Efficient and Smooth Motion Planning Techniques for Nonholonomic Wheeled Robots

Luigi Palmieri

Technische Fakultät
Albert-Ludwigs-Universität Freiburg

Dissertation zur Erlangung des akademischen Grades
Doktor der Naturwissenschaften

Betreuer: Dr. Kai O. Arras

UNI FREIBURG

Dissertation zur Erlangung des Doktorgrades
der Technischen Fakultät der
Albert-Ludwigs-Universität Freiburg im Breisgau

Efficient and Smooth Motion Planning Techniques for Nonholonomic Wheeled Robots

Luigi Palmieri

Betreuer:
Dr. Kai O. ARRAS

6. Juli 2018

Dekan:	Prof. Dr. Oliver Paul
Erstgutachter:	Dr. Kai O. Arras Robert Bosch GmbH
Zweitgutachter:	Prof. Dr. Wolfram Burgard Albert-Ludwigs-Universität Freiburg
Tag der Disputation:	6. Juli 2018

Abstract

In the near future robots will operate in environments full of humans and work-collaborate with them. Human environments are complex and crowded: in these environments the fast generation of high quality robot motion is a fundamental aspect for improving the robot's task efficiency and its human-aware behaviors. Driven by these needs, in this work we consider the goal of smooth and natural real-time motion generation for nonholonomic wheeled mobile robots in complex, cluttered and crowded environments.
We address this goal on different levels as follows:

- We show that by using a steer function, which exploits as much knowledge of the nonholonomic constraints of the system as possible, single-query sampling-based motion planners achieve both high planning efficiency and high trajectory quality in cluttered environments.
- We propose a real-time machine learning approach for the computation of the distance pseudo-metric for sampling-based motion planners: by using a constant-time inference model we are able to well approximate the computation of a trajectory-dependent cost function while avoiding to solve an expensive two-point boundary value problem, thus improving the overall performance of sampling-based motion planners.
- We present a hierarchical combination of a discrete any-angle search with a sampling-based motion planner. Its informed sampling unit generates states whose distribution summarizes geometric information of a smooth any-angle path: the final trajectory generated by the sampling-based motion planner retains the smoothness of the any-angle path.
- We introduce an informed sampling strategy that exploits learned motion priors of flows of dynamic obstacles (i.e. pedestrians) in the form of continuous Gaussian mixture fields. An optimal sampling-based motion planner, by using this information, generates samples that allow the planner to very efficiently generate high-quality solutions in terms of path smoothness, path length as well as natural yet minimum control effort motion through multi-modal representations of Gaussian mixture fields of objects' dynamics: the generated motion better adhere to the perceived and learned environment's dynamics.
- We present a randomized approach that generates quickly a set of homotopically distinct and diverse K paths for a wheeled mobile robot. Generating a set of diverse paths, perhaps homotopically distinct, among static and

dynamic obstacles is an appealing planning strategy for wheeled mobile robots in crowded environments: in fact having a set of diverse paths allows to qualitatively and quickly reason about multiple path hypotheses to the goal.

- With the insights learned during the development of the methods beforehand mentioned, we deploy a social-aware multi-hypothesis planning architecture for wheeled mobile robot on the robot Spencer. The approach quickly generates smooth solutions among people while respecting social norms and being efficient in accomplishing robot-guidance tasks: the approach was deployed and intensively tested in a public crowded airport for several dozens of kilometers.

All the contributions, introduced in this thesis, improve planning efficiency and path quality for wheeled mobile robots in complex and crowded environments.

Zusammenfassung

In naher Zukunft werden sich Roboter in menschlichen Umgebungen aufhalten und mit Menschen zusammen arbeiten. Eine Herausforderung stellen insbesondere Umgebungen dar, in denen sich sehr viele Menschen auf engem Raum befinden und bewegen: In diesen Umgebungen ist die schnelle Generierung von Roboterbewegungen ein grundlegender Aspekt für die Verbesserung der Effizienz des Roboters und seiner Verhaltensweisen, die den Menschen explizit berücksichtigen.

Entsprechend diesen Bedürfnissen und in Anbetracht dessen, verfolgen wir in dieser Arbeit das Ziel, weiche und natürlich wirkende Bewegungen für nichtholonome, fahrende Roboter in komplexen und stark bevölkerten Umgebungen in Echtzeit zu generieren.
Wir erreichen dieses Ziel auf verschiedenen Wegen:

- Wir zeigen, dass durch die Verwendung einer Steuerfunktion, die so viel Wissen der nicht-holonome Einschränkungen des Systems wie möglich ausnutzt, single-query sampling-basierte Bewegungsplaner sowohl hohe Planungseffizienz als auch hohe Trajektorienqualität in komplexen Umgebungen erreichen.

- Wir schlagen einen Echtzeit-Lernansatz für die Berechnung der Distanz-Pseudometrik für sampling-basierte Bewegungsplaner vor: Mit einem Konstantzeit-Inferenzmodell sind wir in der Lage, die Berechnung einer Trajektorien-abhängigen Kostenfunktion gut zu approximieren, indem wir vermeiden, ein teures Zweipunkt-Randwertproblem zu lösen. Infolgedessen wird die Gesamtleistung von sampling-basierten Bewegungsplanern verbessert.

- Wir präsentieren eine hierarchische Kombination einer diskreten any-angle-Suche mit einem sampling-basierten Bewegungsplaner. Sein informiertes sampling erzeugt Konfigurationen, deren Wahrscheinlichkeitsverteilung geometrische Informationen eines glatten any-angle-Wegs zusammenfasst: Die endgültige Trajektorie, die durch den sampling-basierten Bewegungsplaner erzeugt wird, behält die Glätte des any-angle-Wegs bei.

- Wir führen eine informierte Sampling-Strategie ein, die gelernte Wahrnehmungspriors einer dynamischen Umgebung in Form von kontinuierlichen *Gaußschen Mixture Fields* ausnutzt. Ein optimaler sampling-basierter Bewegungsplaner erzeugt durch die Nutzung dieser

Informationen Konfigurationen, die dem Planer eine sehr effiziente Berechnung von hochwertigen Lösungen in Bezug auf Wegglätte und Weglänge erlauben. Der Planer erzeugt auch natürliche Bewegungen mit minimalem Steuerungsaufwand durch multimodale Darstellungen von *Gaußschen Mixture Fields* der Objektdynamiken: Die erzeugten Bewegungen berücksichtigen die wahrgenommene und gelernte Umgebungsdynamik besser.

- Wir präsentieren einen randomisierten Ansatz, der schnell eine Menge von homotopisch unterschiedlichen und vielfältigen K-Wegen für einen mobilen Roboter mit Rädern findet. Die Erzeugung einer Reihe unterschiedlicher und möglicherweise homotopisch verschiedener Pfade inmitten von statischen und dynamischen Hindernissen kann eine sinnvolle Planungsstrategie für mobile Roboter in dichten Menschenmengen darstellen: Das Vorliegen einer gewissen Anzahl an unterschiedlichen Pfaden gestattet es, qualitativ und schnell verschiedene Pfadhypothesen zum Ziel zu explorieren.

- Mit den Erkenntnissen, die bei der Entwicklung der im Vorfeld erwähnten Methoden gewonnen wurden, setzen wir eine Multihypothesen-Planungsarchitektur für mobile Roboter auf dem Roboter Spencer um, die soziale Normen berücksichtigt. Der Ansatz wurde zu diesem Zweck in einem stark frequentierten, öffentlichen Flughafenterminal getestet und hat sich über mehrere Dutzend Kilometer hinweg in der Praxis erwiesen.

Alle Beiträge, die in dieser Arbeit vorgestellt wurden, verbessern Planungseffizienz und Pfadqualität für mobile Roboter in komplexen, dicht gedrängten Umgebungen.

Ai miei genitori e fratelli,
ai miei amici vicini e lontani,
a Chiara.

Acknowledgements

This doctoral dissertation is the result of many collaborations and discussions that I carried on during the PhD years with other researchers. Many of those started with un-planned meetings during workshops, conferences and publicly funded projects. Although its complex and hidden stratification, the world-wide robotics scientific community offers the opportunity to study and face new interesting challenges, and to meet people that are open for helping and collaborating.

I would like to thank many of them, my colleagues and friends, that have been ready and open to discussions that contributed to this thesis. Particularly, I would like to thank:

Timm Linder and Billy Okal that worked with me at the Social Robotics Laboratory and in the EU FP7 project Spencer, for all the long days (and long evenings) spent together to push forward our research, projects and demos. Their support was extremely beneficial especially during the integration weeks of the Spencer project.

Andrey Rudenko, for his outstanding collaboration and his patience to run the very last minute experiments.

Tomasz Kucner, Martin Magnusson, Rudolph Triebel for the fruitful collaborations (also during the integration sessions at Schiphol airport in Amsterdam).

Armin Hornung, Felix Burget, Daniel Maier, Markus Spies (Kuderer), Christoph Sprunk, Stefano de Lucia, Matthias Luber and Diego Tipaldi for their interesting discussions in our laboratories in Freiburg.

Prof. Sven Koenig, an impeccable researcher and professor, for being always ready to thoroughly discuss new ideas.

Prof. Wolfram Burgard for being a reviewer for this thesis, Prof. Moritz Diehl and Prof. Bernhard Nebel for being members of the dissertation committee.

My PhD advisor Kai Arras, for his help during all the PhD years, for the many interesting ideas our discussions brought, for indicating me ways to improve my research work, for his critics and for sharing his competences.

I profoundly thank my parents, brothers and friends that cheered me up and deeply believed in me. My sincere and mere thanks to Chiara for her understanding and support during these years.

Lastly, I thankfully acknowledge that this thesis was partially supported by the European Commission under grant agreement number FP7-ICT-600877 (SPENCER).

Contents

CHAPTER 1

Introduction

"Considerate la vostra semenza:
fatti non foste a viver come bruti,
ma per seguir virtute e canoscenza"

Dante Alighieri,
Divina Commedia, Inferno,
Canto XXVI vv. 18-20

Robotics is "a broad, interdisciplinary subject that involves various facets of sensing and manipulation, as well as thinking that integrates sensing with action", writes Michael Brady in the preface of its book "Robotics Science" (Brady, 1989). Although it is not yet clear if it is a science on its own, several researchers state that robotics is a new scientific field: Bajcsy (Bajcsy, 2007) writes that robotics is a science because "it has to address, develop theoretical foundation of interactive complex physical and dynamic systems". Moreover on February 5th 2008, the Handbook of Robotics (Siciliano and Khatib, 2007) won the Association of American Publishers PROSE Award for Excellence in Physical Sciences & Mathematics, a first international recognition of robotics as a new scientific field. The robotics research community is growing year after year: in the last decades several new scientific journals and conferences having as main topic robotics have appeared. Thanks to the effort of this community, in the last decades we have seen an exponential increase of robots deployed in a huge variety of scenarios, to cope with and solve problems that the human beings alone could not (e.g. industrial manipulators, space robotics, rescue operations, underwater robotics etc., see Fig. 1.1).

A typical robotic architecture is made of several different sub-systems that are interconnected and work together to allow the robot platform to accomplish a desired task by sensing and acting in its environment. Several robotic architectures have been introduced in the literature, e.g. the sub-sumption architecture of Brooks (Brooks, 1986) and Sense-Plan-Act of the robot Shakey (Nilsson, 1969). The Sense-Plan-Act paradigm, is still a valid abstraction for several current robotic architectures. It considers the robotic system divided in three main sub-systems: perception, decision making, motion control and execution (see Fig. 1.2).

Figure 1.1: Example robot applications. **Left:** Curiosity is a Mars rover created by the NASA who drove autonomously on Mars 14.14 km as of 8 September 2016. **Center:** The Rolling Justin robot an advanced mobile dual-arm system developed by DLR. **Right:** ICub robot designed by the Italian Institute of Technology, a robot that has not only manipulation capabilities but it is able to express emotions by using its facial expressions.

The perception components work together with the sensors to provide all the needed information to describe the current robot pose and the environment surrounding the robot; some notable applications belonging to this area are simultaneous localization and mapping, object detection and tracking. For example, a robot manipulator that wants to pick up an object in human environments, needs to know or to infer the object position and to track humans movements and avoid them by inferring their future motion.

The decision making component, composed typically of a task and a motion planning system, is the unit that reasons about the assigned task and the information provided by the perception components, to generate actions, paths or trajectories that the motion control system will execute.

The motion controller transforms the desired actions or plans in the actual commands that the robot actuators can execute. The controller makes sure that the desired plan is accurately executed by not damaging the robotic mechanical system.

Path and motion planning is a fundamental line of research in robotics but also in many other scientific communities (e.g. AI, computer graphics, automatic control systems, computational biology, gaming). The robot motion planning problem can be stated as follow. Given a description of the obstacles in the environment, a start and a goal pose and a notion of the robot properties (geometry, kinematic and dynamic constraints of the robot platform) a motion planner generates a set of actions (controls) that will move the robot from the start to the goal while

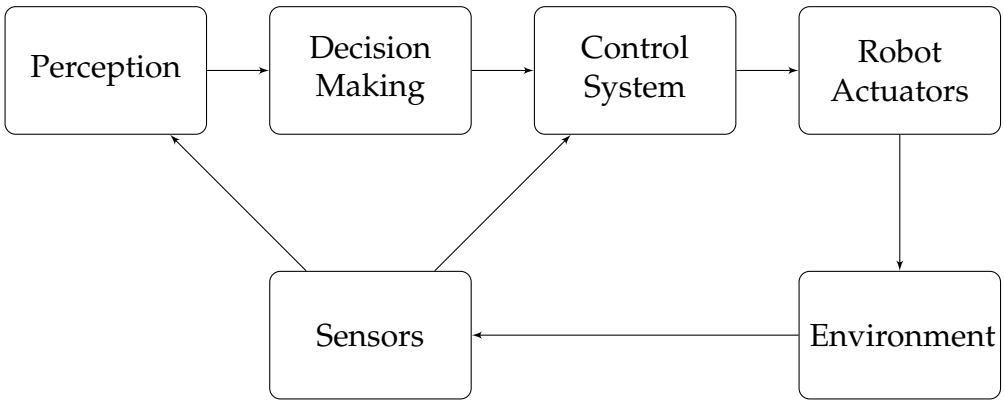

Figure 1.2: A simple Sense-Plan-Act architecture that exemplifies how typically the robot units are connected and work together.

avoiding the obstacles.

If the problem is solved considering only geometric constraints, then we refer to it as path planning or *piano movers' problem* (Schwartz and Sharir, 1983). In general with the term motion planning we refer to a more complex problem that aims to find the controls needed to move the robot from the start to the goal pose by considering the kinematic and dynamic constraints of the platform, this problem is called also *kinodynamic motion planning*, term introduced by Donald et al. (1993). *Nonholonomic motion planning*, firstly presented by Laumond (1987), addresses the problem of finding a trajectory and associated controls to steer a wheeled mobile robot from an initial position to the desired goal by respecting the nonholonomic constraints of the platform: differently by manipulators, many wheeled mobile robots have nonholonomic constraints, e.g. the dimension m of the control space is smaller than the dimension n of the state space, therefore having part of the state space that the robot control system cannot access directly (e.g a differential drive robot cannot move sideways). Several approaches, partially detailed in Chapter 2, have been introduced to solve these problems, each with properties and limits.

1.1 Problem statement

In the not too far future many robotic systems will live-work among humans and collaborate with them: e.g. self-driving cars, service robots, robot guides in public spaces or companies, consumer robots, assistive robot manipulators for goods production and many others. The fast generation of high quality motion, the overall goal of this thesis, is a key aspect particularly in these environments where the planner deals with several new and interesting challenges (see Fig. 1.3). Often an environment with humans is said to be a *dynamic environment*: in each

Figure 1.3: Example nonholonomic motion planning problems. **Left:** An environment with static obstacles (in **grey**), where a differential drive robot needs to find a path that goes to the goal region (in **green**). **Right:** A differential drive robot (Spencer) needs to find efficiently a smooth path in a very crowded environment (i.e. a crowded airport).

time step obstacles and humans in the scenario change their configurations. For robots navigating in dynamic and complex environments, a motion planner is required to be *efficient* and to generate *smooth* robot movements (Kruse et al., 2013). Planning *efficiency* allows the robot to be fast enough to avoid a collision and to quickly accomplish its task. Motion *smoothness* helps to generate natural behaviors for a robot navigating among humans; Kruse et al. (2013) write: "one aspect of natural motion is smoothness. This refers to both the geometry of the path taken as well as the velocity profile". In fact, humans that work with robots expect the latter to generate motion that are smooth and understandable (or *legible*, Dragan et al. (2013); Kruse et al. (2013)). If robot motion are smooth enough a human operator could easily infer the robot intentions and moreover he could also socially accept the robot.
Kruse et al. (2013) also describe the contradiction between *efficiency* and *smoothness* in the context of human-aware navigation: "In HRI, the assumption is that the shortest or most energy efficient path is not necessarily the most desirable. Instead in HRI the intention is to find a path that is also sufficiently safe, comfortable, natural, legible, etc. to persons in the area". We hypothesize that in crowded dynamic environments for a wheeled mobile robot, the planner is expected to behave naturally by *balancing efficiency with social-awareness* and *by reasoning about several possible hypotheses* to reach the goal. Knowing multiple ways to the goal is a much richer information than a single path to the goal, the robot could quickly reason about which route to take to better accomplish its task. As matter of fact, a too social-aware robot motion planner may cause *erratic* and not *legible* robotic behaviors: e.g the robot could be trapped by a crowd and its efficiency may dramatically drop.

Moreover, in human dynamic environments, motion planners' results are highly related to the information provided by perception: often those inputs, although not being very accurate, include a lot of information that could be exploited by the planner *to generate motion that better adhere to what the perception components observed and learned in the environment*. A planner could read and exploit not only the estimated obstacle positions but also covariances or general patterns of obstacles and humans motion in the environment. We hypothesize that in crowded, cluttered and complex environments, perception priors that encode usual crowd or pedestrian flows behaviors are very beneficial to help a motion planner to quickly compute a smooth path.
Following these insights and hypotheses, this thesis proposes several motion planning techniques to answer the following questions:

- How can we improve the efficiency of state-of-the-art motion planners for nonholonomic wheeled mobile robots in difficult and complex environments, such as of large size and full of obstacles?
- How can we exploit place-dependent learned motion priors of flows of dynamic obstacles (i.e. pedestrians), specifically multi-modal Gaussian mixtures models, into kino-dynamic sampling-based motion planners?
- How can we quickly produce several and accurate path hypotheses in dynamic environments with several humans?
- How can we balance smoothness and efficiency during the generation of motion in very crowded environments?

1.2 Scientific Contributions

To answer the aforementioned questions, this thesis introduces several contributions to the field of path and motion planning by leveraging different methods from machine learning, artificial intelligence and control theory. With the goal of smooth and natural real-time motion generations for robots in human environments, we propose several methods that increase planning efficiency and path smoothness of different motion planners. More specifically this work presents (see also Fig. 1.4):

- In Chapter 3, a novel stabilizer for wheeled mobile robots which in combination with sampling based motion planners (used herein as extend function) was shown to produce smoother paths in less time with smaller trees than a set of state-of-the-art baseline methods. Moreover combined with an optimal-sampling based motion planner, the approach produces the

shortest paths and achieves the lowest cost solutions when given more planning time.

- In Chapter 4, an on-line machine learning approach to compute the distance pseudo-metric for nonholonomic sampling based-motion planners. The approach is shown to be faster in planning time by several factors respect to several baselines at negligible loss of path quality.
- In Chapter 5, a hierarchical combination of (discrete) any-angle search with (continuous) sampling-based motion planning for nonholonomic wheeled mobile robots, specifically we consider the case of a differential drive robot and a high-dimensional truck-and-trailer system. The approach, that is probabilistically complete, finds smooth and shorter trajectories significantly faster than four baseline planners without loss of smoothness.
- In Chapter 6, a motion planning approach for wheeled mobile robots under kinodynamic constraints that exploits learned perception priors of dynamic environments in the form of continuous Gaussian multi-modal mixture fields. The approach, which is asymptotically optimal, generates very efficiently high-quality solutions in terms of path smoothness, path length as well as natural yet minimum control effort motion through multi-modal representations of Gaussian mixture fields.
- In Chapter 7, an efficient randomized approach based on weighted random walks, that finds K diverse path hypotheses on the Voronoi diagram of the environment, where each path represents a distinct homotopy class. The approach is significantly faster at finding paths of higher diversity in distinct homotopy classes than two state-of-the-art methods.
- In Chapter 8, a social-aware and efficient motion planner for nonholonomic wheeled mobile robots in very crowded environment. The planning system was developed for the EU-funded project SPENCER and extensively tested at the very crowded Schiphol Amsterdam Airport on 47 km of fully autonomous operation. The planner by means of a multi-hypothesis reasoning balances efficiency with social-awareness and generates locally legible motion that safely and smartly reacts to the dynamics of the environment.

1.3 Publications

Algorithms and ideas of this thesis have been published in peer-reviewed international journals, conference and workshop proceedings. Here a list that presents them in chronological order:

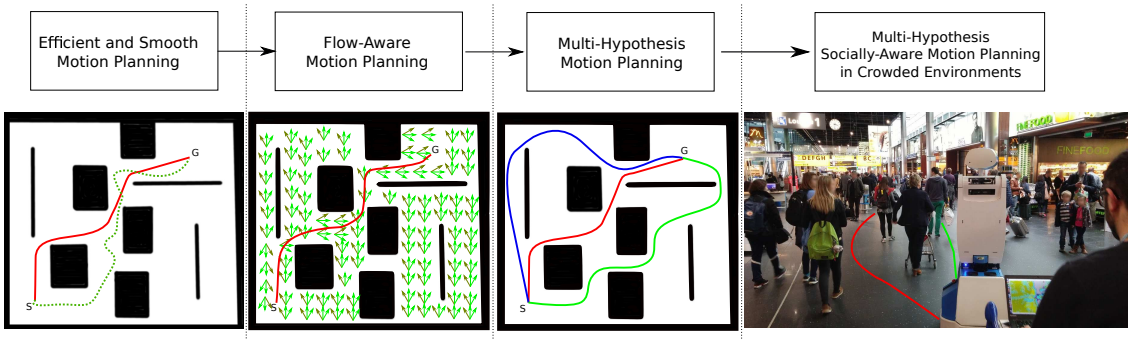

Figure 1.4: Overview of the thesis contributions. **Left:** in Chapters 3, 4 and 5, we describe several approaches to efficiently generate smooth movements for wheeled mobile robots (in **red** an example of smooth path connecting goal and start points, respectively **G** and **S**, the **green-dashed** curve represents a non-smooth path). **Middle-Left:** in Chapter 6, we introduce a method to generate smooth paths considering learned motion priors of pedestrian flows in the form of continuous Gaussian multi-modal mixture fields (**arrows** show an example of these fields). **Middle-Right:** in Chapter 7, we propose a method to quickly compute several path hypotheses to the goal lying in different homotopy classes (paths with different colors are connecting the start position to the goal one). **Right:** considering the insights learned from the developed techniques presented in Chapters 3-7, in Chapter 8 we illustrate a multi-hypothesis socially-aware and efficient motion planner for nonholonomic wheeled mobile robots navigating at high speed in densely crowded environments. In the figure the robot Spencer, successfully deployed at Schiphol Amsterdam Airport, can choose from two possible path hypotheses (**red** and **green** curves).

1.3.1 Peer-Reviewed Journal Articles

- A Fast Random Walk Approach To Find Diverse Paths for Robot Navigation by Palmieri L., Rudenko A. and Arras, K. O.; 2017 *IEEE Robotics and Automation Letters*;

1.3.2 Peer-Reviewed Conference Proceedings

- Kinodynamic Motion Planning on Gaussian Mixture Fields by Palmieri L., Kucner T.P., Magnusson M., Lilienthal A.J. and Arras K.O., 2017, *International Conference on Robotics and Automation*, Singapore;

- RRT-Based Nonholonomic Motion Planning Using Any-Angle Path Biasing by Palmieri L., Koenig S. and Arras K. O., 2016, *International Conference on*

Robotics and Automation, Stockholm, Sweden;

- SPENCER: A Socially Aware Service Robot for Passenger Guidance and Help in Busy Airports by Triebel R., Arras K. O., Alami R., Beyer L., Breuers S., Chatila R., Chetouani M., Cremers D., Evers V., Fiore M., Hung H., Islas Ramirez O., Joosse M., Khambhaita H., Kucner T., Leibe B., Lilienthal A., Linder T., Lohse M., Magnusson M., Okal B., Palmieri L., Rafi U., van Rooij M., Zhang L., 2015, *Field and Service Robotics* (FSR), Toronto, Canada;

- Distance Metric Learning for RRT-Based Motion Planning with Constant-Time Inference by Palmieri L., Arras K. O., 2015, *International Conference on Robotics and Automation*, Seattle, USA

- A Novel RRT Extend Function for Efficient and Smooth Mobile Robot Motion Planning by Palmieri L., Arras K. O., 2014, *International Conference on Intelligent Robots and Systems*, Chicago, USA;

1.3.3 Peer-Reviewed Workshop Proceedings

- RRT-Based Nonholonomic Motion Planning Using Any-Angle Path Biasing by Palmieri L., Koenig S. and Arras, K. O.; 2016 *Symposium on Combinatorial Search*, Tarrytown, NY, USA;

- A Fast Randomized Method to Find Homotopy Classes for Socially-Aware Navigation by Palmieri, L.; Rudenko, A.; Arras, K.O.; 2015 *Proc. of Workshop on Assistance and Service Robotics in a Human Environment, International Conference on Intelligent Robots and Systems*, Hamburg, Germany;

- Distance Metric Learning for RRT-Based Motion Planning for Wheeled Mobile Robots by Palmieri L. and Arras, K.O.; 2014, *Proc. of Machine Learning in Planning and Control of Robot Motion Workshop, International Conference on Intelligent Robots and Systems*, Chicago, USA;

- Efficient and Smooth RRT Motion Planning Using a Novel Extend Function for Wheeled Mobile Robots by Palmieri L., Arras K. O., 2014, *Proc. of the 2nd Workshop on Planning and Robotics, International Conference on Automated Planning and Scheduling*, Portsmouth, USA.

1.4 Collaborations

Partly this work is the result of fruitful collaborations and discussions with other researchers. Several discussions and exchanges with Christoph Sprunk on motion planning techniques and with Ciro Natale and Alessandro Astolfi on control

theory topics contributed to the development of the method proposed in Chapter 3. Thanks to useful suggestions on machine learning techniques with Frank Hutter and Rudolph Triebel, we could achieve the interesting results related to the learned distance pseudo-metric described in Chapter 4. Chapter 5 on the combination of any-angle search with sampling-based motion planners was realized with the precious help of Sven Koenig. Chapter 6 that introduces a sampling-based motion planner on off-line learned continuous Gaussian multi-modal mixture fields was a joint work with Tomasz Kucner, Martin Magnusson and Achim J. Lilienthal. The work on the homotopy classes generation detailed in Chapter7 has been carried on together with Andrey Rudenko and received valuable suggestions by Markus Kuderer. The social-aware motion planner for the SPENCER robot platform detailed in Chapter 8, was developed and improved thanks to the many helpful suggestions of the colleagues of the SPENCER project consortium.

The thesis has been supervised by Kai O. Arras, who contributed to many ideas hereinafter described.

1.5 Outline and Notation

The thesis is divided into nine chapters. This initial chapter (Chapter 1) gives a brief introduction on the motion planning problems that the thesis deals with. Chapter 2 details fundamental motion planning knowledge on which this thesis is built on. Chapter 3 describes a novel steer function for sampling-based motion planners, instead Chapter 4 details the machine learning approach to quickly infer the distance pseudo-metric in sampling-based motion planners. In Chapter 5 we talk about a new algorithm that combines any-angle search with sampling-based motion planners. In Chapter 6 we describe a new sampling-based motion planning algorithm that plans by exploiting perception-priors of multi-modal dynamic flows. In Chapter 7 we introduce a randomized planner for the fast generation of a set of K diverse paths lying into different homotopy classes. Chapter 8 describes a social-aware multi-hypothesis planning architecture for wheeled mobile robot that quickly generates smooth solutions among people while respecting social norms and being efficient in accomplishing robot-guidance tasks. Chapter 9 concludes the thesis with a summary and several recommendations for future work. Appendix A details several notions on nonholonomic systems used in the thesis. Through this work we adopt the notation detailed in Tab.1.1.

Table 1.1: Notation

$\mathcal{X}, \mathcal{Y}, \ldots$	$\triangleq$	Sets
$\lvert\mathcal{X}\rvert$	$\triangleq$	Cardinality of the set $\mathcal{X}$
$\alpha, \beta, a, b, \ldots$	$\triangleq$	Scalars
$\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{a}, \mathbf{b}, \ldots$	$\triangleq$	Vectors
$\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}, \ldots$	$\triangleq$	Matrices
$\boldsymbol{A}_{ij}$	$\triangleq$	Entry of matrix A at row i and column j
$\lVert\mathbf{x}\rVert$	$\triangleq$	l^2 norm of vector $\mathbf{x}$
$\lvert\alpha\rvert$	$\triangleq$	Absolute value of α
$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	$\triangleq$	Normal distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$
$p(X)$	$\triangleq$	Probability of the random variable X
$p(X\lvert Y)$	$\triangleq$	Conditional probability of the random variable X given Y
		Motion planning notation
p	$\triangleq$	Motion planning problem
$\mathbf{x}$	$\triangleq$	Robot configuration or state
$\mathcal{C}$	$\triangleq$	Set of robot states or configurations
$\mathcal{C}_{free}$	$\triangleq$	Set of free-collision robot states or configurations
$\mathcal{C}_{obs}$	$\triangleq$	Set of robot states or configurations in collisions with obstacles
$\mathcal{C}_{goal}$	$\triangleq$	Set of goal states or configuration
$\mathcal{U}$	$\triangleq$	Set of available controls
$\mathbf{u}$	$\triangleq$	Controls
$\mathbf{x}_{start}$	$\triangleq$	Initial robot state or configuration
$\mathbf{x}_{goal}$	$\triangleq$	Goal robot state or configuration
τ	$\triangleq$	Tree structure of states or configurations
$\mathbf{x}_{start}$	$\triangleq$	Initial robot state or configuration
$\boldsymbol{\sigma}$	$\triangleq$	Trajectory of states or configurations
S	$\triangleq$	Set of grid vertices
s_i	$\triangleq$	Node i of the grid set S
$\mathcal{ALG}$	$\triangleq$	An algorithm

CHAPTER 2

Foundations

"Books are not made to be believed, but to be subjected to inquiry. When we consider a book, we mustn't ask ourselves what it says but what it means."

Umberto Eco,
The Name of the Rose

This chapter presents the foundations for the techniques introduced in this thesis. We summarize algorithms for path planning on grids, for the computation of trajectories with sampling-based motion planners that satisfy kinematic and dynamic constraints as well as the most common optimization methods for trajectory generation.

2.1 Introduction

Motion planning is a fundamental research topic in robotics. Several different approaches have been presented to solve the path planning and/or motion planning problem (Latombe, 1991; Choset, 2005; LaValle, 2006). As mentioned in the previous chapter, the term *path planning* refers to the problem of finding a path that brings the robot from an initial position to the goal by knowing only the geometric properties of the robot and of the environment. With the term *motion planning* or *trajectory generation*, we refer to the problem of finding, in a continuous space, a path and the controls to steer a robot system from an initial state to the goal, constrained not only on geometric information but also on the kinodynamic constraints of the system.
Hereinafter we collect and describe recent path and motion planning techniques that are used to generate *smooth* solutions.
The chapter is structured as follows: the descriptions of the configuration space in Section 2.2 and of the any-angle path finding algorithms in Section 2.3 are followed by a brief discussion of discrete search algorithms with motion primitives

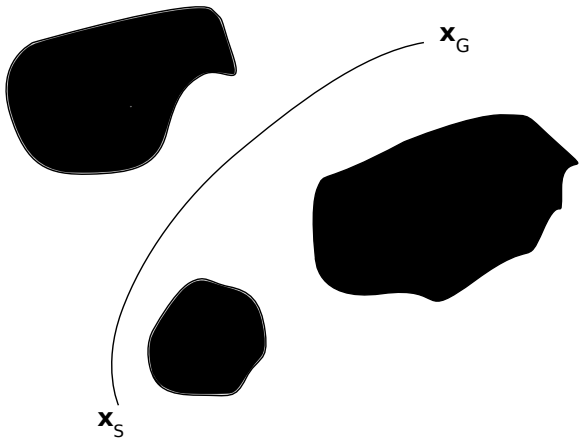

Figure 2.1: Configuration space example. The **dark** regions represent the obstacle space $\mathcal{C}_{obst}$, in **white** the free space $\mathcal{C}_{free}$. A path (**dark** line) connects in $\mathcal{C}_{free}$ the start and goal configurations $\mathbf{x}_S$ and $\mathbf{x}_G$

in Section 2.4. Feasible and optimal sampling-based motion planners are introduced in Section 2.5. In Section 2.6 we detail some of the most known optimization techniques for trajectory generation used in robotics. Section 2.7 presents several metrics used to quantify smoothness in the remaining chapters of the thesis.

