi(\cdot)$ is solved by computing intersections between segments on the velocity diagram, hence it is computationally extremely light. Moreover, the constant velocity assumption

on $v_0^{(w)}$ has been selected for simplicity and for the particular application at hand, since the user cannot be accelerated. Nevertheless, in a more general framework, time varying velocity profiles satisfying velocity constraints could be selected.

The previous graphical solution computes the stop time t_s for a given $v_0^{(w)}$ and $v_{0\gamma}^{(h)}$, where the latter is only known statistically. Therefore, we define the following cost index

$$J_t = \int_{v_{0\,\min}^{(h)}}^{v_{0\,\max}^{(h)}} \Xi(v_{0\gamma}^{(h)}, v_0^{(w)})\, p_\gamma(v_{0\gamma}^{(h)}) \mathrm{d}v_{0\gamma}^{(h)}, \tag{6.10}$$

which represents the average minimum time the vehicle has to wait to avoid the collision and it is a function of the chosen velocity $v_0^{(w)}$. Notice that a change in $v_{0\gamma}^{(h)}$, corresponds to a translation and a scale of the polygon in Figure 6.7. The velocity $v_0^{(w)}$ of the vehicle is then selected from a discrete set of values contained in $[{v_0^{(w)}}_{\min}, {v_0^{(w)}}_{\max}]$ such that (6.10) is minimised. Since in most of the cases the value of $\Xi(\cdot)$ will be zero, there will be multiple minima, say V_w. Hence, the selected $v_0^{(w)}$ will be the closest to a desired comfortable velocity $v_d^{(w)}$, i.e.

$$J_v = \arg\min_{v \in V_w} \left| v - v_d^{(w)} \right|. \tag{6.11}$$

The depicted algorithm determines the optimal $v_0^{(w)}$ in the sense of (6.10) and (6.11) along the selected path and assuming the pedestrian moves from its actual position P_0 to a well defined desired position P_2. If it is possible to modulate the velocity, we can accelerate the vehicle in order to overtake the obstacle (blue line in Figure 6.7). Otherwise, if we do not have enough escaping velocity, it is possible to slow down or stop and let the obstacle pass (green lines in Figure 6.7). All the described computations are performed graphically on the velocity diagram, and a solution always exists: in the worst case, the vehicle stops. However, if for a specific problem the stop time t_s is too high, the probability of having a collision (related to J_t) is too high or if the performance are too poor (a too high cost for J_v), a local re-planning is needed, which is the purpose of the next section.

6.4.2 Relaxing the assumptions

In the previous section, we made the important assumption that the final point P_2 of the pedestrian is known. We can relax this assumption introducing an additional

random variable for P_2. Different papers in the literature give suggestions on the possible probability distributions for P_2 accounting for the social conventions [95, 97]. The analytical computation based on the velocity diagram discussed above can be repeated as a function of P_2, producing a re-formulation of the performance index in (6.10) in which the position of P_2 appears as an additional random variable p_2 to integrate on, and the distribution $p_\gamma(v_{0\gamma}^{(h)})$ is replaced by the joint distribution of p_2 and $v_{0\gamma}^{(h)}$. As far as p_2 is assumed independent from $v_{0\gamma}^{(h)}$, the computation of the integral is straightforward. Notice that the presence of multiple probabilistic destinations similarly affects any other possible simulation-based planner. In the same way, we can deal with a possible variability of the velocity of W. Indeed, while we can "suggest" a possible velocity to the user in different ways (e.g., through haptic devices or a GUI), it is not guaranteed that s/he will closely follow the suggestion, hence random variations around our selected velocity are a possibility.

6.4.3 Local Re-planning

The strategy adopted for the trajectory re-planning is based on a decomposition of the problem into dynamic planning, where the velocity profiles on the path are computed as explained in the previous section, and geometric planning, that will be presented in this section. When a re-planning is requested, the algorithm selects an initial point Q_0 with position (x_0, y_0), angle β_0 and curvature κ_0, and a final point Q_2 with position (x_2, y_2), angle β_2 and curvature κ_2 along the reference path; the re-planned trajectory will depart from Q_0 and will rejoin it at Q_2. The algorithm seeks a new point Q_1 in the proximity of the obstacle to pivot on, in order to find the best trajectory (see Figure 6.8). The connecting curve is a spline of clothoid curves that exhibits G^2 continuity. The different choices of point Q_1 can be explored via a deterministic search (as herein done) or by using stochastic methods. For the vast majority of reasonable application scenarios, the method reliably produces a solution. In the extreme cases where it may fail, its efficiency leaves time to slow down the vehicle and back off to different emergency solutions, e.g. stop on the spot. In principle, the algorithm presented below operates with any pair of entry and exit points Q_0, Q_2 on the reference path. An obvious requirement is that Q_0 and Q_2 be located before and after the obstacle. The low computational cost of the algorithm allows us to test different possible choices or, again, back off

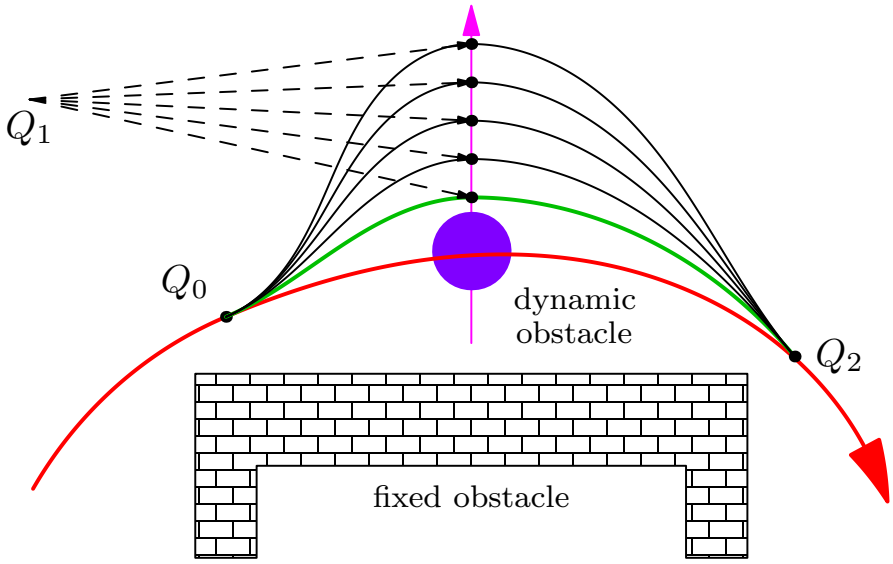


Figure 6.8: A sketch of the local re-planning method, in red the global trajectory that is no longer feasible because of the obstacle (purple circle). In green the optimal escaping manoeuvre, in black feasible candidates for different choices of Q_1 located deterministically aside from the obstacle trajectory. [Published in [12]]

to an emergency strategy if the spline identified by the algorithm fails to satisfy the geometric or the dynamic constraints. However, the application of reasonable heuristics on the selection of Q_0 and Q_2 limits the occurrence of this anomaly. It is useful to observe that if the obstacle is very close to the vehicle, the back off solutions are likely to be adopted. An intuitive and straightforward heuristic choice is to select Q_0 as the current robot position, and Q_2 as a configuration that is located at a reasonably far distance from the obstacle. Once a path is found, the optimal vehicle velocity $v_0^{(w)}$ minimising J_t and J_v in (6.10) and (6.11), respectively, is computed.

6.5 Experimental Validation

Our reactive-planning has been validated in two ways. Firstly, through a set of software simulations, where the trajectories of the dynamic obstacles (i.e. pedestrians) are captured from a real scenario recorded at ETH [98], or computed dynamically according to the Headed Social Force Model (the initial and final pose of each dynamic obstacle can be generated randomly), while the emulated sensing system

allows the simulated walker to generate candidate obstacle trajectories according to the clothoid models discussed in Section 6.3. Secondly, with a direct implementation on our robotic walker *FriWalk*. We have conducted many experiments, both in simulation with real and synthetic data, and also on the field. The next section illustrates some of the outcomes of these experiments.

6.5.1 Software simulations

In the first part of this section, we present two situations where a pair of pedestrians cross the walker's trajectory, in one case they walk both in the same direction, in the other they come from opposite sides, see Figure 6.9 (left). In green is depicted

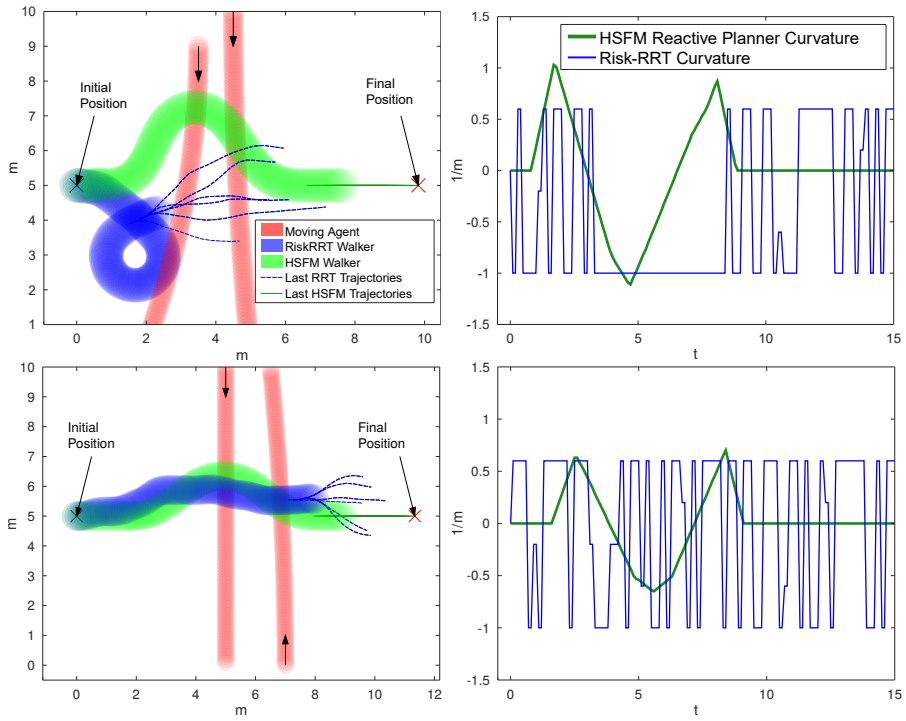


Figure 6.9: A comparison between the proposed approach and the application of Risk-RRT for the dynamic re-planning. [Published in [12]]

the trajectory generated by our reactive planner with the method herein discussed, while in blue the solution provided by the Risk-RRT algorithm [40], which is a state-of-the-art approach for this kind of problem. In the first test case (Figure 6.9, top left), our method produces a smooth deviation from the reference and reconnects to it passing behind the pedestrians. The approach based on Risk-RRT produces an

unnatural loop and then reconnects to the original path. In the second case, both methods give similar natural solution (Figure 6.9, bottom left). However, when the curvature profile of the trajectories is considered, the one obtained by Risk-RRT (blue line in Figure 6.9, right) exhibits a piecewise constant behaviour with frequent jumps to three possible values: this is an undesirable property, causing problems to both the control and the tracking modules deployed on the robot, and reducing the comfort perceived by the user of the navigation assistive robot. In fact, curvature discontinuities and sharp variations induce high spikes in the jerk profile. On the other hand, our method exploits the property that clothoids have linear curvature w.r.t. the arc length, resulting thus in a regular piecewise linear curve (green line in Figure 6.9, right). These comments apply to most of the 100 tests simulated. For a quantitative comparison, on paths of about 15 [m] length, the average value of the squared deviation integral from the reference path was of 0.43 for HSFM and 10.56 for Risk-RRT, while the average of the curvature squared integral was 0.46 and 10.67 respectively. The length of the Risk-RRT path was generally 15% longer than the HSFM path. Moreover, the proposed algorithm has been thoroughly tested on a large number of different simulated scenarios, where the trajectories followed by each of the pedestrians were either real human trajectories from the ETH dataset [98], or computed during the simulation according to the HSFM model. In Figure 6.10 are reported six different scenarios, and the trajectories followed by the robot (in green) and by the dynamic obstacles (in red). It is worth to notice that, during the simulations, the planner takes into account only a limited number of pedestrians moving nearby, in front of the walker. This behaviour is mimicking the actual human behaviour in crowded shared space: indeed, a number of studies have shown how pedestrians take their motion decisions based on the number of walking people in the surroundings [93, 99, 100, 101, 94], which has been also modelled in the literature as the Information Process Space [102, 103], that may be considered as their "field of attention". Moreover, due to the physical limitations of the sensing system available on the robot, consisting in an RGB-D camera with a limited field of view, no more than three or four pedestrians can be detected simultaneously. In addition, groups of pedestrians walking together can be treated as a single entity, i.e. a social group that cannot be split by the robot, by modelling them as a single dynamic obstacle to be avoided.

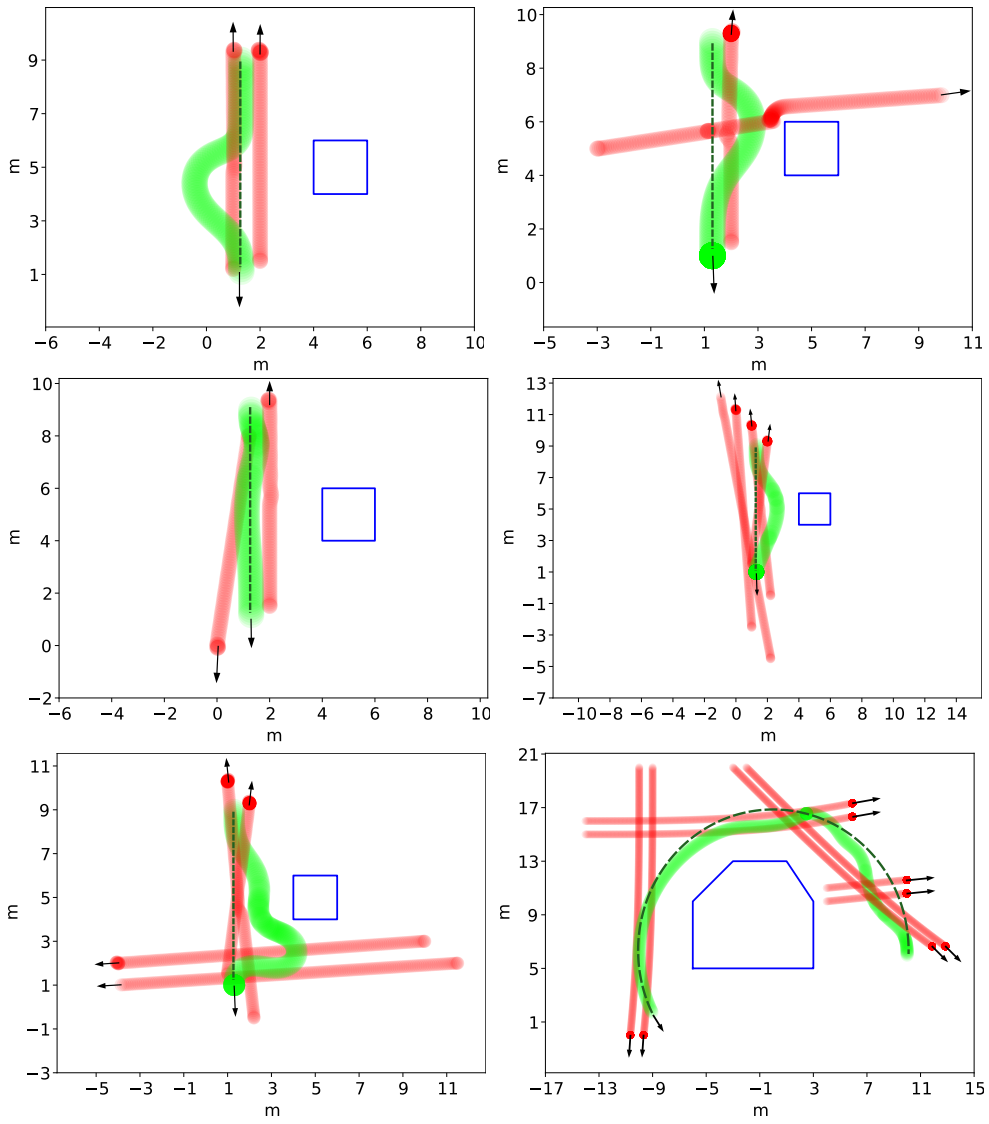


Figure 6.10: A comparison between the proposed approach and the application of Risk-RRT for the dynamic re-planning.

6.5.2 Walker implementation

We have conducted various experiments at the University of Trento, by implementing on the *FriWalk* the presented reactive planner. The results are in accordance with the computer simulations described so far. We worked first in a structured scenario, equipped with the OptiTrack positioning system that tracks precisely the actual movements of the walker and of the pedestrians. This allowed us to record the

experimental data with high accuracy, and to fully test the re-planning algorithm assuming an accurate sensing system. Then, we tested the algorithm in a second, unstructured, scenario, using only the walker's RGB-D camera, without any external service. The computational times on the on–board hardware (based on an Intel I7 5557 Nuc with 8GB of RAM) show that it is possible to employ the proposed reactive planner for real time applications. In fact, we registered a mean execution time of 3 ms for a standard escaping manoeuvre with 5 candidates Q_1 points (as depicted in Figure 6.8), the minimum time was about 1 ms and the maximum 11 ms. The acquisition of the moving obstacles was performed using an Asus Xtion Pro RGB-D sensor, with sampling time of 100 ms, while the re-planning module was executed with a period of 300 ms. Figures 6.11, 6.12, and 6.13 depict the results of some of the experiments: whenever an human enters the field of view of the *FriWalk*, and it determines the unfeasibility of the original path, a new, safe trajectory is generated by the reactive re-planner.

The experimental validation, conducted both in simulation and on the real robot, shows the validity of the proposed approach, that is capable of avoiding collisions with moving pedestrians in a shared space, intervening both on the current maximum speed and, locally, on the geometry of the reference path. The safety of the system is guaranteed by the ongoing availability of a safe, emergency manoeuvre: due to the low velocity of the robot, it is always possible to halt the motion before a collision, and let the other pedestrian pass by, in the extreme circumstances that may prevent the algorithm to find a feasible avoidance manoeuvre. In practice, this scenario occurred extremely rarely during the experiments, only whenever the environment was densely crowded, or due to the sudden and close range appearance of a pedestrian within the field of view of the robot.