2.2 The Configuration Space

Typically path and motion planning techniques consider the robot to be part of the configuration space (Lozano-Pérez, 1983): a configuration is a complete specification of the robot state relative to a fixed coordinate frame, usually a configuration is a vector $\mathbf{x}$ of positions and orientations. The configuration space, $\mathcal{C}$-space, is the space of all the possible configurations that a robot can have: it is separated in two open subset $\mathcal{C}_{free}$ and $\mathcal{C}_{obst}$, respectively, the free configuration space (where the robot configuration is not in collision) and the obstacle configuration space (where the robot configuration is in collision with obstacles). Usually the topology of the configuration space is not that of the Cartesian space: e.g for a wheeled mobile robot that is described as a 2D rigid body with configuration $\mathbf{x} = (x, y, \theta)$ where $x, y \in \mathbb{R}$ and $\theta \in [0, 2\pi)$, the configuration space is the manifold $\mathcal{C} = \mathbb{R}^2 \times \mathbb{S}^1$ (with $\mathbb{S}^1$ the 1-dimensional sphere in $\mathbb{R}^2$); for a robot manipulator with 6 revolute joints the configuration

space is a 6-dimensional torus $\mathbb{C} = \mathbb{T}^6$. In the configuration space, the motion planning problem can be defined as the task of finding a path from a start configuration $\mathbf{x}_S$ to a goal one $\mathbf{x}_G$ that lies entirely in $\mathcal{C}_{free}$ (see Fig. 2.1). Usually problems where also the kinematics and dynamics of the system are included, may be solved considering not only the configuration space but also the phase space: in here any problem with dynamics may be expressed with velocity constraints, projecting them in an additional dimension, on an augmented state space (LaValle, 2006).

2.3 Any-Angle Path Planning on Occupancy Grids

Different combinatorial techniques and representations have been introduced in the robotics and artificial intelligence communities to efficiently solve the problem of finding short paths with a few heading-changes in 2D environments. Common representations of a 2D workspace are **visibility graphs** and **occupancy grids**: in both techniques discrete search techniques are applied on learned (generated) graphs.

Visibility graphs (Lozano-Pérez and Wesley, 1979) are constructed considering obstacles as known polygonal shapes. The graphs contain the start and goal positions and all the obstacle corners as vertices. Each vertex is connected to an other vertex (forming an edge) only if it has line-of-sight to the other vertex. Standard search algorithm (Dijkstra (Dijkstra, 1959) or A* (Hart et al., 1968)) are then applied on the graph to generate a path. Although path planning on visibility graphs generates the true shortest path, the generation of a visibility graph is slow and scales superlinearly: in (de Berg et al., 2008) an algorithm with complexity $\mathcal{O}(n^2 \log n)$ is presented for a set of obstacles with n vertices.

Occupancy grids are a common way to represent the Cartesian space (workspace) for path planning in robotics and artificial intelligence applications. The environment is represented by a tessellation that presents cells blocked by obstacles (or by a not-traversable terrain area) and by unblocked cells (or tiles). A graph is implicitly encoded in the different grid representations (e.g. 4-way tiles, 8-way tiles, hexes). Path planning on grids is faster than planning on visibility graphs (less time is required to build the graph and its number of edges grows only linearly in the number of cells) but it generates solutions with unexpected heading changes (Yap, 2002). To overcome the latter limitation, recently a new class of algorithms called *any-angle* path planning (Nash et al., 2007) has been introduced: they do not constrain the paths to grid edges, thus generating solutions with less heading changes. Moreover they are more efficient than path planning algorithms on visibility graphs (as shown in Daniel et al. (2010)). Theta* (described in Sec.2.3.1) is the first any-angle algorithm for grids introduced in the

community: other solutions exist in literature an interested reader could refer to the following works: Nash et al. (2010), Nash et al. (2009), Daniel et al. (2010) Harabor and Grastien (2013).

For path planning on grids, a path planner can be:

Definition 1. ***Resolution complete:***
A path planner is said to be ***resolution complete****, if given the current grid resolution it is guaranteed to find a path from the start node to the goal node.*

Definition 2. ***Resolution correct:***
A path planner is ***resolution correct****, if given the current grid resolution it is guaranteed to find only unblocked paths from the start node to the goal node.*

Definition 3. ***Resolution optimal:***
A path planner is ***resolution optimal****, if given the current grid resolution it is guaranteed to find the optimal path from the start node to the goal node.*

Next we describe Theta*: as we will show in Chapter 5 *any-angle* paths generated by Theta* are beneficial to inform the search in high dimensional spaces and to quickly generate *smooth* trajectories for such spaces.

2.3.1 Theta*

Theta* is a resolution correct and complete any-angle path planning algorithm, its paths are only slightly longer than true shortest paths. Although Theta* is not guaranteed to find the true shortest paths, that is it is not optimal, Daniel et al. (2010) show that the Basic variant of Theta* generally finds shorter and more realistic looking paths than Field D* (Ferguson and Stentz, 2007), A* with Post smoothing (Botea et al., 2004) and A* on grids. Theta* offers three important properties:

- *efficiency*, it has a good trade-off between computation times and path lengths;
- *simplicity*, it is relatively simple to implement;
- *generality*, it works on every graph embedded in 2D or 3D Euclidean Space.

The basic version of Theta*, reported in Alg. 1-2, maintains two values for every node s of the grid S: the cost-to-come $g(s)$ and the parent $parent(s)$. Theta*, as in A*, makes use of an admissible heuristic h to focus its search and it maintains two global data structure: the *open* list, that contains the list of the vertices to be expanded and the *closed* list which contains vertices already expanded, thus ensuring that each vertex is expanded only once. The key difference between Theta* and A* on grids is that the parent of a vertex can be any vertex that has line

Algorithm 1 *Basic Theta**

```
1: function Basic Theta*(s_start , s_goal)
2: g(s_start) := 0
3: parent(s_start) := s_start
4: open := ∅
5: open.Insert(s_start, g(s_start) + h(s_start))
6: closed := ∅
7: while open ≠ ∅ do
8:     s := open.Pop()
9:     if s = s_goal then
10:        return P
11:    end if
12:    closed := closed ∪ {s}
13:    for all s' ∈ succ(s) do
14:        if s' ∉ closed then
15:            if s' ∉ open then
16:                g(s') := ∞
17:                parent(s') := NULL
18:            end if
19:            UpdateVertex(s, s')
20:        end if
21:    end for
22: end while
23: return failure
```

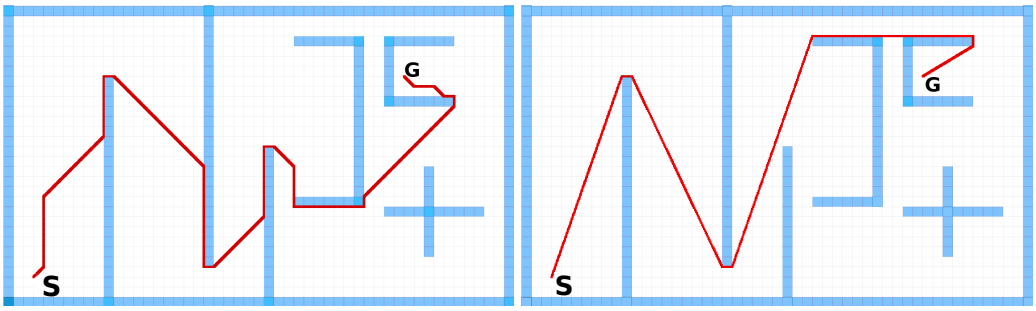

Figure 2.2: Comparison of Theta* and A*. **Left**: The grid path of A* (in **red**) is constrained to grid edges and part of a different homotopy class. It is longer and has more heading changes. **Right:** The any-angle path of Theta* (in **red**) is not constrained to grid edges.

of sight with the current one (see **UpdateVertex**(s, s') function in Alg.2), instead in the case of A* the parent of a vertex has to be a neighbor of the vertex, see Fig. 2.2 for a comparison between A* and Theta*. The result of Theta* is a discrete set **P** of M cells s, $\mathbf{P} = \{s_0, s_1, \ldots, s_{M-1}\}$, that goes from the start cell $s_{start} = s_0$ to the

Algorithm 2 *UpdateVertex*$(s, s\prime)$

```
1: function UpdateVertex(s, s')
2: if lineofsight(parent(s), s') then
3:    if g(parent(s)) + c(parent(s), s') < g(s') then
4:       g(s') := g(parent(s)) + c(parent(s), s')
5:       parent(s') := parent(s)
6:       if s' ∈ open then
7:          open.Remove(s')
8:       end if
9:       open.Insert(s', g(s') + h(s'))
10:   end if
11: else
12:   if g(s) + c(s, s') < g(s') then
13:      g(s') := g(s) + c(s, s')
14:      parent(s') := s
15:      if s' ∈ open then
16:         open.Remove(s')
17:      end if
18:      open.Insert(s', g(s') + h(s')
19:   end if
20: end if
```

goal one $s_{goal} = s_{M-1}$, respectively the cells where the start and the goal poses of the robot are mapped to.

2.4 Discrete Search Approaches with Motion Primitives

Combinatorial planning methods that build a roadmap by exhaustively search over the entire configuration space, work well in general for simple problems as described before, but they scale very poorly with the dimensionality of the problem: for example for a system of dimension d and considering a basic cubic lattices with the same discretization (number of lattice points) k per side, the total number of samples is k^d. Moreover grid based approaches, like A* and Theta* generate only a geometrically feasible or optimal path in the workspace (i.e. a concatenation of Euclidean straight paths).

Discrete search has been used also for differentially constrained systems (e.g. cars and differential drive robots (Likhachev and Ferguson, 2009; Pivtoraiko and Kelly, 2005)) and high dimensional systems (e.g manipulators (Cohen et al., 2014, 2010)). As for grid search in a 2D environment for a point robot, planning also in this case is applied on a graph that describes the configuration space. The

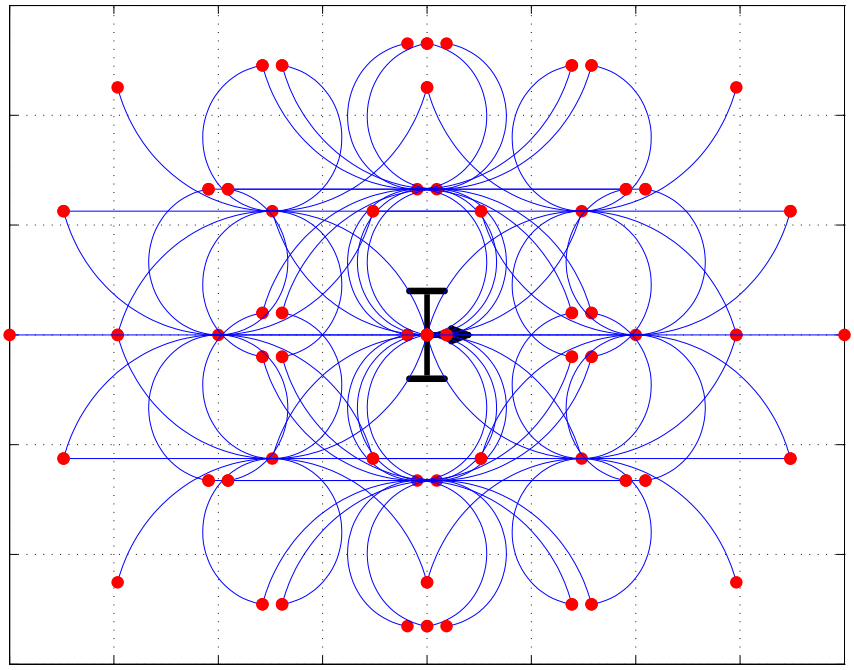

Figure 2.3: Example of a state lattice composed by a set of motion primitives (in **blue**) departing from an initial robot pose (in **black**). In **red** the states belonging to the graph where the planning is performed.

configuration space is discretized into a state lattice, which represents a regular sampling of the state space. In the graph each pair of states is connected if and only if a kinematically feasible action (e.g. motion primitive) can move the system (i.e. robot) from the first configuration to the second one. The graph vertices represent a set of all reachable states of the system computed given a defined resolution.

The state lattices can be computed in two different ways (Pivtoraiko et al., 2009; Likhachev and Ferguson, 2009):

- *forward*: where a set of motion primitives is precomputed off-line and then forward integrated for the state lattice generation, the set of primitives defines then the sampling of the configuration space;

- *inverse*: in this case a desired configuration sampling is defined to assure some geometric properties (e.g dispersion and discrepancy (LaValle et al., 2004)) and then boundary value problem solvers can be used to generate feasible motion from one state to an other.

As in the case of classical grid search also for these planners the properties of *resolution completeness, correctness and optimality* apply. Once the state lattice has

been built, see example in Fig. 2.3 standard search algorithms can be applied to solve the planning problem (e.g A*, Dijkstra).

These methods although being very efficient, they usually generate not smooth solutions and need a further smoothing step (e.g. as in Dolgov et al. (2010)).

2.5 Sampling-based Motion Planning

Path and trajectory planning for high dimensional configuration spaces and for systems with particular kinematic and dynamic constraints is in general a more difficult problem to solve compared to pure geometric path planning on grids.
In the last decades many new methods called *sampling-based motion planners* (e.g. Probabilistic RoadMaps (Kavraki et al., 1996), Rapidly exploring Random Trees (LaValle and Kuffner, 1999) and several other methods) based on the random exploration of the configuration space have been introduced to cope with the aforementioned problems. These methods aim to build/find a graph structure (e.g. roadmap, tree) by exploring the high dimensional configuration space while fulfilling the system constraints, by randomly sampling configurations and checking if different obstacle-free open subspaces can be connected.
In general, sampling-based motion planners are classified into two categories: *multi-query* and *single-query* planners. Probabilistic roadmaps belong to the multi-query class: firstly a graph is built beforehand to represent the connectivity of the configuration space; in *query* phase, given a start and a goal state, which are added to the graph, a standard discrete search algorithm (e.g. A*) is applied on the graph to generate a path. The same graph may be used for different pairs of start and goal poses. RRTs techniques are single-query planners: given a start and a goal, they build a graph structure (e.g. tree) that connects the start to the goal. For each new pair of start and goal states, a new graph needs to be built. in RRT, there is no need of applying any search method on the generated graph to find a path, but simply a backtracking from the node that reached the goal area to the start node.
Sampling-based motion planners offer less strict properties than combinatorial ones, e.g. a sampling-based motion planner is said to be *probabilistically complete* instead of complete.

Definition 4. ***Probabilistic completeness****:*
A sampling-based motion planner is said to be ***probabilistically complete*** *if the probability that an existing solution is found converges to 1 as the number of iterations grows to infinity.*

Differently from discrete search approaches that are *resolution complete* algorithms, sampling-based motion planners that are probabilistically complete

cannot tell us that a problem does not have a solution. Instead the former inform us when a solution is not found at the given resolution.

2.5.1 Rapidly Exploring Random Trees

Let $\mathcal{C} \subset \mathbb{R}^d$ be the state space and $\mathcal{U} \subset \mathbb{R}^m$ the control space, both bounded connected open sets ($d > 0, m > 0$). Given a set $\mathcal{C} \in \mathbb{R}^d$ and a scalar $l \geq 0$, a trajectory (or path) in $\mathcal{C}$ is a continuous function $\boldsymbol{\sigma} : [0, l] \to \mathcal{C}$, where l may indicate the trajectory (or path) length or a time instant. The set of all trajectories (or paths) in $\mathcal{C}$ with nonzero length is denoted by $\Sigma_{\mathcal{C}}$. A dynamical system can be described by a set of differential equation as

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t)) + g(\mathbf{x}(t)) \qquad \mathbf{x}(0) = \mathbf{x}_0 \tag{2.1}$$

where $\mathbf{x}(t) \in \mathcal{C}$, $\mathbf{u}(t) \in \mathcal{U}$, for all t, $\mathbf{x}_0 \in \mathcal{C}$ and f is a function describing the kinematics of the system and g its drift.

Definition 5. ***Feasible Motion Planning problem*** *p:*
The algorithm $\mathcal{ALG}$ solves a feasible kinematic motion planning problem p if given an obstacle space $\mathcal{C}_{obs} \subset \mathcal{C}$, a free space $\mathcal{C}_{free} = \mathcal{C} \setminus \mathcal{C}_{obs}$, an initial state $\mathbf{x}_{start} \in \mathcal{C}_{free}$ and a goal region $\mathcal{C}_{goal} \subset \mathcal{C}_{free}$, it finds the controls $\mathbf{u}(t) \in \mathcal{U}$ with domain $[0, T]$, $T > 0$, such that the unique feasible trajectory $\boldsymbol{\sigma}(t)$ that satisfies equation (2.1), is in the free space $\mathcal{C}_{free} \subseteq \mathcal{C}$ and goes from $\mathbf{x}_{start}$ to a goal $\mathbf{x}_{goal} \in \mathcal{C}_{goal}$.

The RRT (LaValle and Kuffner, 1999) algorithm is a popular algorithm for planning under differential nonholonomic constraints for solving a feasible motion planning problem. For more details on nonholonomic systems and constraints refer to Appendix A.
The RRT procedure is outlined in Algorithm 3. The algorithm builds a tree τ that grows towards unexplored regions of the free space $\mathcal{C}_{free}$. At each iteration n a new configuration $\mathbf{x}_{rand}$ is sampled and possibly connected to the current τ. To connect $\mathbf{x}_{rand}$ to τ, firstly the algorithm looks for the nearest neighbor $\mathbf{x}_{near}$ in τ, then it extends τ from $\mathbf{x}_{near}$ towards $\mathbf{x}_{rand}$ by using an extend (or steer) function, and it adds the new branch $(\mathbf{x}_{new}, \boldsymbol{\sigma}_{new}, \mathbf{u}_{best})$ if and only if the extension is collision free. The algorithm stops once a branch arrives into the goal region $\mathcal{C}_{goal}$, see Fig. 2.4 and Fig. 2.5.
The RRT functions reported in Algorithm 3 are as follows:

- `Sampling`($\mathcal{C}$): by using a predefined sampling distribution $p(\mathcal{C})$ a configuration $\mathbf{x}_{rand}$ is generated from $\mathcal{C}$ (different kind of **sampling distributions** can be used). Many variants of RRT require the distribution to be independent and identically distributed. Different sampling distributions may lead to different planning performance, see Chapter 5.

Algorithm 3 Rapidly-exploring Random Tree (RRT)

1: **function** RRT($\mathbf{x}_{start}$, $\mathbf{x}_{goal}$)
2: τ.AddNode($\mathbf{x}_{start}$)
3: **while** $n \leq$ MAX_ITERATIONS **do**
4: $\mathbf{x}_{rand} \Leftarrow$ Sampling($\mathcal{C}$)
5: $\mathbf{x}_{near} \Leftarrow$ NearestSearch($\tau, \mathbf{x}_{rand}$)
6: **if** InCollision($\mathbf{x}_{near}$) **then**
7: continue
8: **end if**
9: $\mathbf{x}_{new}, \mathbf{u}_{best}, \boldsymbol{\sigma}_{new} \Leftarrow$ Extend($\mathbf{x}_{near}, \mathbf{x}_{rand}$)
10: **if** InCollision($\boldsymbol{\sigma}_{new}$) **then**
11: continue
12: **end if**
13: τ.AddNode($\mathbf{x}_{new}$)
14: τ.AddEdge($\mathbf{x}_{near}, \mathbf{x}_{new}, \boldsymbol{\sigma}_{new}, \mathbf{u}_{best}$)
15: **if** $\mathbf{x}_{new} \in \mathcal{C}_{goal}$ **then**
16: **return** ExtractTrajectory($\mathbf{x}_{new}$)
17: **end if**
18: $n \Leftarrow n + 1$
19: **end while**
20: **return** *failure*

- NearestSearch($\tau, \mathbf{x}_{rand}$): according to a defined **distance metric**, e.g. Euclidean distance, path length or control effort, a vertex $\mathbf{x}_{near}$ is selected from the tree τ. The use of a metric that encodes more information on the kinematic or dynamic system used for the planning, improves the final quality of the solution. In Chapter 4, we give more details on this routine.

- Extend($\mathbf{x}_{near}, \mathbf{x}_{rand}$): this function extends the tree τ from the nearest vertex $\mathbf{x}_{near}$ towards $\mathbf{x}_{rand}$ by returning the new branch, the associated controls and the last state of the branch ($\boldsymbol{\sigma}_{new}, \mathbf{u}_{best}, \mathbf{x}_{new}$). Originally (for more details the reader is referred to LaValle and Kuffner (1999)) the **extend** function (or **steer** function) has been introduced as function that simply uses forward propagation of a chosen control $\mathbf{u}$ (randomly generated or selected as the best among a discrete set) for a given time Δt. Differently one can use techniques that fully solve the two-points boundary value problem (2P-BVP) for the generation of a trajectory that exactly connects $\mathbf{x}_{near}$ to $\mathbf{x}_{rand}$, as it is in the case of PRM where a local planner was used instead of random control propagations. The use of random control propagations, and thus RRT as presented by LaValle and Kuffner (1999), allows to solve kinodynamic motion planning problems for which a steer function that fully solve the 2P-BVP is not available. Chapter3 gives more insights on this approach and

shows how the use of 2P-BVPs leads to better planning performance and path quality than random control propagations.

- `InCollision`(.): it checks if the argument (the trajectory $\boldsymbol{\sigma}_{new}$ or a state $\mathbf{x}_{near}$) is in collision, i.e. $\boldsymbol{\sigma}_{new} \in \mathcal{C}_{obs}$ or $\mathbf{x}_{near} \in \mathcal{C}_{obs}$. Several different approaches can be used to check for collisions (e.g. GJK-EPA Gilbert et al. (1988)), the overall planning efficiency is highly related to the collision checker performance. In this thesis we do not focus on enhancing this component.

An interested reader can refer to Şucan and Kavraki (2010) for further details on the implementation of those functions and their variants.
Planning efficiency and path quality in RRT are strictly related to its sub-routines: sampling distribution, distance metric, extend function. In this work we are going to carefully study their impact on the algorithm performance and propose new methods to improve them.

2.5.1.1 Probabilistic completeness of RRT

RRT is probabilistically complete for geometric path planning in $\mathbb{R}^n, n \in \mathbb{N}^+$, where the sampling distribution is uniform, the distance metric is Euclidean distance and the extend function is a simple straight-line connection.

Probabilistic completeness is well established for systems with geometric constraints (Ladd and Kavraki, 2004) and kinodynamic systems under some strong assumptions, that is: forward simulations (LaValle and Kuffner Jr, 2001) with random controls and propagation time, uniform sampling and optimal steering (Hsu et al., 2002; Frazzoli et al., 2001) and holonomic systems with state-space based interpolation (Caron et al., 2014).

A recent research (Kunz and Stilman, 2014) gave more insights on the probabilistic properties of the kinodynamic version of the algorithm: its probabilistic completeness is highly related to the type of extend function used, more specifically Kunz at al. showed that RRTs with fixed propagation time and best-input extension are not probabilistically complete.

2.5.2 Optimal Sampling-based Motion Planning

RRT solves a *feasible* motion planning problem: it finds a solution without considering or improving its quality. *Optimal* planners aim to find a higher quality solution (where the quality depends on the criteria-cost defined). Let $\mathcal{C} \in \mathbb{R}^d$ be the state space and $\mathcal{U} \in \mathbb{R}^m$ the control space (both bounded connected open sets, $d > 0, m > 0$), a dynamic system can be described by the differential equation

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t)) + g(\mathbf{x}(t)) \qquad \mathbf{x}(0) = \mathbf{x}_0 \tag{2.2}$$

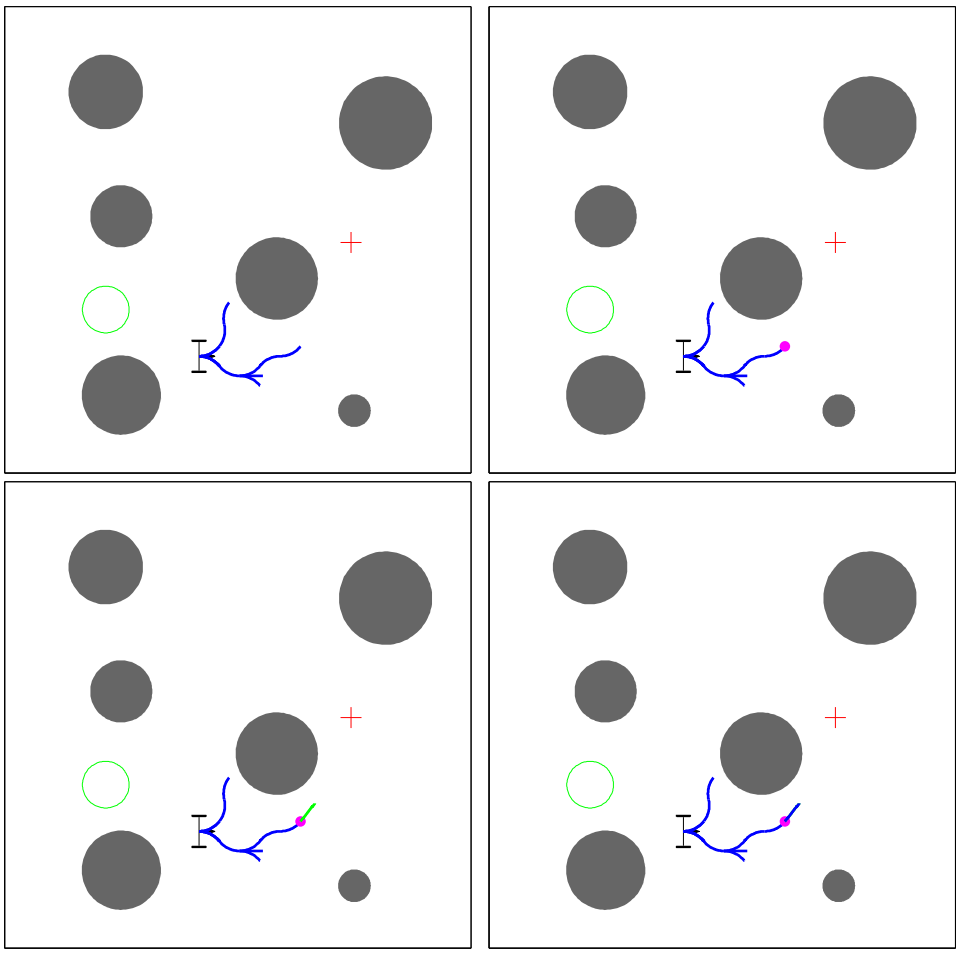

Figure 2.4: Example of an iteration in RRT. The RRT tree τ is drawn in **blue**, the goal region $\mathcal{C}_{goal}$ in a **green** circle, the initial robot configuration $\mathbf{x}_{start}$ in **black**. **Top left:** firstly a configuration $\mathbf{x}_{rand}$ (**red cross**) is sampled. **Top right:** the nearest vertex (in this case in Euclidean sense) is selected (the **violet** dot) from the tree. **Bottom left:** the tree is extended (in **green** the new branch σ_{new}) and checked for collisions. **Bottom right:** If no collisions are detected the new branch is finally added to the tree.

with $\mathbf{x}(t) \in \mathcal{C}$, $\mathbf{u}(t) \in \mathcal{U}$, f describing the system's kinematic constraints and g its drift.

Definition 6. ***Asymptotically-Optimal Motion Planning Problem*** p_{opt} :
Given an obstacle space $\mathcal{C}_{obs} \in \mathcal{C}$, a free space $\mathcal{C}_{free} \in \mathcal{C} \setminus \mathcal{C}_{obs}$, a start state $\mathbf{x}_{start} \in \mathcal{C}_{free}$, a goal state $\mathbf{x}_{goal} \in \mathcal{C}_{goal} \subset \mathcal{C}_{free}$ and a cost function $C_{\sigma} : \Sigma_{\mathcal{C}} \to \mathbb{R} \geq 0$, an asymptotically-optimal algorithm $\mathcal{ALG}$, as the number of its iterations grows to infinity, will find a

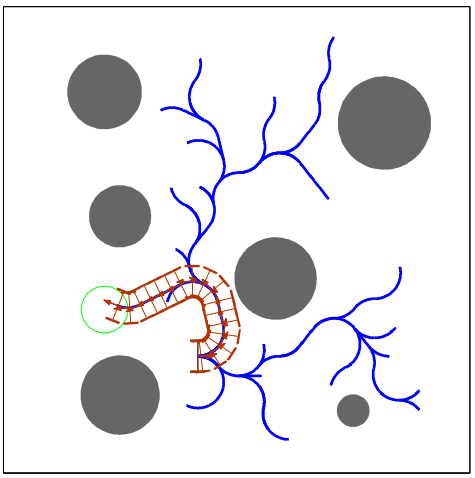

Figure 2.5: Example of an RRT tree grown by using a set of predefined controls (motion primitives). The RRT tree τ is drawn in **blue**, the goal region $\mathcal{C}_{goal}$ in a **green** circle, the initial robot configuration $\mathbf{x}_{start}$ in **black**. The final path is colored in **purple**

collision free minimum cost solution $\boldsymbol{\sigma}^*_{\mathcal{ALG}} \in \mathcal{C}_{free}$

$$\lim_{n\to\infty} \boldsymbol{\sigma}^*_{\mathcal{ALG}} = \texttt{arg}\min_{\boldsymbol{\sigma}\in\Sigma_{\mathcal{C}}} C_{\boldsymbol{\sigma}}(\boldsymbol{\sigma}) \tag{2.3}$$

Recently, optimal variants of the most common sampling based motion planners (RRG, PRM*, RRT*) were introduced by Karaman and Frazzoli (2010b, 2011). In (Karaman and Frazzoli, 2010a, 2013) the same authors extended their algorithms to take into account the kinematic and dynamic constraints of holonomic and nonholonomic systems. The algorithms are based on the representation of the state space in random geometric graphs (Penrose, 2003).

Random geometric graphs are undirected graphs with vertices placed randomly in a space (according to defined probability distributions) and edges added to connect points that are close to each other (according to a particular criterion). Different probability distributions and connection methods between vertices define different random geometric graphs with distinct properties. Many optimal sampling-based algorithms are based on random r-disc graph, see an example in Fig 2.6.

Definition 7. ***Random*** r***-disc graph****:*
A random r-disc graph $\mathcal{G}^{disc}(n,r)$ *in* $\mathbb{R}^d$ $(d > 0)$*, is a graph of* $n > 0$ *nodes* $(X_1,\ldots,X_n)$*, which are independent and uniformly distributed random variables in* $(0,1)^d$ *and are pairwise connected if the Euclidean distance between two vertices is lower than a given radius* $r > 0$*.*

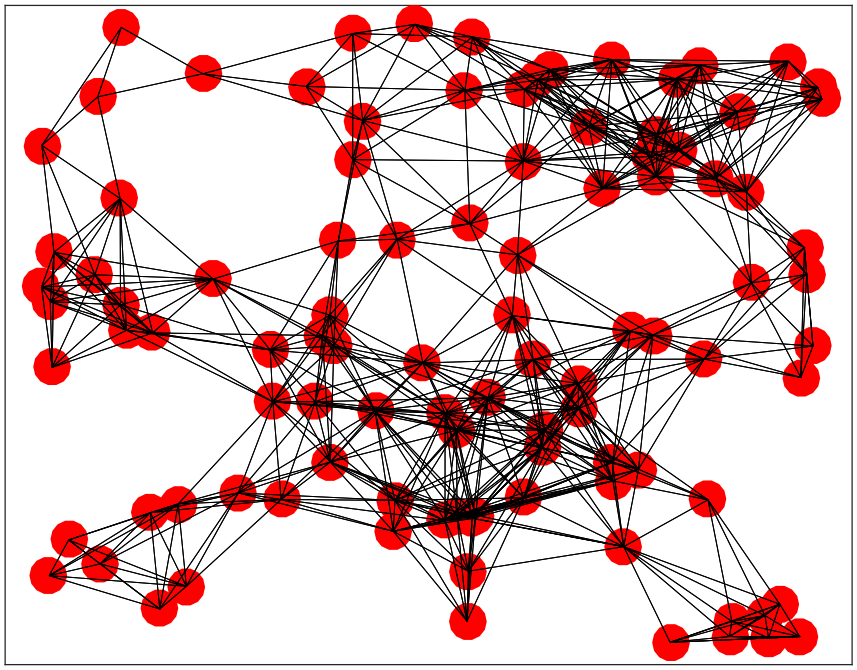

Figure 2.6: Example of random geometric r-disc graph with 100 nodes, with the radius respecting the connectivity property.