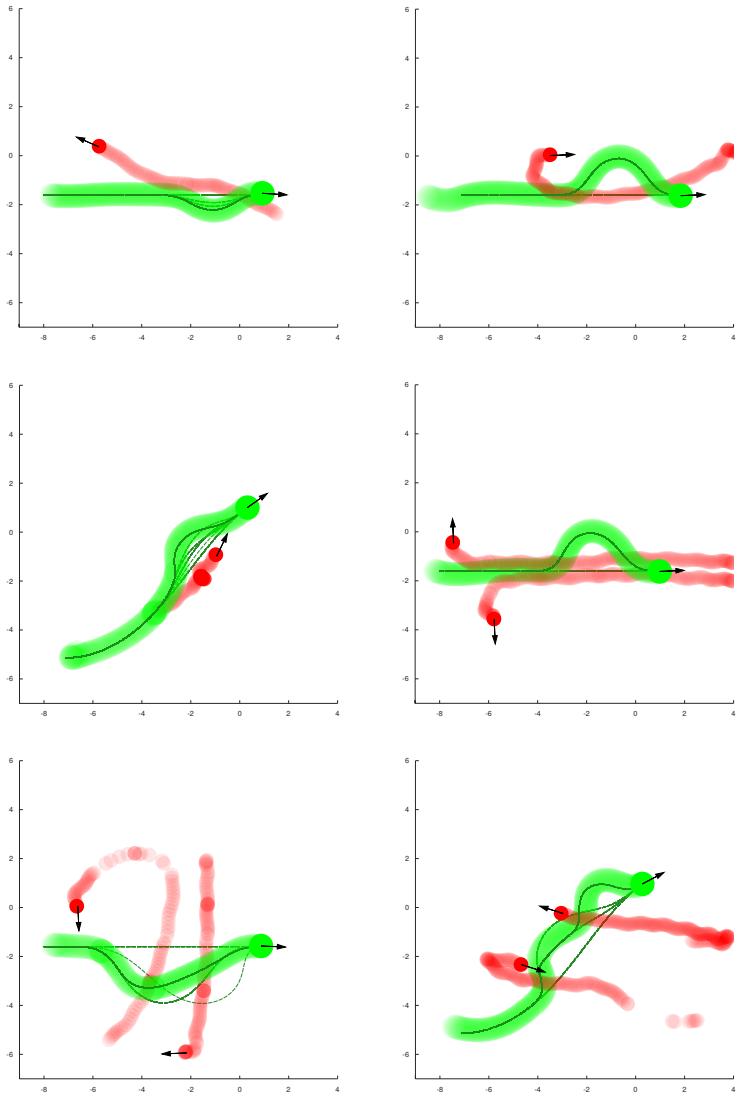


Figure 6.11: Experimental validation of the reactive planner on the *FriWalk*. [Part 1]

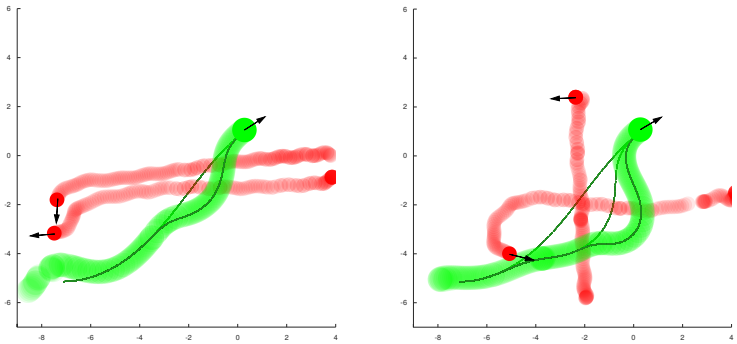


Figure 6.12: Experimental re-planning example with the *FriWalk*. [Part 2]

Figure 6.13: Photos of the reactive planner running on the *FriWalk*.

7 Activity Planning

7.1 General Overview

In the previous chapters, we have illustrated efficient solutions for the motion planning and the reactive replanning specifically focused on assistive robots. These can be considered as the low-level planning components, responsible for the assistance of the user during the navigation between two given locations. However, to develop a fully functional planning system for a robotic assistive platform, also an "high-level" planning component is necessary. The aim of this high-level planner is to provide the users with a set of choices regarding possible activities and tasks to perform, based on their specific interests and profiles, and to automatically select a sequence of locations to visit in order to fulfil an activity.

Therefore, in this chapter we propose a comprehensive planning solution to generate a sequence of activities to be executed in a public space by an older adult with the direct assistance of the robotic walker. The proposed approach is based on a close interaction between the high-level component (activity planning) and the low-level components (motion and reactive planning), with the objective of selecting a subset of intermediate goals (modelled as points of interest, POIs), and of generating a feasible trajectory visiting all of them.

Indeed, it is our belief that the problem of activity planning has a number of unique characteristics, which make it difficult to solve using either high-level or low-level methods alone. High-level methods allow us to model "complex" activities capturing points of interest, user preferences and constraints. In spite of that, the operation in an "open" human and unstructured environment makes the use

of simple abstract models such as state machines or transition systems fairly inadequate. On the other hand, motion planners of different kind are not sufficient to model and capture the complexity of human activities, required for the synthesis of the optimal sequence of high-level tasks (e.g. the visit to some points of interest). An effective solution is therefore necessarily based on a proper convergence between high-level and low-level planning.

We consider scenarios such as shopping malls, museums, etc., where the environment comprises a set of POIs that can be visited by the user gaining some reward. Users have individual preferences and constraints, which may have a different level of criticality (modelled by associating an acceptable probability of violation for each of the constraints). The planner decides a sequence of actions, each one corresponding to a motion from a start to an end point. An action is probabilistically characterised in terms of execution time, travelled distance and other parameters. This is done offline by executing a path planner in a large number of simulated conditions. The probabilistic description of each action recasts the activity planning problem as a Chance Constrained Stochastic Programming (CCSP) with integer decision variables. Our approach is therefore based on the explicit representation of the problem as a CCSP optimization, that allows us to consider multiple probabilistic constraints. When considering problems with a limited size, exact algorithms give the optimal solution within an acceptable time. In general, the exact solution of such a problem with a large number of nodes is intractable. For this reason, we discuss also a hierarchical decomposition technique that allows us to find good suboptimal solutions in a short time for moderately large problems.

7.2 The Activity Planning Problem

The objective of the activity planner is the synthesis of a plan for an activity that will be executed by an older adult with the support of a robotic assistant. The activity takes place in a large public space such as a museum or a shopping mall which contains a collection of points of interest (POIs). We assume that the map and the position of the POIs are known in advance, for instance, a museum can provide a map of the exposition rooms labelled with the list of the items of the collection. We assume reasonable, and also feasible from a computational point of

view, to consider up to around one hundred of POIs. As an example, consider in Figure 7.2 the POIs given by the green dots and their connections (paths). Aim of the activity planner is to select a subset of these POIs and produce a plan, for instance the optimal sequence may be $P1$, $P4$, $P2$ and $P3$. The user profile defines preferences and constraints specific to each user. In particular, the visit of a POI yields an utility to the users that depends on their interests and preferences. A possible way to implement this is by associating a POI with a number of tags, and defining an affinity value with each tag in the user profile. The "score" associated with visiting a POI could be defined as the sum of the affinity values for each tag associated with the POI. In this setting, a possible objective of the planner is to maximise the total score achieved during the activity (i.e., the sum of the scores of all the visited POIs). Additional goals could be related to the user physical condition; for instance, we could have a performance metric related to the distance walked or to the calories burnt. The activity plan has to consider a number of different constraints, which can be roughly classified as follows:

1. **Environment constraints**: due to the geometric configuration of the environment, the robot assistant can move only in certain areas (passageways), while the presence of fixed obstacles can impose constraints on the trajectory that can be followed. Some of the geometric constraints posed by the environment can change over time. For instance, at some time a passageway can be obstructed by a crowd or be temporarily unavailable due to cleaning operations. Thus, given a pair of POIs, they can be directly connected or not, depending on their position.

2. **Physical constraints**: in a configuration in which the user pushes a robotic walker with actuation on the front wheels, the user–walker ensemble can be modelled as a car-like vehicle with rear wheel drive. Instead, if the front wheels are free and the actuation is on the rear, then the ensemble can be modelled as a unicycle-like vehicle [45]. In any case, the motion is subject to nonholonomic constraints, which need to be accounted for when making planning decisions.

3. **User constraints**: the user profile encodes constraints regarding the user's physical or psychological conditions. As an example, the user could have

limitations on the maximum time he/she is allowed to walk before taking a break or, on the contrary, be required to walk for at least a given distance every day. Other constraints can be on the presence of a toilette or of a seat within easy reach. The user constraints do not necessarily have the same level of criticality. For instance, for a user it could be mandatory to avoid crowded areas, while it could be only desirable to take frequent stops during the walk. The way we model criticality is by allowing some of the constraints to be missed with a given probability, whereas mandatory constraints have to be met with probability 1. As also shown in Figure 7.1, the user constraints are in this phase provided by a doctor, formal or informal caregiver, a relative or by the user directly and properly inserted into the user profile.

7.3 Proposed Approach

Our approach is based on a clear separation of concerns between motion planning (i.e., how to physically move the robot between two configurations respecting physical and kinematic constraints) and activity planning (i.e., decide the optimal sequence of high-level actions composing an activity). The activity (e.g., go shopping or visit a museum) can be suggested to the user by an external component (an *Activity Recommendation System*) accounting for her/his preferences, or simply by a doctor or a friend. We split the solution in two phases: in the first, we perform an offline preprocessing of the known environment (map and POIs), to determine the stochastic physical parameters associated with each motion between pairs of connected POIs (e.g. length, travel time). In the second, performed online, we build the (sub-) optimal user-specific plan that satisfies all the constraints, and execute it with the support of the motion planner. The way the activity planner and the motion planner interact is illustrated in Figure 7.1. The first step is to generate and characterise an alphabet of actions. An action $\mathcal{Y}_{i,j}$ simply corresponds to moving between POI number i (denoted as POI_i) and POI number j (POI_j). An action has a duration $\mathcal{T}(\mathcal{Y}_{i,j})$, which depends on the velocity of the user (who ultimately propels the robot) and on the path followed. On its turn, the path depends on the condition of the environment. For these reasons, $\mathcal{T}(\mathcal{Y}_{i,j})$ is modelled as a random variable and its probability distributions are found through simulations. A geometric representation of the environment defining the locations of the POIs and

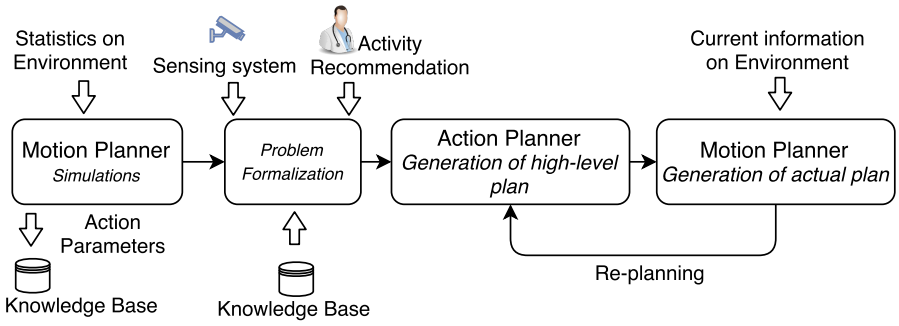


Figure 7.1: The structure of the framework for the activity planner

including information about the (known) layout of the area (a group of buildings, a park, etc.) and of the static obstacles (such as fences and walls) is used as a base model for all the simulations. This static map is randomly populated with dynamic elements, like crowds or temporarily inaccessible areas. An example of these simulations is shown in Figure 7.2, where the blue star represents a particular path obtained in presence of an obstacle along the natural pathway. The dynamic elements, different for each simulation, are generated according to an estimated probability distribution based on statistical observations on the real environment. The simulations can be classified according to different operating scenarios, where a scenario is associated with an average density of dynamic entities (e.g., number of people, number and extension of inaccessible areas, etc.). Another classification is made considering the type of user, ranging from "healthy" users who can exceed an average speed of 1m/s, to heavily impaired users who move at 0.5m/s or less. During each simulation, the motion planner identifies the optimal path joining POI_i and POI_j, avoiding collisions with all the obstacles (both static and dynamic), and optimising the user comfort, as discussed in Chapter 5.

By analysing the data extracted from the simulations, it is possible to find the probability distribution of $\mathcal{T}(\mathcal{Y}_{i,j})$ for each type of users and for each operating scenario. In the same way, it is possible to find the probability distribution of the distance travelled $\mathcal{D}(\mathcal{Y}_{i,j})$ and of the calories spent $\mathcal{C}(\mathcal{Y}_{i,j})$. All the information related to the action $\mathcal{Y}_{i,j}$ is stored in the knowledge base.

When the activity planner is required to produce a plan at run-time, the system retrieves all the information related to the planning domain and merges it with the user profile and choices. The sensors deployed in the environment are queried

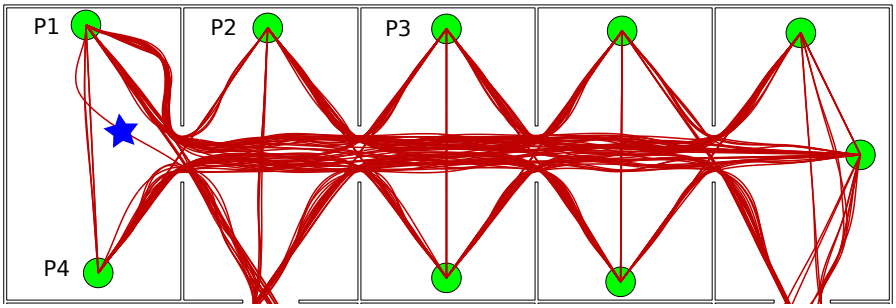


Figure 7.2: An example of the simulated trajectories among pairs of POIs. The blue star indicates a particular path that avoids a (temporarily) forbidden area.

to estimate the current operating scenario for each of the possible actions. The probability distributions are then retrieved and an instance of the planning problem can be formalised, as discussed in the next section, and transformed into a standard format, that can be treated by off-the-shelf optimisation tools. The use of high-level formalisms allows us to avoid strong commitments to a specific solver. By using the suboptimal techniques described below, it is possible to produce good solutions in a matter of a few seconds. The produced plan consists of a sequence of actions that can be handed over to the motion planner in order to be executed.

Except for a number of cases associated with a small probability, the motion planner will then be able to produce a path meeting the user's preferences and constraints. In the unlikely situation in which no acceptable plan can be produced, it is possible to raise an exception and produce a new activity plan that accounts for the new, contingent conditions encountered on the field. In this case, a new activity plan must be produced, which is a modification of the original plan. We must update the current status, by removing the infeasible edges from the list of possible actions, by ignoring the already visited POIs and by updating the remaining resources like the total time or distance travelled, in order to satisfy the constraints. This new problem has the same structure of the original one, but a smaller size.

7.4 Problem Formalisation

Using the approach described in Section 7.3, we abstract the map as a graph where the nodes are the POIs and the edges are the actions $\mathcal{Y}_{i,j}$ connecting POI_i and

POI_j. As discussed above, the current scenario is identified using the remote sensors readings. The Boolean variable vector $\boldsymbol{p}$ represents the POIs, where the element p_i is equal to 1 if the point POI_i is visited, and 0 otherwise. In addition, we introduce the binary variables $Y_{i,j}$ taking 1 if and only if the action $\mathcal{Y}_{i,j}$ is selected in the plan. The visit of a POI or the traversal of an edge can be modelled naturally using Boolean variables, where the variables take the value true or false depending on whether a node is visited or not, or an edge is traversed or not. The presence of integer decision variables leads naturally to an Integer Programming problem.

The objective function of the problem is normally related to the score accumulated with the visit of each point of interest. Suppose that the reward associated with POI_i is for the considered user c_i, then the objective function will contain the term $\sum_i c_i p_i = \boldsymbol{c}^T \boldsymbol{p}$. In addition, if the activity is intended to improve the physical condition of the user, the objective function could contain terms related to the distance walked, i.e. $\sum_{i,j} Y_{i,j} \mathbb{E}\left(\mathcal{D}\left(\mathcal{Y}_{i,j}\right)\right)$, or the calories burnt, i.e. $\sum_{i,j} Y_{i,j} \mathbb{E}\left(\mathcal{C}\left(\mathcal{Y}_{i,j}\right)\right)$, where $\mathbb{E}\left(\cdot\right)$ denotes the expected value.

The next step is to introduce the user profile, which will produce additional hard and soft constraints. The hard constraints are modelled as linear inequalities on specific user requirements, e.g. maximum allowable walked distance, minimum calories, etc. They involve some of the decision variables $\boldsymbol{p}$ and $Y_{i,j}$ for certain indexes i, j and the corresponding nodes and edges. For instance, if a person cannot walk for more than 15 minutes, we could have constraints of the form: $Y_{i,j} = 1 \implies \mathcal{T}(\mathcal{Y}_{i,j}) \leq 15$, which can be written in algebraic form as $Y_{i,j} \cdot \mathcal{T}(\mathcal{Y}_{i,j}) \leq 15$. If the constraint is not critical it takes the form: $Y_{i,j} = 1 \implies \mathbb{P}\left(\mathcal{T}(\mathcal{Y}_{i,j}) \leq 15\right) \geq 1 - \alpha$ (or, equivalently, $\mathbb{P}\left(Y_{i,j} \cdot \mathcal{T}(\mathcal{Y}_{i,j}) \leq 15\right) \geq 1 - \alpha$), where α quantifies the probability of violation. Another set of constraints is needed to enforce a sequential structure to the visit of the different POIs and to avoid sub-tours, that are isolated loops disconnected from the actual solution path. We employ an effective approach based on *lazy constraints* for sub-tour elimination [104]. This kind of approach is widely used to efficiently solve limited size instances of the Travelling Salesman Problem. Lazy Constraints are natively supported by many standard solvers, e.g. Cplex, Gurobi, GLPK. The number of sub-tours in a graph is exponential in the size, so the idea is to initially ignore all the constraints required to forbid solutions presenting sub-tours. During the search of the optimal solution, sub-tour constraints are instead dynamically generated only when necessary. Indeed, when a new solution is

generated, the absence of sub-tours is ensured. When a sub-tour is detected, a new lazy constraint is generated. It takes the form $\sum_{i,j} Y_{i,j} \leq \text{dim} - 1$, where dim is the number of edges in the sub-tour, while the sum is done over the indices (i, j) of the edges that are part of the sub-tour. The problem can be solved also without the lazy constraints, by considering additional variables and static constraints. To this end, we introduce the integer variables $\boldsymbol{t}$, of the same dimension of $\boldsymbol{p}$, that represent the order of the sequence of the visited POIs. These variables are introduced to avoid loops in the solution, by requiring $Y_{i,j} = 1 \implies t_j = t_i + 1$. This constraint can be written in algebraic form as $(Y_{i,j} - 1)m + 1 \leq t_j - t_i \leq (1 - Y_{i,j})m + 1$, where m is a large and positive constant. Whenever $Y_{i,j} = 1$, the constraint is active, since the inequality becomes $1 \leq t_j - t_i \leq 1$, implying that $t_j = t_i + 1$. On the contrary, if $Y_{i,j} = 0$ then $-m + 1 \leq t_j - t_i \leq m + 1$, that is never active (since m is a large constant). In Section 7.6 we show a quantitative comparison of these two methods, based on lazy constraints or on the introduction of additional variables.