An important property of those type of random geometric graphs is their connectivity: a graph is connected when there is a path between every pair of vertices.

Definition 8. ***Connectivity of a random r-disc graph:***
The graph $\mathcal{G}^{disc}(n,r)$ is connected almost surely if the connection radius r holds the following relation: $\zeta_d \; r^d > log(n)/n$, with ζ_d being the volume of the unit ball in d dimensions.

Based on the properties of random geometric graphs, Karaman and Frazzoli introduced RRT*, PRM* and RRG algorithms. RRT* is the optimal variant of RRT, its tree is reshaped according to a given cost function.

2.5.2.1 RRT*

RRT* (Karaman and Frazzoli, 2011) grows its tree towards the entire state space. Differently from RRT, RRT* modifies its tree according to a given cost function. At each iteration, the tree is rewired locally in a way to prefer locally optimal extensions. The algorithm was proven to be asymptotically optimal: as the number of iterations goes to infinity, the algorithm will find the trajectory with the lowest cost. The RRT* rewiring procedure allows the algorithm to find asymptotically for each state in the state space the best trajectory that connects this state to the

Algorithm 4 *RRT**

1: **function** RRT*($\mathbf{x}_{start}$, $\mathbf{x}_{goal}$)
2: τ.AddNode($\mathbf{x}_{start}$)
3: $g(\mathbf{x}_{start}) \Leftarrow 0$
4: $n \Leftarrow 1$
5: **while** $n \leq$ MAX_ITERATIONS **do**
6: $\mathbf{x}_{rand} \Leftarrow$ Sampling($\mathcal{C}$)
7: $\mathbf{x}_{near} \Leftarrow$ NearestSearch($\tau, \mathbf{x}_{rand}$)
8: $\mathbf{u}_{new}, \boldsymbol{\sigma}_{new}, \mathbf{x}_{new} \Leftarrow$ Steer($\mathbf{x}_{near}, \mathbf{x}_{rand}$)
9: **if** InCollision($\boldsymbol{\sigma}_{new}$) **then**
10: continue
11: **end if**
12: τ.AddNode($\mathbf{x}_{new}$)
13: τ.AddEdge($\mathbf{x}_{near}, \mathbf{x}_{new}, \boldsymbol{\sigma}_{new}, \mathbf{u}_{new}$)
14: $g(\mathbf{x}_{new}) \Leftarrow g(\mathbf{x}_{near}) +$ Cost($\mathbf{x}_{near}, \mathbf{x}_{new}$)
15: $\tau \Leftarrow$ Rewire($\tau, \mathbf{x}_{new}, \mathbf{x}_{near}$)
16: **if** $\mathbf{x}_{new} \in \mathcal{C}_{goal}$ **then**
17: $\boldsymbol{\sigma}_{result} =$ ExtractTrajectory($\mathbf{x}_{new}$)
18: **end if**
19: $n \Leftarrow n + 1$
20: **end while**
21: **return** *failure*

starting state.
The RRT* procedure is outlined in Alg.4-5. The algorithm builds a tree τ that grows towards unexplored regions of the free space $\mathcal{C}_{free}$. At each iteration n a new configuration $\mathbf{x}_{rand}$ is sampled and possibly connected to the current τ. To connect $\mathbf{x}_{rand}$ to τ, firstly the algorithm looks for the nearest neighbor $\mathbf{x}_{near}$ in τ, then it extends τ from $\mathbf{x}_{near}$ towards $\mathbf{x}_{rand}$ by using a steer function, and it adds the new branch $(\mathbf{x}_{new}, \boldsymbol{\sigma}_{new}, \mathbf{u}_{best})$ if and only if the extension is collision free. Once the new branch has been generated, then the tree τ is rewired in two steps, see Alg 5 and Fig 2.7: firstly the new vertex $\mathbf{x}_{new}$ is re-connected to a better (in terms of cost) parent by checking all the vertices in a neighborhood $\mathcal{X}_{near}$ of $\mathbf{x}_{new}$ (Lines 5-14); secondly this new connection (with the new parent) is checked if it can further improve the cost-to-come of a vertex $\mathbf{x}_j$ lying in a neighborhood of $\mathbf{x}_{new}$ by associating to $\mathbf{x}_j$ as new parent the state $\mathbf{x}_{new}$, if this is true the new parent vertex $\mathbf{x}_{new}$ is connected to the vertex $\mathbf{x}_j$ (Lines 16-26). The algorithm stops once a branch arrives into the goal region $\mathcal{C}_{goal}$.
RRT* shares many functions with RRT, here we detail only the ones specific for RRT*:

- g($\mathbf{x}$): each vertex in the tree τ keeps its cost-to-come g, the cost to arrive to the vertex $\mathbf{x}$ from $\mathbf{x}_{start}$ through τ.

Algorithm 5 *Rewire*(τ, $\mathbf{x}_{new}$, $\mathbf{x}_{near}$)

function Rewire(τ, $\mathbf{x}_{new}$, $\mathbf{x}_{near}$)
$\mathbf{x}_{min} \Leftarrow \mathbf{x}_{near}$
$g(\mathbf{x}_{min}) \Leftarrow g(\mathbf{x}_{new})$
$\mathcal{X}_{near} \Leftarrow$ NearVertices($\tau, \mathbf{x}_{new}, |\tau|$)
for $\forall\, \mathbf{x}_i \in \mathcal{X}_{near}$ **do**
  $\mathbf{u}_i, \boldsymbol{\sigma}_i, \mathbf{x}_i \Leftarrow$ Steer($\mathbf{x}_i, \mathbf{x}_{new}$)
  if $\boldsymbol{\sigma}_i \in \mathcal{C}_{obs}$ **then**
    continue
  end if
  $c_i = g(\mathbf{x}_i) +$ Cost($\mathbf{x}_i, \mathbf{x}_{new}$)
  if $c_i < g(\mathbf{x}_{min})$ **then**
    $\mathbf{x}_{min} \Leftarrow \mathbf{x}_i$, $\boldsymbol{\sigma}_{min} \Leftarrow \boldsymbol{\sigma}_i$, $\mathbf{u}_{min} \Leftarrow \mathbf{u}_i$
  end if
end for
τ.AddEdge($\mathbf{x}_{min}, \mathbf{x}_{new}, \boldsymbol{\sigma}_{min}, \mathbf{u}_{min}$)
for $\forall\, \mathbf{x}_j \in \mathcal{X}_{near}$ **do**
  $\mathbf{u}_j, \boldsymbol{\sigma}_j, \mathbf{x}_j \Leftarrow$ Steer($\mathbf{x}_{new}, \mathbf{x}_j$)
  if $\boldsymbol{\sigma}_j \in \mathcal{C}_{obs}$ **then**
    continue
  end if
  $c_j = g(\mathbf{x}_{new}) +$ Cost($\mathbf{x}_{new}, \mathbf{x}_j$)
  if $c_j < g(\mathbf{x}_j)$ **then**
    $\mathbf{x}_{parent} \Leftarrow$ Parent($\mathbf{x}_j$)
    τ.RemoveEdge($\mathbf{x}_{parent}, \mathbf{x}_j$)
    τ.AddEdge($\mathbf{x}_{new}, \mathbf{x}_j, \boldsymbol{\sigma}_j, \mathbf{u}_j$)
  end if
end for
return τ

- Cost($\mathbf{x}_i, \mathbf{x}_j$): it computes the cost of the trajectory connecting $\mathbf{x}_i$ to $\mathbf{x}_j$. The cost needs to be defined as a monotonic, additive and Lipschitz continuous function (Karaman and Frazzoli, 2011).

- NearVertices($\tau, \mathbf{x}_{new}, |\tau|$): a set of vertices $\mathcal{X}_{near}$ near to the vertex $\mathbf{x}_{new}$ is selected from the tree τ. The nearness is based on the number of vertices $|\tau|$ in τ and the type of dynamic systems. For generic kinodynamic systems (Karaman and Frazzoli, 2010a), the vertices are selected if they are in the ball of volume $\gamma_{RRT*}(\log(|\tau|)/\ |\tau|)^{\frac{1}{d}}$ (d being the dimension of $\mathcal{C}$). For systems with nonholonomic constraints (Karaman and Frazzoli, 2013), this procedure returns the set of vertices lying into a weighted box of size $\gamma_{RRT*}(\log(|\tau|)/\ |\tau|)^{\frac{1}{D}}$ (where D is the Hausdorff dimension of the distribution generated by the dynamics), the choice of the weighted box is related to

the study of reachability for nonholonomic systems ((Krener, 1974), (Montgomery, 2006), see App. A).

- `Steer`$(\mathbf{x}_i, \mathbf{x}_j)$: this function connects the states $\mathbf{x}_i$ to $\mathbf{x}_j$ by returning a new trajectory, the associated controls and the last state of the trajectory $(\boldsymbol{\sigma}_{new}, \mathbf{u}_{best}, \mathbf{x}_{new})$. In RRT* the steer function fully solves the two-points boundary value problem (2P-BVP) for the generation of a trajectory that exactly connects $\mathbf{x}_{near}$ to $\mathbf{x}_{rand}$. In case of generic dynamic systems the steer function is requested to be optimal (Karaman and Frazzoli, 2010a), in case of nonholonomic system the steer function has to satisfy the topological property (Karaman and Frazzoli, 2013; Sekhavat and Laumond, 1998), see App. A.2.

RRT* is asymptotically optimal with $\gamma_{RRT*} > (2(1+\frac{1}{d}))^{\frac{1}{d}}(\mu(\mathcal{C}_{free})/\ \zeta_d)^{\frac{1}{d}}$, with d being the dimensionality of $\mathcal{C}$ and ζ_d the volume of unit ball (or box in the case of nonholonomic systems) in a space of dimension d.

2.5.2.2 Further Optimal Algorithms

Recently, several novel asymptotically optimal sampling-based algorithms beside of RRT* have been introduced: FMT* (Janson et al., 2015), RRT^X (Otte and Frazzoli, 2016), RRT$^\sharp$ (Arslan and Tsiotras, 2013), Informed RRT* (Gammell et al., 2014), BIT* (Gammell et al., 2015). All of them build (implicitly) a random geometric graph and propose different methods to extrapolate a path/trajectory from it. As pointed out by Arslan (2015), all these methods solve the motion planning problem using dynamic programming (DP) principles over a random geometric graph: the same principles can be seen in standard discrete graph search methods as well including A*, Dijkstra, LPA* (Koenig et al., 2004), D* Lite (Koenig and Likhachev, 2002). The Fast Marching Tree algorithm (FMT*) performs a lazy dynamic programming recursion on a batch of random samples to grow a tree that moves outward in cost-to-come space. RRT$^\sharp$ and RRT^X are using relaxation methods to prioritize and select the best vertex to expand during each tree rewiring. RRT^X, an asymptotically optimal sampling-based motion planner for real-time navigation in dynamic environments, was shown to handle dynamic environments by properly propagating the cost over the tree, by inducing (each time an obstacle appear or disappear) a graph rewiring cascade that quickly updates the graph and repairs its shortest-path-to-goal subtree. Informed RRT* is highly based on RRT*: it behaves like RRT* until a first solution is found, then it starts to sample from an ellipsoid that most likely will contain a better solution (in terms of path length) than the existing one. BIT* (Batch Informed Trees) is an algorithm that by using admissible heuristics and by reusing information gained during the previous steps of the algorithm like in LPA*, guides the growth of

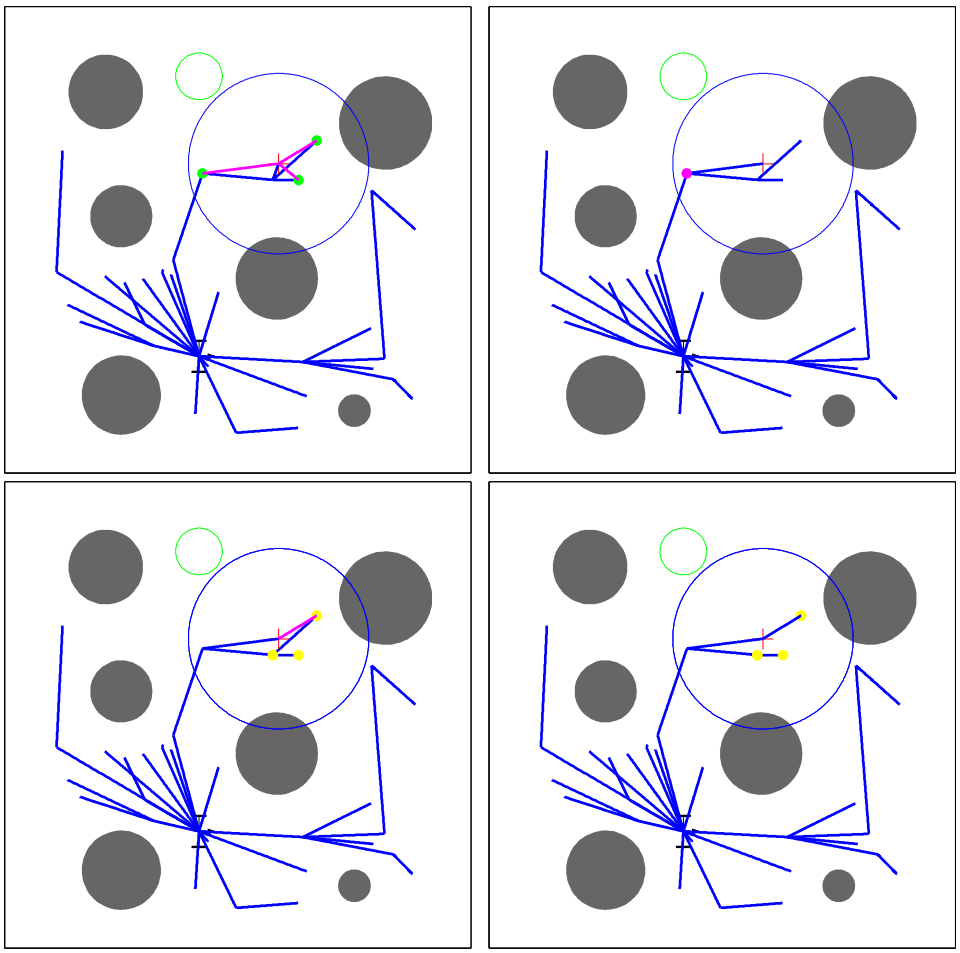

Figure 2.7: Example of an iteration in the rewire procedure of RRT* with straight-line steer function solving the shortest-path problem. The RRT* tree τ is drawn in **blue**, the goal region $\mathcal{C}_{goal}$ in a **green** circle, the initial robot state $\mathbf{x}_{start}$ in **black**. **Top left:** initially in the rewiring procedure the best parent $\mathbf{x}_{min}$ of the current new state $\mathbf{x}_{new}$ (which is the **red** cross), is chosen among several neighboring nodes $\mathbf{x}_i \in \mathcal{X}_{near}$ (in **green**) in the tree. **Top right:** the best parent vertex $\mathbf{x}_{min}$ (**magenta** dot) is selected from the tree and the new state $\mathbf{x}_{new}$ (**red** cross) is connected to it, while removing its connection to its previous parent. **Bottom left:** subsequently the algorithm checks if other tree vertices can benefit from the new added configuration, several nodes (the **yellow** dots) in the tree are checked if the new added configuration (the **red** cross) can be connected (attempted connections are in **magenta**). **Bottom right:** If a new connection has lower cost than the previous existing one, the new one is kept and the tree changed its shape.

the tree towards promising areas of the state space. This algorithm performs an ordered search on batches of samples: each batch is generated in promising areas (as in Informed RRT*) where possibly an improvement to the current solution can be found.

Moreover, very recently a set of different algorithms try to de-randomize sampling-based motion planners (Janson et al., 2018; LaValle et al., 2004): the key idea is to recognize the relationship between classical grid search with sampling-based motion planners. In the latter, one could use as sampling distribution, deterministic sampling sequences that guarantee optimality and completeness in a deterministic way (Janson et al., 2018).

The methods introduced later in the thesis can be applied not only to RRT and RRT* but also to the other detailed optimal sampling based motion planners.

2.6 Trajectory Optimization Methods

In the last decade, many trajectory optimization approaches to path and trajectory planning have been introduced in robotics, partly taking inspiration from other scientific communities such as control theory, machine learning where those methods are well understood.

One of the first path optimizers in robotics is the *elastic band* approach (Quinlan and Khatib, 1993). The elastic band is a dynamic path optimization algorithm that adapts locally (optimizes) an initial plan based on the obstacles' positions and on the environmental changes. The algorithm applies a gradient descent on the elastic force field generated by the obstacles, which is used to compute the displacements to apply to the bubbles composing the elastic band. The elastic strips approach (Brock and Khatib, 2002) extends the elastic band approach by further improving its performances for high dimensional systems. This is achieved by operating in the workspace (while the elastic band approach works in the configuration space). Also in this approach a gradient descent is applied to compute and modify the trajectory on-the-fly.

CHOMP (Zucker et al., 2013) is a gradient descent method for trajectory optimization, based on the computation of a covariant gradient that guides the optimization of a functional, defined in the trajectory space, that sums up a smoothing cost (this related to the square of the velocity) and to the obstacle signed distance field. The authors showed that the algorithm is fast (given fast kinematic and collision evaluations) and generates very smooth solutions, where for smoothness the authors mean smooth variations in acceleration.

Differently from the above mentioned approaches Lamiraux et al. (Lamiraux et al., 2004) introduced a path optimization method for nonholonomic systems. The method modifies the path by pushing it away from the obstacles and meanwhile satisfying the nonholonomic constraints: the initial path is deformed by

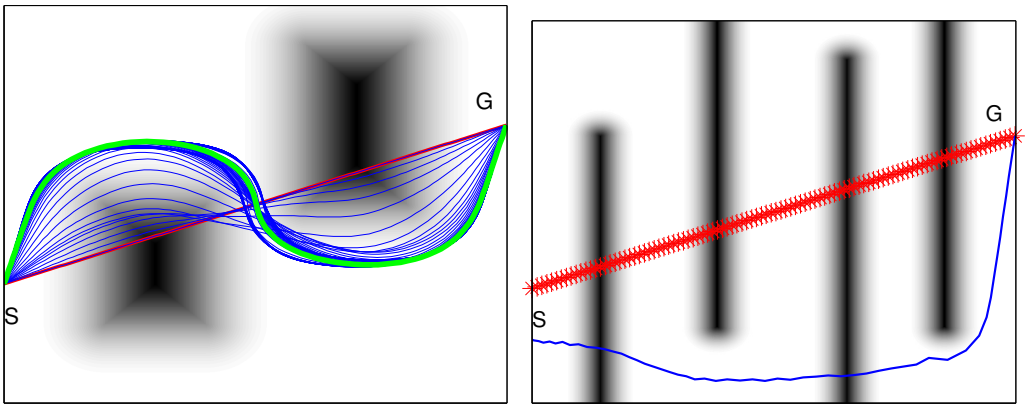

Figure 2.8: Example solutions using CHOMP to generate a purely geometric path in a 2D environment for a given pair of start and goal points (marked with **S** and **G**). **Left**: CHOMP generates a smooth solution (in **green**) if the environment has few constraints. The initial path is given (in **red**). For this simple environment CHOMP does not get stuck in local optima. **Right:** The algorithm is initialized with a path (in **red** stars) that goes through the obstacles (**black** regions); it generates a solution (in **blue**) that does not avoid the obstacles.

perturbing the input functions defined along the path.
Splines are a natural representation for smooth motion, in that the velocity profile is straightforward and fast to compute based on its control points. Lau et al. (2009) define and solve the motion planning problem for a wheeled mobile robot using quintic Bézier spline. The method uses a modified version of RPROP (Riedmiller and Braun, 1993) (i.e a first-order optimization method) to solve the optimization problem.
Several other approaches use second-order Newton optimization methods which are more robust than gradient descent for trajectory optimization. To this category belong different works, that frame the problem as a non-linear constrained optimization method (for a more detailed review see (Toussaint, 2016), (Diehl et al., 2009), (Nocedal and Wright, 2006)).
Hereinafter we will introduce example algorithms that belong to this class. Iterative LQG (iLQG (Todorov and Li, 2005)) which linearizes the dynamics and generates a quadratic cost function about a given feasible initial trajectory and then uses LQR to compute a control policy. This approach was successfully used to generate locally optimal trajectories for robots with many degrees of freedom.
The Iterated LQR Smoothing (van den Berg, 2014), introduce a fast-converging iterative procedure to compute a locally-optimal feedback control policy for systems with non-linear dynamics and non-quadratic cost, where in each iteration the minimum cost trajectory for the linear-quadratic control problem is computed

by exploiting a backward and a forward pass as in Kalman smoothing.
TrajOpt (Schulman et al., 2014) is based on Sequential Quadratic Programming (SQP) which formulates the constrained optimal motion planning problem in each iteration as a convex optimization problem, by linearizing the non-linear system on the current iteration point. The method was shown to efficiently compute smooth solutions for high-dimensional systems, although it suffers of convergence problems for system with high non-linearities due to linearization issues.
Fig. 2.8 shows known limits (as also described by Schulman et al. (2014)) of trajectory optimization methods (in this example we use CHOMP). Often in complex environments if initialized with a path that goes through obstacles, this kind of algorithms may get stuck in infeasible regions and cannot generate a solution (Fig. 2.8-right). Also, as reported by the authors of several optimization methods described above, when given a better initial path this class of algorithms may often get stuck in a local optima, which is a feasible solution but does not improve much the given initial path (or trajectory).

These methods, which are local search approaches, are not well-suited to solve the global motion planning problem as in the case of sampling-based motion planners: often they suffer from local minima due to the high degree of non-linearity and non-convexity of the objective function.

2.7 Quantify Smoothness

Smoothness, although being an intuitive concept, is less straightforward to measure precisely. Usually one can guess that smooth movements are composed of a few of submovements, which do not show high accelerations in the their joins. Unsmooth movements can be due to a loss of continuity between the submovements, providing shaking movements to the entire robot structure.

2.7.1 Smoothness Metrics

Balasubramanian et al. (2012) survey a number of metrics to quantify movement smoothness, in the thesis we adopt some of the measures presented by the cited authors that are relevant in our context.
We firstly introduce smoothness metrics related to the velocity and the acceleration of the robot movements. Let v_{max} be the maximum magnitude of the robot velocity vector $\mathbf{v}$ (nominal feedforward commands (De Luca et al., 2001)), $\tilde{\mathbf{v}} = \frac{\mathbf{v}(t)}{v_{max}}$ the normalized velocity, and $[t_1, t_2]$ the time interval over which the movement is performed.

1. η_{nmaJ}, the average of the mean absolute jerk normalized by v_{max}, for which

Exp.	η_{nmaJ}	η_{spal}	R	η_{pm}
line-arc	-0.0024	-0.0022	0.0011	1
line-arc-line	-0.0049	-0.0033	0.0023	2
sine wave	-0.0032	-0.0029	0.0023	8

Table 2.1: Smoothness metrics - Example Paths Results

the best value is zero:

$$\eta_{nmaJ} = -\frac{1}{v_{max}(t_2 - t_1)} \int_{t_1}^{t_2} \left| \frac{d^2\mathbf{v}}{dt^2} \right| dt,$$

2. average of the speed arc length η_{spal}, for which the best value is zero

$$\eta_{spal} = -ln \left(\int_{t_1}^{t_2} \sqrt{\left(\frac{1}{t_2 - t_1} \right)^2 + \left(\frac{d\tilde{\mathbf{v}}}{dt} \right)^2} dt \right),$$

3. average number of peaks η_{pm}

$$\eta_{pm} = -|\mathcal{V}_{peaks}|,$$

with $\mathcal{V}_{peaks} = \{\mathbf{v}(t) : \frac{d\mathbf{v}}{dt} = 0, \frac{d^2\mathbf{v}}{dt^2} < 0\}$ being the set of local velocity maxima.

Secondly we detail a metric that is better suited for measuring *geometric* trajectory smoothness and thus human-perceived smoothness (namely how sharp the turns are): roughness R, defined as the square of the change in curvature κ of the robot, integrated along the trajectory and normalized by the trajectory length L,

$$R = \int_{t_0}^{t_l} \left| \frac{1}{L} \frac{d\kappa}{dt} \right|^2 dt.$$

2.7.2 Behavior of the Metrics

We report here a few basic path examples where we study the metrics' behavior: *line and arc*, *line - arc - line*, *sine wave*, see Fig. 2.7.2. Table 2.1 reports the results for the basic paths. η_{nmaJ} and η_{spal}, together with the number of peaks η_{pm} increase in the cases where there are more changes in velocities (e.g. slowing down or increasing the velocity in *sine wave* and *line-arc-line*). Roughness R is higher in paths with more turns or discontinuities (e.g. *sine wave* and *line-arc-line*).

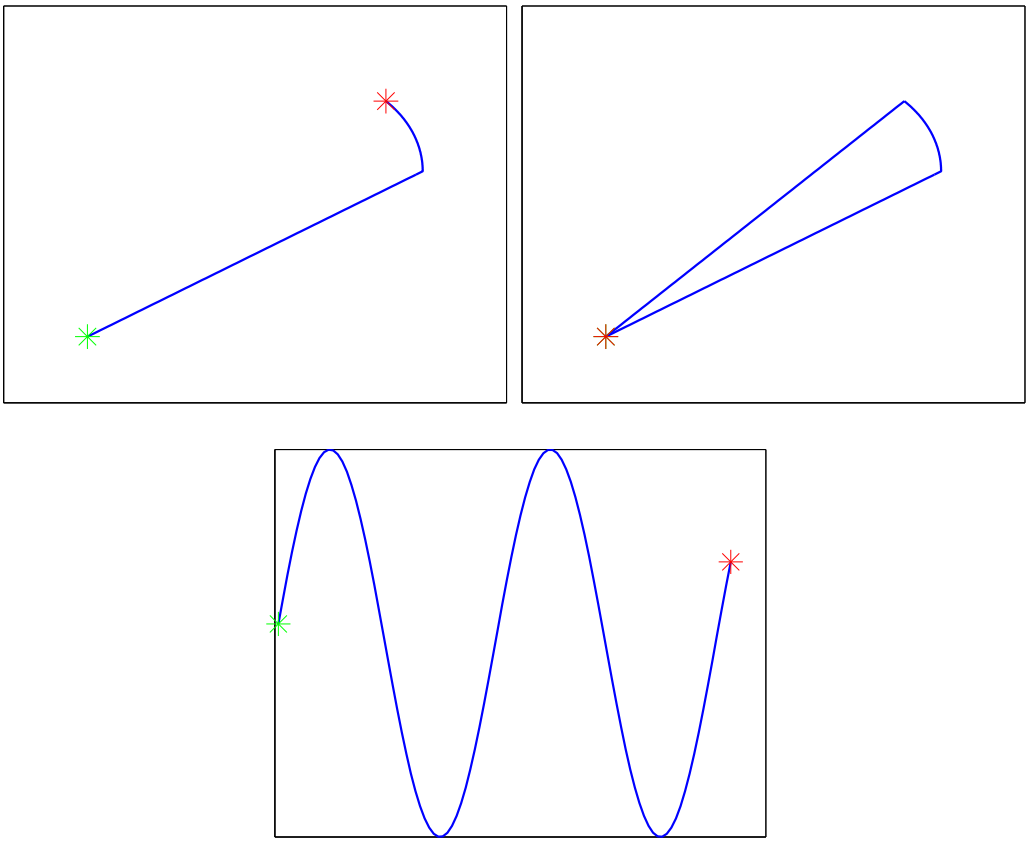

Figure 2.9: Three basic paths where to test the smoothness metrics, namely starting from the **upper-left** to **bottom-centered** figure: *line-arc*, *line-arc-line*, *sine wave*. The **green stars** represent the start points while the **red ones** are the goals: note that start and goal points overlap in the *line-arc-line* path.

2.8 Summary

In this chapter we briefly described state-of-the-art motion planning algorithms. A lot of progress has been in done in the recent years to improve the solution quality of those algorithms. The before described classes of algorithms offer different properties, characteristics that an expert user should be aware of to decide which planner to use depending on the contexts (or scenarios, environments) which mainly define the motion planning requirements. For example, instead of grid-based approaches, sampling based motion planners are the natural choice for systems with particular nonholonomic constraints and very complex cost-functions. For such kind of problems optimization methods (which are faster) may suffer a lot of linearization issues and local optima.

Hereinafter we will focus our attention on motion planners that solve the global

motion planning problem and propose many new techniques to further improve their efficiency and path quality, only in Chapter 8 we will use also a local search planner (or optimization method) to locally adapt an initial global path.

CHAPTER 3

POSQ: A Novel Extend Function for RRT*

"Keep the gradient!"

Bruno Siciliano
Robotics and Automation
Magazine, March 2008

With the intention of generating smooth and real-time robot motion in complex environments, in this chapter we introduce a novel RRT and RRT extend function for wheeled mobile robots. The approach efficiently generates smooth and feasible paths that connect any pairs of states by computing closed-loop forward simulations based on the kinematic model of the robot. We extend the control law of an existing discontinuous state feedback controller to make it usable as an RRT extend function and prove that all relevant stability properties are retained. We study the properties of the new approach as extend function for RRT and RRT* and compare it systematically to a spline-based approach and a large and small set of motion primitives. The results show that our approach generally produces smoother paths to the goal in less time with smaller trees. For RRT*, the approach produces also the shortest paths and achieves the lowest cost solutions when given more planning time.*

3.1 Introduction

Planning with rapidly-exploring random trees (RRT) (LaValle and Kuffner, 1999) has become a popular approach to robot motion planning. RRT planners are single-query sampling-based planners that grow a tree of configurations to eventually cover the entire state space. A probabilistically optimal RRT variant named RRT* has been introduced by Karaman and Frazzoli (2010b). RRT* trees grow based on the notion of a cost: under the assumptions given in (Karaman and Frazzoli, 2011) the solution converges to the optimum as the number of samples approaches infinity.

For robots with kinematic or kinodynamic constraints, the *extend function*, the function that grows the tree by finding collision-free trajectories towards new sampled configurations, becomes a key component. Its task is to connect any pair of states under differential constraints which represents by itself a local planning problem also referred to as the *two-point boundary value problem* (2P-BVP). As described in Chapter 2, originally the **extend** function (or **steer** function) has been introduced, by LaValle and Kuffner (1999), as function that simply uses forward propagation of a chosen control $\mathbf{u}$ (randomly generated or selected as the best among a discrete set) for a given time Δt. Differently one can use techniques that fully solve the two-points boundary value problem (2P-BVP) for the generation of a trajectory that exactly connects $\mathbf{x}_{near}$ to $\mathbf{x}_{rand}$, as it is in the case of PRM (Kavraki et al. (1996)) where a local planner was used instead of random control propagations. The use of random control propagations, and thus RRT as presented by LaValle and Kuffner (1999), allows to solve kinodynamic motion planning problems for which a steer function that fully solve the 2P-BVP is not available. We anticipate that using a *steer* function that fully solve the 2P-BVP, leads to better planning performance and path quality.

Here, we consider the motion planning problem for nonholonomic wheeled robots in the configuration space $\mathcal{C} = \mathcal{R}^2 \times \mathcal{S}^1$ with the goal of particularly smooth and natural real-time motion generation for robots in human environments. To this end, we propose a new extend function for RRT and RRT* which enables the planner to efficiently generate smooth paths. The contribution of this chapter, presented in (Palmieri and Arras, 2014a), is as follows:

- We propose an extend function based on closed-loop predictions for a nonholonomic wheeled mobile robot. It efficiently solves the 2P-BVP by exponentially converging to the goal state from any start state. We extend the control law of the original approach by Astolfi (1999) with a term that leads to quasi-constant path velocities along local path concatenations – a key ability for RRT extend functions. We also prove the relevant stability properties under our modification.