Finally, to make the problem consistent, we need some additional constraints. First, we need to define an initial and a final POI (which can be trivially the entry and exit points of the environment). Second, for all intermediate points we need to have an incoming and an outgoing edge: $p_i = 1 \implies \exists j, h \text{ s.t. } Y_{j,i} = 1 \wedge Y_{i,h} = 1$. These constraints can be equivalently written as: $p_i = \sum_j Y_{j,i}$, $p_i = \sum_h Y_{i,h}$.

All the constraints discussed above, except for the probabilistic ones, have a linear algebraic structure (which would lead us to an integer linear program). The presence of probabilities for the non-critical constraints disrupts this structure, transforming the problem into a form of Chance-Constrained Stochastic Programming (CCSP) [105]. Indeed, such probabilistic constraints can be expressed as $\mathbb{P}(\cdot) \geq 1 - \alpha$, where the argument is a linear inequality. Combining such inequalities, we can write the probabilistic constraints as the product of a matrix with a subset of the decision variables. The term $1 - \alpha$ is the criticality level decided by the user for that requirement. In conclusion, we have modelled the problem as a CCSP with a linear objective function, a set of linear inequalities and additional stochastic constraints. We call $\boldsymbol{x}$ the vector of all the integer decision variables to be optimised ($\boldsymbol{p}$, $\boldsymbol{Y}$ and $\boldsymbol{t}$), and we collect all the inequalities coming from soft, hard and consistency constraints in $\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}$. In this way, it is possible to recast

the complete optimisation problem in canonical form:

$$\begin{aligned} \max_{\boldsymbol{x}\in\mathbb{N}^n} \boldsymbol{c}^T\boldsymbol{x} \qquad s.t. \quad & \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}, \\ & \mathbb{P}\left(\tilde{\boldsymbol{D}}\boldsymbol{x} \leq \boldsymbol{v}\right) \geq \mathbf{1} - \boldsymbol{\alpha}. \end{aligned} \tag{7.1}$$

where $\boldsymbol{x} \in \mathbb{N}^n$, $\boldsymbol{c} \in \mathbb{R}^n$, $\boldsymbol{A} \in \mathbb{R}^{k\times n}$, $\tilde{\boldsymbol{D}} \in \mathbb{R}_+^{N\times n}$, is the matrix that collects the probabilistic properties of nodes and edges, $\boldsymbol{v} \in \mathbb{R}_+^N$, $\boldsymbol{b} \in \mathbb{R}^k$ and $\boldsymbol{\alpha} \in (0,1)^N$. Finally, $\mathbb{P}\left(\tilde{\boldsymbol{D}}\boldsymbol{x} \leq \boldsymbol{v}\right) \geq \mathbf{1}-\boldsymbol{\alpha}$ is the term that formalises the non-critical constraints, which can be satisfied in a probabilistic measure only. As a remark, n denotes the number of optimisation variables, k the number of deterministic constraints, N the number of probabilistic constraints.

7.5 Proposed solutions of the Problem

The CCSP optimisation problem (7.1) is an effective approach for solving optimisation problems under uncertainties [106]. In general, a CCSP is hard to solve, since it is not possible to directly apply off-the-shelf numeric solvers, due to the presence of probabilistic constraints [107, 108]. However, in particular cases, if the distribution of the random matrix $\tilde{\boldsymbol{D}}$ shares some mild mathematical properties (e.g. Gaussian pdf), the problem can be reformulated as a deterministic Integer Programming (e.g. an instance of a Second Order Cone Programming). In the general case, i.e. when the distribution is not known, the problem can be approximated with the Sample Average Approximation (SAA) approach within an interval of confidence [105]. Although it would be possible to solve the probabilistic inequality $\tilde{\boldsymbol{D}}\boldsymbol{x} \leq \boldsymbol{v}$ in joint probability [108], we consider instead the inequality as a set of individual chance constraints, since they have heterogeneous origins. After this first simplification step, we can deal with a scalar quantity for each probabilistic constraint, that is $\mathbb{P}\left(\boldsymbol{d}^{(i)^T}\boldsymbol{x} \leq v^{(i)}\right) \geq 1 - \alpha^{(i)}$, for $i = 1, 2, \ldots, N$, where $\boldsymbol{d}^{(i)}$ is the i^{th} row of $\tilde{\boldsymbol{D}}$ and $\alpha^{(i)}$ is the criticality level associated with constraint i.

7.5.1 First Solution: Second Order Cone Programming

Under the hypothesis that $\boldsymbol{d}^{(i)}$ is a random vector independent and normally distributed, $\boldsymbol{d}^{(i)} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, it is possible to convert the CCSP (7.1) into a deterministic Second Order Cone Programming (SOCP). This formulation preserves

convexity of the relaxed problem and is efficiently solved by many numeric tools (e.g. CPLEX, Gurobi, Fico-Xpress, and others). Let $\boldsymbol{d}^{(i)^T}\boldsymbol{x} \leq v^{(i)}$, if $\boldsymbol{d}^{(i)} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, then $\boldsymbol{d}^{(i)^T}\boldsymbol{x} - v^{(i)}$ is distributed as $\mathcal{N}(\boldsymbol{\mu}^T\boldsymbol{x} - v^{(i)}, \boldsymbol{x}^T\boldsymbol{\Sigma}\boldsymbol{x})$. Therefore

$$\mathbb{P}\left(\boldsymbol{d}^{(i)^T}\boldsymbol{x} \leq v^{(i)}\right) = \Phi\left(\frac{v^{(i)} - \boldsymbol{\mu}^T\boldsymbol{x}}{\sqrt{\boldsymbol{x}^T\boldsymbol{\Sigma}\boldsymbol{x}}}\right), \tag{7.2}$$

where Φ is the standard Gaussian cumulative distribution function (see [109]). We are interested in (7.2) greater than $1-\alpha^{(i)}$, thus, by inverting the previous relation, we obtain the deterministic bound:

$$v^{(i)} - \boldsymbol{\mu}^T\boldsymbol{x} \geq \Phi^{-1}\left(1-\alpha^{(i)}\right)\left|\left|\sqrt{\boldsymbol{\Sigma}}\boldsymbol{x}\right|\right|_2, \tag{7.3}$$

which represents a quadratic cone constraint. We have thus restated problem (7.1), for $i = 1, 2, \ldots, N$, as

$$\begin{aligned} \max_{\boldsymbol{x}\in\mathbb{N}^n} \boldsymbol{c}^T\boldsymbol{x} \qquad & s.t. \\ & \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}, \\ & v^{(i)} - \boldsymbol{\mu}^T\boldsymbol{x} \geq \Phi^{-1}\left(1-\alpha^{(i)}\right)\left|\left|\sqrt{\boldsymbol{\Sigma}}\boldsymbol{x}\right|\right|_2. \end{aligned} \tag{7.4}$$

7.5.2 Second Solution: Sample Average Approximation

As stated previously, if the distribution of the random matrix $\tilde{\boldsymbol{D}}$ is not known, not available, or cannot be handled efficiently, it is possible to approximate the problem (7.1) with the Sample Average Approximation (SAA) [108]. We start by formulating an equivalent version of (7.1) in terms of the probability of violation of the stochastic constraint:

$$\begin{aligned} \max_{\boldsymbol{x}\in\mathbb{N}^n} \boldsymbol{c}^T\boldsymbol{x} \qquad & s.t. \quad \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}, \\ & \mathbb{P}\left(\tilde{\boldsymbol{D}}\boldsymbol{x} > \boldsymbol{v}\right) \leq \alpha. \end{aligned} \tag{7.5}$$

As discussed above, we can consider the stochastic constraint as a set of individual scalar probabilistic inequalities, e.g. $\mathbb{P}\left(\boldsymbol{d}^{(i)^T}\boldsymbol{x} > v^{(i)}\right) \leq \alpha^{(i)}$, for $i = 1, \ldots, N$. At this stage we consider $\{\hat{\boldsymbol{d}}^{(i,j)}\}_{j=1}^{M}$ to be M independent identically distributed samples of the random vector $\boldsymbol{d}^{(i)}$. Let $\hat{p}_M^{(i)}(\boldsymbol{x})$ be the proportion of times that

$(\hat{\boldsymbol{d}}^{(i,j)})^T\boldsymbol{x} > v^{(i)}$, in other words:

$$\hat{p}_M^{(i)}(\boldsymbol{x}) = \frac{1}{M}\sum_{j=1}^{M} \mathbb{1}_{(0,\infty)}((\hat{\boldsymbol{d}}^{(i,j)})^T\boldsymbol{x} - v^{(i)}), \tag{7.6}$$

where $\mathbb{1}_{(0,\infty)} : \mathbb{R} \to \{0,1\}$ is the indicator function that evaluates to 1 if the argument is positive and zero otherwise. The optimisation problem associated to the M samples $\hat{\boldsymbol{d}}^{(i,j)}$ is

$$\begin{aligned} \max_{\boldsymbol{x}\in\mathbb{N}^n} \boldsymbol{c}^T\boldsymbol{x} \qquad s.t. \quad & \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}, \\ & \hat{p}_M^{(i)}(\boldsymbol{x}) \leq \gamma^{(i)}, \end{aligned} \tag{7.7}$$

for $i = 1, 2, \ldots, N$ and $\gamma^{(i)} \in [0,1]$ the criticality level of the SAA problem. We can express the inequality $\hat{p}_M^{(i)} \leq \gamma^{(i)}$ by introducing new Boolean slack variables $\boldsymbol{z} \in \{0,1\}^{N\times M}$, and approximate problem (7.1) with the SAA problem:

$$\begin{aligned} \max_{\boldsymbol{x}\in\mathbb{N}^n} \boldsymbol{c}^T\boldsymbol{x} \qquad s.t. \quad & \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}, \\ & (\hat{\boldsymbol{d}}^{(i,j)})^T\boldsymbol{x} + v^{(i)}\boldsymbol{z}^{(i,j)} \leq v^{(i)}, \\ & \frac{1}{M}\mathbb{1}^T\boldsymbol{z}^{(i)} \leq \gamma^{(i)}, \end{aligned} \tag{7.8}$$

for $i = 1, \ldots, N$ and $j = 1, \ldots, M$. More in depth, in order to satisfy $(\hat{\boldsymbol{d}}^{(i,j)})^T\boldsymbol{x} + v^{(i)}\boldsymbol{z}^{(i,j)} \leq v^{(i)}$, if there is no violation, then $(\hat{\boldsymbol{d}}^{(i,j)})^T\boldsymbol{x} \leq v^{(i)}$ and $\boldsymbol{z}^{(i,j)}$ is set to 0. On the other hand, if there is a violation, only $\boldsymbol{z}^{(i,j)} = 1$ makes the inequality true. The last inequality of (7.8) counts the violations and ensures that the average frequency is below the criticality level $\gamma^{(i)}$.

Remark 1 *According to [108] the criticality level $\boldsymbol{\gamma} = [\gamma^{(1)}, \ldots, \gamma^{(N)}]^T$ may be different from $\boldsymbol{\alpha} = [\alpha^{(1)}, \ldots, \alpha^{(N)}]^T$, however it is proven that for $\boldsymbol{\gamma} = \boldsymbol{\alpha}$ the solution of the approximated problem* (7.8) *converges to the solution of the original problem* (7.1) *with probability 1 as the size M of the samples increases. In the literature, to the best of the Authors' knowledge, there is not a rule to choose a proper value for M. Authors in [108] suggest an empirical strategy which consists in choosing $\boldsymbol{\gamma} = \boldsymbol{\alpha}/2$, solving many instances of problem* (7.8) *with different (small) values of M and keeping the best solution. This is easier than performing a single instance of* (7.8) *with a large number of samples M. Suggested values for M are in*

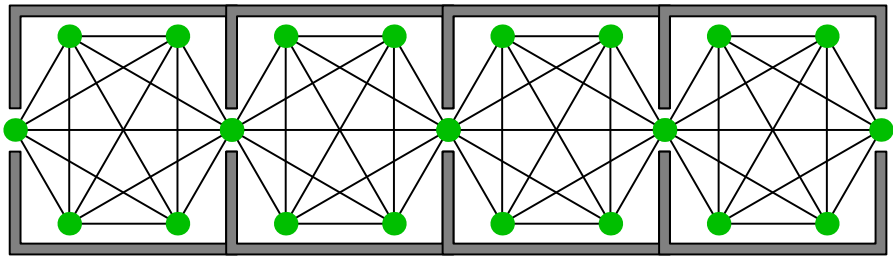


Figure 7.3: A portion of the museum with the abstract graph of the connections among the POIs. Inside each room, the corresponding subgraph is complete.

the range 50-120.

7.5.3 Hierarchical Decomposition and Optimisation

The above discussed SOCP and SAA approaches used to solve the activity planning can become computationally prohibitive if the number of points of interest and of their connections is large. On the other hand, if the size of a SOCP or SAA problem is small, solvers are capable of finding the optimal solution in small amounts ot time. The proposed idea is therefore to solve many small instances of different SOCP or SAA problems as building blocks for a global optimisation. In order to keep the whole computational time in the order of a few tens of seconds, we introduce a hierarchical decomposition of the nodes of the graph associated to the problem. We cluster the points of interest and connect those clusters with few edges. The idea is that a cluster is a portion of the museum with a certain number of rooms, each with some POIs, see Figure 7.3. In cases where such natural clustering is not straightforward (e.g. rooms, floor, etc.), it is always possible to partition an open space on the basis of the (relative) distances of the POIs. As discussed in the experimental section, the optimisation problem combining the results of the smaller subproblems can be solved very quickly, in the order of hundredths of a second. The cluster can represent a physical portion of the map, that should be visited as a single unit. In Figure 7.4 we show the complete graph (left) and its clustered version (right). The hierarchical decomposition allows us to shrink the full problem of Figure 7.4 into a small graph with 4 nodes and 6 edges. This high-level problem has to optimise the total score of the visited POIs and is constrained by some user requirements, for instance total time and maximum distance walked. These constraints are reflected in the single SOCP or SAA instances associated with each

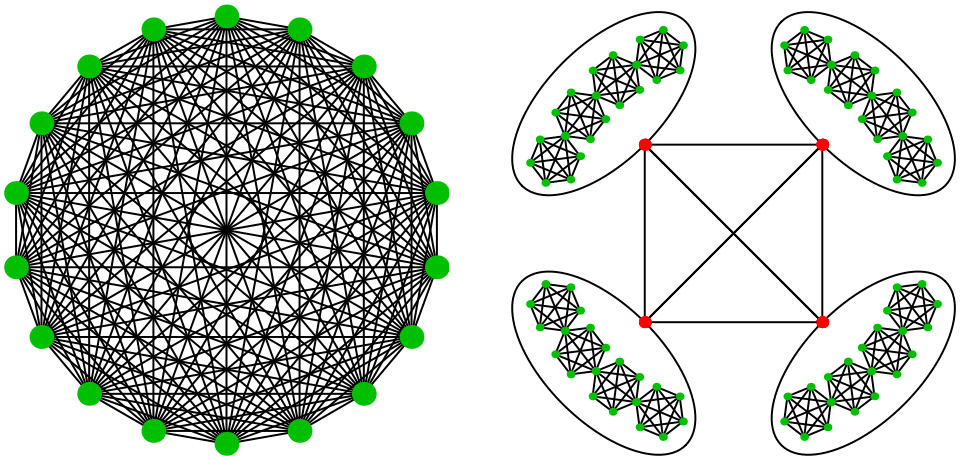


Figure 7.4: Left: the complete graph where all the POIs are connected with all other POIs. This case is computationally prohibitive if the number of POIs or connections is large. Right: the clustered version of the problem; each of the 4 clusters has the shape of the graph depicted in Figure 7.3.

cluster: we solve each problem multiple times with varying local constraints. In this way, the high level optimisation consists in finding for each cluster the optimal amount of time and walked distance that produce the best global score. We test the single subproblem on a discretisation of $N_C \in \mathbb{N}$ possible constraints (e.g. times, distances), thus the final global solution will be only suboptimal: the finer the discretisation, the better the approximation of the global solution. For instance, if the total allowable amount of time for the visit is 3 hours, we optimise the visit of each cluster with different time constraints, say for a maximum of 30, 60 and 90 minutes. It is important to notice that this decomposition allows us to tackle all the subproblems independently, which opens the way to a complete parallelisation. This decomposition has the advantage that the solutions of the small subproblems can be stored and used later to compose user-tailored plans.

The remainder of this section is dedicated to the mathematical formulation of the high-level hierarchical problem. Let n_χ be the number of clusters, then we define the matrices $\boldsymbol{D}, \boldsymbol{T} \in \mathbb{R}^{n_\chi \times N_C}$ of the local constraints. The elements of the matrices are defined as follows: D_{ij} (respectively T_{ij}) is the j^{th} maximum allowed walked distance (respectively dwell time) for cluster i. Analogously, $\boldsymbol{S} \in \mathbb{R}^{n_\chi \times N_C}$ is the matrix of the optimal scores associated to each subproblem and each discretisation. Finally, let $\boldsymbol{\chi} \in \{0,1\}^{n_\chi \times N_C}$ be the Boolean matrix such that χ_{ij} represents the

visit of cluster i with constraints j. The high level problem is therefore modelled as the following Integer Programming:

$$\begin{aligned} &\max_{\boldsymbol{\chi}} \sum_{i=1}^{n_\chi} \sum_{j=1}^{N_C} S_{ij}\chi_{ij} && s.t. \\ &\sum_{i=1}^{n_\chi} \sum_{j=1}^{N_C} T_{ij}\chi_{ij} \leq T_{\max}, \quad \sum_{i=1}^{n_\chi} \sum_{j=1}^{N_C} D_{ij}\chi_{ij} \leq D_{\max}, \\ &\sum_{j=1}^{N_C} \chi_{ij} \leq 1 && i = 1, \ldots, n_\chi. \end{aligned} \tag{7.9}$$

The last N_C equations are necessary to guarantee that each cluster is selected at most once.

7.6 Experimental Validation

In this section we present a numerical evaluation of the discussed activity planner, performed through the solution of a large number of different planning instances. Since the proposed application is to find the optimal plan for the visit to public spaces, such as museums or shopping malls, the first part of this section presents a concrete example of application of the developed activity planner to a concrete map inspired from a real museum. Then, in the second part of the section, we perform an extensive validation on a large number of synthetic, plausible scenarios of increasing size and complexity.

7.6.1 Realistic Scenario

To show a realistic example of applicability of our activity planner, we are considering now a map inspired from a real science museum (i.e., the MUSE in the city of Trento, Italy*), with different areas dedicated to different topics and categories of POIs. The considered layout is composed of two separate floors, connected by lifts. In Figure 7.5 is shown the map of the environment, with different thematic areas and a number of points of interest within each area. The scores given by the user to the various POIs (circles) are proportional to the size of the POIs (radius of the

*https://www.muse.it/en/Pages/default.aspx

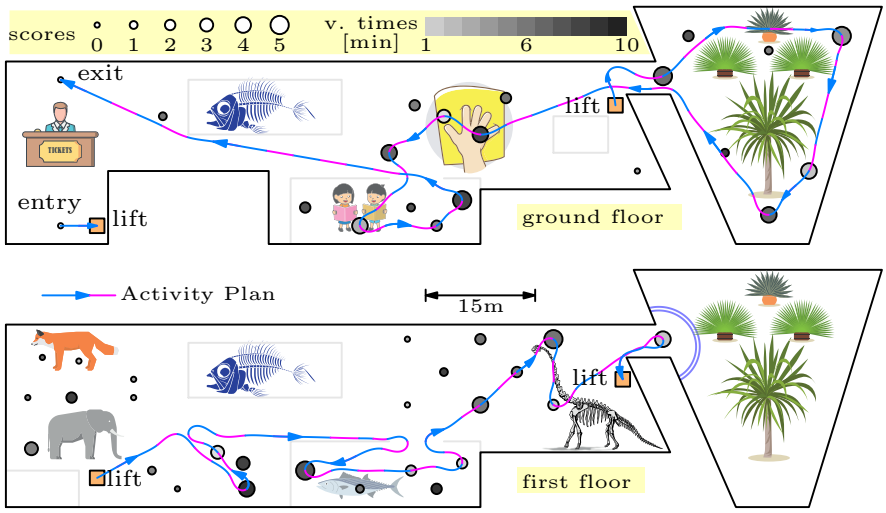


Figure 7.5: Map of the two (ground and first) floors of the museum. The synthesised activity plan is depicted as the blue/magenta solid line, with the direction indicated by the arrow. There are lifts (orange squares) to move the user from a floor to the other. The various thematic areas are sketched with a representative logo, like plants in the greenhouse and dinosaurs for the fossils. The POIs are represented as filled circles: the radius is proportional to the score ranging from 0 to 5; the visit time corresponds to the filling colour, ranging from light to dark grey, according to the colormap depicted in the legend.

circle), while the estimated visit time is proportional to the darkness of the filling, as reported in the legend colormap. An example of a synthesised optimal activity plan is displayed over the map. This plan corresponds to a constraint on the maximum length of the path of 1000 meters and a maximum visit time of two hours, both to be respected within 95% probability. The plan starts at the entrance of the museum (ground floor) and proceeds towards the lift (orange square in the top map of Figure 7.5) that brings to the first floor. Next, a few POIs about the mammals are visited, but more attention is paid to the area of fishes and dinosaurs. Finally, the panoramic balcony over the tropical greenhouse is reached before moving back to the ground floor using the lift. Then, the plan proceeds inside the greenhouse, where only the most rewarding POIs are visited. It can be noticed how some of the POIs are skipped even if they are on the way, due to their high visit time and low score. The final part of the plan is spent at the sensorial zone and within the kids area, before ending the visit. The computational time for this museum example

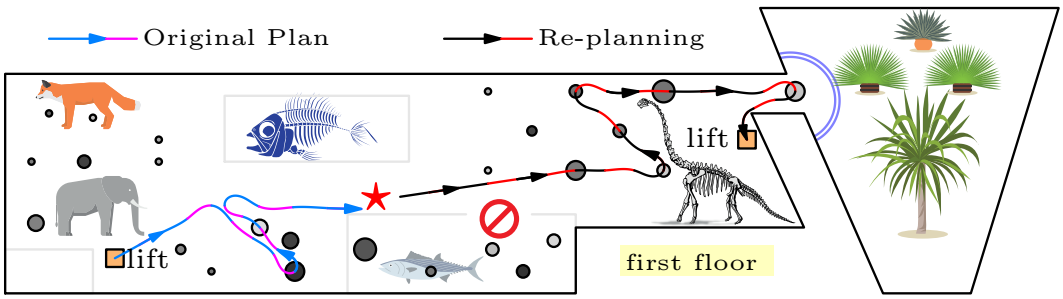


Figure 7.6: Re-planned path due to the unavailability of the fishes' exposition w.r.t. the original plan of Figure 7.5.

is about 22 seconds and the solver converges to the actual optimal solution. The overall length is 482 meters within 95% probability, while the overall visit time is four seconds below the upper limit of two hours within 95% probability, thus showing that in this case the problem is dominated by the time constraint. To show a case of replanning under unforeseen events, which is discussed at the end of Section 7.3, let us suppose that the programmed visit at the area of fishes cannot be carried out due to a maintenance service. The original plan becomes unfeasible, therefore the activity planner is invoked to produce an alternative plan (shown with the dashed black and red line in Figure 7.6). The re-planned activity adds a POI in the dinosaurs' area on the first floor, while the remaining part of the plan on the ground floor remains unaltered. The computational time required to synthesise the new plan is around 5 seconds.

7.6.2 Extensive Validation

In this section, we perform an extensive validation of the activity planner, by generating a large number of likely abstract map layouts. The environment is divided into clusters, representing specific physical areas (such as floors, sections of the museum, ...). Each cluster is composed by a sequence of rooms and each room contains points of interest. The layout is depicted, as a graph, in Figure 7.3 and in Figure 7.4. The points of interest within a room are all connected (i.e., the graph is complete) and the rooms are linked with an entrance and an exit point. The global constraints are the maximum visit time and the maximum walked distance. We solve this scenario with eight different techniques (see Table 7.1), discussed above: the SOCP and SAA problems, solved completely (full problem)

and with the hierarchical decomposition (identified with "_h"). The experiments are carried out with and without the lazy constraints (identified with "_ns", i.e. without sub-tour elimination constraints), as described in Section 7.4. We consider three base scenarios: Museum A consists of a museum with 4 clusters, 4 rooms per cluster and 4 POIs per room; Museum B has 5 clusters, 5 rooms per cluster and 5 POIs per room; finally, Museum C has 5 clusters, 8 rooms per cluster and 10 POIs per room. The experiments for each museum have been repeated 300 times, with different random parameters: scores and visit times of the POIs uniformly distributed, travelling times normally distributed, for a total of 900 different tested scenarios.

For the hierarchical decomposition, we choose two discretisations of N_C: in Table 7.1 $N_C = 9 (= 3 \times 3)$, that is with 3 values for the time and 3 values for the distance, while in Table 7.2 $N_C = 30 (= 6 \times 5)$. As highlighted in Section 7.5.3, the finer is the discretisation, the better is the solution, at the price of an increased computation time, as clearly evidenced by the higher computational times reported in Table 7.2 with respect to the corresponding hierarchical solutions in Table 7.1. The optimal scores for each combination of local constraints is obtained from the solution of the CCSP problem (7.1), which can be solved by method SOCP (7.4) or by SAA (7.8). There are four tuning parameters in the problem: N_C, $\boldsymbol{\alpha}$, $\boldsymbol{\gamma}$ and M. The number of discretisations N_C can be chosen according to the available computational resources. The criticality levels $\boldsymbol{\alpha}$ and $\boldsymbol{\gamma}$ do not affect the performance and represent the risk that a constraint is not satisfied. The user chooses the value $\boldsymbol{\alpha}$ (in the examples $\boldsymbol{\alpha}$ is chosen constant $\alpha \mathbb{1}$ for $\alpha = 0.05$), whereas Remark 1 discusses the choice of $\boldsymbol{\gamma} = \boldsymbol{\alpha}/2$. For the SAA problem, the fourth parameter, M, can be chosen in the range $50 - 120$ as explained in Remark 1. After extensive experiments we have found out that $M = 75$ gives a good compromise between accuracy (w.r.t. SOCP equivalent) and performance. Therefore, in the proposed examples, we generate a number of $M = 75$ random samples (from an estimated Gaussian distribution) that model various scenarios. The first step of the hierarchical solution is to solve the $n_\chi \times N_C$ (i.e. number of clusters times number of different discretisation) SOCP/SAA subproblems, obtaining the optimal scores $\boldsymbol{S}$. Then we solve the high level problem (7.9) to find the (suboptimal) global solution. The Integer Programming solver used in the simulation is Gurobi with Julia interface. It is worthwhile to note that since we apply standard numerical

algorithms, the completeness, complexity and time performance are inherited by the solver. The computational times are reported in Table 7.1, with the standard deviations in bracket. To keep the computational time contained, we set a timeout of 30 seconds for the solution time of the solver and we take the best solution so far computed, if it exists. Notice that the computational times include also the time required to solve the high level problem (7.9) (in the hierarchical case), which is negligible as it takes tens of milliseconds, and the time required to build the problem, that for big problems is not negligible.

The quality of the solution can be measured by comparing the result of the actual integer programming problem with its continuous relaxation. Since the latter is convex, any type of solver for SAA (linear program) or SOCP (second-order cone programming) produces the global optimum. Indeed, the non-convexity of the activity planning problem depends only on the integrality constraint of the decision variables. The optimal solution of the relaxed problem provides a conservative upper bound solution of the original problem, which can be used as means to evaluate the quality of the computed solution. During the solution process for this category of problems, the solver computes a sequence of relaxed versions of the problem, determining a non-increasing sequence of upper bounds for the optimal solution. The solver stops either when the gap between upper bound and current optimal solution reaches 0 or when a time out expires. In our case, the optimum is always achieved within a 30s time-out for both the small and medium sized scenarios, which correspond to a normalised score of 100% in Table 7.1. For the large sized problems, the time-out expires but the gap between the solution and the upper bound given by the relaxed problem is smaller than 15% on average. The solution quality is satisfactory from a practical point of view, even for a large amount of 400 POIs, which is way beyond the typical size of the problem for the envisioned application.

From these experiments we conclude the following remarks: the SOCP is in general faster than the SAA in the hierarchical subproblems; the number of 75 samples used for the SAA approximation gives in most cases the same result of the SOCP equivalent; with a small number of discretisations N_C we readily obtain a good sub-optimum final target; the hierarchical approach allows us to solve in a reasonable amount of time larger problems that cannot be solved directly, and to obtain acceptable solutions (see the statistics of Museum C). As discussed in

Table 7.1: Results for the three museums, mean times in seconds, std deviation in bracket, score in percent w.r.t. best found.

	Museum A		Museum B		Museum C	
Problem	time	score	time	score	time	score
saa	6 (1)	100	14 (5)	100	355 (55)	100
saa_h	21 (3)	77	66 (8)	72	320 (40)	80
saa_ns	20 (10)	100	111 (59)	100	426 (82)	98
saa_ns_h	30 (3)	77	108 (14)	72	990 (253)	80
socp	5 (4)	100	21 (11)	100	344 (55)	100
socp_h	10 (1)	77	31 (4)	72	191 (31)	80
socp_ns	19 (14)	100	104 (57)	100	413 (73)	91
socp_ns_h	21 (3)	77	80 (15)	72	652 (200)	80

Table 7.2: Results for the three museums, mean times in seconds, std. deviation in bracket, score in percent w.r.t. best found.

	Museum A		Museum B		Museum C	
Problem	time	score	time	score	time	score
saa_h	75 (8)	96	260 (24)	96	1633 (84)	82
saa_ns_h	104 (10)	96	394 (42)	96	2922 (341)	82
socp_h	34 (4)	96	118 (15)	96	1259 (49)	82
socp_ns_h	70 (8)	96	274 (36)	96	2245 (231)	82

Section 7.5.3, the hierarchical decomposition opens the way to the parallel solution of the subproblems. Moreover, the stored partial results can be reused for people with similar profiles.

Assistive Robotics Applications: Conclusions

In this first part of the thesis, we discussed the main issues related to the development of a complete planning framework for assistive robotics deployed in public spaces, such as museums, shopping malls, etc. and capable of providing a complete support during the execution of social activities. We identified three main interacting components: at the low level, there are the global and the reactive motion planner, responsible respectively for the generation of a global reference path between two assigned locations on the known map, and for its dynamic, real-time adjustment during the navigation to overcome unforeseen events, such as the presence of dynamic obstacles interfering with the reference trajectory. At the high level, there is the activity planner, responsible for the synthesis and the suggestion to the user of social activities to perform according to his/her profiles and constraints.

Motion and Reactive Planning

The motion planner seeks to generate smooth, natural and human-like paths optimising the comfort perceived by the user, by first determining a collision-free, minimum length route from the start to the goal location, and then by interpolating a sequence of waypoints along this route by a clothoid spline with G^2 continuity, in order to yield smooth paths. We defined a non-linear optimisation problem, minimising the overall jerk, that is directly related to the actual functional optimised by humans while walking.

During the navigation, the reactive planner predicts a set of probabilistic trajectories for the surrounding pedestrians, generated based on a simple, but accurate and extremely efficient short horizon approximation of a real human motion model, the HSFM. If the probability of collision with some of these dynamic obstacles is above a given threshold level, a set of alternative, local adjustments to the local trajectory, both in terms of reference velocity and path geometry (preserving the original properties of G^2 continuity and smoothness) are considered. The smallest local adjustment yielding to a probability of collision below the

threshold is chosen as the new reference trajectory. It is worth to notice that, whenever an alternative feasible solution cannot be found, e.g. in overcrowded scenarios, there is always a feasible, backup strategy that the robot can apply: to stop and let the dynamic obstacle move away. The purpose of the reactive planner is to reduce the number of this uncomfortable, complete stops of the vehicle to a minimum. Indeed, in the experiments, this backup strategy was employed only sporadically, only in densely populated environments, whenever a safe, feasible local adjustment was nonviable.

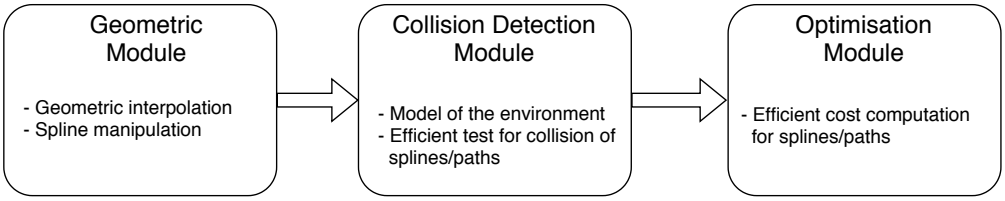
Activity Planning

The activity planner, based on the preferences and constraints specific for each user, synthesises a sequence of tasks defining a social activity to be performed by the elder with the support of the assistive robot. From the point of view of the robot, tasks consist in motions between points of interest (POIs) to be visited. Therefore, the environment is modelled by the activity planner as a graph, where the nodes correspond to the POIs, while edges represent motions between pairs of nodes. The probabilistic distributions of the physical parameters associated with each edge of the graph, such as the length and travel time, are estimated according to a large number of simulations performed in different realistic scenarios. This information could be complemented by additional data provided by some sensors deployed in the environment, such as surveillance cameras, that could be employed to estimate the amount of crowdedness. Therefore, an instance of an activity planning problem is modelled as a Chance Constrained Stochastic Programming: probabilistic constraints are provided regarding the physical parameters associated with the plan (e.g. the overall duration should be lower than 60 minutes with a probability of 95%). To define the objective of the optimisation problem by taking into account the preferences of the user, to each POI is associated a certain score, depending on its category. The target of the problem is therefore the maximisation of the overall score given by the user to the POIs that belong to the plan.

Part II

Automotive Applications

The second part of the thesis shows how the different core building blocks specifically developed during the implementation of the planning infrastructure for an assistive robot, can be adapted and applied to a completely different context, that is the synthesis of an optimal-time trajectory for an autonomous racing car, and its dynamic adjustment to cope with external factors such as the presence of unforeseen obstacles along the lane. We report here the same diagram illustrated in the introduction to the first part of the thesis, showing the different, core building blocks used in different ways to solve the different kind of motion planning problems.



As discussed in the next chapter, splines of clothoid curves provide a good approximation for the optimal trajectory of a racing vehicle. Therefore, the first two core building blocks (i.e. the geometric module and the collision detection module) developed and discussed in the first part of the thesis, are still applicable and will be used also for this second part. Indeed, the only missing module required by the global and local trajectory planner is the one responsible for the determination of the cost to assign to a given path. In this setting, we want to minimise the travel time for a car moving along a given spline (knowing the initial velocity and the parameters of the vehicle).

This is the focus of the next chapter of the thesis, where an efficient solution will be presented to compute the time-optimal manoeuvre for a car moving along a given clothoid spline. Then, the successive two chapters deal with the determination of optimal trajectories for autonomous racing cars, and on the reactive re-planning of such trajectories, respectively.

8

Trajectory Optimization on Clothoids

In this chapter we show how it is possible to obtain good approximations of the optimal trajectory for a racing vehicle by using a spline of clothoid curves, that allow us to apply the same kind of efficient algorithms presented in Chapter 4, and extensively applied also in the first part of the thesis. In addition, we discuss efficient solutions to compute the optimal (i.e. minimum time) trajectory to travel a path modelled as a clothoid spline, considering a dynamic model of the vehicle with bounds on the longitudinal acceleration, lateral slipping and aerodynamic drag.

8.1 Model of the Vehicle

The use of actual vehicle dynamics in analytic methods is often very difficult, hence simplified models are usually adopted to render the optimal trajectory synthesis a tractable problem. As observed by Fraichard et al. [84], one of the most widely adopted assumptions is to consider only trajectories with a continuous curvature, due to the continuous actuation of the driver on the steering wheel. The simplest possible cases of zero curvature, i.e. straight line, and constant curvature, i.e. arc of circle, are sometimes used for path synthesis. For example, the optimal path synthesis for the Dubins car [54] comprises exactly a sequence of lines and circles. However, a more general, but still tractable, path description with piecewise linear functions for path curvature, i.e., clothoids, can be used to synthesise optimal paths for a wider set of vehicle models (indeed, the arc of circle and the straight lines are special cases of clothoids). For instance, an actual car vehicle having the velocity

modelled as a linear ODE and limitations of the lateral velocity produces exactly a clothoid. Similar results can be obtained also for the classic model of a car-like robot, whose kinematic model in the cartesian xy-coordinates is given by:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\psi} \\ \dot{\bar{\delta}} \end{bmatrix} = \begin{bmatrix} \cos\psi \\ \sin\psi \\ \frac{\tan\bar{\delta}}{l} \\ 0 \end{bmatrix} v + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \bar{\omega}, \tag{8.1}$$

where ψ is the orientation (yaw angle) of the vehicle with respect to a right-handed reference frame having the Z axis perpendicular to the (X, Y) plane of motion, $\bar{\delta}$ is the steering angle, v is the forward velocity of the vehicle, $\bar{\omega}$ is the normalised angular velocity of the steering wheel and $l > 0$ is the wheelbase. For this model, [84] shows that the car-like robot time optimal trajectory is given by a sequence of clothoids whenever the velocity v is considered constant. In [83, 110] it has been shown that even with a varying velocity v, the curvature of the path for the model (8.1) can be safely approximated by a curve presenting a piecewise linear curvature, which corresponds to the definition of a clothoid or of a sequence of clothoids.