- We systematically compare our approach to two alternative extend functions, namely motion primitives (two sets of different size) and splines. The experiments demonstrate that our approach outperforms both methods in many relevant metrics: smoother paths and shorter planning time (with RRT), shorter paths (with RRT*), and significantly smaller trees (both). We also find that our method can benefit most from the incremental path improvement ability of RRT* resulting in the lowest cost solutions when given more planning time.

- To the best of our knowledge, we present the first systematic study of the impact of different extend functions on RRT and RRT* performance and

path quality. Its necessity is corroborated by the significant variations of key metrics only caused by the use of different extend functions.

The chapter is structured as follows. The next Section reviews the related work and typical extend functions. In Section 3.3 we describe our approach which is then experimentally evaluated in Section 3.4. We discuss the results in Section 3.5, and Section 3.6 concludes the chapter.

3.2 Extend functions in Sampling-based Motion Planners

We briefly describe the motion planning problem under differential constraints and different extend functions. Let $\mathcal{C} \subset \mathbb{R}^d$ be the configuration space and $\mathcal{U} \subset \mathbb{R}^m$ the control space, both bounded connected open sets. A nonholonomic wheeled mobile robot can be described by a differential equation as

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t)) + g(\mathbf{x}(t)) \qquad \mathbf{x}(0) = \mathbf{x}_0 \tag{3.1}$$

where $\mathbf{x}(t) \in \mathcal{C}$, $\mathbf{u}(t) \in \mathcal{U}$, for all t, $\mathbf{x}_0 \in \mathcal{C}$, f describes the kinematics of the system and g its drift. The RRT algorithm, outlined in Algorithm 3, solves a feasible kinematic motion planning problem p: given an obstacle space $\mathcal{C}_{obs} \subset \mathcal{C}$, a free space $\mathcal{C}_{free} = \mathcal{C} \setminus \mathcal{C}_{obs}$, an initial state $\mathbf{x}_{start} \in \mathcal{C}_{free}$ and a goal region $\mathcal{C}_{goal} \subset \mathcal{C}_{free}$, find a control $\mathbf{u}(t) \in \mathcal{U}$ with domain $[0, T]$, $T > 0$, such that the unique trajectory $\boldsymbol{\sigma}$ satisfies equation (3.1), is in the free space $\mathcal{C}_{free} \subseteq \mathcal{C}$ and goes from $\mathbf{x}_{start}$ to a goal $\mathbf{x}_{goal} \in \mathcal{C}_{goal}$.
RRT*, see Alg. 4, instead asymptotically solves an optimal motion planning problem (see Sec.2.5.2.1): as the number of its iterations grows to infinity, it will find a collision free minimum cost solution $\boldsymbol{\sigma}^*_{\mathcal{ALG}} \in \mathcal{C}_{free}$

$$\lim_{n\to\infty} \boldsymbol{\sigma}^*_{\mathcal{ALG}} = \texttt{arg}\min_{\boldsymbol{\sigma}\in\Sigma_{\mathcal{C}}} C_{\boldsymbol{\sigma}}(\boldsymbol{\sigma}) \tag{3.2}$$

RRT and RRT* use an extend function to grow their trees.

3.2.1 Extend or Steer Function

The purpose of the *extend* (or *steer*) function is to connect new states to the tree: using the notation of Chapter 2, it grows a branch from $\mathbf{x}_{near}$ toward $\mathbf{x}_{rand}$. The terminal state of the new branch, $\mathbf{x}_{new}$, may differ from $\mathbf{x}_{rand}$ depending on the extend function used (largely if motion primitives are used). $\mathbf{x}_{new}$ is then added to the tree τ together with the intermediate points of the new local path and the selected $\mathbf{u}$. The expansion fails if a collision along the path occurs.
We will now review previously used extend functions: *motion primitives* (LaValle

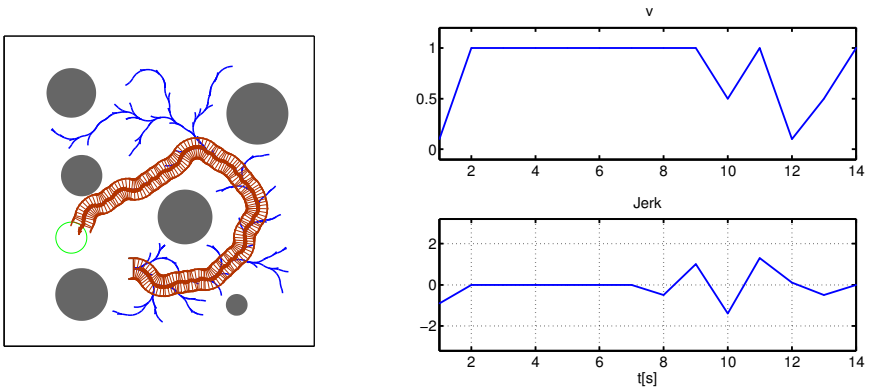

Figure 3.1: **Left:** Example RRT tree (in **blue**) and path (in **ruby**) generated using motion primitives. **Right:** Example velocity and jerk profiles generated by using the motion primitives approach. The profiles show how the concatenation of primitives leads to discontinuous movements.

and Kuffner, 1999; Frazzoli et al., 2005; Kalisiak and van de Panne, 2006, 2007) detailed in Sec.3.2.1.1, and steer functions described in Sec.3.2.1.2 (Perez et al., 2012; Webb and van den Berg, 2013; Hwan J et al., 2011; Yang et al., 2014; Kuwata et al., 2009; Park, 2016).

3.2.1.1 Motion primitives

Motion primitives have originally been proposed for RRT-based planning under differential constraints. LaValle and Kuffner (1999) implement the extend function as a forward simulation of a set of predefined discretized controls, so called motion primitives. The approach satisfies the constraints, is efficient to compute and easy to implement: the tree is extended with the primitive that is found to come closest to the new sampled configuration $\mathbf{x}_{rand}$. Given an initial state $\mathbf{x}_0$, an integration time Δt, an integration time step t_s and a input $\mathbf{u}_i$ from a discrete set of controls $\mathcal{U} = \{\mathbf{u}_1, .., \mathbf{u}_m\}$, a trajectory $\mathbf{x}_i(t)$ is generated by integrating Eq. (3.1)

$$\mathbf{x}_i(t) = \int_0^{\Delta t} f(\mathbf{x}_i(t), \mathbf{u}_i(t)) + g(\mathbf{x}(t))\, dt + \mathbf{x}_0, \quad i = 1, ..., m. \tag{3.3}$$

All the controls in $\mathcal{U}$ are checked, and the one that brings the expansion closest to $\mathbf{x}_{rand}$ according to some distance metric, is stored together with the associated local trajectory that will be added to the tree τ. To minimize the time needed to extend the tree, the motion primitives can be precomputed off-line. Although being used very often, it was recently showed by Kunz and Stilman (2014) that RRTs with fixed time integration and best-input extension are not probabilistically complete. Other ways to generate forward simulations (a.k.a motion primitives) exist: fixed integration time and random input, variable integration time

and best control input, variable integration time and random control input (Kunz and Stilman (2014)). According to Kunz and Stilman (2014), using the latter two approaches as extend function has not been proved to be probabilistically complete.

Motion primitives have several shortcomings, as this Chapter will show: they do not fully solve the two-point boundary value problem as, in the case of a wheeled mobile robot, the orientation of $\mathbf{x}_{rand}$ is ignored, the extension of the tree even by the closest motion primitive may still be far-off from $\mathbf{x}_{rand}$. Moreover the concatenation of primitives may lead to sequences of discontinuous inputs and trajectories with several heading discontinuities (see also Fig. 3.1). A first attempt to improve the smoothness and continuity of motion primitives concatenations was introduced by Frazzoli et al. (2005) who proposes a finite-state machine called a Maneuver Automaton to allow a correct concatenation of motion primitives to complex motion trajectories. The method still suffers of a sub-optimality gap induced by reducing feasible paths to the sequential combination of predefined motion primitives.

3.2.1.2 Solving the Two-Point Boundary Value Problem

An alternative method for extending the tree is to employ a full-fetched local planner that generates trajectories $\boldsymbol{\sigma} \in \mathcal{C}_{free}$ and the corresponding continuous controls $\mathbf{u}(t)$. This local planner also known as *steer function*, is called each time that RRT or RRT* attempt to connect two vertices. This approach is used to a great degree in RRT*, see Chapter 2. Next we detail some recent *steer* functions proposed for RRT and RRT*, namely *optimal controllers* (Perez et al., 2012; Webb and van den Berg, 2013), *shooting methods* (Hwan J et al., 2011), *splines* (Yang et al., 2014), and *closed-loop controllers* (Kuwata et al., 2009; Park, 2016):

- *optimal controllers*: Perez et al. (2012) use an optimal infinite-horizon LQR controller to connect pairs of states. The method linearizes the domain dynamics locally, which is interesting from an efficiency point of view, but will in general not reach the target state exactly. Webb and van den Berg (2013) use a finite-horizon optimal controller as local planner. They can optimize a certain class of cost functions to trade off between time and control effort. Although optimal control techniques may produce high-quality solutions to the two-point boundary value problem, they typically suffer from high computational costs and numerical issues that can make them unsuitable for motion planning in real-time.

- *shooting methods*: Hwan J et al. (2011) use a so called shooting method to numerically solve the two-point boundary value problem to obtain an extend function for RRT*. The method allows for time-optimal maneuvers of a high-speed off-road vehicle. As with optimal control techniques, shooting

methods may have issues with numerical stability and computational costs for our application.

- *splines*: In a recent work, Yang et al. (2014) use splines as RRT extend function. The authors take cubic Bézier splines that guarantee curvature continuity of paths and are able to satisfy upper-bounded curvature constraints. With our goal of smooth and natural motion generation, we consider splines to be a potentially interesting approach and include a spline-based extend function into our experimental comparison. However, instead of cubic Bézier splines which are limited to curves with continuous curvature, we will use η^3 splines, introduced by Piazzi et al. (2007) that produce curves with a continuous derivative of the curvature, therefore generating even smoother paths than cubic Bézier splines.

- *closed loop-predictions*: Kuwata et al. (2009) introduce closed-loop RRT (CL-RRT), a modified RRT for real-time local lane following with a car using an extend function based on a closed-loop model. Given a sampled control input, the method runs a forward simulation using the vehicle and controller models to predict and then evaluate extend trajectories. Park (2016) presents an approach to extend the RRT* tree for a differential drive robot. The method introduces a control-Lyapunov function to select and steer the robot between two poses during the RRT* iterations. Our approach resembles this method: although their method produces qualitatively interesting results, it is not compared to existing approaches in quantitative experiments as we do for our approach in the remaining part of the chapter.

3.3 The Approach: POSQ

In our approach instead of using motion primitives, we aim to connect two states during the RRT-RRT* search by fully solving the 2P-BVP. In place of using splines, shooting methods or optimal controllers, which generally require more computational effort or carry limitations due to linearization (as detailed in Sec. 3.2.1.2), we propose an extend function that computes closed-loop forward simulations based on the kinematic model of a nonholonomic wheeled mobile robot.

Our approach generates the trajectory $\boldsymbol{\sigma}$ and controls $\mathbf{u}(t)$, $t \in [0, T]$, $T > 0$, that connect any given pair of 2D poses. Thus, it solves the two-point boundary value problem for such kinematic systems, see Fig. 3.2. The tree is grown in the configuration space $\mathbb{R}^2 \times \mathbb{S}^1$ where each configuration $\mathbf{x}$ consists of the Cartesian position of the wheeled mobile robot and its orientation, i.e. (x, y, θ). The approach, originally proposed by Astolfi (1999), solves the problem of exponential stabilization of the kinematic and dynamic model of the wheeled mobile robot.

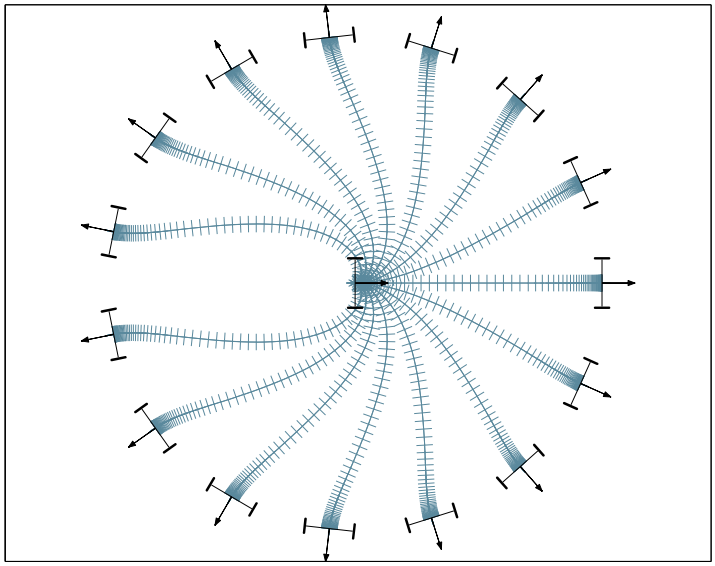

Figure 3.2: Paths of the controller when steering the robot from the center to the poses on the circle

It is not an optimal controller but has provable local and global stability, a lightweight implementation, and generates cusp-free trajectories. For extend functions, optimality may be less relevant than efficiency, smoothness and the ability to fully solve the two-point boundary value problem. This is particularly true for RRT* for nonholonomic dynamical systems (Karaman and Frazzoli, 2013), see Chapter 2.
We briefly summarize the original approach (Astolfi, 1999) and describe our extension in the next subsection.
Let ρ be the Euclidean distance between the initial pose and the goal pose ($\mathbf{x}_{near}$ and $\mathbf{x}_{rand}$ in an RRT notation), ϕ the angle between the x-axis of the robot reference frame $\{X_R\}$ and the x-axis of the goal pose frame $\{X_G\}$, α the angle between the y-axis of the robot reference frame and the vector Z connecting the robot with the goal position, v the translational and ω the angular robot velocity (Fig. 3.3). Then, the method makes a Cartesian-to-polar coordinate transform to describe the kinematics using the open loop model

$$\begin{aligned} \dot{\rho} &= -\cos\alpha\, v, \\ \dot{\alpha} &= \frac{\sin\alpha}{\rho} v - \omega, \\ \dot{\phi} &= -\omega. \end{aligned} \tag{3.4}$$

and the feedback law

$$\begin{aligned} v &= K_\rho \rho, \\ \omega &= K_\alpha \alpha + K_\phi \phi. \end{aligned} \tag{3.5}$$

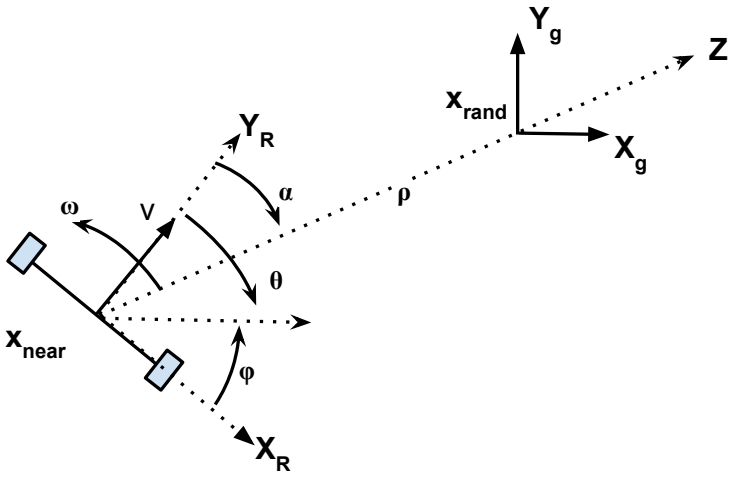

Figure 3.3: Notation and robot to goal relations

As Astolfi (1999) shows, this feedback law guarantees smooth trajectories without cusps.

3.3.1 Our kinematic control law

The original approach, however, generates trajectories of decaying forward velocity bringing the robot to a stop at each goal. The concatenation of such local paths would result in final paths of unnatural and slow movements. Thus, we modify the feedback law so as to have quasi constant forward velocity at a desired maximum value across multiple expansions. We will prove that this modification retains local stability and that the robot's heading converges asymptotically to the desired equilibrium point.
Considering the open loop model in Eq. (3.4) obtained by the polar coordinate transform, we define the non-linear feedback law

$$\begin{aligned} v &= K_\rho \tanh(K_v \rho), \\ \omega &= K_\alpha \alpha + K_\phi \phi \,. \end{aligned} \tag{3.6}$$

In case $\rho = 0$ and the robot's heading differs from the one requested, a turn on the spot is performed to align the robot's heading to latter. In the case start and goal poses to connect are the same, our steer function returns directly the start pose. Substituting the control law (3.6) into the open loop model we obtain the

following closed loop model

$$\begin{aligned}\dot{\rho} &= -K_\rho \cos\alpha \tanh(K_v\rho),\\ \dot{\alpha} &= K_\rho \frac{\sin\alpha}{\rho}\tanh(K_v\rho) - K_\alpha\alpha - K_\phi\phi,\\ \dot{\phi} &= -K_\alpha\alpha - K_\phi\phi\,.\end{aligned} \tag{3.7}$$

We now describe the conditions for which local stability holds and prove heading convergence.

3.3.1.1 Local Stability

We can locally approximate the closed loop model (3.7) by

$$\begin{aligned}\dot{\rho} &= -K_\rho K_v\rho,\\ \dot{\alpha} &= -(K_\alpha - K_\rho K_v)\alpha - K_\phi\phi,\\ \dot{\phi} &= -K_\alpha\alpha - K_\phi\phi.\end{aligned}$$

Proposition 1 (Local Stability)**.** *The local approximation of system in Eq.3.7 is locally exponentially stable if and only if the eigenvalues of the matrix describing the linear approximation of the model have all negative real parts. For that, we need to have*

$$\begin{aligned}K_v &> 0\\ K_\rho &> 0\\ K_\phi &< 0\\ K_\alpha + K_\phi - K_\rho K_v &> 0\,.\end{aligned} \tag{3.8}$$

Proposition 2 (Trapping Region)**.** *Considering the closed loop model (3.7), assume* $\alpha(0) \in D_1 =]-\frac{\pi}{2}, \frac{\pi}{2}]$*, and* $\phi(t) \in\ ]n\pi, n\pi]$ *for all* t*. Then, if*

$$K_\alpha + 2nK_\phi - \frac{2}{\pi}K_\rho K_v > 0$$

holds one has $\alpha(t) \in D_1 =]-\frac{\pi}{2}, \frac{\pi}{2}]$ *for all* $t > 0$ *which means that the robot trajectory will always stay in this region: we have defined a* trapping region*. Thus, together with the condition* $K_\rho K_v > 0$*, the robot will move monotonically towards the origin.*

3.3.1.2 Asymptotically Convergence

We want the robot to move towards to goal $\mathbf{x}_{rand}$ but notice in (3.7) that the goal position (the origin) can not be reached because ρ is a singularity point. Thus, we define an arbitrarily small number to which ρ converges, $\rho \to \epsilon$, with $\epsilon > 0$, $\rho > \epsilon$.

Let us focus on the following reduced subsystem which describes how the orientation evolves

$$\begin{aligned} \dot{\alpha} &= -K_\alpha \alpha - K_\phi \phi + K_\rho \sin\alpha \frac{\tanh(K_v \rho)}{\rho}, \\ \dot{\phi} &= K_\alpha \alpha - K_\phi \phi \,. \end{aligned} \tag{3.9}$$

Given that $\dot{\rho}$ is strictly negative, we want to find the conditions for which the above vector field has a unique equilibrium point ($\alpha = 0$, $\phi = 0$) to which all trajectories converge asymptotically for all $\rho > \epsilon$. This is equivalent to minimizing the orientation error as well as stopping the robot at $\mathbf{x}_{new}$, ϵ meters away from $\mathbf{x}_{rand}$.

Proposition 3. *The system in Eq.3.9 converges asymptotically to the origin.*

Proof. If we consider the candidate Lyapunov function

$$V(\alpha,\phi) \quad = \quad (-K_\alpha \alpha \; + \; K_\phi \phi)^2 \; + \; 2 \quad K_\phi K_\rho \quad (\cos\alpha \; - \; 1) \quad \tanh(K_v \rho), \tag{3.10}$$

we can show that

$$\dot{V}(\alpha,\phi) \;=\; 2 \; K_\phi K_\rho \alpha \;\; \sin\alpha \;\; \tanh(K_v \rho) \left[K_\phi + K_\alpha - K_\rho \frac{\sin\alpha}{\alpha} \;\; \tanh(K_v \rho) \right]. \tag{3.11}$$

Given that condition (3.8) holds, V is positive and $\dot{V}$ is non-positive in all $S_2 = \left\{ (\alpha,\phi) \in R^2 \mid \alpha \in \,]-\frac{\pi}{2}, \frac{\pi}{2}], \; \phi \in (-2\pi, 2\pi] \right\}$, so the state of the system converges asymptotically to the origin according to the LaSalle invariance principle.

It exists a positively invariant set

$$M = \left\{ (\alpha,\phi) \in S_2 | \;\; V \leq \frac{9}{4} k_\phi^2 \pi^2 - 2 K_\phi K_\rho \right\},$$

such that $M \subset S_2$, and $S_1 \subseteq M$ where

$$S_1 = \left\{ (\alpha,\phi) \in R^2 \mid \alpha \in \,]-\frac{\pi}{2}, \frac{\pi}{2}], \; \phi \in (-\pi, \pi] \right\}.$$

M contains only the equilibrium point ($\alpha = 0$, $\phi = 0$). Thus, all trajectories starting in S_1 and contained in S_2 converge asymptotically to the origin according to the Poincare-Bendixson Theorem. □

Notice that in the algorithm, we are not solving asymptotically the stabilization problem like in the original approach (Astolfi, 1999). The new control law allows us, during expansion of the tree from $\mathbf{x}_{near}$ towards $\mathbf{x}_{rand}$, to minimize the error in orientation and stop when the local trajectory is close enough to the goal $\mathbf{x}_{rand}$. A $\gamma > 0$ threshold can be defined as minimum Euclidean distance that stops the

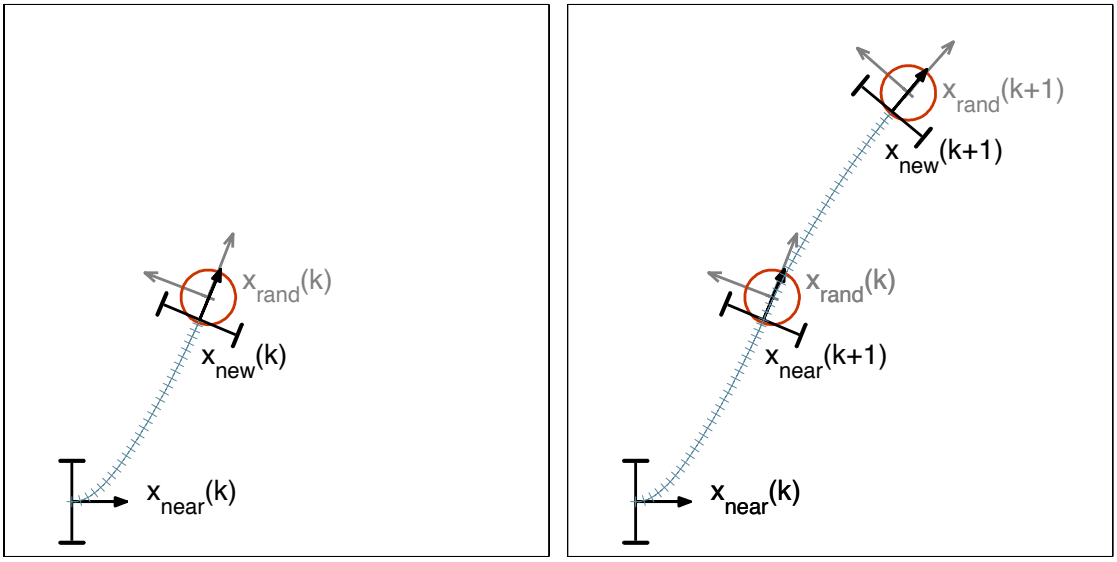

Figure 3.4: A simple two-step expansion example. With k being the time index of successive extensions, the proposed controller extends the tree from $\mathbf{x}_{near}(k)$ to $\mathbf{x}_{rand}(k)$ until the local trajectory enters the disk of radius γ at $\mathbf{x}_{new}(k)$. The procedure is repeated for $k+1$.

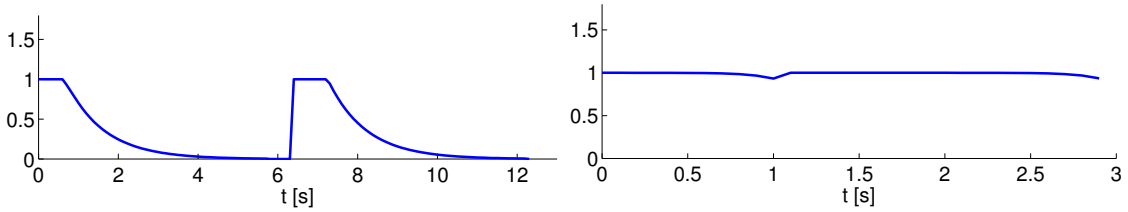

Figure 3.5: Translational robot velocity in [m/s] with the original control law from (Astolfi, 1999) (left) and our control law (right) across the concatenation of two extensions. For the same two goal poses the new law allows for much faster movements.

expansion towards $\mathbf{x}_{rand}$. It is guaranteed that the terminal state $\mathbf{x}_{new}$ is not further away from $\mathbf{x}_{rand}$ than γ, which thus becomes a tunable error bound. Threshold γ can be seen as the radius of a circle centred at $\mathbf{x}_{rand}$, see Fig. 3.4. In practice, γ is chosen to be a few centimeters.

The new control law does not remove the velocity decay toward the goal but makes it significantly sharper. So sharp, that even small values for γ cause the decay to disappear and allow for quasi-constant forward velocities along the previously explained extension procedure. See Fig. 3.5 for a comparison. The method is named *POSQ* as it acts like a pose controller. In App.B, we further detail a control law for a dynamic version of the differential drive system described by polar coordinates.

3.3.2 Topological Property

A key aspect for a steer function in RRT* is the fulfillment of the topological property (Sekhavat and Laumond, 1998), see App.A.2.
The POSQ steer function stabilizes the unicycle model, which is *small-time locally controllable* and *locally accessible*. The dimension of the involutive closure of the distribution associated to the unicycle system is equal to the degrees of the unicycle system (see Laumond et al. (1998)), so it satisfies the Chow Theorem and the Box-Ball Theorem (Montgomery, 2006). Let us define a metric d_{max} between any two paths $\boldsymbol{\sigma}_1(t)$ and $\boldsymbol{\sigma}_2(t)$, with $t \in [0, 1]$:

$$d_{max} = \max_{t=[0\ 1]} d\left(\boldsymbol{\sigma}_1(t), \boldsymbol{\sigma}_2(t)\right) \tag{3.12}$$

where d is an Euclidean metric on the configuration space $\mathcal{C}$. Our control law satisfies the topological property, it is able to connect two configurations whatever the distance between the initial and goal configuration is. Given that the conditions for local stability (Eq.3.8) and asymptotically convergences (Prop.3), the sufficient conditions of the topological property (see App.A.2) are satisfied:

- POSQ is continuous w.r.t. the topology associated with d_{max};
- the path generated by POSQ with the start $\mathbf{x}_{start}$ and goal $\mathbf{x}_{goal}$ poses being the same is reduced to the start pose $\mathbf{x}_{start}$.

Notice that the second condition is obviously a necessary condition. We perform planning considering as sampling space the entire free space, the Lie Hull of the distribution generated by the unicycle model spans the tangent space at every point of the space $\mathcal{C}$, so the the maximal integral manifold of the dynamics is equal to the entire space $\mathcal{C}$ (see Laumond et al. (1998)).

3.4 Experiments

In the experiments, we evaluate the new extend function and compare it to two alternative methods, namely motion primitives (two sets of different size) and splines. We quantify the impact of our steer function and of the two baselines on planning performance in terms of time, tree size and path quality in three different simulated environments. We use both RRT and RRT* as planning algorithms. Regarding our steer function, we use only the kinematic control law, because the dynamic version (detailed in App.B), although being still smooth, results in slightly more expensive planning time. The POSQ parameters are $K_\rho = 1$, $K_\phi = -1$, $K_\alpha = 6$, $K_v = 3.8$, $\gamma = 0.15$ (in case of RRT* experiments $\gamma = 0.05$). The two sets of motion primitives $\mathcal{U}_{small}$ with 10 controls and $\mathcal{U}_{large}$

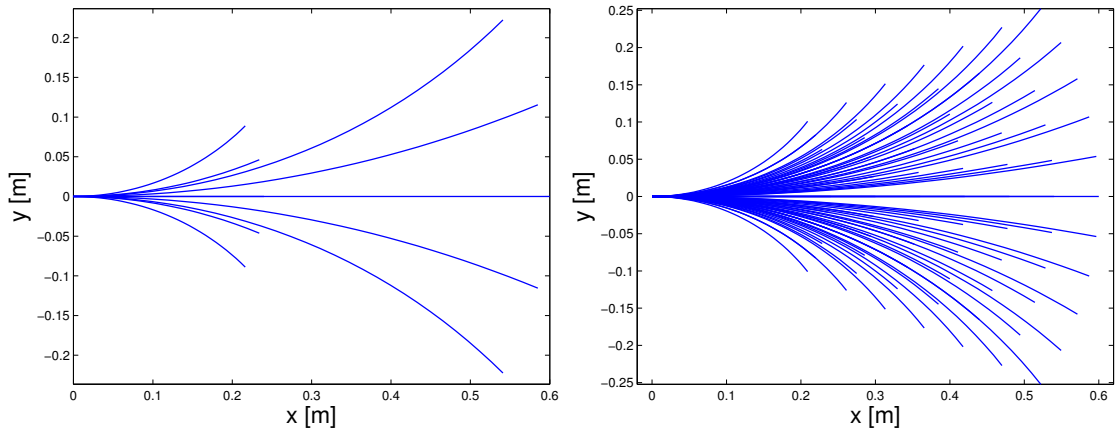

Figure 3.6: The motion primitive sets, $\mathcal{U}_{small}$ (left) with 10 motion primitives and $\mathcal{U}_{large}$ (right) with 77 motion primitives.

with 77 controls are shown in Fig. 3.6. For the spline-based extend function we use η^3 splines (Piazzi et al., 2007), seventh order polynomial spline whose paths have continuous tangent vectors, curvature and curvature derivatives along the arc length.

All the steer functions share the same integration time step t_s and velocity limits. For each combination of extend function/map/planning algorithm we perform 100 runs and compute the average and standard deviation of all metrics. We use uniform sampling and the Euclidean distance as distance metric.

Different cost functions may be used in RRT*: the algorithm formally needs to have a monotonic, Lipschitz continuous and positive real cost function Karaman and Frazzoli (2011). The function that we use, is derived from the work of LaValle and Kuffner (1999) and has two terms, one for the approximated path length and one that measures heading changes along the path, both with equal weights ($w_d = w_q$)

$$C = \sum_{i=0}^{N_e-1} w_d ||\mathbf{P}_{i+1} - \mathbf{P}_i|| + w_q \left(1 - |\mathbf{q}_{i+1} \cdot \mathbf{q}_i|\right)^2 .$$

$N_e + 1$ are the intermediate points $\mathbf{P}_i$ of the local path and $\mathbf{q}_i$ the associated quaternions.

The RRT* γ_{RRT^*} used to compute the neighborhood radius is constant at a high value with respect to the map size.

Our implementation is based on the C++ SMP template library (Karaman, 2011). All experiments were running on an ordinary PC with 2.67 GHz Intel Core i7 and 10 GB of RAM.