Using the auxiliary control input $\omega = (\delta^2 + 1)\,\bar{\omega}$, assuming that $\tan\bar{\delta} = \delta$, the original model (8.1) can be described with this new set of ODEs [83]

$$\begin{bmatrix} \dot{s} \\ \dot{n} \\ \dot{\alpha} \\ \dot{\delta} \\ \dot{v} \end{bmatrix} = \begin{bmatrix} \frac{v\cos\alpha}{1 - nk(s)} \\ v\sin\alpha \\ \frac{v}{l}\tan\delta - \dot{s}k(s) \\ \omega \\ a - c_0 v - c_1 v^2 \end{bmatrix}. \tag{8.2}$$

where $s(t)$ is the curvilinear abscissa (a customary choice for trajectory planning problems), n is the normal displacement of the vehicle with respect to the reference trajectory at the abscissa $s(t)$ (see Figure 8.1), $\alpha(t)$ is the angle of the local tangent to the clothoid path with respect to the longitudinal direction of the vehicle named X_v-axis, $a(t)$ is the longitudinal acceleration of the vehicle, $c_0 > 0$ and $c_1 > 0$ are the laminar friction and the aerodynamic drag coefficients and $k(s) = \kappa + \kappa' s$ is

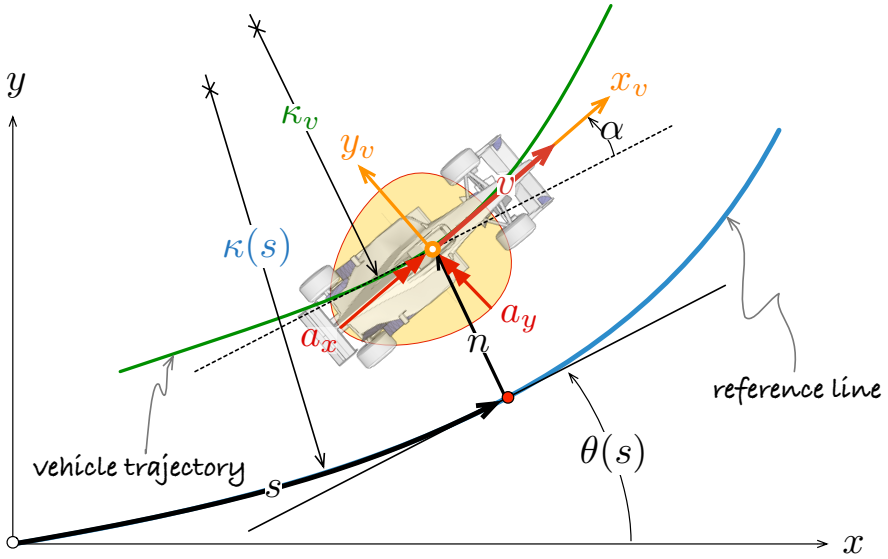


Figure 8.1: Curvilinear coordinates (s, n, α) defined with respect to a clothoid reference trajectory. X and Y represent the absolute frame axes, $V(t)$ is the absolute velocity of vehicle centre of mass, which is tangent to the trajectory, and $\beta(t)$ is the chassis slip angle. Finally, $\alpha(t)$ is the orientation of the vehicle w.r.t. reference line (i.e. clothoid) local tangent. [Published in [14]]

the curvature of the reference clothoid having $\kappa, \kappa' \in \mathbb{R}$ as curvature parameters.

8.2 Optimal Trajectory

The vehicle accelerations have to satisfy both longitudinal and lateral constraints. The former derive from the limited power of the engine and can be expressed as $-\underline{a} \leq a(t) \leq \overline{a}$. The latter, instead, derive from the limited friction of the tyres on the ground. These constraints are commonly modelled by the so called *friction ellipse* (see Figure 8.1), which limits the feasible manoeuvres of the vehicle by modelling the lateral acceleration $a_l(t) = k(s)v^2(s)$ and the longitudinal acceleration $a(t)$, as follows:

$$\left(\frac{v\dot{v}}{a_x}\right)^2 + \left(\frac{k(s)v^2(s)}{a_y}\right)^2 \leq 1, \tag{8.3}$$

where a_x and a_y are the maximum accelerations allowed for the longitudinal and lateral accelerations, respectively. The limits on $a(t)$ are introduced by means of the ODE $v\dot{v}$. In the computation of the optimal control, to obtain an analytic

closed form solution, the ellipse is simplified with a rectangle, i.e. $|k(s)v^2(s)| \leq a_y$. Using the rectangle inscribed in the ellipse, the generated solution is conservative and has an optimal time that is higher with respect to the true optimum, obtained using the whole ellipse. On the other hand, the application of more restrictive bounds guarantees the satisfaction of the original constraints.

An efficient solution for this optimal control problem is discussed in [111, 83, 110]. The approach is based on the execution of five steps. During each of this steps, the original clothoid segments are split into subsegments, in correspondence with switching points, i.e. points where some of the constraints become active or inactive. During each stage, each segment is labelled with a status, depending on the kind of active constraints. The final solution will then consist of arcs run at maximum acceleration, minimum acceleration and at the maximum acceleration profile compatible with the constraints (also this case can be expressed analytically [111]).

- **Preprocessing**: during this phase, segments are partitioned and classified depending on whether only longitudinal constraints are possible, or also lateral constraints may be active.

- **Saturation Analysis**: after the preprocessing step, there may be some segments classified as having the lateral constraint active, but for which it would be actually impossible to reach the corresponding velocity limit, e.g. due to a physical limitation of the actuators, or due to the friction and aerodynamic drag. During this phase, the segments are furtherly split and the classification is updated accordingly, considering also the longitudinal constraints. Therefore, at the end of this phase, all the segments for which the lateral constraint may be active in the final solution are labelled correctly.

- **Forward Sweep**: in this phase, the velocity is integrated forward in space, starting from the first segment, using the maximum acceleration allowed for each segment, and saturating the new velocity whenever it becomes larger than the maximum allowed velocity computed during the previous steps.

- **Backward Sweep**: during this phase, the velocity profile is analysed starting from the last segment, and moving backwards to the first. To remove discontinuities in the velocity profile, whenever the starting velocity of the new segment is smaller the final velocity of the previous segment, the optimal

braking control is computed, and the segments and the velocity profile are updated accordingly.

- **Boundary Conditions**: at the end of the backward sweep phase, we have determined the optimal travel time on the given path, and the maximum allowed initial and final velocity. The last phase of the algorithm consists in the application of the boundary conditions, i.e. the prescribed initial and final velocity. If this values are greater than the maximum allowed velocities, the problem is unfeasible. Otherwise, the velocity profile is updated by integrating and applying the new values for the initial and final velocity, considering the acceleration limits and the velocity profile computed during the previous steps of the algorithm.

In Figure 8.2 is shown an example of a velocity profile generated after the execution of each step of the algorithm.

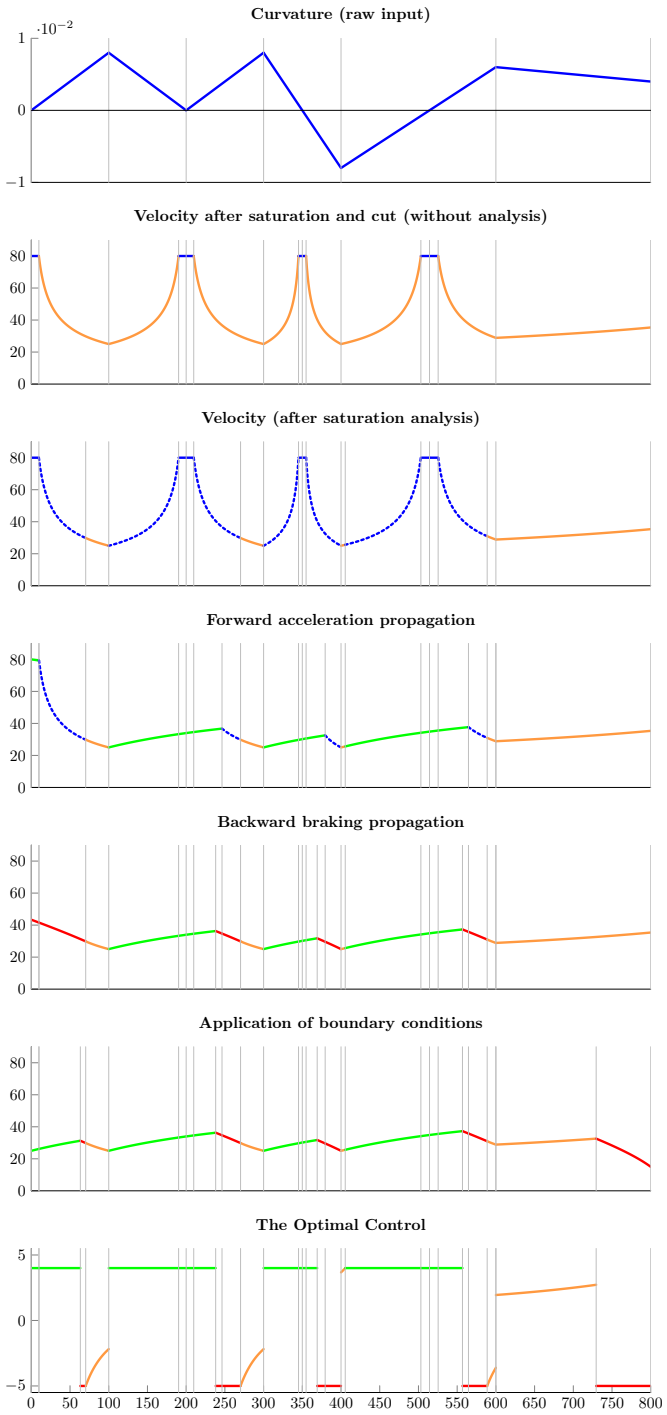


Figure 8.2: The output velocity profile generated by the different phases of the optimization algorithm.

9 Minimum Time Trajectory

9.1 General Overview

In this chapter we present an efficient approach for the planning of trajectories for autonomous racing cars, trying to minimize the travel time.

Since we are considering racing vehicles, the trajectory starts from a given position at the beginning of the track, while the final configuration is not fixed, but is instead chosen freely along the finish line. Moreover, the planned trajectory has to always remain within the boundaries of the track. The vehicle is a car–like, and is described by a dynamic model that considers bounds on the longitudinal acceleration, lateral slipping and aerodynamic drag, as illustrated in Chapter 8. In addition, other geometric constraints can derive from the occasional presence of an obstacle on the lane. As discussed in Chapter 8, our approach is based on the generation of paths consisting in sequences of clothoids.

The proposed approach is based on a modular solution in which three different problems are identified and solved separately. The first problem is, given two points in the Euclidean space, two vectors expressing the direction of the velocity vector and two curvature values at these endpoints, to find a clothoid spline linking the two points and satisfying the imposed constraints. The second problem is, given a segment of clothoid, to verify that it respects the geometric constraints, i.e. that it is fully contained in the boundaries of the track, and not in collision with any obstacle. The third problem is, given a path made by a sequence of interconnected clothoids, and a car–like vehicle described by its dynamic model, to find the optimal acceleration profile to move the vehicle from the start to the

end of the path in minimum time. We have already illustrated the basic, efficient building blocks that are required to solve all these three different problems. Indeed, the first and the second problems correspond to a G^2 Hermite interpolation and to a test for intersection involving clothoids; efficient solutions for both of them have been already illustrated in Chapter 4, and applied also for the development of motion planning solutions for assistive robotics. On the other hand, an efficient solution to the last problem, i.e. the determination of the minimum time solution given a clothoid spline and the parameters of the vehicle, has been presented in Chapter 8.

To find a complete solution to the general problem, we will therefore propose an algorithm that combines and uses all these three modules to identify the "optimal lap" on a racing track.

Inspired from tree based search algorithms commonly adopted in the field of motion planning, our solution is based on the growth of a tree of solutions incrementally expanding until the end of the track is reached. During the expansion of the tree, a set of different candidate trajectories are generated and compared using the travel time as cost function. After each iteration of the algorithm, only a small subset of best solutions are retained in the search tree. At the end of the search, the best solution is elected as the reference trajectory.

9.2 Optimal Trajectory Planning Algorithm

The proposed modular approach is based on a "master" algorithm based on an iterative search, with a dynamic cost function (representing the minimum travel time solution).

At the beginning of the algorithm, the projected reference position along the centre line is set to the start line of the circuit, and a node corresponding to the initial configuration of the vehicle is inserted into the tree. Then, a set of waylines, orthogonal to the lane, placed at different lookahead distances (measured along the centre line) with respect to the current car position are generated. For each wayline, a set of points is sampled at a uniform distance, then, for each of these points, a "tentacle", consisting in a G^2 clothoid spline, is grown from each of the current tree leaves (see Figure 9.1). After the generation of each tentacle, collision detection is performed by first building the corresponding tree of axis-aligned bounding boxes,

and by detecting possible intersections with the boundaries of the track, or with known obstacles lying along the lane. Among the set of all the feasible tentacles reaching a certain point along a wayline, we keep only the one with the lowest overall travel time. The computation of this information entails the invocation of the third submodule. For each node of the tree we keep track of the arrival velocity and the minimum optimal time to reach it. In this way, the computation of the minimum final time at the end of a tentacle can be performed incrementally, going back along the tree of nodes by one level at each step, and trying to compute the velocity profile along the corresponding subpath. The initial velocity is set to the arrival velocity associated with the initial point of the subpath. The procedure is repeated on longer and longer subpaths (up to the root of the tree), until we find a feasible velocity profile. The minimum time is then the sum of the time to reach the initial point of the subpath, plus the optimal time to cover the subpath.

When all the sampled points along the waylines have been processed, to each of them is associated the best tentacle connecting it to the current search tree (if at least one feasible connection exists), and the leaf node in the search tree from whom the tentacle originates. At this point, the algorithm proceeds by moving forward the current projected reference on the centre line by a certain step, and by determining for each of the current tentacles the pose, velocity and arrival time corresponding to that projection on the centre line (i.e. the point along the tentacle closest to the reference point on the centre line). A new node is generated in correspondence with this pose, velocity and arrival time, and stored in the search tree. Once all the new "frontier" nodes have been added to the tree, a new iteration of the algorithm begins, and the same procedure is applied until the end of the lane is reached.

The pseudocode illustrating this solution is presented in Algorithm 4.

9.3 Experimental Validation

In order to validate the effectiveness of the proposed modular solution, we apply it to four different real racing circuits, that are Silverstone (Great Britain), Monza (Italy), Monaco and Valencia (Spain), sampled from public on-line maps.

We compare our solution with a state-of-the-art approach for the synthesis of a time optimal trajectory on a racing circuit, based on the formalisation of the

Algorithm 4: Pseudocode of the proposed algorithm to determine the minimum time trajectory for a racing car on a given track

```
1  function FindPath
2      n0 = (x_0, y_0, θ_0, κ_0, v_0 0, {})
3      s = 0
4      tentacles = {}
5      frontier = {n0}
6      while s < path.length do
7          forall (id, dst) ∈ nextwaylines(s) do
8              forall src ∈ frontier do
9                  p = path(src, dst)
10                 if ¬collision(p) then
11                     vp = velProfile(src, p)
12                     if cost(tentacles[id]) > vp.totalTime then
13                         tentacles[id] = (src.path, p, vp)
14                     end
15                 end
16             end
17         end
18         frontier = {}
19         s = s + s0
20         ref = midlane(s)
21         forall t ∈ tentacles do
22             proj = projection(t, ref)
23             vel = velocity(t.vp, proj)
24             t = time(t.vp, proj)
25             n = (proj.x, proj.y, proj.θ, proj.κ, vel, t, proj.path)
26             push(frontier, nc)
27         end
28     end
29     optTime = inf
30     optPath = {}
31     forall node ∈ frontier do
32         if n.time < optTime then
33             optTime = n.time
34             optPath = n.path
35         end
36     end
37     return (optTime, optPath)
38 end
```

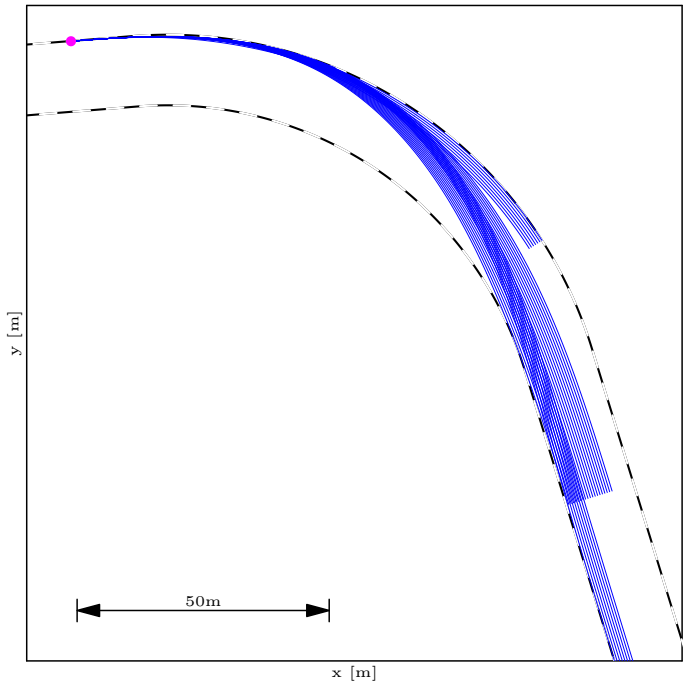


Figure 9.1: Example of candidate "tentacles" grown by the algorithm, connecting a starting node (magenta circle) with sample points at different lookahead distances.

problem as an Optimal Control, and on its solution by means of the numerical OCP solver Pins. This software is based on an indirect method with penalties and barriers to handle generic mixed state and control constraints [112, 113, 63]. Model (8.2) is used to define the OCP, therefore there is no imposed trajectory. We fix the initial position on the centre line, at the beginning of the track, and the initial velocity at 50 km/h. We use the same approximated friction ellipse (8.3) with lateral acceleration $|k(s)v^2(s)| \leq a_y$ considered by our algorithm. In Figure 9.2 is shown a comparison of the output of our algorithm and Pins on a portion of the circuit of Monza. In Table 9.1 are reported the computational times and the lap times on the different circuits.

It can be see that our solution is close to the optimal one, generated by the numerical solver, in terms of travel time, but there is some gap. Also the performance in terms of computational time for the numeric solver is lower (but comparable) with respect to our approach.

However, an advantage of the proposed method is its robustness and flexibility. Indeed, to ensure the convergence of the numeric solver, a good tuning of the

	Comp. Time [s]		Travel Time [s]	
	Pins	Ours	Pins	Ours
Silverstone	61.16	109.41	75.15	76.86
Monza	61.52	88.00	74.68	75.72
Montecarlo	49.11	57.37	57.32	58.61
Valencia	51.73	79.87	65.63	66.44

Table 9.1: Comparison of the OCP numerical solver Pins and the proposed algorithm

parameters and of the weights is necessary to allow the algorithm to converge and to find an optimal solution. In addition, our approach allows far more flexibility in the modelling of the environment. For instance, the addition of a set of obstacles placed along the lane is trivial using our algorithm, and, as explained in Chapter 4, by using adequate spatial partitioning data structures to represent the environment, the impact on the computational time is almost negligible. To validate this claim, we perform a large number of tests with random obstacles placed along the lane, and the algorithm is able to produce a valid solution with computational times that are a few seconds higher with respect to the ones reported in Table 9.1. In Figure 9.3 is shown an example of how the original path is modified by the presence of an obstacle along the way. In addition, another important advantage of the proposed solution lies in the fact that the processing of each tentacle is independent from the others, therefore the algorithm is well tailored to be massively parallelised on modern hardware providing a large number of computing cores.

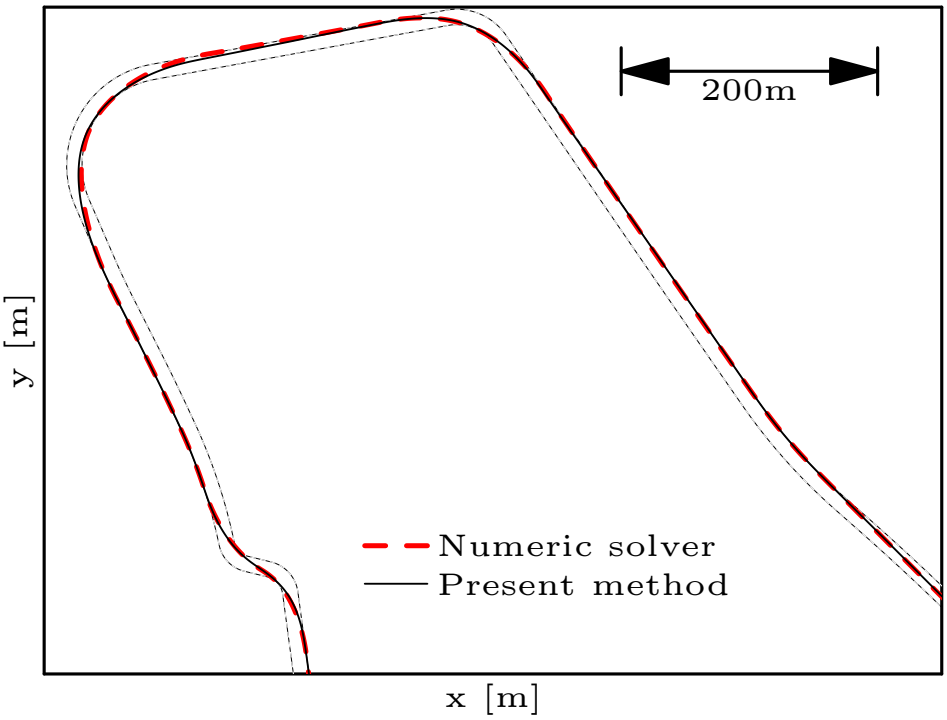


Figure 9.2: Comparison of the result of the numeric OCP solver Pins (dashed line) and of the present algorithm (solid line) on a portion of the circuit of Monza.

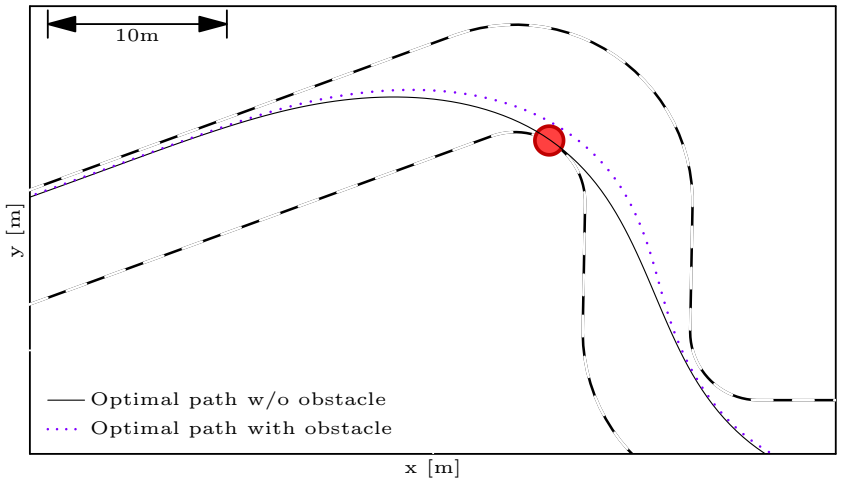


Figure 9.3: An obstacle is posed along the lane, the solid line represents the optimal trajectory without the obstacle, the dotted line is the optimised trajectory that avoids the obstacle.

10

Optimal Trajectory Replanning

10.1 General Overview

In the previous chapter we have presented an efficient and robust solution to find a minimum time trajectory on a track for a racing vehicle. In this chapter, instead, we address the problem of how to reactively re-plan on-line local modifications to the global, minimum time reference trajectory to avoid collisions with obstacles lying along the road. Since by assumption the vehicle is moving along a pre-defined trajectory, the proposed technique seeks to minimise the amount of local deviation and ensures that the new trajectory joins the previously planned one shortly after the obstacle.

We consider the same kind of dynamic and geometric constraints applied for the generation of the global trajectory in Chapter 9. The occasional presence of obstacles and of slower vehicles generates additional geometric constraints.

Our approach is based on the assumptions that the geometry of the track (i.e. road or lane limits) is known, that the vehicle is equipped with a sensing system able to reveal obstacles and anomalous conditions in the surroundings (e.g., slippery areas on the road), and that the trajectory re-planning has to be executed in real–time and adapted every time an unforeseen situation is detected. Another fundamental aspect that we are considering is the fact that the algorithm should run on a lean hardware, in order to reduce the costs and simplify the system engineering.

We apply an approach similar to the one presented in Chapter 9 to generate a global minimum time trajectory. The idea is to decompose the general planning

problem into a geometric and a dynamic optimisation component, handled separately by different algorithms.

When an obstacle is detected, the algorithm selects a point P_0 on the optimal trajectory with position (x_0, y_0), angle θ_0, curvature κ_0 and speed v_0 and a point P_2, with position (x_2, y_2), angle θ_2, curvature κ_2. The re-planned trajectory will depart from the reference at point P_0 and will rejoin it at point P_2. Notably, the speed in P_2 cannot be assigned, because, in general, it may not be reachable, e.g. if the global trajectory is time optimal. The algorithm seeks a new point P_1 in the proximity of the obstacle to pivot on to generate a set of candidate, alternative trajectories (see Figure 10.1). For each candidate point, a geometric subproblem is solved to find the corresponding path and to verify its feasibility (the re-planned spline has to be contained within the track limits). For each candidate path, a dynamic optimisation is executed to find the time optimal manoeuvre, if it exists, using the semi-analytic solution presented in Chapter 8.

10.2 Proposed Approach

For safety reasons, the reactive replanning algorithm first tries to compute an emergency braking manoeuvre, to determine whether it is possible to stop the vehicle before crashing into the obstacle, moving along the prescribed reference trajectory. This can be computed by determining the optimal velocity profile on the subpath from the current location of the car to the location of the obstacle, and imposing the final velocity to 0.

After the computation of an emergency manoeuvre, the algorithm tries to determine a local deviation from the reference trajectory that allows the vehicle to overtake the obstacle, as shown in Figure 10.1.

10.2.1 Reactive Replanning

The first important aspect that we need to consider is how to properly select the points P_0 and P_2, corresponding to the locations where the replanned trajectory joins the reference. In principle, the algorithm presented in this chapter operates with any pair of entry and exit points P_0–P_2 on the global trajectory. An obvious requirement is that P_0 and P_2 should be located before and after the obstacle. The

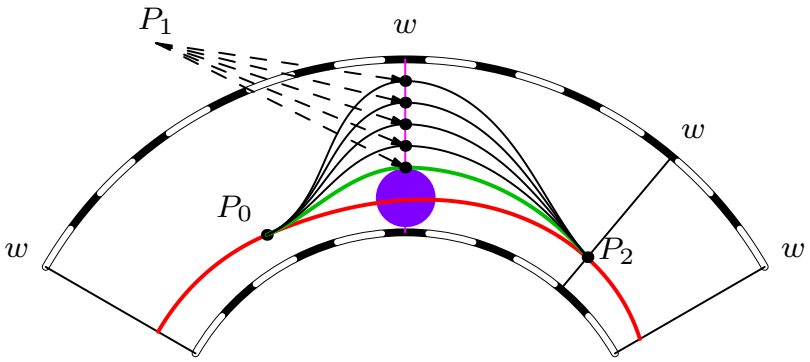


Figure 10.1: Structure of the track with a sketch of the used notation. The waylines w divide the track into sectors. In red the global trajectory that is now unfeasible due to the obstacle (purple circle). In green the optimal escaping manoeuvre, in black feasible candidates for different choices of P_1. [Published in [15]]

low computational cost of the algorithm allows us to test different possible choices or back off to an emergency strategy if the spline identified by the algorithm fails to satisfy the geometric or the dynamic constraints. However, the application of reasonable heuristics on the selection of P_0 and P_2 limits the occurrence of this anomaly.

An intuitive and straightforward way to select the points P_0 and P_2 is to identify them as the intersections of the global trajectory with a finite set of waylines superimposed to the track (see Figure 10.1). The distance between consecutive waylines can be tailored to the geometric features of the track and to the dynamic properties of the vehicle. For instance, waylines can be closer in curved sectors of the track and farther in straight sectors. When an obstacle is detected, P_0 can be chosen coincident with the current position of the vehicle. P_2 can be chosen on the intersection between the global trajectory and the first wayline beyond the obstacle.

Geometric Problem

The reactive replanner has to generate local deviations to the global trajectory preserving the G^2 continuity constraint.

As an useful remainder for the reader, we will report here the equations to enforce G^2 continuity for clothoid segments connecting P_i to P_{i+1}, discussed in

Chapter 4:

$$
\begin{aligned}
H_{i,0} &:= x_i + L_i X_0\big(\kappa_i' L_i^2, \kappa_i L_i, \theta_i\big) - x_{i+1}, && (10.1)\\
H_{i,1} &:= y_i + L_i Y_0\big(\kappa_i' L_i^2, \kappa_i L_i, \theta_i\big) - y_{i+1}, && (10.2)\\
H_{i,2} &:= (1/2)\kappa_i' L_i^2 + \kappa_i L_i + \theta_i - \theta_{i+1}, && (10.3)\\
H_{i,3} &:= \kappa_i' L_i + \kappa_i - \kappa_{i+1}. && (10.4)
\end{aligned}
$$

To enforce G^2 continuity, all the equations $H_{i,j}(\theta_i, \kappa_i, \kappa_i', L_i, \theta_{i+1}, \kappa_{i+1})$ for $j = 0, 1, 2, 3$ must be equal to 0.

While for points P_0 and P_2 both the position, the tangent and the curvature are given, at the intermediate point P_1 only the position is fixed, while the tangent and the curvature need to be chosen suitably by our algorithm. To reduce the computational cost, we resort to the G^2 Hermite Interpolation routine only twice, once all the parameters have been fixed, and the final interpolation has to be performed. We rely instead on the more efficient G^1 Hermite Interpolation routine to determine the missing parameters. The idea is to construct two clothoid arcs neglecting the curvature at P_0 and P_2, but with G^2 continuity at P_1. We then use the resulting angle and curvature at P_1, to generate the complete, G^2 spline interpolating the path from P_0 to P_1 and from P_1 to P_2.

In our case, the G^1 interpolation problem between P_0 an P_1 can be seen as a function of one unknown variable θ_1, since all the other parameters are fixed. Therefore we obtain $\kappa_0'(\theta_1)$, $\overline{\kappa}_0(\theta_1)$ and $L_0(\theta_1)$. Analogously, the G^1 interpolation problem between P_1 an P_2 yields the functions $\kappa_1'(\theta_1), \kappa_1(\theta_1)$ and $L_1(\theta_1)$. Since θ_1 corresponds to both the final orientation of the first segment and the initial orientation of the second segment, the resulting arcs match with G^1 continuity in P_1. Thus, we have a family of splines made by two clothoids parametrised with θ_1, and we can adjust this value in order to satisfy the G^2 constraint, i.e. the curvatures of the two arcs should match in P_1. This condition is

$$
h(\theta_1) := \kappa_0'(\theta_1) L_0(\theta_1) + \kappa_0(\theta_1) - \kappa_1(\theta_1) = 0, \qquad (10.5)
$$

which is simply equation (10.4) specialised to our scope. To find an angle θ_1 that

satisfies $h(\theta_1) = 0$, a standard Newton–Raphson scheme suffices. This approach works well in practice, and convergence is achieved in just 2 or 3 iterations. Once the tangent and curvature at P_1 have been determined, two G^2 Hermite Interpolation problems are solved between P_0 and P_1, and between P_1 and P_2, to generate a clothoid spline smoothly joining the original reference trajectory with G^2 continuity.

For each of the candidate alternative paths, we check if it is collision free and, in such case, we evaluate its time optimal speed profile. The whole computation process is extremely efficient, so it is possible to generate many different escaping manoeuvres and select the one with minimum travel time, as shown in Figure 10.1, where the green curve is selected among the set of black curves.

10.2.2 Replanning Algorithm

The pseudo-code of the proposed approach is shown in Algorithm 5.

The input is given by the initial and final configurations of the vehicle, thus position, angle and curvature, and by the initial velocity. The final velocity is neglected, since it is almost impossible to reconnect to the global trajectory with the optimal speed. Since the algorithm loops on all the available waypoints, they are also part of the input. The output is given by the best escaping manoeuvre computed by the reactive re-planning (if one exists). Notice that Algorithm 5 can be used also to safely stop the vehicle if an overtaking manoeuvre does not exist. The first part of the algorithm involves the search of the missing angle and curvature at P_1, namely θ_1 and κ_1, and is done by solving equation (10.5) (see line 4 and Algorithm 6). This computation requires to solve the G^1 interpolation problem for the two clothoids that connect P_0 to P_1 and P_1 to P_2. The resulting values for θ_1 and κ_1 are then used by the g2Hip function to compute the final smooth spline that connects P_0 to P_2 with G^2 continuity. The spline is also checked for collisions with the obstacle and the track limits. If the spline is valid, the time optimal speed profile is computed (fixing the initial velocity to the current velocity of the vehicle). When all the possible trajectories have been processed, the algorithm terminates and returns the solution with the lowest travel time.

Algorithm 5: Find a collision free partial path

Function ***AvoidObstacle***
- **Input :** conf_0, initial config. of the vehicle (x_0, y_0, θ_0, κ_0);
  conf_2, final config. of the vehicle (x_2, y_2, θ_2, κ_2);
  v_0, initial velocity of the vehicle;
  waypoints, intermediate waypoints.
- **Output :** {time, traj}, collision free traj. with lowest cost.

time $\leftarrow \infty$; traj $\leftarrow$ [ ];
foreach wp $\in$ waypoints **do**
  $[\theta_1, \kappa_1] \leftarrow$ findThetaKappa(conf_0,conf_2,wp);
  $\text{spline}_1 \leftarrow$ g2Hip(conf_0,{wp, θ_1, κ_1});
  if not collisionFree(spline_1) **then continue**;
  $\text{spline}_2 \leftarrow$ g2Hip({wp, θ_1, κ_1},conf_2);
  if not collisionFree(spline_2) **then continue**;
  currTraj $\leftarrow$ [$\text{spline}_1, \text{spline}_2$];
  {ok, currMan} $\leftarrow$ minTime(currTraj,v_0);
  if not ok **then continue**;
  [currTime] $\leftarrow$ time(currMan);
  if currTime < time **then**
    time $\leftarrow$ currTime; traj $\leftarrow$ currTraj
  end
end
return {time, traj};

Algorithm 6: Find intermediate angle and curvature

Function ***FindThetaKappa***
- **Input :** conf_0, initial config. of the vehicle (x_0, y_0, θ_0, κ_0);
  conf_2, final config. of the vehicle (x_2, y_2, θ_2, κ_2);
  wp, intermediate waypoint (x_1, y_1);
  {max_{iter}, tol}, termination criteria
- **Output :** {θ_1, κ_1}

$\theta_1 \leftarrow (\theta_0 + \theta_2)/2$;
for iter *from* 1 *to* max_{iter} **do**
  $[\bar{\kappa}, \bar{\kappa}', L, \bar{\kappa}_{\theta_1}, \bar{\kappa}'_{\theta_1}, L_{\theta_1}]_0 \leftarrow$ g1Hip(x_0,y_0,θ_0,x_1,y_1,θ_1);
  $[\kappa, \kappa', L, \kappa_{\theta_1}, \kappa'_{\theta_1}, L_{\theta_1}]_1 \leftarrow$ g1Hip(x_1,y_1,θ_1,x_2,y_2,θ_2);
  $h \leftarrow \bar{\kappa}_0 + \bar{\kappa}'_0 L_0 - \kappa_1$;
  $h' \leftarrow \bar{\kappa}_{0\theta_1} + \bar{\kappa}'_{0\theta_1} L_0 + \bar{\kappa}'_0 L_{0\theta_1} - \kappa_{1\theta_1}$;
  if $|h| <$ tol **then return** {θ_1, κ_1} ;
  $\theta_1 \leftarrow \theta_1 - h/h'$;
end
return {}

10.3 Experimental Validation

To validate the proposed approach, we applied it to adjust given time optimal trajectories (computed as discussed in Chapter 9) on given racing tracks to overcome some unexpected obstacles lying along the way.

As an example, in Figure 10.3 is shown an example of an escaping manoeuvre generated to avoid a circular obstacle of radius 5 m placed on the track. This example is based on the Formula 1 track of Silverstone, as shown in Figure 10.2.