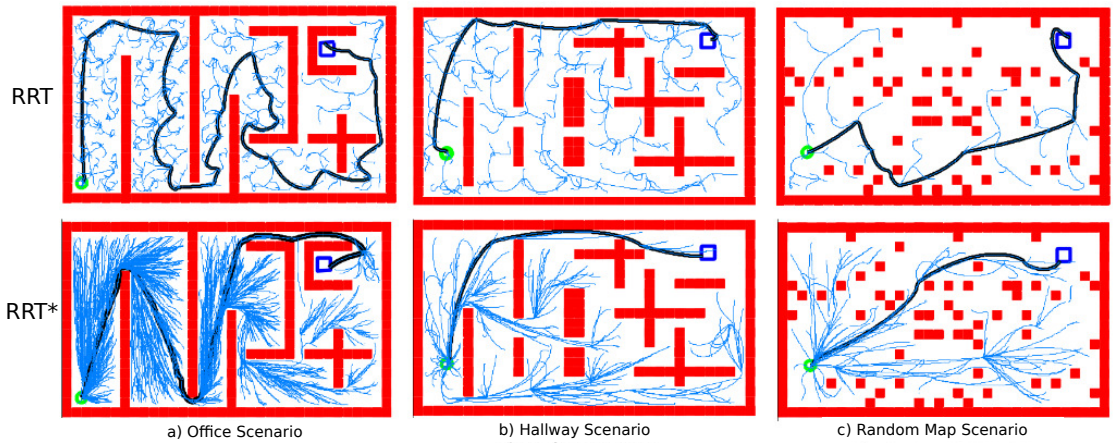

Figure 3.7: The three environments and example solutions found with the proposed extend function for RRT (top row) and RRT* (bottom row, showing the first solution). The start state (green circle) is always in the bottom left, the goal region (blue square) in the top right.

3.4.1 Metrics

To quantify planning performance we compute the averages and standard deviations of the following metrics: tree size as the number of vertices (N_v), time to find a solution (RRT) or a first solution (RRT*) (T_s), and path length in meters (l_p). Regarding smoothness we adopt some of the metrics described in Section 2.7, namely: η_{nmaJ}, the average of the mean absolute jerk normalized by the maximum velocity, average of the speed arc length η_{spal}, average number of peaks η_{pm}.

3.4.2 Test Environments

Planning is carried out in three simulated environments (Fig. 3.7). In the *office environment*, there are few alternative ways to the goal. It has several local minima, the goal lies behind a U-shaped obstacle, and an asymmetry makes that the shortest path go through a narrow passage. The *hallway scenario* contains more open spaces and alternative paths to the goal. The *random map environment* contains 100 randomly placed square obstacles. There are many homotopy classes, some require more or less maneuvers than others. The map size in all scenarios is $50m \times 30m$.

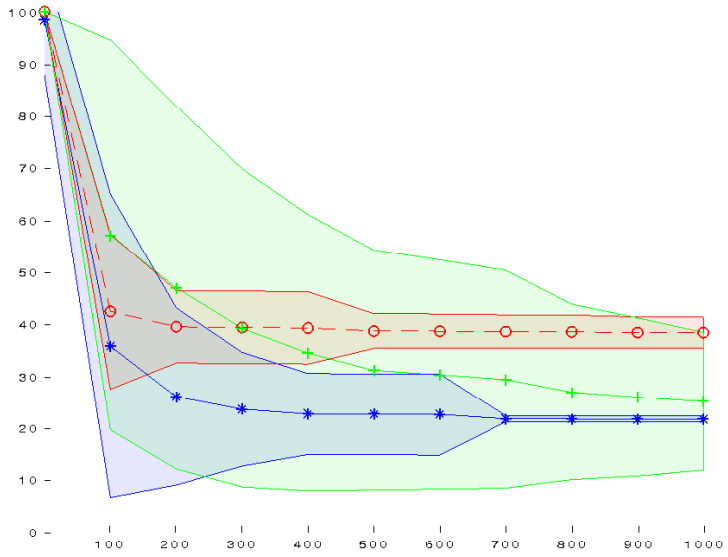

Figure 3.8: The RRT* cost C computed over 1000 seconds for the Random Map Scenario. The trends are displayed (mean and standard deviation): in **blue** with "$*$" as markers the POSQ results, in **red** with "**o**" as markers the motion primitive ones and in **green** with "**+**" as markers the splines.

3.5 Results and Discussion

The RRT results are given in Table 3.1-3.2, the RRT* results in Table 3.3-3.4. The best values in each metric are highlighted in bold. With RRT, the proposed extend function *POSQ* outperforms motion primitives and splines in all metrics except path length. It produces smoother paths and finds the goal in less time with significantly smaller trees. The low number of tree vertices and the smaller planning times are mainly due to the ability of our approach to better follow the Voronoi bias and deeply enter unexplored regions of the configuration space. This is unlike, for example, motion primitives that require the concatenation of many small local expansions for the same exploration effort. In fact, all continuous extend functions that fully solve the two-point boundary value problem possess this property as also confirmed by the similar trends in the results of the spline-based steer function. The motion primitive approach find shorter paths which is not surprising given the much denser trees from the multitude of small-sized extensions.

With RRT* as planner, our approach outperforms the other methods in tree size, path length and two of three smoothness measures. The fact that our method

finds the shortest paths, and so the lowest cost in all the cases, suggests that it is particularly easy to rewire in the sense of the cost function, quite in contrast to motion primitives. This is also pointed out by Webb and van den Berg (2013) who state that the RRT* rewiring procedure is well suited for continuous extension approaches where reachability of a state is not compromised. Figure 3.8 is another indication in this direction. It shows an example cost trend when given more planning time. POSQ can benefit most from the incremental path improvement of RRT*. While the proposed steer function is the smoothest in terms of the η_{spal} and η_{pm} measures, it falls behind the motion primitive approach in the jerk-related metric η_{nmaJ}. This may be explained by the much denser trees with several factors more vertices that allow solutions with fewer maneuvers. Regarding the time to find the first solution, T_s, the results are inconclusive. The high variance is mainly due to the large number of homotopy classes, particularly in the random map and hallway environments. POSQ and the spline-based approach (the two continuous approaches) grow the tree deeply into unexplored regions and discover many different ways to the goal, also inefficient ones at times. However, as discussed before, such first solutions can be improved when given more planning time, particularly well by POSQ.

Office scenario						
Extenders	$\mathbf{N_v}$		$\mathbf{T_s}$ $[s]$		$\mathbf{l_p}$ $[m]$	
POSQ	**1667**	$\pm$ 713	**0.197**	$\pm$ 0.09	150.739	$\pm$ 18.446
MP $\mathcal{U}_{small}$	13335	$\pm$ 3283.4	2.235	$\pm$ 0.597	134.108	$\pm$ 8.687
MP $\mathcal{U}_{large}$	14090	$\pm$ 3523.1	2.583	$\pm$ 0.720	**133.504**	$\pm$ 8.85
η^3 splines	2369	$\pm$ 939.9	0.274	$\pm$ 0.108	159.78	$\pm$ 14.88
Hallway scenario						
Extenders	$\mathbf{N_v}$		$\mathbf{T_s}$ $[s]$		$\mathbf{l_p}$ $[m]$	
POSQ	**520.4**	$\pm$ 379.2	**0.050**	$\pm$ 0.021	85.857	$\pm$ 16.740
MP $\mathcal{U}_{small}$	2458.3	$\pm$ 868.09	0.388	$\pm$ 0.124	72.918	$\pm$ 10.072
MP $\mathcal{U}_{large}$	2358.6	$\pm$ 922.2	0.367	$\pm$ 0.127	**71.734**	$\pm$ 9.254
η^3 splines	548.3	$\pm$ 514.5	0.0526	$\pm$ 0.026	86.659	$\pm$ 18.49
Random map scenario						
Extenders	$\mathbf{N_v}$		$\mathbf{T_s}$ $[s]$		$\mathbf{l_p}$ $[m]$	
POSQ	**277.2**	$\pm$ 351.5	**0.031**	$\pm$ 0.022	62.465	$\pm$ 9.003
MP $\mathcal{U}_{small}$	1095.1	$\pm$ 664.2	0.176	$\pm$ 0.104	**56.448**	$\pm$ 5.242
MP $\mathcal{U}_{large}$	1124.6	$\pm$ 646.4	0.168	$\pm$ 0.09	57.27	$\pm$ 5.3
η^3 splines	519.6	$\pm$ 718.6	0.044	$\pm$ 0.035	66.686	$\pm$ 9.514

Table 3.1: RRT results - planning efficiency, tree size, path length.

3.6 Conclusions

In this chapter we have presented a novel RRT extend function for nonholonomic wheeled mobile robots. We have evaluated its impact on planning performance

Office scenario						
Extenders	η_{nmaJ}		η_{spal}		η_{pm}	
POSQ	$-\mathbf{0.051}$	± 0.006	$-\mathbf{4.464}$	± 0.166	**0**	± 0
MP $\mathcal{U}_{small}$	-0.062	± 0.0006	-5.8648	± 0.064	37.8	± 6.7
MP $\mathcal{U}_{large}$	-0.062	± 0.0007	-5.8901	± 0.066	21.09	± 6.1
η^3 splines	-0.55	± 0.06	-6.88	± 0.16	**0**	± 0
Hallway scenario						
Extenders	η_{nmaJ}		η_{spal}		η_{pm}	
POSQ	$-\mathbf{0.039}$	± 0.007	$-\mathbf{3.602}$	± 0.282	**0**	± 0
MP $\mathcal{U}_{small}$	-0.0631	± 0.001	-5.237	± 0.138	22.32	± 5.7
MP $\mathcal{U}_{large}$	-0.0632	± 0.001	-5.2528	± 0.13	11.12	± 4.4
η^3 splines	-0.382	± 0.089	-5.93	± 0.359	**0**	± 0
Random map scenario						
Extenders	η_{nmaJ}		η_{spal}		η_{pm}	
POSQ	$-\mathbf{0.027}$	± 0.007	$-\mathbf{2.881}$	± 0.345	**0**	± 0
MP $\mathcal{U}_{small}$	-0.0638	± 0.001	-4.977	± 0.099	18.51	± 4.8
MP $\mathcal{U}_{large}$	-0.0637	± 0.001	-5.029	± 0.098	9.48	± 4.10
η^3 splines	-0.3013	± 0.082	-5.4851	± 0.337	**0**	± 0

Table 3.2: RRT results - smoothness.

Office scenario						
Extenders	$\mathbf{N_v}$		$\mathbf{T_s}$ $[s]$		$\mathbf{l_p}$ $[m]$	
POSQ	**1825**	± 785.7	**315.1**	± 187.3	**105.33**	± 4.96
MP $\mathcal{U}_{small}$	13571	± 3601.8	731.3	± 301.93	131.88	± 8.35
MP $\mathcal{U}_{large}$	14146	± 2562.3	933.5	± 258.50	134.257	± 8.849
η^3 splines	3438	± 4254.9	646.5	± 483.62	116.4909	± 6.337
Hallway scenario						
Extenders	$\mathbf{N_v}$		$\mathbf{T_s}$ $[s]$		$\mathbf{l_p}$ $[m]$	
POSQ	**697.9**	± 704	66.9	± 109.8	**54.16**	± 3.26
MP $\mathcal{U}_{small}$	2385.3	± 987	**51.3**	± 31.8	71.0350	± 9.5819
MP $\mathcal{U}_{large}$	2529.7	± 1020.4	68.4711	± 4.12	71.3390	± 8.4327
η^3 splines	3787.2	± 14583	637	± 2921	57.302	± 4.2401
Random map scenario						
Extenders	$\mathbf{N_v}$		$\mathbf{T_s}$ $[s]$		$\mathbf{l_p}$ $[m]$	
POSQ	**400.8**	± 551.2	43.16	± 90.54	**46.11**	± 2.4
MP $\mathcal{U}_{small}$	1028.1	± 597	**13.11**	± 12.48	56.66	± 5.08
MP $\mathcal{U}_{large}$	998.7	± 535.8	15.4411	± 14	56.9105	± 4.7867
η^3 splines	815.7	± 2421.1	157.8	± 715.8	48.5212	± 2.7370

Table 3.3: RRT* results - planning efficiency, tree size, path length.

Office scenario						
Extenders	η_{nmaJ}		η_{spal}		η_{pm}	
POSQ	-0.3261	± 0.135	$-\mathbf{5.128}$	± 0.29	**23.1**	± 11.1
MP $\mathcal{U}_{small}$	$-\mathbf{0.0622}$	± 0.001	-5.865	± 0.06	38.59	± 8.1
MP $\mathcal{U}_{large}$	-0.0623	± 0.001	-5.897	± 0.07	30.65	± 5.6
η^3 splines	-22.3	± 6.81	-8.49	± 0.09	47.30	± 27
Hallway scenario						
Extenders	η_{nmaJ}		η_{spal}		η_{pm}	
POSQ	-0.1430	± 0.126	$-\mathbf{3.6479}$	± 0.56	**3.1**	± 4.26
MP $\mathcal{U}_{small}$	-0.0631	± 0.0007	-5.212	± 0.129	16.2	± 5.2
MP $\mathcal{U}_{large}$	$-\mathbf{0.0630}$	± 0.0002	-5.2485	± 0.12	11.3	± 3.99
η^3 splines	-9.365	± 11.91	-7.08	± 0.59	12.9	± 13.9
Random map scenario						
Extenders	η_{nmaJ}		η_{spal}		η_{pm}	
POSQ	-0.0896	± 0.103	$-\mathbf{2.95}$	± 0.697	**1.3**	± 2.71
MP $\mathcal{U}_{small}$	$-\mathbf{0.0636}$	± 0.0007	-4.98	± 0.10	18.7	± 4.95
MP $\mathcal{U}_{large}$	-0.0637	± 0.0005	-5.02	± 0.08	10.09	± 4.2
η^3 splines	-7.67	± 11.87	-6.46	± 0.97	5.0	± 7.54

Table 3.4: RRT* results - smoothness.

and path quality and found that it outperforms motion primitives and a spline-based approach in many relevant metrics. It enables RRT to find smoother paths in less time with smaller trees, and it enables RRT* to find shorter paths with smaller trees while being on par in planning time and smoothness. We also found that our method can benefit most from the cost-guided rewiring procedure of RRT* resulting in the lowest cost solutions when given more planning time. Using a different cost function in RRT* obviously would result in different paths' shapes: in this thesis we do not investigate this aspect which could be studied in a future work.

CHAPTER 4

Distance Pseudo-Metric Learning for RRT Motion Planners with Constant-Time Inference

"With four parameters I can fit an elephant, and with five I can make him wiggle his trunk."

John von Neumann
as reported by Dyson (2004)

In Chapter 3, we have introduced the steer function POSQ, which efficiently solves the two-point boundary value problem for wheeled mobile robots, and showed that it improves the planning performance and the path quality of RRT and RRT planners. An other key component in RRT-based motion planning is the distance pseudo-metric: the latter deeply affects coverage of the state space, path quality and planning time. In this chapter, with the aim to speed up planning time without deteriorating path quality, we introduce a learning approach to approximate the distance pseudo-metric for RRT-based planners. By exploiting the smooth and efficient POSQ steer function, we train a simple non-linear parametric model with constant-time inference that is shown to predict distances accurately in terms of regression and ranking performance. In an extensive analysis we compare our approach to an Euclidean distance baseline, consider four alternative regression models and study the impact of domain-specific feature expansion. The learning approach is shown to be faster in planning time by several factors at negligible loss of path quality.*

4.1 Introduction

A key component in the extension of the tree in RRT is the distance pseudo-metric, or cost-to-go pseudo-metric, used to select the nearest vertex from where to grow the tree. In RRT* this function has an even more important role as it

guides the rewiring procedure. To do so, the pseudo-metric has to be computed as many times as there are vertices in the near-neighbour ball (Karaman and Frazzoli, 2011) or near-neighbour box (Karaman and Frazzoli, 2013).

For general kinodynamic systems, it is hard to determine the true cost-to-go function. It implies solving a two-point boundary value problem (2P-BVP), which may be as expensive as solving a motion planning query on its own or an optimal control problem. This is why, in the original RRT paper, LaValle and Kuffner (1999) suggest to use approximations of the optimal cost-to-go as functions of path length, difference between initial and final orientation, and translational and rotational velocities. Cheng and LaValle (2001) shows that such a sub-optimal distance metric enables a motion planner to entirely cover the configuration space and to solve hard planning problems.

In this chapter we introduce a novel method to approximate the cost-to-go metric by a simple, offline-learned regression model with constant-time inference. We consider differential drive robots in the configuration space $\mathbb{R}^2 \times \mathbb{S}^1$, although the same approach can be extended to systems with higher dimensions. Our distance metric estimates the cost of local paths from the novel extender POSQ which solves the 2P-BVP and produces smooth cusp-free trajectories (Palmieri and Arras, 2014a). As described in Chapter 3, POSQ is able to connect any pair of $\mathbb{R}^2 \times \mathbb{S}^1$ poses and produces RRT trees that quickly cover the entire state space and reach the goal region. The latter is not true for forward propagation approaches of discretized controls (motion primitives) as also discussed by Glassman and Tedrake (2010). Furthermore, the POSQ extender makes no linearization or approximation, is efficient to compute and was shown to produce smoother paths in shorter time with smaller trees than motion primitives and a spline-based extender approach (Palmieri and Arras, 2014a). In this chapter, with the goal of making the planner even more efficient, we make the following contributions (published in (Palmieri and Arras, 2014b) and (Palmieri and Arras, 2015)):

- We show how the distance pseudo-metric for the case of the POSQ extender can be learned offline using a set of domain-specific features and a simple basis function model with constant-time inference. In addition of being very fast to compute, the learned model is accurate in terms of regression and ranking performance at negligible loss of path quality.

- We present a comprehensive comparison to an Euclidean distance baseline and four alternative regression models namely neural network regression, LWPR, SVM regression, and random forest regression and analyze the impact of domain-specific feature expansion.

- The comparison demonstrates that the Euclidean distance – although fast to compute – is highly inaccurate in terms of ranking and regression and leads to paths of poor quality and length. The experiments also show that

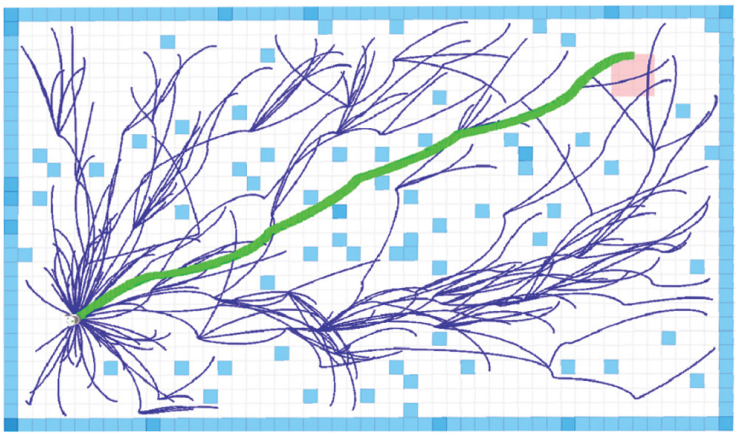

Figure 4.1: An example tree and path generated with our learned distance metric. The robot starts at the bottom-left and plans a path to the goal region marked in red. The first-solution path is shown in green.

our learning approach is able to cover the state space in the same way than the ground truth function.

The chapter is structured as follows: Section 4.2 details the related work, in Section 4.3 we describe how the distance metric is learned. Experiments and their results are described in Section 4.4. Section 4.5 concludes the chapter.

4.2 Related Work

A number of different distance pseudo-metrics have been used in previous research (Amato et al., 2000; Kuffner, 2004; Glassman and Tedrake, 2010; Perez et al., 2012; Webb and van den Berg, 2013; Li and Bekris, 2011).
Variants of the Euclidean distance have been studied by Amato et al. (2000); Kuffner (2004). Amato et al. (2000) compare several distance metrics defined in the configuration space and show that for nonholonomic systems, the weighted Euclidean distance – commonly used for holonomic systems – is unable to correctly cover the space. They give recommendations on how to select a metric based on efficiency and effectiveness.
Kuffner (2004) defines a proper distance metric for the configuration space of a 3D rigid body, the Lie group $\mathbb{SE}(3)$. The author proposes a weighted metric with a translation component that uses a standard Euclidean norm and a scalar function that returns an approximate measure of the distance between the initial and final orientation.
Distance metrics have been derived by linearizing the system dynamics in (Glassman and Tedrake, 2010; Perez et al., 2012; Webb and van den Berg, 2013). Glassman and Tedrake (2010) describe how the Voronoi bias of RRT only applies when

a proper metric is defined for the case of an extend function that forward simulates a dynamical system. They use an affine quadratic regulator design and show that it can be used to approximate the exact minimum-time distance pseudo-metric at a reasonable computational cost.
Perez et al. (2012) use an optimal infinite-horizon LQR controller to connect pairs of states. The method linearizes the domain dynamics locally. In this case the cost-to-go pseudo-metric is defined as the solution of the Riccati's equation used in the LQR extender. Webb and van den Berg (2013) use a finite-horizon optimal controller as local planner. They can optimize a certain class of cost functions that trades off time and control effort. A drawback of these linearization methods is that the approximations are valid only as long as the linearization is valid. When non-linearities increase, the accuracy of the metric degrades. They may also suffer from high computational costs and numerical issues.
Li and Bekris (2011) approximate the optimal cost-to-go pseudo-metric by an off-line learning method: the distance between two states is approximated by the cost of an A* path between their closest sampled states on a learned graph. The graph is generated off-line by using forward propagation of the system dynamics. For speeding up the method, they map the offline samples into a higher-dimensional Euclidean space. The method makes approximations on two levels: the graph is built using a discretized set of controls, not solving the 2P-BVP, and the mapping of the samples which compromises the method's ability to cover the state space.
Varricchio et al. (2016) propose a novel k-d tree build and query strategies associated to sub-Riemannian metrics. The authors demonstrate significant improvements in the running time of nearest-neighbor search queries when compared to k-d tree structures that consider Euclidean distance. Differently from our approach, this method avoids an expensive linear search, nevertheless for high-dimensional systems, computing boxes (and their weights, see ball-box theorem from Appendix A) that approximate the reachability space of the system can be very difficult to compute a priori.
A recent idea, which has been developed independently by three research groups at the same time (Palmieri and Arras, 2014b; Bharatheesha et al., 2014; Allen et al., 2014), is to approximate the distance pseudo-metric by supervised learning of a non-linear regression model.
Bharatheesha et al. (2014) approximate the optimal cost-to-go using locally weighted projection regression (LWPR). The optimal cost-to-go is obtained by iteratively solving a linear quadratic regulator problem where non-linear dynamics are still linearized around a nominal trajectory instead of a point. The learning approach is incremental which makes that cost prediction scales with the increasing number of nodes in the RRT tree.
Allen et al. (2014) use locally weighted linear regression (LWR) to predict optimal cost-limited reachable sets of dynamical systems in real-time and a binary SVM classifier to learn the non-linear boundary between a state's reachable and non-

reachable set. While achieving good regression accuracy, inference times scale quadratically with the number of LWR training samples and linearly with the number of SVM support vectors, respectively.

4.3 Our Approach

In RRT-based planning, a tree is grown by connecting randomly sampled configurations $\mathbf{x}_{rand}$ to their nearest vertex $\mathbf{x}_{near}$ in the tree. For the selection of $\mathbf{x}_{near}$, the algorithm evaluates the distances from all or a subset of tree vertices to $\mathbf{x}_{rand}$.
The evaluation of this distance metric is a frequent operation deep within every RRT algorithm and a speed up at this point would have a strong impact onto planning times.
The idea is, instead of computing an extension path and then evaluating its cost, to learn a parametric regression model that directly predicts the cost. This is fast to compute and we can expect a speed up even though the forward simulation by POSQ (presented in Chapter 3) is already efficient to implement.
Formally, we have a regression model

$$y \approx g(\mathcal{X}, \boldsymbol{\beta}) \tag{4.1}$$

with $\mathcal{X}$ being the set of independent variables (features or attributes) and $\boldsymbol{\beta}$ the parameters of g.
In this section, we first define the class of distance metrics considered here, design a set of features, choose a regression model and a learning algorithm to fit its parameters.

4.3.1 The distance metric

Following the work by LaValle and Kuffner (1999), we consider a class of distance metrics $C(\mathbf{x}_1, \mathbf{x}_2)$ defined as a linear combination of path length and sum of heading changes between states $\mathbf{x}_1$ and $\mathbf{x}_2$,

$$C(\mathbf{x}_1, \mathbf{x}_2) = \sum_{i=0}^{N_e-1} w_d ||\mathbf{P}_{i+1} - \mathbf{P}_i|| + w_q \left(1 - |\mathbf{q}_{i+1} \cdot \mathbf{q}_i|\right)^2 . \tag{4.2}$$

$\mathbf{P}_i$ are the $N_e + 1$ intermediate points of the path and $\mathbf{q}_i$ the associated quaternions. The time needed to compute this cost expression depends on the distance between $\mathbf{x}_1$ and $\mathbf{x}_2$ and the integration time step Δt (leading to more or fewer intermediate points $\mathbf{P}_i$). Fig. 4.2 shows the cost distribution for paths generated by the POSQ controller.

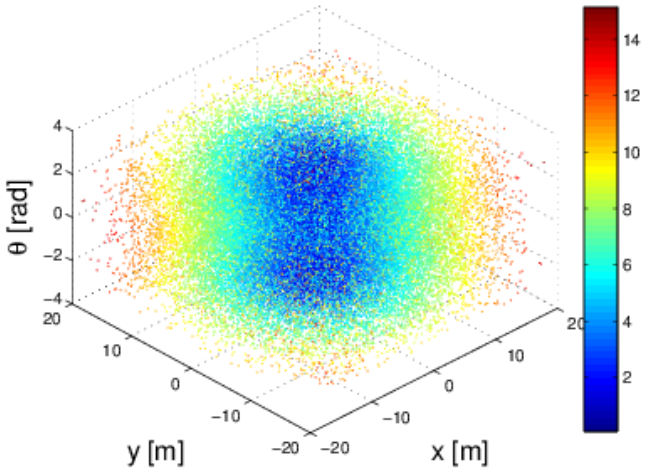

Figure 4.2: The cost-to-go function $C(\mathbf{0}, \mathbf{x})$ for paths generated by the POSQ controller.

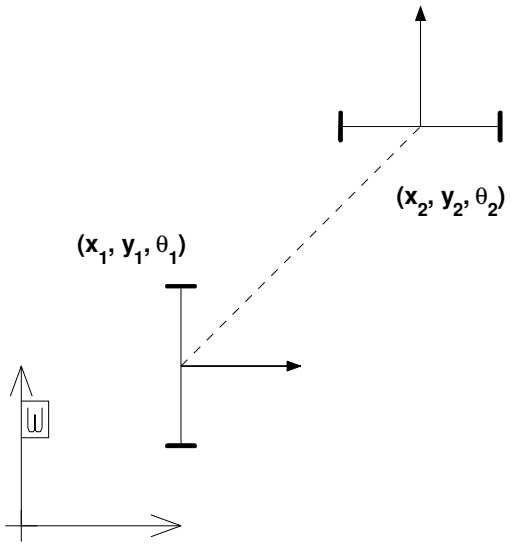

Figure 4.3: Example two robot poses used for features computation. Both poses refer to the same world reference frame **W**.

4.3.2 Features

Let the set of independent variables $\mathcal{X}$ be the vector of features $\mathbf{f}$ that we define in this section. Naively, we could directly use the inputs of the extend function, the two poses $\mathbf{x}_1$ and $\mathbf{x}_2$, as features $\mathbf{f} = (x_1, x_2, y_1, y_2, \theta_1, \theta_2)$ since they fully define the problem. However, this choice encodes the relevant information only very implicitly. We expect interesting interactions between those features which we

Description	**Expression**
Displacement in x	$\Delta x = x_2 - x_1$
Displacement in y	$\Delta y = y_2 - y_1$
Displacement in θ	$\Delta\theta = \theta_2 - \theta_1$
Euclidean distance between poses	$d = \|\mathbf{x}_2 - \mathbf{x}_1\|$
x-projection of the orientation change	$\cos\Delta\theta$
y-projection of the orientation change	$\sin\Delta\theta$
Orientation change multiplied by Euclidean distance	$d\,\Delta\theta$
x-projection of the orientation change multiplied by Euclidean distance	$d\,\cos\Delta\theta$
y-projection of the orientation change multiplied by Euclidean distance	$d\,\sin\Delta\theta$
Angular difference between $\mathbf{x}_1$ and connecting line of the two poses	$\text{atan}\frac{\Delta y}{\Delta x} - \theta_1$
Angular difference between $\mathbf{x}_2$ and connecting line of the two poses	$\text{atan}\frac{\Delta y}{\Delta x} - \theta_2$
Ratio between the previous two features	$\frac{\text{atan}(\Delta y/\Delta x)-\theta_1}{\text{atan}(\Delta y/\Delta x)-\theta_2}$
Angular difference between $\mathbf{x}_1$ and connecting line multiplied by Euclidean dist.	$d\,(\text{atan}\frac{\Delta y}{\Delta x} - \theta_1)$
Angular difference between $\mathbf{x}_2$ and connecting line multiplied by Euclidean dist.	$d\,(\text{atan}\frac{\Delta y}{\Delta x} - \theta_2)$

Table 4.1: Input features computed considering two different robot poses (x_1, y_1, θ_1), (x_2, y_2, θ_2) (see Fig. 4.3).

seek to make explicit by performing feature expansion to obtain more meaningful inputs. But instead of an uninformed, generic method such as quadratic expansion or kernel methods, we can take advantage of our domain knowledge to capture those interactions. For example, it is obvious that the Euclidean distance $d_E(\mathbf{x}, \mathbf{y}) = \sqrt{(y_2 - y_1)^2 + (x_2 - x_1)^2}$ will be a dominant feature for predicting the cost of paths that connect $\mathbf{x}_1$ and $\mathbf{x}_2$. Finally, in multiple validation runs, we have found a set of fourteen features to characterize the cost-to-go function, see Table 4.1. The features make the geometry of POSQ paths under the cost model Eq. 4.2 more explicit and, as will be shown in the experiments, facilitate the learning process.

4.3.3 Learning

We choose a basis function model (BFM) for learning to predict path costs, fitted to the training set $\mathcal{S} = \{\mathbf{s}_i\}_{i=1}^{N}$ using Levenberg-Marquardt. The model is defined as

$$y = \sum_{m=1}^{M} \Phi_m(\mathbf{f}, \boldsymbol{\beta}) \tag{4.3}$$

where M is the number of basis functions Φ. Given our initial goal of speeding up planning time, this choice appears promising for its simplicity and constant-time inference, independent on the number of training samples. Concretely, we choose quadratic basis functions given by

$$y = \sum_{m=1}^{M_f} \beta_{m_1}(f_m - \beta_{m_2})^2 \tag{4.4}$$

where M_f is the number of features.

Training samples $\mathbf{s}_i = [\mathbf{f}_i,\ c_i]$ are pairs of feature vectors and ground truth costs. Here, we randomly generate pose pairs $(\mathbf{x}_1, \mathbf{x}_2)$ in the configuration space, compute their 14-dimensional feature vector $\mathbf{f}_i$ and determine the corresponding ground truth cost from the POSQ extend function, $c_i = C(\mathbf{x}_1, \mathbf{x}_2)$.

4.4 Experiments and Results

In the experiments we compare our approach to an Euclidean distance baseline, consider four alternative regression models and study the impact of the domain-specific feature expansion described above. We perform two sets of experiments, first we evaluate the prediction accuracy of the different methods in terms of regression and ranking metrics, and second, we analyze how the learned distance metrics impacts planning time, path quality, and state space coverage. Concretely, we consider

- The proposed basis function model with the fourteen domain-specific features (BFM) in Table 4.1.
- The basis function model with naive features $\mathbf{f} = (x_1, x_2, y_1, y_2, \theta_1, \theta_2)$ (BFM naive).
- A neural network model (NN) with two hidden layers, 30 neurons in the first and 20 in the second one, has been trained with the back-propagation algorithm Werbos (1989). The architecture has been found through 5-fold cross validation.

- A random forest regression model (Rnd Forest) (Breiman, 2001) with 100 trees each with 5 terminal leaves. Both hyperparameters have been found through cross validation and provide a good trade off between prediction time and accuracy.

- A ν-SVR model (Schölkopf et al., 2000) with a RBF kernel $Q_{ij} = \exp(-\gamma||\mathbf{x}_i - \mathbf{x}_j||^2)$ with parameters $C = 1e^{-8}$ and $\gamma = 1e^{-3}$ which have been found via cross validation.

- A locally weighted projection regression (LWPR, Vijayakumar and Schaal (2000)) model with Gaussian kernels and eight receptive fields. Initial parameters and hyperparameter have again been found through cross validation.

- We also consider the Euclidean distance (Eucl. Dist.), the most commonly used distance metric, as an approximation of the true cost.

4.4.1 Regression and Ranking Performance

To evaluate the prediction accuracy of the learned regression models, we use the following metrics: median of the residuals, mean squared error normalized by the residuals' variance (NMSE), and the coefficient of determination. We also determine the average runtime t_{pred} of a single prediction.

Note that although we framed our task as a regression problem it is actually a *learning-to-rank problem*. When searching the tree for the nearest state $\mathbf{x}_{near}$ given $\mathbf{x}_{rand}$, we are actually interested in the correct ranking of the tree vertices under the cost model rather than the predicted costs as such. The typical strategy is then to choose the best ranked (lowest cost) vertex as $\mathbf{x}_{near}$. Therefore, we also evaluate the model with respect to its ability to correctly rank a set of states and use the following ranking metrics: Kendall τ coefficient, Kendall τ_d distance and Spearman ρ coefficient (Spearman, 2010). Kendall τ and Spearman ρ coefficients are both correlation measures between two-ordinal level variables equal to 1 if the two rankings agree perfectly and equal to -1 if they disagree perfectly. Kendall τ_d distance measures the number of disagreements between two ordered lists equal to 0 if the two ranks are equal. Here, we consider the ranking of the five best vertices.

We learn all models with the same 50,000 training samples and validate with 10,000 samples in terms of regression performance. For ranking, we predict the cost and evaluate the metrics for a grid of poses over the entire configuration space without obstacles with a resolution of $0.1m$ in x, y and $\pi/4$ rad in θ.

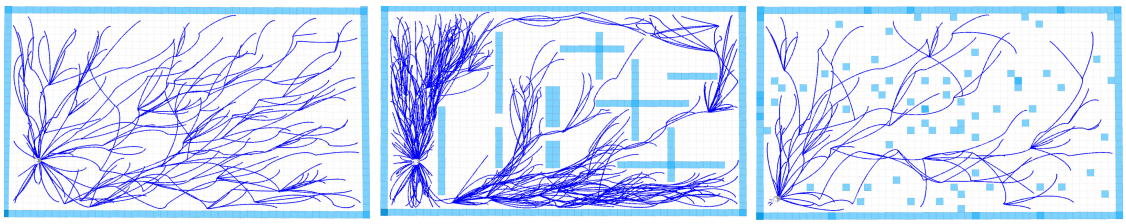

Figure 4.4: The three environments and example trees obtained with our approach. **Left:** *open space* scenario. **Center:** *hallway* scenario. **Right:** *random map* scenario

4.4.2 Regression and Ranking Results

The results for the regression and ranking metrics are reported in the Table 4.2. The poor performance of the naive features approach clearly demonstrates the necessity to design informative features for this learning task. Also the Euclidean distance fails to approximate the true distance metric accurately in terms of both regression and ranking error. Furthermore, the results show the relation between a model's ranking and regression performance: less accurate regression does not prevent perfect ranking (of the best five states). Among the four methods with perfect ranking results, the BFM approach is a clear winner with two orders of magnitude better runtime performance. The worst model in this sense is ν-SVR where inference time scales with the number of support vectors which in our experiments exceeds 16,000.

4.4.3 Planning Performance

We now investigate how the learned regression model impacts planning time as well as path quality and – further below – coverage ability.
To this end, we compare the learned distance metric with two ground truth baselines, the second best model in the previous experiment (NN), and, due to its common usage, the Euclidean distance.
The baselines are the ground truth cost of the POSQ extender with the regular high-resolution integration time of $\Delta t = 0.1$ *sec* (POSQ 0.1) and a lower integration time of $\Delta t = 0.5$ *sec* (POSQ 0.5). The latter is to analyze the performance of a faster but "rougher" version of POSQ with fewer path points. For the sake of a fair comparison the Euclidean distance uses a *k-d* tree data structure to speed up nearest neighbor search.
We consider three simulated test environments (Fig. 4.4) that stress different properties of a planner. The *open space* scenario has no obstacles, it serves to study the planner's behaviour when the tree can grow freely. The *hallway* scenario contains many open space areas, alternative paths to the goal and local minima. The *random map* scenario contains 100 randomly placed square obstacles. There are

many homotopy classes, some require more or less maneuvers along paths than others. The map size in all scenarios is $50m \times 30m$.
To quantify planning performance we compute the averages of the following metrics: time for a single extension ($\mathbf{t}_{ext}$), time to find a solution ($\mathbf{T}_{path}$), and path length in meters ($\mathbf{l}_{path}$).
Regarding smoothness also in this case, we adopt some of the metrics described in Section 2.7, namely: η_{nmaJ}, the average of the mean absolute jerk normalized by the maximum velocity, average of the speed arc length η_{spal}, average number of peaks η_{pm}.

For each environment and method we perform 100 runs and compute the average of all metrics. We use uniform sampling in the entire state space. All experiments were carried out in a C++ implementation on a single core of an regular laptop with 2.70 GHz Intel i5 and 12 GB RAM.

4.4.4 Planning Performance Results

The results, given in Tables 4.3-4.4, show the expected speed advantage of the Euclidean distance over all other methods but also that this cost metric produces by far the longest and least smooth paths. Among the learned distance metrics, the BFM model generally improves both runtime metrics by several factors at a negligible loss of path smoothness. This remains true even for the "rough" version of the POSQ extender, POSQ 0.5.
The speed up in planning time of the learned distance metric is most dramatic in the *open space* scenario. This is because without obstacles, the other RRT heuristics that influence tree growth (extend function, collision checking and random state generation) have no effect and the improvement is fully visible.
In the *hallway* scenario, the BFM approach is still able to find a solution more than twice as fast while in the cluttered *random map* environment, the learning method is on par with the POSQ extender. The reason is that in cluttered environments a lot of time is spent for collisions checking and short extensions for which the acceleration by the learning approach is less visible. Path smoothness remains largely unaffected, the values are all within the same order of magnitude than the original approach.

4.4.5 State Space Coverage

Different distance metrics may lead to different state space coverage behaviors (Glassman and Tedrake, 2010). We thus compare the ability of our learning approach to cover the state space with the previous four methods in the *random map* scenario.
To this end, we divide the entire state space into a grid of 3D cells and determine state space coverage as the ratio of grid cells covered by the tree. We perform 100

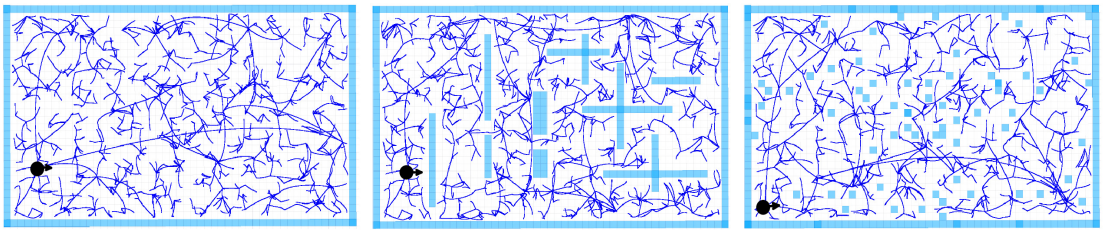

Figure 4.5: Example trees obtained using the Euclidean distance in the three scenarios. The initial pose of the robot is shown by a black dot. Although state space coverage is good, the planner generates trees that lead to poor solutions in terms of path length and quality.

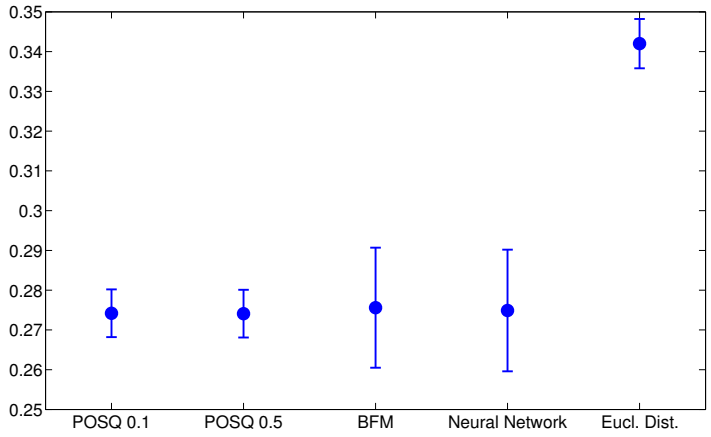

Figure 4.6: Means and standard deviations of the state space coverage metric after 5000 iterations.

runs of 5,000 iterations for each method.
The results are shown in the Fig. 4.6. The Euclidean distance is best at covering the state space but it does so because it fails to approximate the true cost and picks incorrect (quasi random) nearest vertices. The resulting trees lead to poor solutions in terms of path length and quality (see Fig. 4.5 for example trees). Thanks to its good approximation abilities, the learned BFM-based model distance metric does not degrade in terms of state space coverage. On average, its trees are able to cover the same amount of grid cells than the baseline method.

4.5 Conclusions

In this chapter, we have presented a new learning approach to approximate the distance pseudo-metric (or cost-to-go metric) for RRT-based motion planning for

wheeled mobile robots in $\mathcal{C} = \mathcal{R}^2 \times \mathcal{S}^1$. Instead of computing local extension paths and then evaluating their cost when growing the tree, we learn a parametric regression model that directly predicts the cost. In a comparison with four alternative regression models we could show that a simple basis function model with constant-time inference is the best choice in terms of regression and ranking accuracy as well as planning time and path quality. The resulting speed up in planning time is significant particularly in less cluttered environments. Using a set of domain-specific features, we have also demonstrated the need to design informative features for this learning task.
Despite good results for planning time and state space coverage, our experiments have shown the Euclidean distance to be a poor choice for the approximation of the true cost with respect to regression and ranking accuracy, and consequently, path quality and length.

This approach and partly those results generalize to different 2P-BVP solvers. This is particularly true, in terms of planning time speed-up, for complex solvers, i.e. in the case of optimal controllers or high dimensional spaces, which are in general slow to compute trajectories between different states.

Regression Performance

Metric	**BFM**	**BFM naive**	**NN**	**Rnd Forest**	ν**-SVR**	**LWPR**	**Eucl. Dist.**
Median res.	0.030	7.8376	**1.607 e^{-6}**	0.0054987	0.000261	0.6055	0.167
NMSE	0.005	0.8843	**7.729 e^{-7}**	0.0011037	0.028991	0.0806	0.0118
Determination	0.999	0.8040	**1**	0.99976	0.99349	0.9821	0.9895

Runtime Performance

Metric	**BFM**	**BFM naive**	**NN**	**Rnd Forest**	ν**-SVR**	**LWPR**	**Eucl. Dist.**
t_{pred} [sec]	6.22 e^{-07}	4.09 e^{-07}	1.24 e^{-05}	6.81 e^{-05}	0.0012	1.87 e^{-05}	**3.853 e^{-07}**

Ranking Performance

Metric	**BFM**	**BFM naive**	**NN**	**Rnd Forest**	ν**-SVR**	**LWPR**	**Eucl. Dist.**
τ	**1**	-0.2	**1**	**1**	**1**	0.40	-0.40
τ_d	**0**	0.6	**0**	**0**	**0**	0.30	0.70
ρ	**1**	-0.3	**1**	**1**	**1**	0.40	-0.50

Table 4.2: Regression and ranking performance

Open space scenario			
Method	$\mathbf{t}_{ext}\ [s]$	$\mathbf{T}_{path}\ [s]$	$\mathbf{l}_{path}\ [m]$
POSQ 0.1	0.01019	14.5215	47.6487
POSQ 0.5	0.006093	5.3704	50.4071
BFM	0.001325	1.0902	48.608
NN	0.003556	2.6722	48.2047
Eucl. Dist.	0.001353	0.1417	56.0124
Hallway scenario			
Method	$\mathbf{t}_{ext}\ [s]$	$\mathbf{T}_{path}\ [s]$	$\mathbf{l}_{path}\ [m]$
POSQ 0.1	0.01082	92.9419	54.6069
POSQ 0.5	0.007975	70.6947	56.0821
BFM	0.00179	26.2349	61.5938
NN	0.005318	65.2463	63.3869
Eucl. Dist.	0.0007944	1.4498	83.0162
Random map scenario			
Method	$\mathbf{t}_{ext}\ [s]$	$\mathbf{T}_{path}\ [s]$	$\mathbf{l}_{path}\ [m]$
POSQ 0.1	0.01180	37.7666	49.8373
POSQ 0.5	0.01089	20.88	50.129
BFM	0.003976	23.3264	53.2168
NN	0.003783	19.452	53.5202
Eucl. Dist.	0.001503	0.6197	61.7909

Table 4.3: Results for planning efficiency and path length

Open space scenario			
Method	η_{nmaJ}	η_{spal}	η_{pm}
POSQ 0.1	$-7.32075\ e^{-05}$	-0.802622	0.62
POSQ 0.5	$-6.77003\ e^{-05}$	-0.73929	0.38
BFM	$-6.44915\ e^{-05}$	-0.759564	0.31
NN	$-6.71913\ e^{-05}$	-0.777919	0.55
Eucl. Dist.	-0.0034127	-2.28714	0.36
Hallway scenario			
Method	η_{nmaJ}	η_{spal}	η_{pm}
POSQ 0.1	$-7.44230\ e^{-05}$	-0.844397	0.90
POSQ 0.5	$-8.01868\ e^{-05}$	-0.8833	3.23
BFM	$-4.28115\ e^{-05}$	-0.9543	0.42
NN	$-7.22698\ e^{-05}$	-1.01336	0.39
Eucl. Dist.	-0.00705978	-3.36954	0.52
Random map scenario			
Method	η_{nmaJ}	η_{spal}	η_{pm}
POSQ 0.1	$-6.04633\ e^{-05}$	-0.794841	0.45
POSQ 0.5	$-6.12429\ e^{-05}$	-0.795457	0.68
BFM	$-6.85253\ e^{-05}$	-0.844749	3.89
NN	$-6.98598\ e^{-05}$	-0.865452	0.33
Eucl. Dist.	-0.00487451	-2.76645	2.63

Table 4.4: Smoothness results

CHAPTER 5

Theta*-RRT: Any-angle Path Biasing for RRT Nonholonomic Motion Planning

Geometry does not teach us to draw these lines, but requires them to be drawn; for it requires that the learner should first be taught to describe these accurately, before he enters upon geometry; then it shows how by these operations problems may be solved

Isaac Newton
Philosophiae Naturalis
Principia Mathematica

In the previous chapters we have introduced improvements for two key functions in RRT and RRT based planning, namely a novel steer function and a learning-based method to compute the distance pseudo-metric. An other key routine in RRT and RRT* is the sampling unit. With the goal of further improving the efficiency and the path quality of the single-query planners detailed in the previous chapters, we introduce a new technique to generate informed samples for RRT-based planning. RRT and RRT*'s planning times can scale poorly for high-dimensional systems such as wheeled robots with complex nonholonomic constraints, especially if samples are drawn blindly and in an uninformed way in the entire configuration space. This has motivated researchers to study hierarchical techniques that grow the RRT trees in more focused ways. Along this line, we introduce Theta*-RRT that hierarchically combines (discrete) any-angle search with (continuous) RRT motion planning for nonholonomic wheeled robots. Theta*-RRT is a variant of RRT that generates a trajectory by expanding a tree of geodesics toward sampled states whose distribution summarizes geometric information of the any-angle path computed considering only the workspace. We show experimentally, for both a differential drive system and a high-dimensional truck-and-trailer sys-*

tem, that Theta-RRT finds shorter trajectories significantly faster than four baseline planners (RRT, A*-RRT, RRT*, A*-RRT*) without loss of smoothness, while A*-RRT* and RRT* (and thus also Informed RRT*) fail to generate a first trajectory sufficiently fast for systems with complex nonholonomic constraints. We also prove that Theta*- RRT retains the probabilistic completeness of RRT for all small-time controllable systems that use an analytical steer function.*

5.1 Introduction

Any-angle search is a family of discrete search techniques which, unlike A* or Dijkstra's algorithm, find paths that are not constrained to grid edges. Daniel et al. (2010) introduce Theta*, an any-angle search technique whose paths are only slightly longer than true shortest paths. The authors show that the basic variant of Theta* finds shorter paths than Field D*, A* with post smoothing and A* on grids, see Fig. 2.2.
To improve the performance of sampling-based motion planners, recent research has combined them with discrete search techniques (Plaku et al., 2007, 2010; Bekris and Kavraki, 2008; Brunner et al., 2013) that generate paths considering only workspace information. None of these studies, however, combine any-angle search with RRT variants although its properties (such as finding shorter paths than A* with fewer heading changes) are likely beneficial for the performance of the combination.
Following the approaches detailed in Chapters 3-4, in this chapter with the goal of further improving the efficiency and the path quality of the single-query planners, we introduce Theta*-RRT (Palmieri et al., 2016) a new technique to generate informed samples for RRT-based planning. Theta*-RRT is a hierarchical technique that combines (discrete) any-angle search with (continuous) RRT motion planning. It improves the efficiency of RRT in high-dimensional spaces substantially by transferring the properties of the any-angle path to the final trajectory. Theta*-RRT considers a continuous control space during planning: it uses steer functions instead of random control propagations to exploit as much knowledge of the nonholonomic constraints of the system as possible and to ensure both high planning efficiency and high trajectory quality. Since heuristics can also be misleading and degrade planning performance, we prove that Theta*-RRT retains the probabilistic completeness of RRT for all small-time controllable systems that use an analytical steer function. We evaluate the approach using a 3D differential drive robot and a 8D truck-and-trailer system and four baseline planners: RRT, RRT* (and thus also Informed RRT* (Gammell et al., 2014) which behaves like RRT* until a first trajectory is found), A*-RRT, and A*-RRT* (Brunner et al., 2013). The evaluation shows that Theta*-RRT is significantly faster and produces shorter

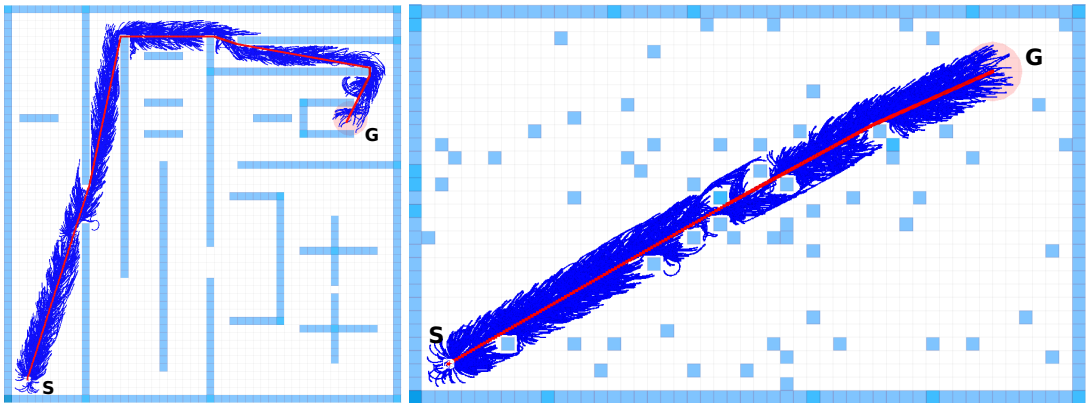

Figure 5.1: Theta*-RRT trees in two example environments used in the experiments. **Left:** *Maze* environment. **Right**: *Random map* environment. The trees (in **blue**) grow smoothly towards the goal in a subspace centered around the any-angle path (in **red**).

high-quality trajectories than those of the baselines.
The chapter is structured as follows: We describe related work in Sec. 5.2-5.3 and Theta*-RRT in Sec. 5.4. We present experiments in Sec. 5.5 and discuss their results in Sec. 5.6. Probabilistic completeness of Theta*-RRT is proven in Sec. 5.7.

5.2 Sampling Measure

Sampling configurations is a key routine in RRT-based planning. Hsu et al. (2007) detail the importance of the sampling unit in sampling-based motion planning (the authors detail its properties for PRM but the same can be fairly extended for single-query based algorithms e.g. RRT). The same authors define a sampling strategy as:

Definition 9 (Sampling Strategy (Hsu et al., 2007))**.** *A sampling strategy is specified as a pair* (π, S)*, where:*

- π *is a probability measure that prescribes how sampled configurations are distributed over the configuration space* $\mathcal{C}$
- S *is is a source of uniformly distributed, random or deterministic numbers.*

π is often defined, in context of sampling-based motion planner as a uniform distribution. Several techniques have been introduced to generate good non-uniform sampling distributions that improve the overall planner performance (i.e. path quality and length and planning efficiency, see for the work by Şucan and Kavraki (2010) for a review).

Hsu et al. (2007) give an intuition on how important is the sampling unit. Quoting the authors:

> *"Suppose that while constructing a roadmap, the planner could maintain a representation (H, η), where H is the set of all hypotheses over the shape of $\mathcal{C}_{free}$ and η is a probability measure that assigns to each hypothesis in H the probability of it being correct. Then, in each iteration of the planner, the optimal sampling measure π^* would be the one that minimizes the expected number of remaining iterations until the roadmap connects $\mathbf{x}_{start}$ and $\mathbf{x}_{goal}$, whenever these two configurations lie in the same connected component of $\mathcal{C}_{free}$. In principle, π^* could be inferred from (H, η). In practice, maintaining (H, η) explicitly would be too expensive. So, existing PRM planners use heuristics to approximate the optimal sampling measure."*

With Theta*-RRT we go into this direction and find a sampling heuristic that would allow the planner to efficiently generate high quality solutions.

5.3 Related Work

Prior research, like in Theta*-RRT, has combined discrete search with continuous sampling-based motion planning to approximate the optimal sampling measure. For example, Plaku et al. (2007, 2010) propose a planner where a search-based planner finds a sequence of decomposition regions, into the workspace, that are then used to guide how RRT grows the tree into the configuration space. Bekris and Kavraki propose the Informed Subdivision Tree technique (Bekris and Kavraki, 2008) that uses a heuristic to direct the tree growth and improve the coverage of the state space. In contrast to these two planners, Theta*-RRT biases the tree growth most likely in the homotopy class found by Theta* and considers a continuous control space (by utilizing steer functions instead of a discrete set of randomly generated control propagations) to exploit as much knowledge of the nonholonomic constraints as possible.

Brunner et al. (2013) propose a two-phase motion planner where A* finds a geometrically feasible path, which then biases the tree growth of RRT*. This planner is applied only to a high-dimensional holonomic robot, where the RRT* vertices (sampled from a Gaussian distribution centered around the A* path) are connected using motion interpolation. In contrast, Theta*-RRT focuses on more complex nonholonomic systems and uses steer functions.

Cowlagi and Tsiotras (2012) propose a planner that constructs a discrete control set using expensive model-predictive control techniques. In contrast, Theta*-RRT adopts a continuous control space.

Rickert et al. (2014) propose the EET planner for holonomic systems that sacrifices probabilistic completeness by using workspace information to continuously

adjust the sampling behavior of the planner. In contrast, Theta*-RRT is probabilistically complete.

Moreover our approach takes inspiration from the *two level planning approach* detailed by Sekhavat et al. (1997); Laumond et al. (1998); Sekhavat et al. (1998), where the nonholonomic motion planning problem for small-time controllable systems is solved by approximating an initial geometric feasible path into a kinematically feasible one, by using repeatedly a steer function to connect subdivisions of the initial path. In our approach, differently from the latter, the geometric path is used to bias the sampling-unit and we do not try to connect directly points on the geometric path: this operation may result in paths with unexpected cups.

5.4 Combining Any-Angle Search with RRT

Let $\mathcal{C} \subset \mathbb{R}^d$ be the state space, $\mathcal{U} \subset \mathbb{R}^m$ the control space, and $\mathcal{C}_{obs} \subset \mathcal{C}$ and $\mathcal{C}_{free} = \mathcal{C} \setminus \mathcal{C}_{obs}$ the obstacle and free spaces, respectively. A nonholonomic (control) system Σ on state space $\mathcal{C}$ is a differential system such that

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t))\, u + g(\mathbf{x}(t)) \qquad \mathbf{x}(0) = \mathbf{x}_{start}, \tag{5.1}$$

where $\mathbf{x}_{start} \in \mathcal{C}$ and, for all t, $\mathbf{x}(t) \in \mathcal{C}$ and $\mathbf{u}(t) \in \mathcal{U}$. g describes the drift, and f describes the system dynamics (for more details on the properties of nonholonomic systems see App.A). Theta*-RRT is a feasible motion planner for small-time controllable nonholonomic systems: It finds controls $\mathbf{u}(t) \in \mathcal{U}$ for $t \in [0, T]$ such that the unique trajectory $\boldsymbol{\sigma}$ that satisfies Equation (5.1) connects a given start state $\mathbf{x}_{start} \in \mathcal{C}_{free}$ to a given goal state $\mathbf{x}(T) = \mathbf{x}_{goal} \in \mathcal{C}_{goal} \subset \mathcal{C}_{free}$ in the free space $\mathcal{C}_{free}$.

5.4.1 Geodesic Distance for Nonholonomic Wheeled Robots

Let us consider small-time controllable nonholonomic systems.

Definition 10. *System Σ is locally controllable from $\mathcal{C}$ if the set of states reachable from $\mathcal{C}$ by an admissible trajectory contains a neighborhood of $\mathcal{C}$. It is small-time controllable from $\mathcal{C}$ if, for any time T, the set of states reachable from $\mathcal{C}$ before time T contains a neighborhood of $\mathcal{C}$.*

For small-time controllable nonholonomic wheeled robots, we define the geodesic distance $D_{\mathbf{P}}(\mathbf{x}_1, \mathbf{x}_2)$ of two states $\mathbf{x}_1$ and $\mathbf{x}_2$ to a path $\mathbf{P}$ through $\mathbb{R}^2 \times \mathbb{S}^1$. Consider a path $\mathbf{P}$ and let $\mathbf{x}'_1$ and $\mathbf{x}'_2$ be the orthogonal projections of $\mathbf{x}_1$ and $\mathbf{x}_2$ onto $\mathbf{P}$ and their Euclidean distances be $d_1 = \|\mathbf{x}_1 - \mathbf{x}'_1\|$ and $d_2 = \|\mathbf{x}_2 - \mathbf{x}'_2\|$ (respectively). Then, the geodesic distance $D_{\mathbf{P}}(\mathbf{x}_1, \mathbf{x}_2)$ is the sum of the lengths of the

geodesics from each of the two states to path $\mathbf{P}$, that is,

$$\begin{aligned} D_{\mathbf{P}}(\mathbf{x}_1, \mathbf{x}_2) &= w_e\,(d_1 + d_2) + w_\theta \left(1 - |\mathbf{q}_{\mathbf{x}_1} \cdot \mathbf{q}_{\mathbf{x}'_1}|\right) \\ &\quad + w_\theta \left(1 - |\mathbf{q}_{\mathbf{x}_2} \cdot \mathbf{q}_{\mathbf{x}'_2}|\right) \end{aligned}$$

(for parameters w_e and w_θ), where $\mathbf{q}_{\mathbf{x}_1}$ and $\mathbf{q}_{\mathbf{x}_2}$ are the quaternions of states $\mathbf{x}_1$ and $\mathbf{x}_2$, and $\mathbf{q}_{\mathbf{x}'_1}$ and $\mathbf{q}_{\mathbf{x}'_2}$ the quaternions of the segments of path $\mathbf{P}$ to which $\mathbf{x}'_1$ and $\mathbf{x}'_2$ belong. The geodesic distance of two states is the smaller, the closer they are to path $\mathbf{P}$ in Euclidean distance, heading orientations and steering orientations.

5.4.2 Our Technique: Theta*-RRT

Theta*-RRT (detailed in Algorithm 6) first generates a geometrically feasible any-angle path $\mathbf{P}$ using only geometric information about the workspace. Then, it computes the trajectory by growing a tree τ of smooth local geodesics around path $\mathbf{P}$ (path-biasing heuristic) satisfying the system's nonholonomic constraints.

Algorithm 6 *Theta*-RRT*

function `Theta*-RRT`($\mathbf{x}_{start}$, $\mathbf{x}_{goal}$)
$\mathbf{P} \Leftarrow$ `AnyAngleSearch`($\mathbf{x}_{start}$, $\mathbf{x}_{goal}$)
if $\mathbf{P} = \emptyset$ **then**
  return *failure*
end if
τ.`AddNode`($\mathbf{x}_{start}$)
$g(\mathbf{x}_{start}) \Leftarrow 0$
$k \Leftarrow 1$
while $k \leq$ `MAX_ITERATIONS` **do**
  $\mathbf{x}_{rand} \Leftarrow$ `AnyAngleSampling`($\mathcal{C}, \mathbf{P}$)
  $\mathbf{x}_{near} \Leftarrow$ `NearestNeighborSearch`($\tau, \mathbf{x}_{rand}, \mathbf{P}$)
  $\mathbf{u}_{new}, \boldsymbol{\sigma}_{new} \Leftarrow$ `Steer`($\mathbf{x}_{near}, \mathbf{x}_{rand}$)
  if $\boldsymbol{\sigma}_{new} \in \mathcal{C}_{obs}$ **then**
    `continue`
  end if
  τ.`AddNode`($\mathbf{x}_{rand}$)
  τ.`AddEdge`($\mathbf{x}_{near}, \mathbf{x}_{rand}, \mathbf{u}_{new}$)
  $g(\mathbf{x}_{rand}) \Leftarrow g(\mathbf{x}_{near}) + C(\mathbf{x}_{near}, \mathbf{x}_{rand})$
  if $\mathbf{x}_{rand} \in \mathcal{C}_{goal}$ **then**
    return `ExtractTrajectory`($\mathbf{x}_{rand}$)
  end if
  $k \Leftarrow k + 1$
end while
return *failure*

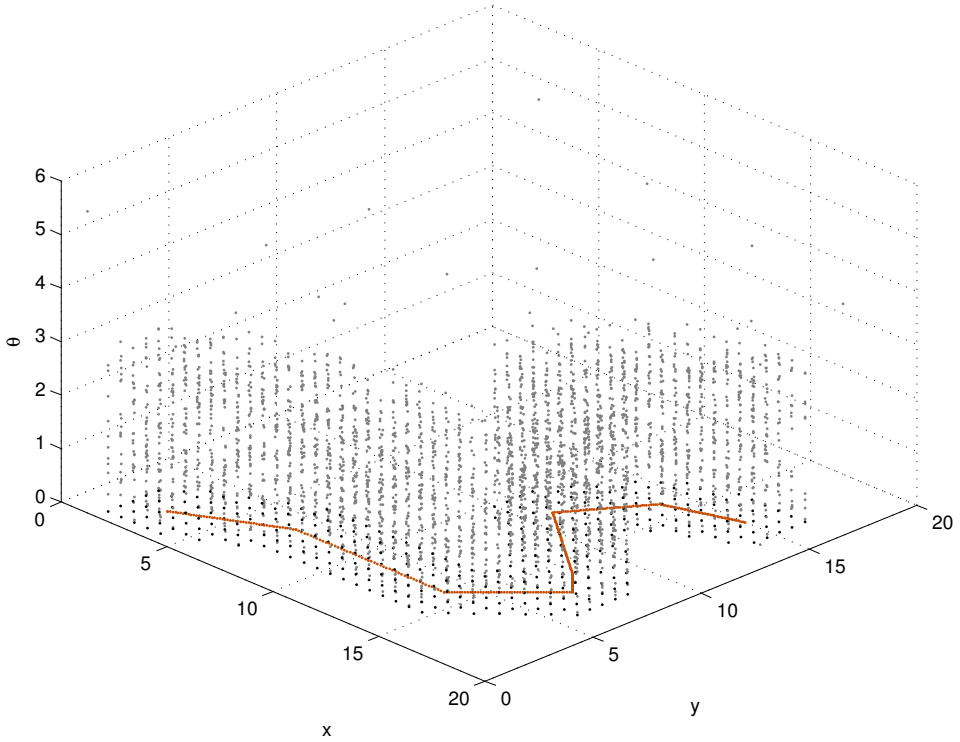

Figure 5.2: Example samples obtained by the path-biasing any-angle sampling strategy for a differential drive system, with configurations (x, y, θ). The samples (**grey** dots) are most likely generated around an any-angle path (2D projection in **orange**) which has been computed considering only the workspace information (xy plane).