We tested our solution on a large number of test cases on the same racing circuits used in Chapter 9, by randomly placing obstacles on different portions of the track, and by applying our algorithm to adjust the optimal reference trajectory accordingly. The proposed replanning algorithm is very fast (in the order of a few milliseconds on a Core I5 2.3 GHz machine, for all the experiments), and applicable for real-time adjustments of the reference trajectory.

The present method could be extended to consider also simple moving obstacles, for example by applying an approach similar to the one developed in Chapter 6, and by determining a safe combination of path geometry and velocity profile. Moreover, in the case of obstacles moving at much slower velocities in comparison with the controlled car, a simple and conservative, but valid approach, is to expand the obstacle along its motion direction, in such a way to cover and render inaccessible the entire portion of the road that it sweeps during the completion of the overtaking manoeuvre. Unfortunately, on the other hand, this solution by itself is not sufficient to handle the overtaking of another racing vehicle. Indeed, in this scenario, the simplifying assumption that the other vehicle would passively move along its reference trajectory, without trying to defend its position by impeding the manoeuvre of the overtaking car, would be really strong and unrealistic. Indeed, to effectively handle this scenario, an higher-level, strategic planner would be required to determine the best tactics and sequence of moves to perform, for example by applying classical game theoretical models. This problem is outside of the scope of this thesis, but represents an interesting opportunity for future research.

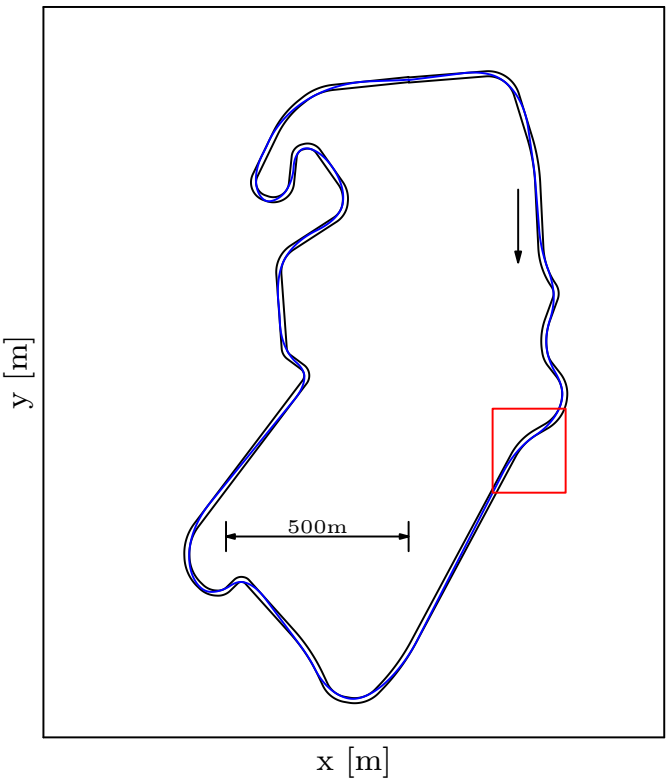


Figure 10.2: The circuit track of Silverstone, UK. In red the portion of the track that is zoomed in Figure 10.3. [Published in [15]]

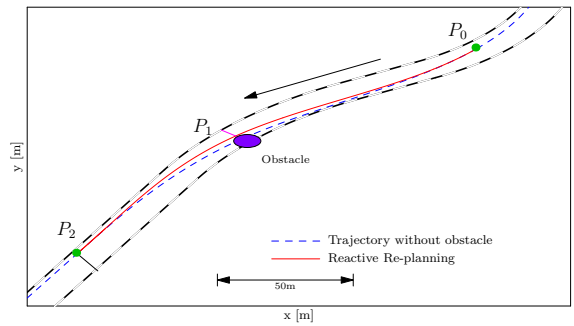


Figure 10.3: Re-planning in the presence of an obstacle. The original trajectory (in blue dashed) is no more feasible because of the obstacle. The re-planning (solid red line) can be done between two points on the waylines P_0 and P_2, because before P_0 and after P_2 the trajectory is close to the old one without the obstacle and allows a smooth reconnection. [Published in [15]]

Automotive Applications: Conclusions

This second part of the thesis was focused to show the adaptability and applicability of the core building blocks and algorithms developed in the first part of the thesis, specifically tailored to the implementation of a complete planning framework for assistive robotic applications, to a completely different context, such as the generation of optimal trajectories for racing vehicles driving at their limits. The software infrastructure that we developed to plan time-optimal trajectories for racing cars requires the interaction of three core components: a "geometric" module, responsible for the generation of a smooth path connecting two given poses. A "collision-detection" module, that, given the geometry of the road and a path, determines whether the path lies entirely within the road, and whether it intersects with any obstacle. Finally, a "trajectory optimisation" module, that, given a path and the parameters of both the vehicle and the road, determines the optimal velocity profile allowing the vehicle to travel the reference path in minimum time. To implement the first two modules, we resorted to the solutions developed and presented in the first part of the thesis, i.e. clothoid spline interpolation algorithms to synthesise smooth paths connecting pairs of given configurations, and spatial partitioning data structures to effectively determine possible intersections between a path and obstacles or road boundaries. On the other hand, we applied an existing state-of-the-art solution to determine the optimal manoeuvre for a vehicle moving on a clothoid spline in minimum time. Finally, we showed how it is possible to combine these three core modules both to synthesise an optimal reference global trajectory, and to dynamically adjust it in real-time to overcome the presence of unpredicted obstacles obstructing the way.

11

Conclusions and Future Work

In this thesis, we proposed a set of efficient algorithms and approaches to solve different kind of planning problems, always focusing on the computational efficiency, to produce solutions that are applicable also for mobile robotic platforms providing limited amounts of computational power. We identified common subproblems shared by the different kind of problems at hand, such as the smooth geometric interpolation of two given configurations, or the efficient check for collision, and we applied and developed efficient solutions for each of them.

11.1 Assistive Robotics

We developed a path panning solution specifically tailored to provide support to older and disabled users during the navigation of public spaces. Our approach is based on the separate solution of two different subproblems, i.e. the generation of a sequence of reference waypoints, and the smooth interpolation of these waypoints to produce paths optimising the comfort perceived by the user. The latter goal is achieved by using a nonholonomic car–like model that naturally generates smooth curves (clothoid splines), and by optimising a cost functional directly related with different dimensions of the user comfort.

Then, we presented an efficient solution to reactively modify the planned global path, in order to avoid pedestrians during the navigation in an environment populated by humans. The proposed key idea is the adoption of the HSFM, a model mathematically expressing human motion behaviours, to probabilistically predict possible paths followed by the pedestrians. We have shown how to approximate such

paths using splines of clothoid curves. The uncertainty on the velocity followed by the pedestrians, on the points that he wants to reach and on the possible deviations of the user from the suggested speed are modelled as stochastic variables. Therefore, we define a probabilistic performance index and seek the velocity profile that minimises it over all the predicted trajectories. Different alternative deviations from the reference are considered, and the one minimising the performance index and the deviation from the global plan is selected.

Finally, we developed a solution based on the strong cooperation between the low-level motion planning modules and the high-level activity planning module, to produce detailed activity plans that fit the users' interests, maximise their experience and meet their physical and psychological requirements. This is an essential asset for a robot offering assistance for the navigation of complex environments and for the execution of social activities. The motion planning modules are used to characterise elementary motion actions and to turn an activity plan into an actual sequence of motions implementable by the user-robot ensemble. The activity planning module, on the other hand, is used to generate the optimal concatenation of motion actions. Since each action is characterised in probabilistic terms and we associate failure probabilities to non mandatory requirements, our problem turns out to be a Chance Constrained Stochastic Programming with integer decision variables. In addition, we propose a suboptimal, hierarchical technique that produces very good solutions in a matter of few seconds also for moderately large environments.

The planning solutions developed for ACANTO, and mainly focusing on assistive robotics, can actually be generalised and adopted also for different applications. Indeed, as an example, we applied the proposed reactive replanning system also to robots employed in an autonomous warehouse, for the transportation of pallets between different locations. The application of our replanning system, allowed the robot to navigate in an environment shared with human operators, without the need to stop every time when the reference path was occluded. In the same way, the framework proposed for the planning of activities, based on a strict collaboration between the low-level and the high-level planning components, and on the statistical characterisation of the physical parameters associated with motion actions, has been applied within the context of autonomous warehousing. In this case, we integrated our low-level planning system with an external task planner and scheduler, to organise the handling and movement of the different pallets, in

order to maximize the number of satisfied requests.

11.2 Automotive Applications

We showed how it is possible to solve the challenging problem of determining the time optimal trajectory for a car–like dynamic model of a racing vehicle on a given track, by decomposing it into three smaller subproblems, that are efficiently solved by dedicated modules. The two geometric subproblems of interpolating two given configurations by a clothoid spline, and of detecting potential collisions with the boundaries of the track, are exactly the same that we had to deal with during the development of motion planning solutions for assistive robotics, and for which we already developed efficient solutions. We presented an efficient, state-of-the-art solution for the third subproblem, consisting in the determination of the optimal control for a car–like dynamic model moving on a clothoid spline, with bounds on the velocity, longitudinal acceleration, lateral slipping and aerodynamic drag. We proposed then a "master" algorithm, combining the three modules and applying an incremental search to generate the optimal lap. Finally, we presented an algorithm to find a minimum time deviation from the global, time optimal trajectory for a robotic car to avoid unforeseen obstacles lying on the track. We applied the same kind of modular decomposition adopted for the generation of the global trajectory. The approach is based on the deterministic generation of a set of alternative candidate paths, synthesized by sampling points on a "wayline" collocated in proximity of the obstacle, and on the selection of the minimum-time feasible solution. This algorithm is very efficient, taking just a few milliseconds to compute a solution, and can be realistically implemented on an embedded platform.

11.3 Future Work

Several possible research directions lie before us for further exploration and exploitation. In particular, a research topic of particular relevance regards the reactive re-planning strategy adopted to modify the global reference trajectory during the autonomous navigation in an environment populated by other humans. Indeed, in this thesis we have proposed a robust and efficient solution, assuming that all the responsibility for the avoidance of collisions has to be taken by the robot.

However, an interesting and challenging approach, is based on the recognition of the possibility that also the pedestrians that we are trying to avoid may be collaboratively and actively trying to adjust their trajectory in order to avoid the robot (at least up to some degree). In this setting, we are required first of all to understand whether the pedestrian is actively collaborating, or if we need to assume the full responsibility for the avoidance. In a collaborative setting, the robot has to understand the intentions of the human, establish a policy to accomplish the avoidance manoeuvre accordingly (e.g. pass on the left), and determine adjustments to the reference trajectory in accordance with this policy. The process has to be repeated iteratively, until a consensus is reached by the robot and the pedestrian on a common, shared avoidance strategy.

Another interesting extension applicable to various of the proposed motion planning approaches, is the possibility to massively parallelise the solution of a large number of different subproblem instances (e.g. the interpolation of pairs of configurations with clothoid curves), and to distribute the computational burden to a large number of processors. Indeed, with the development of general-purpose computing on GPUs, the deployment and adoption of this kind of hardware platforms on robotics and automotive systems is becoming more and more popular, also thanks to the development of ad-hoc embedded boards equipped with GPU cores.

Bibliography

[1] "Active ageing: A policy framework," *The Aging Male*, vol. 5, pp. 1–37, mar 2002.

[2] M. Andreetto, S. Divan, D. Fontanelli, and L. Palopoli, "Path following with authority sharing between humans and passive robotic walkers equipped with low-cost actuators," *IEEE Robotics and Automation Letters*, vol. 2, pp. 2271–2278, oct 2017.

[3] M. Andreetto, S. Divan, D. Fontanelli, and L. Palopoli, "Harnessing steering singularities in passive path following for robotic walkers," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, may 2017.

[4] M. Andreetto, S. Divan, F. Ferrari, D. Fontanelli, L. Palopoli, and F. Zenatti, "Simulating passivity for robotic walkers via authority-sharing," *IEEE Robotics and Automation Letters*, vol. 3, pp. 1306–1313, apr 2018.

[5] V. Magnago, M. Andreetto, S. Divan, D. Fontanelli, and L. Palopoli, "Ruling the control authority of a service robot based on information precision," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, may 2018.

[6] V. Magnago, L. Palopoli, R. Passerone, D. Fontanelli, and D. Macii, "A nearly optimal landmark deployment for indoor localisation with limited sensing," in *2017 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, IEEE, sep 2017.

[7] V. Magnago, P. Bevilacqua, L. Palopoli, R. Passerone, D. Fontanelli, and D. Macii, "Optimal landmark placement for indoor positioning using context information and multi-sensor data," in *2018 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, IEEE, may 2018.

[8] V. Magnago, L. Palopoli, R. Passerone, D. Fontanelli, and D. Macii, "Effective landmark placement for robot indoor localization with position uncertainty

constraints," *IEEE Transactions on Instrumentation and Measurement*, pp. 1–13, 2019.

[9] "ACANTO: A CyberphysicAl social NeTwOrk using robot friends." http://www.ict-acanto.eu/acanto. February 2015. EU Project.

[10] "The Roborace robocar." http://roborace.com/.

[11] P. Bevilacqua, M. Frego, E. Bertolazzi, D. Fontanelli, L. Palopoli, and F. Biral, "Path planning maximising human comfort for assistive robots," in *Control Applications (CCA), 2016 IEEE Conference on*, pp. 1421–1427, IEEE, 2016.

[12] P. Bevilacqua, M. Frego, D. Fontanelli, and L. Palopoli, "Reactive planning for assistive robots," *IEEE Robotics and Automation Letters*, vol. 3, pp. 1276–1283, April 2018.

[13] P. Bevilacqua, M. Frego, D. Fontanelli, and L. Palopoli, "Activity planning for assistive robots," *IEEE Transactions on Industrial Informatics*, 2019. Submitted.

[14] M. Frego, P. Bevilacqua, E. Bertolazzi, F. Biral, D. Fontanelli, and L. Palopoli, "Trajectory planning for car-like vehicles: A modular approach," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, pp. 203–209, Dec 2016.

[15] E. Bertolazzi, P. Bevilacqua, F. Biral, D. Fontanelli, M. Frego, and L. Palopoli, "Efficient re-planning for robotic cars," in *2018 European Control Conference (ECC)*, pp. 1068–1073, June 2018.

[16] J. H. Reif, "Complexity of the mover's problem and generalizations," in *20th Annual Symposium on Foundations of Computer Science (sfcs 1979)*, IEEE, oct 1979.

[17] J. T. Schwartz and M. Sharir, "On the "piano movers'" problem i. the case of a two-dimensional rigid polygonal body moving amidst polygonal barriers," *Communications on Pure and Applied Mathematics*, vol. 36, pp. 345–398, may 1983.

[18] J. F. Canny, *Complexity of Robot Motion Planning (ACM Doctoral Dissertation Award)*. The MIT Press, 1988.

[19] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[20] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

[21] 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, no. 4, pp. 566–580, 1996.

[22] R. Geraerts and M. H. Overmars, "A comparative study of probabilistic roadmap planners," in *Springer Tracts in Advanced Robotics*, pp. 43–57, Springer Berlin Heidelberg, 2004.

[23] B. Lau, C. Sprunk, and W. Burgard, "Kinodynamic motion planning for mobile robots using splines," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, oct 2009.

[24] J. wung Choi, R. Curry, and G. Elkaim, "Path planning based on bézier curve for autonomous ground vehicles," in *Advances in Electrical and Electronics Engineering - IAENG Special Edition of the World Congress on Engineering and Computer Science 2008*, IEEE, oct 2008.

[25] S. Quinlan and O. Khatib, "Elastic bands: connecting path planning and control," in *[1993] Proceedings IEEE International Conference on Robotics and Automation*, IEEE Comput. Soc. Press.

[26] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the RRT*," in *International Conference on Robotics and Automation (ICRA)*, pp. 1478–1483, IEEE Press, 2011.

[27] S. Karaman and E. Frazzoli, "Optimal kinodynamic motion planning using incremental sampling-based methods," in *49th IEEE Conference on Decision and Control (CDC)*, IEEE, dec 2010.

[28] D. J. Webb and J. van den Berg, "Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics," in *2013 IEEE International Conference on Robotics and Automation*, IEEE, may 2013.

[29] S. Karaman and E. Frazzoli, "Sampling-based optimal motion planning for non-holonomic dynamical systems," in *2013 IEEE International Conference on Robotics and Automation*, IEEE, may 2013.

[30] K. Yang, S. Moon, S. Yoo, J. Kang, N. L. Doh, H. B. Kim, and S. Joo, "Spline-based rrt path planner for non-holonomic robots," *Journal of Intelligent & Robotic Systems*, vol. 73, no. 1, pp. 763–782, 2014.

[31] M. Likhachev and D. Ferguson, "Planning long dynamically feasible maneuvers for autonomous vehicles," *The International Journal of Robotics Research*, vol. 28, pp. 933–945, jun 2009.

[32] M. Pivtoraiko, R. A. Knepper, and A. Kelly, "Differentially constrained mobile robot motion planning in state lattices," *Journal of Field Robotics*, vol. 26, pp. 308–333, mar 2009.

[33] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, D. Johnston, S. Klumpp, D. Langer, A. Levandowski, J. Levinson, J. Marcil, D. Orenstein, J. Paefgen, I. Penny, A. Petrovskaya, M. Pflueger, G. Stanek, D. Stavens, A. Vogt, and S. Thrun, "Junior: The stanford entry in the urban challenge," in *Springer Tracts in Advanced Robotics*, pp. 91–123, Springer Berlin Heidelberg, 2009.

[34] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Practical search techniques in path planning for autonomous driving," *Ann Arbor*, vol. 1001, no. 48105, pp. 18–80, 2008.

[35] N. E. Du Toit and J. W. Burdick, "Probabilistic collision checking with chance constraints," *IEEE Trans. on Robotics*, no. 4, pp. 809–815, 2011.

[36] F. Large, D. Vasquez, T. Fraichard, and C. Laugier, "Avoiding cars and pedestrians using velocity obstacles and motion prediction," in *Intelligent Vehicles Symposium, 2004 IEEE*, pp. 375–379, IEEE, 2004.

[37] A. Alempijevic, R. Fitch, and N. Kirchner, "Bootstrapping navigation and path planning using human positional traces," in *Robotics and Automation (ICRA), IEEE Int. Conf. on*, pp. 1242–1247, IEEE, 2013.

[38] P. Trautman, J. Ma, R. M. Murray, and A. Krause, "Robot navigation in dense human crowds: Statistical models and experimental studies of human–robot cooperation," *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 335–356, 2015.

[39] Q. Zhu, "Hidden markov model for dynamic obstacle avoidance of mobile robot navigation," *IEEE Transactions on Robotics and Automation*, vol. 7, pp. 390–397, Jun 1991.

[40] C. Fulgenzi, C. Tay, A. Spalanzani, and C. Laugier, "Probabilistic navigation in dynamic environment using rapidly-exploring random trees and gaussian processes," in *Intelligent Robots and Systems. IROS 2008. IEEE/RSJ International Conference on*, pp. 1056–1062, IEEE, 2008.

[41] D. Althoff, J. J. Kuffner, D. Wollherr, and M. Buss, "Safety assessment of robot trajectories for navigation in uncertain and dynamic environments," *Autonomous Robots*, vol. 32, pp. 285–302, Apr 2012.

[42] A. Colombo, D. Fontanelli, D. Gandhi, A. DeAngeli, L. Palopoli, S. Sedwards, and A. Legay, "Behavioural templates improve robot motion planning with social force model in human environments," in *Proc. IEEE Int. Conf. on Emerging Technologies & Factory Automation (ETFA)*, (Cagliari, Italy), pp. 1–6, IEEE, 10-13 Sep. 2013.

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

[44] G. Arechavaleta, J.-P. Laumond, H. Hicheur, and A. Berthoz, "On the nonholonomic nature of human locomotion," *Autonomous Robots*, vol. 25, no. 1-2, pp. 25–35, 2008.

[45] G. Arechavaleta, J.-P. Laumond, H. Hicheur, and A. Berthoz, "An optimality principle governing human walking," *IEEE Transactions on Robotics*, vol. 24, pp. 5–14, Feb 2008.

[46] G. Ferrer and A. Sanfeliu, "Proactive kinodynamic planning using the extended social force model and human motion prediction in urban environments,"

in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1730–1735, IEEE, 2014.

[47] F. Farina, D. Fontanelli, A. Garulli, A. Giannitrapani, and D. Prattichizzo, "Walking ahead: The headed social force model," *PLOS ONE*, vol. 12, pp. 1–23, 01 2017.

[48] M. Saha, T. Roughgarden, J.-C. Latombe, and G. Sánchez-Ante, "Planning tours of robotic arms among partitioned goals," *The International Journal of Robotics Research*, vol. 25, no. 3, pp. 207–223, 2006.

[49] E. Plaku, L. E. Kavraki, and M. Y. Vardi, "Motion planning with dynamics by a synergistic combination of layers of planning," *IEEE Transactions on Robotics*, vol. 26, pp. 469–482, June 2010.

[50] M. Van Den Briel, R. Sanchez, M. B. Do, and S. Kambhampati, "Effective approaches for partial satisfaction (over-subscription) planning," in *AAAI*, pp. 562–569, 2004.

[51] A. Gunawan, H. C. Lau, and P. Vansteenwegen, "Orienteering problem: A survey of recent variants, solution approaches and applications," *European Journal of Operational Research*, vol. 255, no. 2, pp. 315–332, 2016.

[52] M. Schilde, K. F. Doerner, R. F. Hartl, and G. Kiechle, "Metaheuristics for the bi-objective orienteering problem," *Swarm Intelligence*, vol. 3, no. 3, pp. 179–201, 2009.

[53] D. Gavalas, C. Konstantopoulos, K. Mastakas, and G. Pantziou, "A survey on algorithmic approaches for solving tourist trip design problems," *Journal of Heuristics*, vol. 20, no. 3, pp. 291–328, 2014.

[54] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *American Journal of mathematics*, pp. 497–516, 1957.

[55] R. Sanfelice, S. Yong, and E. Frazzoli, "On minimum-time paths of bounded curvature with position-dependent constraints," *Automatica*, vol. 50, no. 2, pp. 537–546, 2014.

[56] L. S. Pontryagin, *Mathematical theory of optimal processes*. Routledge, 2018.

[57] J. Hendrikx, T. Meijlink, and R. Kriens, "Application of optimal control theory to inverse simulation of car handling," *Vehicle System Dynamics*, vol. 26, no. 6, pp. 449–461, 1996.

[58] D. Tavernini, M. Massaro, E. Velenis, D. I. Katzourakis, and R. Lot, "Minimum time cornering: the effect of road surface and car transmission layout," *Vehicle System Dynamics*, vol. 51, no. 10, pp. 1533–1547, 2013.

[59] R. Lot and F. Biral, "A curvilinear abscissa approach for the lap time optimization of racing vehicles," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 7559–7565, 2014.

[60] N. Dal Bianco, R. Lot, and M. Gadola, "Minimum time optimal control simulation of a gp2 race car," *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 232, no. 9, pp. 1180–1195, 2018.

[61] G. Perantoni and D. J. Limebeer, "Optimal control for a formula one car with variable parameters," *Vehicle System Dynamics*, vol. 52, no. 5, pp. 653–678, 2014.

[62] K. Berntorp, B. Olofsson, K. Lundahl, and L. Nielsen, "Models and methodology for optimal trajectory generation in safety-critical road–vehicle manoeuvres," *Vehicle System Dynamics*, vol. 52, no. 10, pp. 1304–1332, 2014.

[63] N. Dal Bianco, E. Bertolazzi, F. Biral, and M. Massaro, "Comparison of direct and indirect methods for minimum lap time optimal control problems," *Vehicle System Dynamics*, vol. 57, no. 5, pp. 665–696, 2019.

[64] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, vol. 2, pp. 995–1001, IEEE, 2000.

[65] C. Urmson et al., "Autonomous driving in urban environments: Boss and the urban challenge," *Journal of Field Robotics*, vol. 25, no. 8, pp. 425–466, 2008.

[66] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Path planning for autonomous vehicles in unknown semi-structured environments," *The International Journal of Robotics Research*, vol. 29, no. 5, pp. 485–501, 2010.

[67] Y. Kuwata, S. Karaman, J. Teo, E. Frazzoli, J. How, and G. Fiore, "Real-time motion planning with applications to autonomous urban driving," *Control Systems Technology, IEEE Transactions on*, vol. 17, pp. 1105–1118, Sept 2009.

[68] J. hwan Jeon, R. V. Cowlagi, S. C. Peters, S. Karaman, E. Frazzoli, P. Tsiotras, and K. Iagnemma, "Optimal motion planning with the half-car dynamical model for autonomous high-speed driving," in *2013 American Control Conference*, pp. 188–193, IEEE, 2013.

[69] A. Arab, K. Yu, J. Yi, and D. Song, "Motion planning for aggressive autonomous vehicle maneuvers," in *2016 IEEE International Conference on Automation Science and Engineering (CASE)*, pp. 221–226, IEEE, 2016.

[70] Y. Gao, T. Lin, F. Borrelli, E. Tseng, and D. Hrovat, "Predictive control of autonomous ground vehicles with obstacle avoidance on slippery roads," in *ASME 2010 dynamic systems and control conference*, pp. 265–272, American Society of Mechanical Engineers, 2010.

[71] A. Gray, Y. Gao, T. Lin, J. K. Hedrick, H. E. Tseng, and F. Borrelli, "Predictive control for agile semi-autonomous ground vehicles using motion primitives," in *2012 American Control Conference (ACC)*, pp. 4239–4244, IEEE, 2012.

[72] E. Frazzoli, M. A. Dahleh, and E. Feron, "Maneuver-based motion planning for nonlinear systems with symmetries," *IEEE transactions on robotics*, vol. 21, no. 6, pp. 1077–1091, 2005.

[73] E. Velenis, P. Tsiotras, and J. Lu, "Modeling aggressive maneuvers on loose surfaces: The cases of trail-braking and pendulum-turn," in *2007 European Control Conference (ECC)*, pp. 1233–1240, IEEE, 2007.

[74] J. V. Frasch, A. Gray, M. Zanon, H. J. Ferreau, S. Sager, F. Borrelli, and M. Diehl, "An auto-generated nonlinear mpc algorithm for real-time obstacle

avoidance of ground vehicles," in *2013 European Control Conference (ECC)*, pp. 4136–4141, IEEE, 2013.

[75] A. Liniger, A. Domahidi, and M. Morari, "Optimization-based autonomous racing of 1: 43 scale rc cars," *Optimal Control Applications and Methods*, vol. 36, no. 5, pp. 628–647, 2015.

[76] E. Bertolazzi and M. Frego, "G^1 fitting with clothoids," *Mathematical Methods in the Applied Sciences*, vol. 38, no. 5, pp. 881–897, 2015.

[77] E. Bertolazzi, M. Frego, and P. Bevilacqua, "Clothoids." https://github.com/ebertolazzi/Clothoids.

[78] E. Bertolazzi and M. Frego, "On the G^2 Hermite interpolation problem with clothoids," *Journal of Computational and Applied Mathematics*, vol. 341, pp. 99–116, 2018.

[79] C. Ericson, *Real-Time Collision Detection*. Boca Raton, FL, USA: CRC Press, Inc., 2004.

[80] P. Guigue and O. Devillers, "Fast and robust triangle-triangle overlap test using orientation predicates," *Journal of Graphics Tools*, vol. 8, no. 1, pp. 25–32, 2003.

[81] Y. Xing, X. P. Liu, and S. Xu, "Efficient collision detection based on aabb trees and sort algorithm," in *IEEE ICCA 2010*, pp. 328–332, June 2010.

[82] 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*, IEEE, oct 2009.

[83] M. Frego, E. Bertolazzi, F. Biral, D. Fontanelli, and L. Palopoli, "Semi-analytical minimum time solutions for a vehicle following clothoid-based trajectory subject to velocity constraints," in *2016 European Control Conference (ECC)*, pp. 2221–2227, IEEE, June 2016.

[84] T. Fraichard and A. Scheuer, "From reeds and shepp's to continuous-curvature paths," *Robotics, IEEE Transactions on*, vol. 20, pp. 1025–1035, Dec 2004.

[85] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed rrt*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2997–3004, IEEE, 2014.

[86] A. Wchter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, pp. 25–57, apr 2005.

[87] E. Bertolazzi and M. Frego, "Interpolating clothoid splines with curvature continuity," *Mathematical Methods in the Applied Sciences*, vol. 41, no. 4, pp. 1723–1737, 2017.

[88] C. R. Dyer, "The space efficiency of quadtrees," *Computer Graphics and Image Processing*, vol. 19, p. 89, may 1982.

[89] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, dec 1959.

[90] L. Palopoli, A. Argyros, J. Birchbauer, A. Colombo, D. Fontanelli, *et al.*, "Navigation assistance and guidance of older adults across complex public spaces: the dali approach," *Intelligent Service Robotics*, vol. 8, no. 2, pp. 77–92, 2015.

[91] P. Panteleris and A. A. Argyros, "Vision-based slam and moving objects tracking for the perceptual support of a smart walker platform," in *ECCV Workshops (3)*, pp. 407–423, 2014.

[92] K. Mombaur, A. Truong, and J.-P. Laumond, "From human to humanoid locomotion – an inverse optimal control approach," *Autonomous robots*, vol. 28, no. 3, pp. 369–383, 2010.

[93] I. Nishitani, T. Matsumura, M. Ozawa, A. Yorozu, and M. Takahashi, "Human-centered x–y–t space path planning for mobile robot in dynamic environments," *Robotics and Autonomous Systems*, vol. 66, no. Supplement C, pp. 18–26, 2015.

[94] C.-P. Lam, C.-T. Chou, K.-H. Chiang, and L.-C. Fu, "Human-centered robot navigation–towards a harmoniously human–robot coexisting environment," *IEEE Trans. on Robotics*, vol. 27, no. 1, pp. 99–112, 2011.

[95] M. Luber, J. A. Stork, G. D. Tipaldi, and K. O. Arras, "People tracking with human motion predictions from social forces," in *Proc. IEEE ICRA 2010*, (Anchorage, USA), 2010.

[96] N. E. Du Toit and J. W. Burdick, "Robot motion planning in dynamic, uncertain environments," *IEEE Transactions on Robotics*, vol. 28, no. 1, pp. 101–115, 2012.

[97] M. Bennewitz, W. Burgard, G. Cielniak, and S. Thrun, "Learning motion patterns of people for compliant robot motion," *The International Journal of Robotics Research*, vol. 24, no. 1, pp. 31–48, 2005.

[98] S. Pellegrini, A. Ess, K. Schindler, and L. van Gool, "You'll never walk alone: Modeling social behavior for multi-target tracking," in *2009 IEEE 12th International Conference on Computer Vision*, pp. 261–268, Sept. 2009. http://www.vision.ee.ethz.ch/datasets/index.en.html.

[99] R. Mead and M. J. Matarić, "Autonomous human–robot proxemics: socially aware navigation based on interaction potential," *Autonomous Robots*, vol. 41, pp. 1189–1201, Jun 2017.

[100] J. Rios-Martinez, A. Spalanzani, and C. Laugier, "From proxemics theory to socially-aware navigation: A survey," *International Journal of Social Robotics*, vol. 7, pp. 137–153, Apr 2015.

[101] D. R. Parisi, P. A. Negri, and L. Bruno, "Experimental characterization of collision avoidance in pedestrian dynamics," *Phys. Rev. E*, vol. 94, p. 022318, Aug 2016.

[102] J. Rios-Martinez, A. Renzaglia, A. Spalanzani, A. Martinelli, and C. Laugier, "Navigating between people: A stochastic optimization approach," in *Robotics and automation (ICRA), 2012 IEEE international conference on*, pp. 2880–2885, IEEE, 2012.

[103] K. Kitazawa and T. Fujiyama, "Pedestrian vision and collision avoidance behavior: Investigation of the information process space of pedestrians using an eye tracker," in *Pedestrian and evacuation dynamics 2008*, pp. 95–108, Springer, 2010.

[104] D. L. Applegate, R. E. Bixby, V. Chvatal, and W. J. Cook, *The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics)*. Princeton, NJ, USA: Princeton University Press, 2007.

[105] J. Luedtke and S. Ahmed, "A sample approximation approach for optimization with probabilistic constraints," *SIAM Journal on Optimization*, vol. 19, no. 2, pp. 674–699, 2008.

[106] M. Zare, T. Niknam, R. Azizipanah-Abarghooee, and A. Ostadi, "New stochastic bi-objective optimal cost and chance of operation management approach for smart microgrid," *IEEE Transactions on Industrial Informatics*, vol. 12, pp. 2031–2040, Dec 2016.

[107] G. C. Calafiore and L. E. Ghaoui, "On distributionally robust chance-constrained linear programs," *Journal of Optimization Theory and Applications*, vol. 130, pp. 1–22, Jul 2006.

[108] B. K. Pagnoncelli, S. Ahmed, and A. Shapiro, "Sample average approximation method for chance constrained programming: Theory and applications," *Journal of Optimization Theory and Applications*, vol. 142, pp. 399–416, Aug 2009.

[109] E. Camponogara, A. B. de Oliveira, and G. Lima, "Optimization-based dynamic reconfiguration of real-time schedulers with support for stochastic processor consumption," *IEEE Transactions on Industrial Informatics*, vol. 6, pp. 594–609, Nov 2010.

[110] E. Bertolazzi and M. Frego, "Semi-analytical minimum time solution for the optimal control of a vehicle subject to limited acceleration," *Journal of Computational and Applied Mathematics (JCAM)*, 2016. Submitted.

[111] M. Frego, E. Bertolazzi, F. Biral, D. Fontanelli, and L. Palopoli, "Semi-analytical minimum time solutions with velocity constraints for trajectory following of vehicles," *Automatica*, vol. 86, pp. 18–28, 2017.

[112] E. Bertolazzi, F. Biral, and M. Da Lio, "Real-time motion planning for multibody systems: Real life application examples," *Multibody System Dynamics*, vol. 17, no. 2-3, pp. 119–139, 2007.

[113] F. Biral, E. Bertolazzi, and P. Bosetti, "Notes on numerical methods for solving optimal control problems," *IEEJ Journal of Industry Applications*, vol. 5, pp. 154–166, 2015.

DOCTORAL PROGRAM IN
INFORMATION AND COMMUNICATION TECHNOLOGY

Doctoral candidate

Paolo Bevilacqua

Cycle	**31**
Thesis	**Efficient Motion Planning for Wheeled Mobile Robotics**
Advisor	**Luigi Palopoli (University of Trento)**
Co-advisor	**Daniele Fontanelli (University of Trento)**

1. List of publications

- Bevilacqua, P., Frego, M., Bertolazzi, E., Fontanelli, D., Palopoli, L., & Biral, F. (2016, September). Path planning maximising human comfort for assistive robots. In Control Applications (CCA), 2016 IEEE Conference on (pp. 1421-1427). IEEE.
- Frego, M., Bevilacqua, P., Bertolazzi, E., Biral, F., Fontanelli, D., & Palopoli, L. (2016, December). Trajectory planning for car-like vehicles: a modular approach. In *Decision and Control (CDC), 2016 IEEE 55th Conference on* (pp. 203-209). IEEE.
- Bevilacqua, P., Frego, M., Fontanelli, D., & Palopoli, L. (2018). Reactive planning for assistive robots. *IEEE Robotics and Automation Letters*, *3*(2), 1276-1283.
- Bertolazzi, E., Bevilacqua, P., Biral, F., Fontanelli, D., Frego, M., & Palopoli, L. (2018, June). Efficient Re-planning for Robotic Cars. In *2018 European Control Conference (ECC)* (pp. 1068-1073). IEEE.
- Magnago, V., Bevilacqua, P., Palopoli, L., Passerone, R., Fontanelli, D., & Macii, D. (2018, May). Optimal landmark placement for indoor positioning using context information and multi-sensor data. In *2018 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)* (pp. 1-6). IEEE.

Under review:

- Bevilacqua, P., Frego, M., Fontanelli, D., & Palopoli, L. (2019). Activity Planning for Assistive Robots. *IEEE Transactions on Industrial Informatics.*

2. Research/study activities

During the Doctoral programme, my research has been focused on the development of efficient motion planning solutions for different kind of applications.

My Ph.D. has been funded by the H2020 European Project ACANTO, that aims to develop an assistive robot providing support and assistance to older users during the execution of social activities in public spaces. The ultimate goal of this project was to promote an “active ageing” of elderly people, with the support of new, advanced ICT technologies. Therefore, most of the developed planning tools have been developed specifically for assistive robotics.

During the last year of Ph.D., I participated also to the European project AWARD, focused on the development of robotic platforms and software components for autonomous warehousing.

From February to June 2018, I spent four months at the Hori-Fujimoto laboratory (University of Tokyo), under the supervision of professor Hiroshi Fujimoto. There, my research was concentrated on the adaptation and application of the efficient motion planning techniques developed during my Ph.D. to autonomous electric cars.

Finally, I have been the teaching assistant for the course “Laboratory of Applied Robotics” for three academic years (2016/17 – 2017/18 – 2018/19).