The probability measure $\pi_{Theta*-RRT}$ of its sampling strategy, repeatedly samples a state $\mathbf{x}_{rand}$ mainly from a subspace $\mathcal{C}_{local} \subset \mathcal{C}_{free}$ centered around path $\mathbf{P}$, see Fig. 5.2-5.3: in this phase the workspace information, i.e. the any-angle path and its heading changes, informs the algorithm while guiding the tree through the remaining part of the configuration space. The algorithm then makes $\mathbf{x}_{rand}$ a new tree vertex and connects it to $\mathbf{x}_{near}$, which is selected among several ones as the vertex that connects with minimum cost to $\mathbf{x}_{rand}$. The cost depends on the length and smoothness of the trajectory from the candidate tree vertex to state $\mathbf{x}_{rand}$ and the geodesic distance of both vertices to the any-angle path.

The subroutines of Algorithm 6 are described below:

`AnyAngleSearch`($\mathbf{x}_{start}$, $\mathbf{x}_{goal}$) uses Basic Theta*, detailed in Alg.1-2, to search an eight-neighbor grid from start grid vertex s_{start} to goal grid vertex s_{goal}, where S is the set of all grid vertices. $s_{start} \in S$ is the grid vertex that corresponds to the start vertex $\mathbf{x}_{start}$, and $s_{goal} \in S$ is the grid vertex that corresponds to the goal vertex $\mathbf{x}_{goal}$. We assume obstacles cells to be inflated so as to reflect the robot

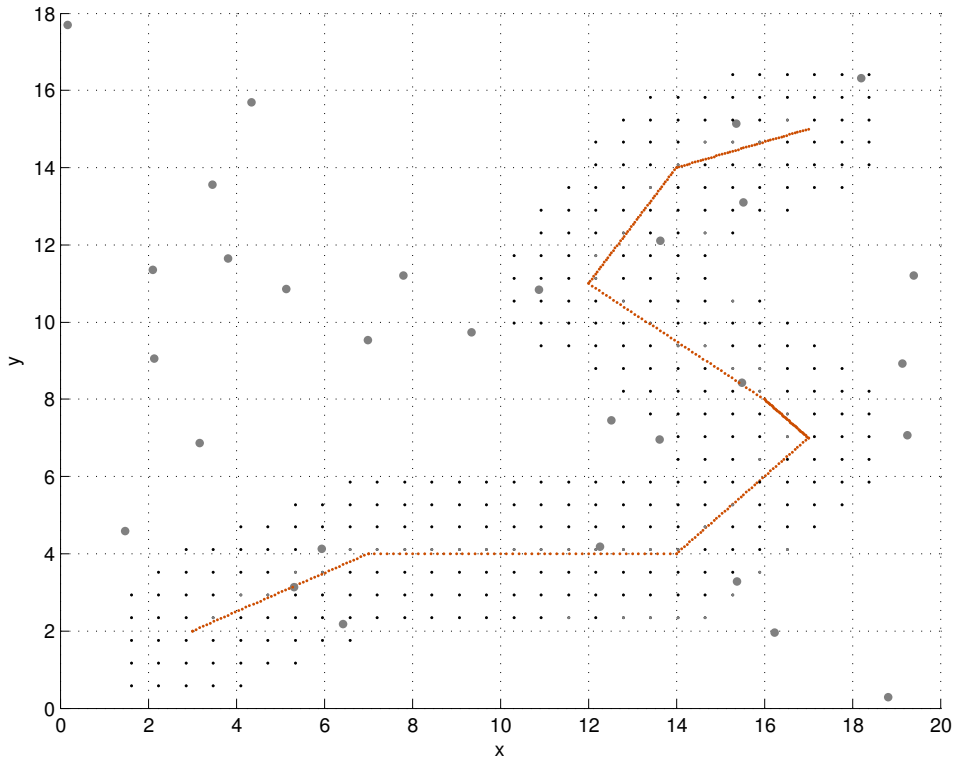

Figure 5.3: 2D-Projection of the example samples showed in Fig. 5.2. The samples (**grey** dots) are most likely generated around an any-angle path (2D projection in **orange**) which has been computed considering only the workspace information (xy plane): a fraction of the samples (bigger **grey** dots) have been generated also in the remaining part of the configuration space.

shape. Theta* uses the consistent straight-line distances as heuristics. It returns an any-angle path $\mathbf{P} = \{\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_N\}$ (a discrete Cartesian path) if one exists and the empty path otherwise.

`AnyAngleSampling`$(\mathcal{C}, \mathbf{P})$ samples mainly from a connected subspace $\mathcal{C}_{local} \subset \mathcal{C}$ according to a distribution $\pi_{Theta*-RRT}$ that conveys geometric information of path $\mathbf{P}$ and returns the sampled state $\mathbf{x}_{rand}$. Concretely, the Cartesian components of the samples are generated uniformly from a strip with width W (for parameter W, called position bias) centered around path $\mathbf{P}$. To convert the any-angle path into a smooth trajectory, the heading orientation x_θ and steering orientation x_δ of the samples are generated uniformly from angular intervals centered around a mean orientation $\bar{\gamma}$, which is a linear combination of the orientations of the

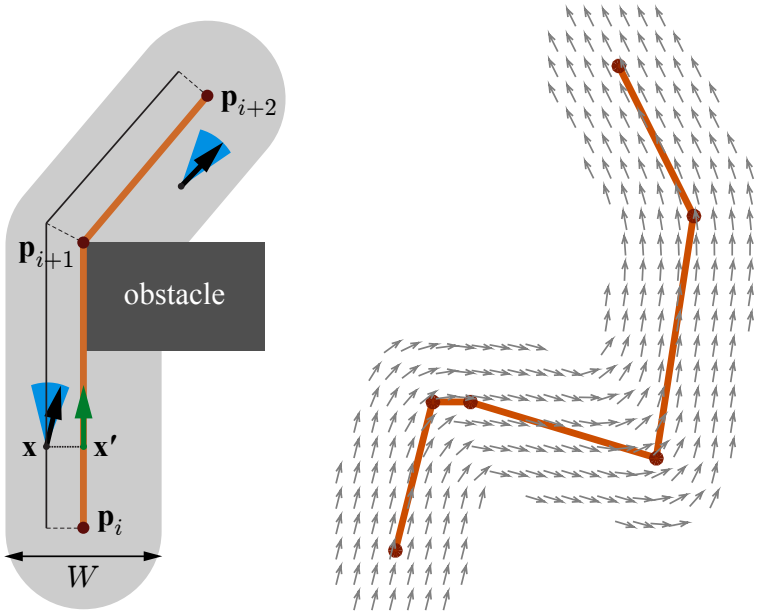

Figure 5.4: Path-biased sampling strategy. **Left:** Example strip (in **grey**) around an any-angle path (in **orange**), in which samples are randomly generated. **Black** arrows are samples, and **green** arrows are their projections onto the any-angle path. **Blue** sectors are the angular ranges from which the sample orientations are drawn. The mean sector orientations are computed as weighted averages of the orientations of the any-angle path segments. The individual weight contributions are evaluated in geodesic path coordinates along the offset black line. **Right:** Resulting mean orientations around an example any-angle path.

segments of path **P**, that is,

$$\bar{\gamma} = \sum_{i=1}^{N} w_i \, \gamma_{\mathbf{p}}^i, \tag{5.2}$$

where $\gamma_{\mathbf{p}}^i$ is the orientation of segment $\overline{\mathbf{p}_i\mathbf{p}_{i+1}}$. The weights w_i are calculated from trapezoidal membership functions that are associated with each segment. The functions are centered around the centers of their segments with tails that overlap into the neighboring segments such that their values at the path vertices $\mathbf{p}_i$ are exactly 0.5 and their slopes are no less than a minimal slope δ_S (for parameter δ_S). The influence of each membership function on a given sample $\mathbf{x}$ is computed along geodesic path coordinates, obtained by offsetting path **P** with the perpendicular distance of $\mathbf{x}$ to **P** (see Fig. 5.4, left). The orientations x_θ and x_δ of the samples are then generated uniformly from the interval $(\bar{\gamma} - \Delta\theta, \bar{\gamma} + \Delta\theta)$ (for parameter $\Delta\theta$, called orientation bias). The components of the samples that are

not related to the workspace (such as velocities and accelerations) are generated uniformly. Moreover with a frequency $f_{uniform}$, this function generates uniformly a sample from the entire $\mathcal{C}_{free}$ (for parameter $f_{uniform}$).

`NearestNeighborSearch`(τ, $\mathbf{x}_{rand}$, $\mathbf{P}$) returns the tree vertex $\mathbf{x}_{near}$ that connects with minimum cost $C(\mathbf{x}_{near}, \mathbf{x}_{rand})$ to state $\mathbf{x}_{rand}$. Instead of determining tree vertex $\mathbf{x}_{near}$ directly, Theta*-RRT determines a set of tree vertices $\mathcal{C}_{near}$ within distance δ_R from $\mathbf{x}_{rand}$ (for parameter δ_R). If this set is empty, it returns the tree vertex nearest to $\mathbf{x}_{rand}$. Otherwise, it returns the tree vertex from set $\mathcal{C}_{near}$ that connects with minimum cost $C(\mathbf{x}_{near}, \mathbf{x}_{rand})$ to state $\mathbf{x}_{rand}$, that is,

$$\mathbf{x}_{near} = \arg\min_{\mathbf{x} \in \mathcal{C}_{near}} C(\mathbf{x}, \mathbf{x}_{rand}) \tag{5.3}$$

with

$$C(\mathbf{x}, \mathbf{x}_{rand}) = g(\mathbf{x}) + C_{\boldsymbol{\sigma}} + D_{\mathbf{P}}(\mathbf{x}, \mathbf{x}_{rand}), \tag{5.4}$$

where $g(\mathbf{x})$ is the sum of the costs from the tree root $\mathbf{x}_{start}$ to the tree vertex $\mathbf{x}$ and $D_{\mathbf{P}}(\mathbf{x}, \mathbf{x}_{rand})$ the geodesic distance of states $\mathbf{x}$ and $\mathbf{x}_{rand}$ from path $\mathbf{P}$. The cost $C_{\boldsymbol{\sigma}}$ measures the length and smoothness of the trajectory $\boldsymbol{\sigma}$ from tree vertex $\mathbf{x}$ to state $\mathbf{x}_{rand}$ returned by the steer function. It is defined as

$$C_{\boldsymbol{\sigma}} = \sum_{i=0}^{N_e-1} w_d ||\boldsymbol{\sigma}_{i+1} - \boldsymbol{\sigma}_i|| + w_q \left(1 - |\mathbf{q}_{i+1} \cdot \mathbf{q}_i|\right)^2$$

(for parameters w_d and w_q), where $N_e + 1$ is the number of intermediate states $\boldsymbol{\sigma}_i$ on trajectory $\boldsymbol{\sigma}$ and $\mathbf{q}_i$ are the associated quaternions. The cost $C_{\boldsymbol{\sigma}}$ can be computed on-line or very efficiently with a regression approach (Palmieri and Arras, 2015).

`Steer`($\mathbf{x}_{near}$, $\mathbf{x}_{rand}$) returns controls $\mathbf{u}_{new}$ and a trajectory $\boldsymbol{\sigma}_{new}$ from state $\mathbf{x}_{near}$ to state $\mathbf{x}_{rand}$ with terminal time T. The analytical steer function connects any pair of states and respects the *topological property* (Laumond et al., 1998), that is, for any $\epsilon > 0$ there exists some $\eta > 0$ such that, for any two states $\mathbf{x}_{near} \in \mathcal{C}$ and $\mathbf{x}_{rand} \in \mathcal{C}$ with $||\mathbf{x}_{near} - \mathbf{x}_{rand}|| < \eta$, it holds that $||\mathbf{x}_{near} - \boldsymbol{\sigma}_{new}(t)|| < \epsilon$ for all $t \in [0, T]$. If $\boldsymbol{\sigma}_{new}$ is collision-free, it is added to τ as the tree branch (or edge) that connects $\mathbf{x}_{near}$ to $\mathbf{x}_{rand}$.

5.5 Experimental Setup

We now investigate how well Theta*-RRT performs against the baseline planners RRT, A*-RRT, RRT* and A*-RRT*. All planners extend their trees using steer functions. RRT and RRT* sample in the entire state space. RRT uses $C_{\boldsymbol{\sigma}}$ as distance metric, and RRT* uses $C_{\boldsymbol{\sigma}}$ as cost function. A*-RRT and A*-RRT* sample along A* paths. A*-RRT generates samples and selects the tree vertex that connects to

the sampled state with minimum cost in the same way as Theta*-RRT. A*-RRT* generates the samples from a Gaussian distribution centered around the A* path as introduced by Brunner et al. (2013) and uses $C_{\boldsymbol{\sigma}}$ as cost function.

All experiments are carried out with a C++ implementation on a single core of an ordinary PC with a 2.67 GHz Intel i7 processor and 10 GB RAM. The weights of the cost function and the parameters of the distance metric are $w_d = w_e = w_q = w_\theta = 0.5$ and $\delta_S = \delta_R = 4m$. $f_{uniform}$ is set to 1 over 5000 samples.

5.5.1 Nonholonomic Systems

We consider two small-time controllable nonholonomic systems, namely a $3D$ differential drive system and an $8D$ truck-and-trailer system.

Differential drive system: We use a unicycle system with state (x, y, θ), where $(x, y) \in \mathbb{R}^2$ is the Cartesian position and $\theta \in [-\pi, \pi)$ is the heading orientation. After a Cartesian-to-polar coordinate transformation, see Fig. 5.5, the equations of motion are

$$\begin{aligned} \dot{\rho} &= -\cos\alpha\, v \\ \dot{\alpha} &= \frac{\sin\alpha}{\rho} v - \omega \\ \dot{\phi} &= -\omega, \end{aligned} \tag{5.5}$$

where v and ω are the translational and the angular velocities, respectively. For this system, we use the efficient and smooth steer function POSQ (Palmieri and Arras, 2014a). Width and length of the robot are set to $0.4m$ and $0.6m$, respectively.

Truck-and-trailer system: For this system, we use the extended state $(x, y, \theta_0, \theta_1, v, \dot{v}, \delta, \dot{\delta})$, where $(x, y) \in \mathbb{R}^2$ are the coordinates of the trailer axle's midpoint, θ_0 and θ_1 the orientations of the trailer and truck, respectively, v the translational velocity of the truck, $\dot{v}$ its acceleration, δ the steering angle of the truck and $\dot{\delta}$ its derivative, see Fig. 5.6. The equations of motion are

$$\begin{aligned} \dot{x} &= v\cos(\theta_1 - \theta_0)\cos\theta_0 \\ \dot{y} &= v\cos(\theta_1 - \theta_0)\sin\theta_0 \\ \dot{\theta}_0 &= \frac{v}{d_0}\sin(\theta_1 - \theta_0) \\ \dot{\theta}_1 &= \frac{v}{d_1}\ \tan(\delta). \end{aligned} \tag{5.6}$$

For this system, we use the η^4 splines (Ghilardelli et al., 2014) as steer function since they are known to generate high-quality trajectories for truck-and-trailer systems. We set $d_0 = d_1 = 1$, width and length of the trailer to $0.4m$ and $0.6m$ and the truck width to $0.4m$.

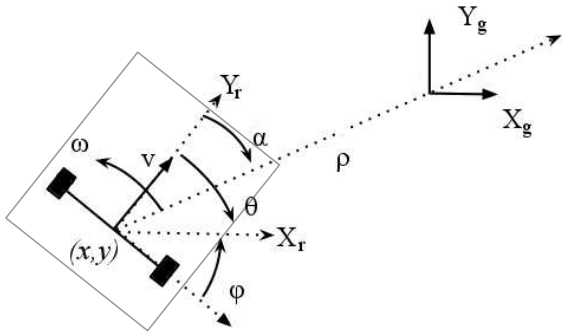

Figure 5.5: Differential drive system in polar coordinates: ρ is the Euclidean distance between the Cartesian coordinates of the robot pose (x, y, θ) and of the goal state, ϕ the angle between the x-axis of the robot reference frame $\{X_r\}$ and the x-axis of the goal state frame $\{X_g\}$, α the angle between the y-axis of the robot reference frame and the vector connecting the robot with the goal position, v the translational and ω the angular robot velocity.

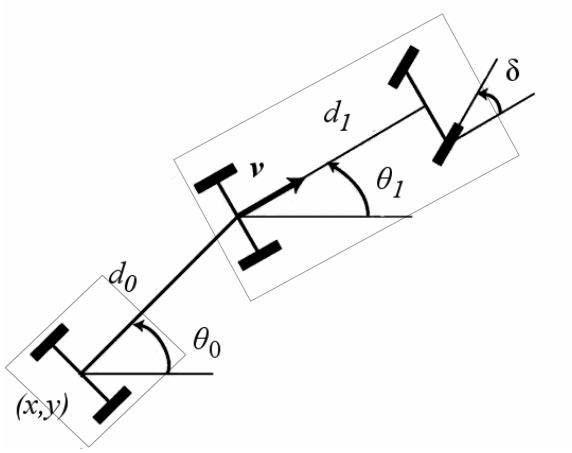

Figure 5.6: Truck-and-trailer system: $(x, y) \in \mathbb{R}^2$ are the coordinates of the trailer axle's midpoint, θ_0 and θ_1 the orientations of the trailer and truck, v the translational velocity of the truck, δ its steering angle, d_1 the distance between the front axle and the rear axle of the truck, and d_0 the distance between the trailer axle and the hitch joint on the rear truck axle.

5.5.2 Environments

To stress-test the planners and study how they behave in environments of varying complexity, we design three simulated test environments shown in Figs. 5.1

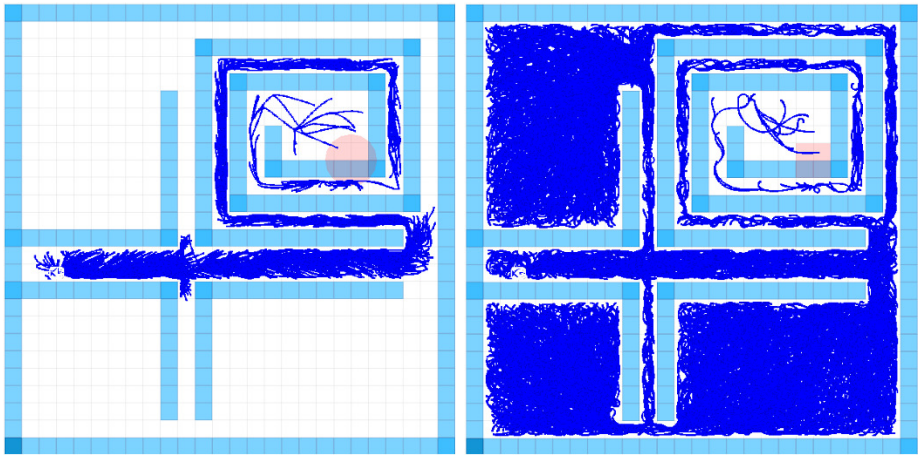

Figure 5.7: *Narrow corridor* environment with the goal position (in **red**) and the trees (in **blue**). **Left:** Tree of Theta*-RRT. **Right:** Tree of RRT. Theta*-RRT generates a smaller tree than RRT, which makes Theta*-RRT faster.

and 5.7. The *maze* environment in Fig. 5.1 contains many different homotopy classes, has local minima (such as U-shaped obstacles) and narrow passages. Its size is 50m×50m. The *random* environment contains randomly generated square obstacles, its size is 50m×30m. The *narrow corridor* environment in Fig. 5.7 stresses the ability of the planners to generate smooth trajectories in narrow corridors, its size is 25m×25m. The grid cell size for the any-angle search is 1 m in all environments.

5.5.3 Performance Metrics

For each planner and environment, we perform 100 runs for the differential drive system and 50 runs for the truck-and-trailer system. We are solely interested in the first trajectories found. We compute the means and standard deviations of the following four performance metrics for all planning problems that are solved within the planning time limit of 1,000 seconds: tree size N_v (measured in the number of stored tree vertices), planning time T_s (measured in milliseconds or seconds) and resulting trajectory length l_p (measured in meters). Regarding smoothness, instead of using performance metrics based on the velocity profile of the robot (as in the previous chapters such as the average speed arc lengths, velocity profile peaks or normalized jerk), here we use a metric that is better suited for measuring *geometric* trajectory smoothness and thus human-perceived smoothness (namely how sharp the turns are) as also presented in Chapter 2 roughness R, defined as the square of the change in curvature κ of the robot, integrated along the trajectory and normalized by the trajectory length L,

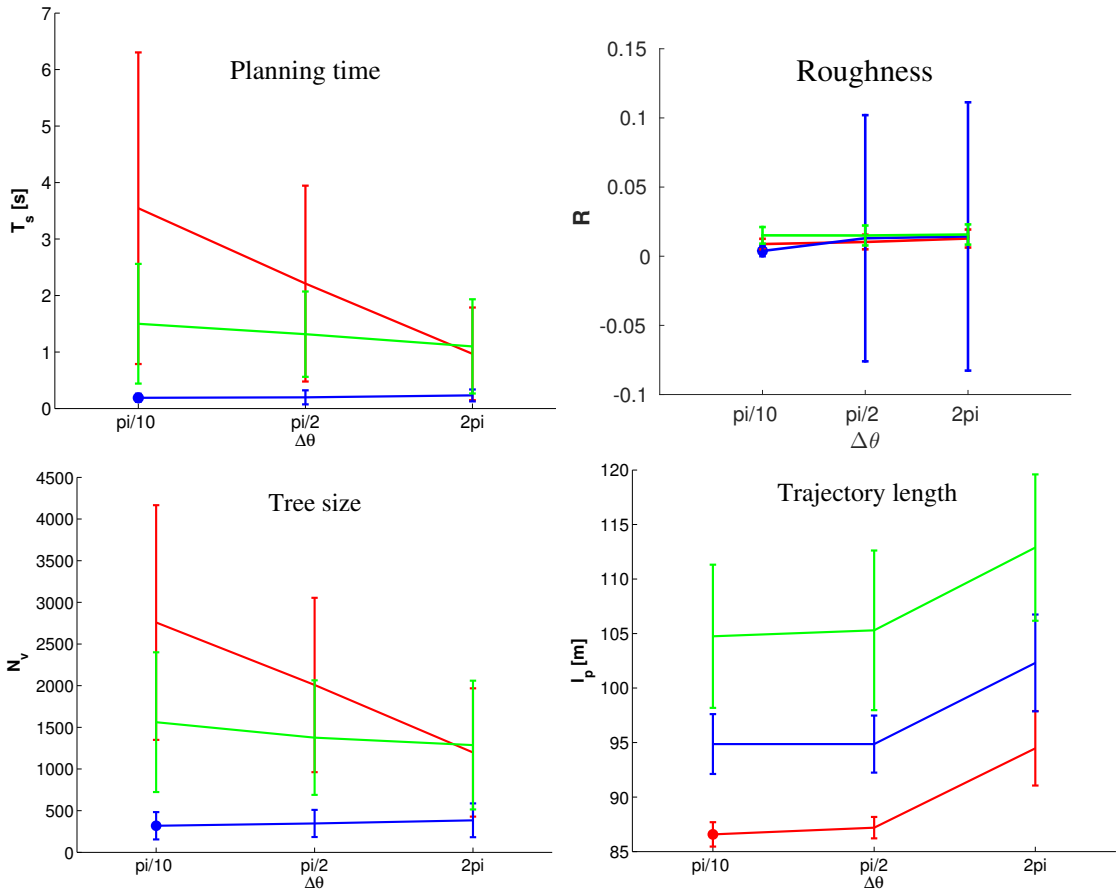

Figure 5.8: Performance trends for different strengths of the position bias W ($W = 1m$ in **red**, $W = 4m$ in **blue** and $W = 10m$ in **green**) and orientation bias $\Delta\theta$ in the *maze* environment for the metrics planning time, roughness, tree size and trajectory length (smaller values are better for all performance metrics).

$$R = \int_{t_0}^{t_l} \left| \frac{1}{L} \frac{d\kappa}{dt} \right|^2 dt.$$

A smaller roughness indicates smoother trajectories. We also compute the percentage of trajectories found within the planning time limit.

5.5.4 Theta*-RRT Parameters

Prior to the main experiment, we analyze the impact of the parameters W and $\Delta\theta$ on the performance of Theta*-RRT. Position bias W is related to the geometry of the wheeled robot and should be set to a value no less than the maximum

value of its length and width. We use the *maze* environment and the ranges $W = \{1m, 4m, 10m\}$ and $\Delta\theta = \{\frac{\pi}{10}, \frac{\pi}{2}, 2\pi\}$. For each pair of parameter values, we compute the mean and standard deviation of the four performance metrics over multiple runs. Fig. 5.8 shows the results for the differential drive system. The results for the truck-and-trailer system are qualitatively similar. We observe three trends: *(i)* With a larger orientation and position bias (that is, smaller $\Delta\theta$ and W), the trajectories tend to be shorter and smoother, which is expected since the trajectories then follow the any-angle paths more closely. *(ii)* With a smaller orientation bias (that is, larger $\Delta\theta$), the tree sizes and planning times tend to be smaller. The optimum is at the medium value $W = 4m$ where the value of $\Delta\theta$ has almost no influence (but the optimum is at the smallest value $\Delta\theta = \frac{\pi}{10}$). Given these trends, we select the medium position bias $W = 4m$ and the strong orientation bias $\Delta\theta = \frac{\pi}{10}$.

5.6 Experimental Results

The experimental results for Theta*-RRT and the four baseline planners are given in Tables 5.1-5.6. Smaller values are better for all performance metrics. The best values are highlighted in boldface. Theta*-RRT outperforms the four baselines with respect to all performance metrics, with only two exceptions. It is a close second with respect to trajectory smoothness to RRT* for the differential drive system in the *random* environment and to A*-RRT for the truck-and-trailer system in the *narrow corridor* environment. We make the following observations:

(i) The path-biasing heuristic of Theta*-RRT avoids the time-consuming exploration of the entire state space and thus results in small tree sizes and planning times. This advantage comes at the cost of having to find an any-angle path first but Tab. 5.7 shows that the runtime of the discrete search is negligible compared to the overall planning time. Theta*-RRT thus has an advantage over RRT and RRT* that explore large parts of the state space, especially in environments with local minima and narrow passages. For this reason, RRT* (and even A*-RRT*) fail to find any trajectory within 1,000 seconds for the high-dimensional truck-and-trailer system in all runs in two of the three environments.

(ii) The path-biasing heuristic of Theta*-RRT results in trajectories that fall into good homotopy classes and are thus short. Theta*-RRT thus has an advantage over A*-RRT and A*-RRT*, whose path-biasing heuristics suffer from the A* paths typically being in worse homotopy classes than the Theta* paths, which results in longer trajectories and thus also larger planning times and tree sizes.

(iii) The sampling strategy of Theta*-RRT results in smooth trajectories. Theta*-RRT thus has an advantage over A*-RRT*, whose sampling strategy is not quite as sophisticated.

Additionally, we tested Theta*-RRT in a real-world setting by deploying it on a

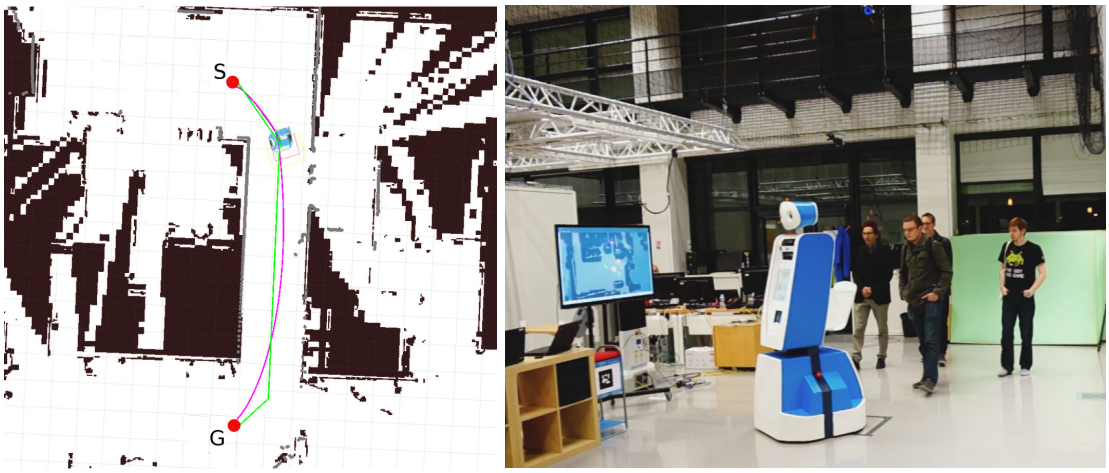

Figure 5.9: Theta*-RRT on a real differential drive robot. **Right:** The robot guides a group of people. **Left:** The dots S and G (in **red**) represent the start and goal positions (respectively). The any-angle path (in **green**) is generated first, followed by the smooth trajectory (in **purple**)

.

passenger guidance robot for complex and busy airport environments (Fig. 5.9).

Random environment						
Planner	$\mathbf{N_v}$		$\mathbf{T_s}$ $[s]$		$\mathbf{l_p}$ $[m]$	
Theta*-RRT	54	$\pm$ 59	**0.011**	$\pm$ 0.007	**43.11**	$\pm$ 1.485
A*-RRT	49	$\pm$ 46	0.020	$\pm$ 0.01	44.16	$\pm$ 1.76
RRT	137	$\pm$ 150	0.09	$\pm$ 0.05	65.25	$\pm$ 13.54
RRT*	168	$\pm$ 154	9.57	$\pm$ 13.73	43.84	$\pm$ 1.45
A*-RRT*	**32**	$\pm$ 34	0.40	$\pm$ 0.93	52.88	$\pm$ 19.0
Maze environment						
Planner	$\mathbf{N_v}$		$\mathbf{T_s}$ $[s]$		$\mathbf{l_p}$ $[m]$	
Theta*-RRT	**319**	$\pm$ 164	**0.19**	$\pm$ 0.07	**94.86**	$\pm$ 2.74
A*-RRT	1470	$\pm$ 777	3.73	$\pm$ 4.8	98.45	$\pm$ 1.12
RRT	2615	$\pm$ 960	4.85	$\pm$ 4.62	139.16	$\pm$ 21.63
RRT*	658	$\pm$ 37	16.13	$\pm$ 0.34	129.61	$\pm$ 5.65
A*-RRT*	356	$\pm$ 193	47.66	$\pm$ 48.47	96.57	$\pm$ 5.37
Narrow corridor environment						
Planner	$\mathbf{N_v}$		$\mathbf{T_s}$ $[s]$		$\mathbf{l_p}$ $[m]$	
Theta*-RRT	**1206**	$\pm$ 258	**8.16**	$\pm$ 3.63	**77.78**	$\pm$ 1.44
A*-RRT	1799	$\pm$ 715	18.84	$\pm$ 15.2	77.8	$\pm$ 1.33
RRT	8488	$\pm$ 1639	180.45	$\pm$ 58.55	78.36	$\pm$ 1.82
RRT*	45310	$\pm$ 7012	2667.5	$\pm$ 481.7	79.47	$\pm$ 0.9
A*-RRT*	3236	$\pm$ 572	309.4	$\pm$ 119.4	78.37	$\pm$ 0.65

Table 5.1: Experimental results: Planning efficiency for the differential drive system.

Roughness **R**						
Planner	*Random* env.		*Maze* env.		*Narrow corridor* env.	
Theta*-RRT	0.001	$\pm$ 0.003	**0.0038**	$\pm$ 0.0038	**0.0027**	$\pm$ 0.004
A*-RRT	0.003	$\pm$ 0.004	0.015	$\pm$ 0.007	0.023	$\pm$ 0.01
RRT	0.009	$\pm$ 0.008	0.018	$\pm$ 0.01	0.0069	$\pm$ 0.005
RRT*	**0.0007**	$\pm$ 0.00140	0.024	$\pm$ 0.01	0.03	$\pm$ 0.008
A*-RRT*	0.0057	$\pm$ 0.0098	0.013	$\pm$ 0.009	0.0125	$\pm$ 0.006

Table 5.2: Experimental results: Trajectory quality for the differential drive robot system.

Problems solved			
Planner	*Random* env.	*Maze* env.	*Narrow corridor* env.
Theta*-RRT	100%	100%	100%
A*-RRT	100%	100%	100%
RRT	100%	100%	100%
RRT*	100%	100%	100%
A*-RRT*	100%	100%	100%

Table 5.3: Experimental results: Problems solved for the differential drive robot system.

5.7 Probabilistic Completeness of Theta*-RRT

The results clearly demonstrate the benefit of Theta*-RRT. However, its path-biasing heuristic – as any heuristic – can mislead and even degrade the performance of RRT, for example when the any-angle path is infeasible to follow under kinodynamic constraints, although a geometric solution (in the inflated grid world) exists. In such cases, the probabilistic completeness, a key property of RRT, is lost. In this section, we prove that Theta*-RRT retains the probabilistic completeness for all small-time controllable nonholonomic systems which use an analytical steer function. Our proof follows the one introduced by LaValle and Kuffner Jr (2001) but we consider a special class of nonholonomic systems, namely systems that are *small-time controllable*, see Definition 10. Moreover our proof takes inspiration from the *two level planning approach* and its properties detailed by Sekhavat et al. (1997); Laumond et al. (1998); Sekhavat et al. (1998).

Theorem 1. *Consider a small-time controllable nonholonomic system. Define a non-zero and non-uniform continuous sampling distribution f_s over $\mathcal{C}_{free}$ generated by the path-biasing technique. Let Theta*-RRT use an analytical steer function that connects any*

Random environment					
Planner	$\mathbf{N_v}$		$\mathbf{T_s}$ $[s]$		$\mathbf{l_p}$ $[m]$	
Theta*-RRT	**52.2**	$\pm$ 48.3	**0.0547**	$\pm$ 0.0790	**44.331**	$\pm$ 2.8418
A*-RRT	75.7	$\pm$ 52.4	0.1019	$\pm$ 0.0984	51.74	$\pm$ 7.89
RRT	836	$\pm$ 378	1.32	$\pm$ 0.84	66.96	$\pm$ 14.7
RRT*	3957	$\pm$ 2756	816.16	$\pm$ 656.58	52.39	$\pm$ 13.12
A*-RRT*	3582	$\pm$ 3138	949.6	$\pm$ 823.7	49.30	$\pm$ 12.79
Maze environment						
Planner	$\mathbf{N_v}$		$\mathbf{T_s}$ $[s]$		$\mathbf{l_p}$ $[m]$	
Theta*-RRT	**522**	$\pm$ 167	**2.57**	$\pm$ 1.50	**98.59**	$\pm$ 4.95
A*-RRT	661	$\pm$ 181	4.56	$\pm$ 2.0858	101.79	$\pm$ 8.26
RRT	4858	$\pm$ 1276	38.88	$\pm$ 15.83	126.34	$\pm$ 16.52
RRT*	0	$\pm$ 0	0	$\pm$ 0	0	$\pm$ 0
A*-RRT*	0	$\pm$ 0	0	$\pm$ 0	0	$\pm$ 0
Narrow corridor environment						
Planner	$\mathbf{N_v}$		$\mathbf{T_s}$ $[s]$		$\mathbf{l_p}$ $[m]$	
Theta*-RRT	**1513**	$\pm$ 492	**20.87**	$\pm$ 13.77	**77.10**	$\pm$ 6.75
A*-RRT	2139	$\pm$ 573	33.46	$\pm$ 16.74	79.66	$\pm$ 5.94
RRT	1794	$\pm$ 5473	733.98	$\pm$ 438.28	83.77	$\pm$ 7.04
RRT*	0	$\pm$ 0	0	$\pm$ 0	0	$\pm$ 0
A*-RRT*	0	$\pm$ 0	0	$\pm$ 0	0	$\pm$ 0

Table 5.4: Experimental results: Planning efficiency for the truck-and-trailer system.

Roughness **R**				
Planner	*Random* env.	*Maze* env.		*Narrow corridor* env.	
Theta*-RRT	**0.0057** $\pm$ 0.0060	**1.0073**	$\pm$ 0.7226	2.15	$\pm$ 1.12
A*-RRT	3.4993 $\pm$ 8.8502	1.1317	$\pm$ 1.0372	**1.9352**	$\pm$ 0.8722
RRT	2.17 $\pm$ 2.00	2.0788	$\pm$ 1.2985	2.31	$\pm$ 1.47
RRT*	0.54 $\pm$ 1.01	0	$\pm$ 0	0	$\pm$ 0
A*-RRT*	0.1013 $\pm$ 0.2647	0	$\pm$ 0	0	$\pm$ 0

Table 5.5: Experimental results: Trajectory quality for the truck-and-trailer system.

Problems solved			
Planner	*Random* env.	*Maze* env.	*Narrow corridor* env.
Theta*-RRT	100%	100%	100%
A*-RRT	58%	100%	100%
RRT	100%	100%	100%
RRT*	76%	0% – **failed**	0% – **failed**
A*-RRT*	100%	0% – **failed**	0% – **failed**

Table 5.6: Experimental results: Problems solved for the truck-and-trailer system.

Environments	$\mathbf{T}_{\mathbf{Theta}^*}$ $[ms]$	$\mathbf{T}_{\mathbf{A}^*}$ $[ms]$
Random	5.34	8.07
Maze	12.06	19.91
Narrow corridor	45.14	37.74

Table 5.7: Experimental results: Planning times of Theta* and A*

pair of states in $\mathcal{C}$. Then, Theta-RRT is probabilistically complete since the probability of connecting the start state $\mathbf{x}_{start} \in \mathcal{C}_{free}$ to the goal state $\mathbf{x}_{goal} \in \mathcal{C}_{free}$, if possible, approaches one asymptotically.*

Proof. Let $\mathcal{B}(\mathbf{x}_i, \rho)$ denote the ball of radius $\rho > 0$ centered on $\mathbf{x}_i \in \mathcal{C}_{free}$. Consider all the tree vertices $\cup_{i=0,..,k}\ \mathbf{x}_i \in \tau$ at iteration k. Since the volume $\Omega = \cup_{i=0,..,k}\ \mathcal{B}(\mathbf{x}_i, \rho \geq \delta_R > 0)$ is non-zero for the Lebesgue metric the event of sampling a state $\mathbf{x}_{rand} \in \Omega$ will happen with probability one as the number of iterations goes to infinity. Given that the system is small-time controllable, the connection (performed by the *steer* function) of $\mathbf{x}_{rand}$ to $\mathbf{x}_{near}$ (chosen among multiple vertices in τ), will be successful and therefore (if collision free) $\mathbf{x}_{rand}$ will be added to τ. The set $\tilde{\mathcal{C}}_k = \{\mathbf{x} \in \mathcal{C}_{free} \setminus \forall \mathbf{x} \in \tau\}$ represents the uncovered part of the space $\mathcal{C}_{free}$ by τ. By induction following the above property, as k approaches infinity, $\mu(\tilde{\mathcal{C}}_k)$ (the volume of $\tilde{\mathcal{C}}_k$) approaches zero, therefore the state $\mathbf{x}_{goal}$ will be added to τ with probability one. □

Since the proof does not exploit any geometric properties of $\mathcal{C}_{local}$, Theorem 1 extends to RRT with any path-biasing heuristic as long as it uses analytical steer functions for systems that are small-time controllable.

5.8 Conclusions

In this chapter, we introduced Theta*-RRT, a hierarchical technique that combines (discrete) any-angle search with (continuous) RRT motion planning for small-time controllable nonholonomic wheeled robots. We evaluated the approach using two different nonholonomic systems in three different environments and compared it to four different baseline planners, namely RRT, A*-RRT, RRT* and A*-RRT*. The results show that Theta*-RRT finds shorter trajectories at least one order of magnitude faster than the baselines without loss of smoothness in very complex environments and for high dimensional systems, while A*-RRT* and RRT* (and thus also Informed RRT* (Gammell et al., 2014)) fail to generate a first trajectory sufficiently fast in environments with complex nonholonomic constraints. We also proved that Theta*-RRT retains the probabilistic completeness of RRT for all small-time controllable systems that use an analytical steer function: we showed that high dimensional problems with difficult nonholonomic

constraints can be solved without sacrificing the original probabilistic completeness of RRT by properly exploiting, with an any-angle path biasing technique, the workspace information.

CHAPTER 6

Kinodynamic Motion Planning on Gaussian Mixture Fields

"Wahrlich es ist nicht das Wissen,
sondern das Lernen, nicht das Besitzen
sondern das Erwerben, nicht das
Da-Seyn, sondern das Hinkommen,
was den grössten Genuss gewährt."

Johann Carl Friedrich Gauss
Letter to Farkas Bolyai

In this chapter, we extend the work presented in the previous chapters by introducing a motion planning approach for wheeled mobile robots under kinodynamic constraints to environments where the objects' dynamics are described by learned perception priors in the form of continuous Gaussian mixture fields. Our Gaussian mixture fields are statistical multi-modal motion models of discrete objects or continuous media in the environment that encode e.g. the dynamics of air or pedestrian flows. We approach this task using a recently proposed circular linear flow field map based on semi-wrapped GMMs whose mixture components guide sampling and rewiring in an RRT algorithm using an enhanced POSQ steer function for nonholonomic mobile robots. In our experiments with three alternative baselines, we show that this combination allows the planner to very efficiently generate high-quality solutions in terms of path smoothness, path length as well as natural yet minimum control effort motion through multi-modal representations of Gaussian mixture fields.*

6.1 Introduction

So far we have introduced modifications to the main functions in RRT and RRT* based motion planners, aiming to efficiently generate smooth robot kinodynamically feasible motion in complex and cluttered static environments. In this chapter, we consider the robot in a dynamic environment and present an RRT* based approach, that generates robot motion based on learned perception priors of the

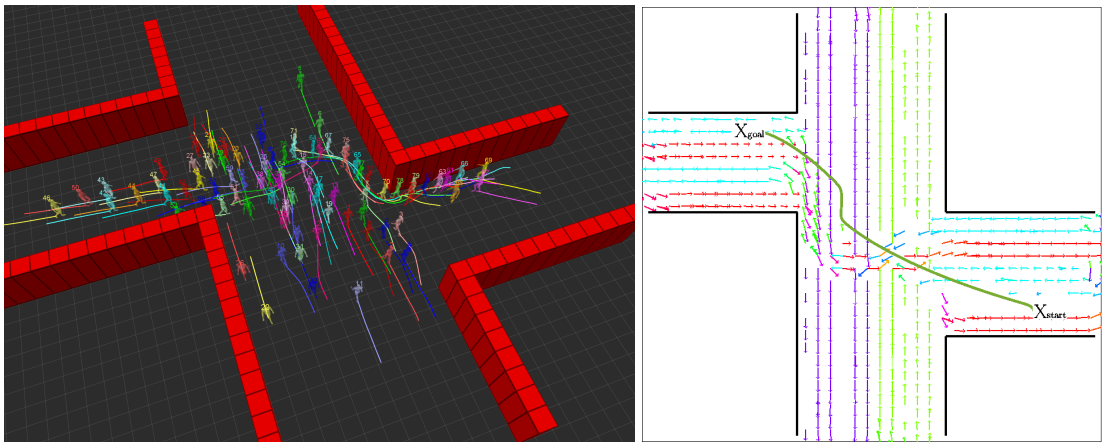

Figure 6.1: We present a planning approach that generates smooth and optimal trajectories over a field of Gaussian mixtures. **Left:** The figure shows an example from the studied simulated *intersect* scenario where multiple flows of people encounter each other. **Right:** An example path (in **green**) generated among the Circular Linear Flow Field (CLiFF) map which associates a Gaussian mixture model to each location, whose components encode multiple weighted flow directions.

obstacles and humans' dynamics. Robot operation environments are often rich in semantics, affordances and dynamically moving objects that follow typical motion patterns. Knowledge of such features in addition to the basic geometry of the workspace represents valuable information for a motion planner to generate better solutions in terms of path quality, safety, replanning frequency or social normativeness. Mobile service robots in human environments, for example, may exploit information about typical pedestrian flows to avoid high-density areas and to take advantage of such flows to reach a destination.

Here we present a planning approach that accounts for typical motion of dynamic objects or continuous media such as pedestrian flows, modeled as a field of semi-wrapped Gaussian mixtures to represent the underlying multi-modal vector field.

In addition to the model to represent environment dynamics, we use an asymptotically optimal sampling-based motion planner that implicitly considers the robot's kinematic and its nonholonomic constraints by using a steer function to plan over a continuous state space. The combination leads to a novel algorithm, named *CLiFF-RRT**, that plans kinodynamically feasible paths under a CLiFF-map model (the model will be introduced in Sec.6.3), trading off classical path quality metrics with the compliance to the environment dynamics. In the experiments, we compare our approach to RRT, RRT* and an uninformed variant of

the algorithm and show that CLiFF-RRT* is significantly faster than the baselines and produces solutions that best comply to the flow directions as modeled by the map. The algorithm also achieves shorter and smoother paths and retains the probabilistic completeness and asymptotic optimality properties of RRT*.
The chapter is structured as follows: in Sec.6.2 we detail related state-of-the-art methods, in Sec.6.3 we briefly present the CLiFF-map model and describe the algorithm and its properties in Sec. 6.4. We present experiments in Sec. 6.5 and discuss their results in Sec. 6.6.

6.2 Related work

Past approaches have considered motion planning over regular (unimodal) vector fields. Kularatne et al. (2016) present a graph-based approach that generates time optimal and energy efficient motion plans for autonomous surface and underwater vehicles in time-varying flow fields. The kinematic constraints of the vehicles are accounted for in the cost function. Moreover in navigation among humans the robot is not always interested in finding the most efficient solution but rather one that respects some social norms. Otte et al. (2016) describe a graph-based algorithm to solve the problem of real-time path planning in time-varying wind fields. The anytime algorithm finds an $\alpha\,\beta$ solution quickly which is then incrementally improved given more time. Lolla et al. (2014) generate paths for swarms of underwater vehicles, deployed in the Philippine Archipelago region, over dynamic water flow fields using a level set approach. The method, based on a 2D grid representation of the environment, finds time-optimal paths while respecting the kinematic constraints of the system. Ko et al. (2014) present an RRT-based path planner over a vector field defined in the configuration space. To find a path, the algorithm tries to minimize an *upstream criterion* which quantifies the control effort to go against a vector field. Tree growth is guided by this criteria resulting in extensions that are more probably aligned with the vector field directions.
Recurring patterns of human motion that people typically follow in an environment have been learned and used for planning by Bennewitz et al. (2003); O'Callaghan et al. (2011); Fulgenzi et al. (2010); Rios-Martinez et al. (2011). Such patterns can be seen as sparse vector fields as they are only defined in parts of the state space where humans have been repeatedly observed. Based on the Risk-RRT algorithm (Fulgenzi et al., 2010), Rios-Martinez et al. (2011) use Gaussian processes (GP) to predict motion of humans and generate paths with an RRT-based planner that minimize the risk of disturbing and colliding with surrounding people.
O'Callaghan et al. (2011) present a method that generates paths by following learned motion patterns of people using GPs. The method computes a nav-

igational map based on the motion patterns whose cells incorporate velocity vectors. The robot navigates through the environment by querying the learned map and obtaining the next direction to follow. Bennewitz et al. (2003) learn a collection of human motion patterns using Gaussian mixtures and EM. For each observed human the most probable pattern is determined and used to make predictions of future motion for planning. The method uses A* on a 2D grid with cell costs discounted by the probability that a person is in a cell at a given time.
Unlike (Kularatne et al., 2016; Otte et al., 2016; Lolla et al., 2014; Ko et al., 2014) we use a more powerful probabilistic representation than vector fields named Circular Linear Flow Field (CLiFF) map (Kucner et al., 2016). It associates a Gaussian mixture model to each location whose components encode multiple weighted flow directions. The model captures the dependency between motion speed (a linear variable) and direction (a circular variable) using semi-wrapped Gaussian mixture models introduced by Roy et al. (2014).

6.3 The CLiFF-Map Model

The Circular Linear Flow Field map (CLiFF-map [1]) Kucner et al. (2017) describes motion patterns as a field of Gaussian mixtures whose local elements are probability distribution of (instantaneous) velocities $\boldsymbol{V} = (\theta, \rho)$, where $\theta \in [0, 2\pi)$ is the orientation and $\rho \in \mathbb{R}^+$ the speed. This is a heterogeneous vector with one circular random variable (θ) and one linear (ρ). For their representation we choose the *semi-wrapped normal distribution*: a bivariate normal distribution on a cylinder where one of the dimensions is defined along the cylinder's height while the other is wrapped around it's circumference,

$$\mathcal{N}^{\mathcal{SW}}_{\boldsymbol{\mu},\boldsymbol{\Sigma}}(\boldsymbol{V}) = \sum_{k\in\mathbb{Z}} \mathcal{N}_{\boldsymbol{\mu},\boldsymbol{\Sigma}}\left(\begin{bmatrix}\theta\\ \rho\end{bmatrix} + 2\pi\begin{bmatrix}k\\ 0\end{bmatrix}\right). \tag{6.1}$$

To model multi-modal events such as human motion patterns, CLiFF-maps employ semi-wrapped Gaussian mixture models (SWGMM),

$$p(\boldsymbol{V}|\boldsymbol{\xi}) = \sum_{j=1}^{J} \pi_j \mathcal{N}^{\mathcal{SW}}_{\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j}(\boldsymbol{V}) \tag{6.2}$$

with $\sum_{j=1}^{J} \pi_j = 1$. A SWGMM ($\boldsymbol{\xi}$) is a weighted sum of J semi-wrapped normal distributions, that capture the local distribution of velocities. A CLiFF-map

[1] Note that the CLiFF representation has been introduced by Kucner et al. (2017). In this work we combine the CLiFF map concept with a kinodynamic motion planner.

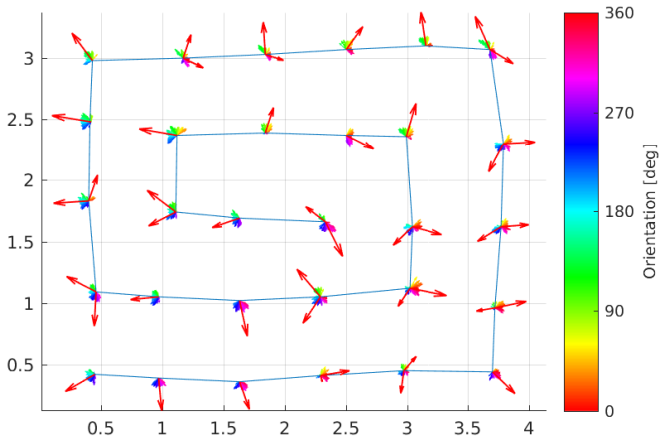

Figure 6.2: Visualisation of CLiFF distributions obtained for a set of wind measurements. The red arrows represent the directions of the modes while color coded ones represent the raw measurements. Wada et al. (2010)

(see Fig. 6.2), denoted as $\mathcal{D}$, is a field of $N_{\mathcal{D}}$ tuples that characterize local motion patterns of dynamic obstacles as

$$\mathcal{D} = \{(\boldsymbol{\xi}_s, \boldsymbol{l}_s) | s \in \mathbb{Z}^+ \wedge \boldsymbol{l}_s \in \mathbb{R}^2\}, \tag{6.3}$$

where $\boldsymbol{\xi}_s$ denotes a SWGMM that describes a local motion pattern at position $\boldsymbol{l}_s$. We use Mean Shift (MS) to estimate the initial position of clusters for EM, which estimates the parameters of SWGMM ($\boldsymbol{\xi}$).

The von Mises distribution, broadly used for modeling uncertain circular data (e.g. Calderara et al. (2011)), is not suitable due to the heterogeneity of the considered variables. Attempts to overcome this include e.g. building Independent von Mises–Gaussian distributions (Roy et al. (2012)) but such distributions still assume no correlation between magnitude and orientation of velocity vectors – an invalid assumption in most real world cases.

6.4 Our Approach

In this section we describe our method to plan a robot's motion under a CLiFF-map model. To this end, we introduce an *extended upstream criterion* which measures the effort to navigate through a CLiFF-map, followed by the description of the algorithm and its properties.

6.4.1 Extended Upstream Criterion

The goal of our algorithm is to find planning solutions that trade off classical motion planning metrics such as path length and path smoothness with the compliance to the environment dynamics. In order to quantify the latter, we extend the *upstream criterion*, proposed by Ko *et al.* Ko et al. (2014) for unimodal vector fields, to fields of Gaussian mixtures so as to account for the multi-modal nature of the CLiFF-map representation.

Given a state $\mathbf{x}_i$ that falls into a cell $\boldsymbol{l}_i$ to which a mixture $\boldsymbol{\xi}_i$ with J_i semi-wrapped normal components is associated, the upstream metric is computed as

$$U_d(\mathbf{x}_i, \boldsymbol{\xi}_i) = \sum_{j=1}^{J_i} \Big(\|\boldsymbol{\mu}_{ji}\| - \langle \boldsymbol{\mu}_{ji}, \mathbf{x}'_i \rangle \Big)$$

with $\langle , \rangle$ being the inner product, $\boldsymbol{\mu}_{ji}$ the first-order moment of the jth component of mixture $\boldsymbol{\xi}_i$, and $\mathbf{x}'_i$ the unit vector describing the direction of the path at $\mathbf{x}_i$. When $\mathbf{x}_i$ is mapped to a cell $\boldsymbol{l}_i$ that has no distributions, we perform nearest-neighbor interpolation and compute $U_d(\mathbf{x}_i, \boldsymbol{\xi}_i)$ using the closest mixture $\boldsymbol{\xi}_i$. The criterion yields low costs for paths that comply to the directions of CLiFF-map mixture components and high costs for paths in opposite directions.

6.4.2 CLiFF-RRT*

For planning, we choose (and modify) RRT* as a natural choice for optimal motion planning under kinodynamic constraints. Let $\mathcal{X} \in \mathcal{R}^d$ be the configuration space and $\mathcal{U} \in \mathcal{R}^m$ the control space, the dynamics of the robot can be described by the differential equation $\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t)), \mathbf{x}(0) = \mathbf{x}_0$, with $\mathbf{x}(t) \in \mathcal{X}$, $\mathbf{u}(t) \in \mathcal{U}$ and f describing the system's kinematic constraints.

RRT* Karaman and Frazzoli (2011) is a probabilistically complete single-query sampling-based planner that asymptotically finds optimal solutions for a motion planning problem. Given an obstacle space $\mathcal{X}_{obs} \in \mathcal{X}$, a free space $\mathcal{X}_{free} \in \mathcal{X} \setminus \mathcal{X}_{obs}$, a start state $\mathbf{x}_{start} \in \mathcal{X}_{free}$ and a goal state $\mathbf{x}_{goal} \in \mathcal{X}_{goal} \subset \mathcal{X}_{free}$, the algorithm expands into $\mathcal{X}_{free}$ a tree τ whose edges are trajectories σ_i (with $\sigma_i(j)$ being state j of trajectory i) that satisfy the kinematic constraints of the considered system.

In summary, we approach the task as a hierarchical motion planning problem in that we first generate a discrete path $\mathbf{P}_{\mathcal{D}}$ that selects mixtures at relevant locations, and then use those mixtures to bias the sampling and rewiring procedures in RRT*. The first step makes sure that an initially feasible path is found quickly given a CLiFF map while the second step, generates and incrementally improves a trajectory $\mathbf{x}_{\mathcal{D}}$ that satisfies the kinodynamic vehicle constraints. The result is CLiFF-RRT* in Alg. 7 whose steps are explained next.

`SelectMixtures`($\mathcal{X}$, $\mathbf{x}_{start}$, $\mathbf{x}_{goal}$): in this step we select the semi-wrapped mixtures $\mathcal{N}^{SW}$ that allow the system to move from $\mathbf{x}_{start}$ to $\mathbf{x}_{goal}$ while respecting

Algorithm 7 *CLiFF-RRT**

function CLiFF-RRT*($\mathbf{x}_{start}$, $\mathbf{x}_{goal}$)
$\mathbf{P}_{\mathcal{D}} \Leftarrow$ SelectMixtures($\mathbf{x}_{start}$, $\mathbf{x}_{goal}$)
if $\mathbf{P}_{\mathcal{D}} = \emptyset$ **then**
  return *failure*
end if
τ.AddNode($\mathbf{x}_{start}$)
$g(\mathbf{x}_{start}) \Leftarrow 0$
$n \Leftarrow 1$
while $n \leq$ MAX_ITERATIONS **do**
  $\mathbf{x}_{rand} \Leftarrow$ CLiFFSampling($\mathcal{C}, \mathbf{P}_{\mathcal{D}}$)
  $\mathbf{x}_{near} \Leftarrow$ NearestSearch($\tau, \mathbf{x}_{rand}, \mathbf{P}_{\mathcal{D}}$)
  $\mathbf{u}_{new}, \boldsymbol{\sigma}_{new}, \mathbf{x}_{new} \Leftarrow$ Steer($\mathbf{x}_{near}, \mathbf{x}_{rand}$)
  if $\boldsymbol{\sigma}_{new} \in \mathcal{C}_{obs}$ **then**
    continue
  end if
  τ.AddNode($\mathbf{x}_{new}$)
  τ.AddEdge($\mathbf{x}_{near}, \mathbf{x}_{new}, \boldsymbol{\sigma}_{new}, \mathbf{u}_{new}$)
  $g(\mathbf{x}_{new}) \Leftarrow g(\mathbf{x}_{near}) +$ Cost($\mathbf{x}_{near}, \mathbf{x}_{new}$)
  $\tau \Leftarrow$ Rewire($\tau, \mathbf{x}_{new}, \mathbf{x}_{near}$)
  if $\mathbf{x}_{new} \in \mathcal{C}_{goal}$ **then**
    $\mathbf{x}_{\mathcal{D}} =$ ExtractTrajectory($\mathbf{x}_{new}$)
  end if
  $n \Leftarrow n + 1$
end while
return *failure*

the learned environment's dynamics. We run a Dijkstra search over the graph $\mathcal{G}$ in which each node $n_{\mathcal{G},i}$ is associated to each mixture component $\big(\forall(\boldsymbol{\xi}_i, \boldsymbol{l}_i) \in \mathcal{D}\big)$: for each map cell $\boldsymbol{l}_i$ we compute edges that go between all pairs of the SWGMM components of $\boldsymbol{l}_i$ and those of the cells in its 4-neighborhood. To each edge $e(n_{\mathcal{G},i}, n_{\mathcal{G},j})$, with $n_{\mathcal{G},i}$ and $n_{\mathcal{G},j}$ being two neighboring nodes, we associate the following cost:

$$c(e(n_{\mathcal{G},i}, n_{\mathcal{G},j})) = d(e(n_{\mathcal{G},i}, n_{\mathcal{G},j})) + U_d(\mathbf{x}_{n_{\mathcal{G},j}}, \boldsymbol{\xi}_{n_{\mathcal{G},j}}) \tag{6.4}$$

where $d(e(n_{\mathcal{G},i}, n_{\mathcal{G},j}))$ is the squared Euclidean distance between the nodes, $\mathbf{x}_{n_{\mathcal{G},j}}$ and $\boldsymbol{\xi}_{n_{\mathcal{G},j}}$ respectively the state and the mixture associated to the node $n_{\mathcal{G},j}$.

The search generates a concatenation $\mathbf{P}_{\mathcal{D}}$ (i.e. a path) of $N_{\mathbf{P}_{\mathcal{D}}}$ CLiFF-map tuples $(\boldsymbol{\xi}_i, \boldsymbol{l}_i)$ from $\boldsymbol{l}_0$ (with $\mathbf{x}_{start} \in \boldsymbol{l}_0$) to $\boldsymbol{l}_{N_{\mathbf{P}_{\mathcal{D}}-1}}$ (with $\mathbf{x}_{goal} \in \boldsymbol{l}_{N_{\mathbf{P}_{\mathcal{D}}-1}}$) which is forwarded to the sampling unit.

CLiFFSampling($\mathcal{C}, \mathbf{P}_{\mathcal{D}}$): it draws $\mathbf{x}_{rand}$ samples in $\mathcal{X}$. The parameter $\alpha \in [0, 1]$ sets the probability of the biasing towards the $N_{\mathbf{P}_{\mathcal{D}}}$ CLiFF-map mixtures $\boldsymbol{\xi}_i$ of $\mathbf{P}_{\mathcal{D}}$:

$$\mathbf{x}_{rand} \sim \sum_{i=0}^{N_{\mathbf{P}_\mathcal{D}}-1} \sum_{j=1}^{J_i} \pi_j \mathcal{N}^{SW}_{\boldsymbol{\xi}_i}(\boldsymbol{\mu}_{j\boldsymbol{\xi}_i}, \boldsymbol{\Sigma}_{j\boldsymbol{\xi}_i})$$

With a probability of $(1-\alpha)$, samples are drawn from a uniform distribution defined on the entire state space $\mathcal{X}$. Note that the samples contains, not only the robot pose but also its velocity.

`NearestSearch`(τ, $\mathbf{x}_{rand}$, $\mathbf{P}_\mathcal{D}$): it returns the node $\mathbf{x}_{near}$ that connects to $\mathbf{x}_{rand}$ with minimum cost-to-go $C(\mathbf{x}_{near}, \mathbf{x}_{rand}, \mathbf{P}_\mathcal{D})$ within distance δR (as parameter) from the latter:

$$\mathbf{x}_{near} = \texttt{arg}\min_{\mathbf{x}\in\mathcal{X}_{\delta R}} g(\mathbf{x}) + \texttt{Cost}(\mathbf{x}, \mathbf{x}_{rand}) \tag{6.5}$$

with $g(\mathbf{x})$ being the cost-to-come to vertex $\mathbf{x}$ from root $\mathbf{x}_{start}$ through the current tree τ. If no nodes are found within this distance, the closest vertex in terms of Euclidean distance is returned.

`Cost`($\mathbf{x}_i$, $\mathbf{x}_j$): returns the cost of the trajectory σ that connects node $\mathbf{x}_i$ to node $\mathbf{x}_j$. Our algorithm aims to find trajectories which are smooth and short, respect the environment dynamics and minimize the upstream criteria (the control effort to move with the vector field) with respect to the off-line learned mixtures $\forall\boldsymbol{\xi}_i \in \mathcal{D}$. For these reasons we use the following cost function:

$$\begin{aligned} C(\mathbf{x}_p, \mathbf{x}_z, \mathbf{P}_\mathcal{D}) = &\sum_{i=1}^{N_p} \|\sigma(i) - \sigma(i-1)\| + |(1 - |\mathbf{q}_i \cdot \mathbf{q}_{i-1}|)| \\ &+ \sum_{i=1}^{N_p} U_d(\sigma(i), \boldsymbol{\xi}_i) \end{aligned}$$

where $\sigma(i)$ are intermediate states of the trajectory σ connecting $\mathbf{x}_p$ to $\mathbf{x}_z$, $\mathbf{q}_i$ are related quaternions, and $U_d(\sigma(i), \boldsymbol{\xi}_i)$ being the upstream functional value at $\sigma(i)$. A supervised learning approach can be used to improve the efficiency of the cost computation as in Palmieri and Arras (2015).

`Steer`($\mathbf{x}_i$, $\mathbf{x}_j$): it generates a trajectory σ and the set of controls $\mathbf{u}$ needed to steer the system from $\mathbf{x}_i$ to $\mathbf{x}_j$. The analytical steer function connects any pair of states and respects the *topological property* as described in Karaman and Frazzoli (2011); Palmieri et al. (2016). Moreover, the generated trajectory needs to respect the initial and the final velocity contained in the samples $\mathbf{x}_i$, $\mathbf{x}_j$.

`Rewire`($\tau, \mathbf{x}_{new}, \mathbf{x}_{near}$): rewires the tree τ, see Alg. 8, as in the original RRT* Karaman and Frazzoli (2011), using the above described `Steer` and `Cost` functions. The rewiring is done at each iteration on a set of vertices found by a near neighbor search as in Karaman and Frazzoli (2013): it finds the set of all the states in τ that lie within a box centered on $\mathbf{x}_{near}$ whose volume scales as $\gamma^D \frac{\log(n_U)}{n_U}$, with

Algorithm 8 *Rewire*(τ, $\mathbf{x}_{new}$, $\mathbf{x}_{near}$)

function Rewire(τ, $\mathbf{x}_{new}$, $\mathbf{x}_{near}$)
$\mathbf{x}_{min} \Leftarrow \mathbf{x}_{near}$
$g(\mathbf{x}_{min}) \Leftarrow g(\mathbf{x}_{new})$
$\mathcal{X}_{near} \Leftarrow$ NearNeighborSearch($\tau, \mathbf{x}_{new}, |\tau|$)
for $\forall\ \mathbf{x}_i \in \mathcal{X}_{near}$ **do**
  $\mathbf{u}_i, \boldsymbol{\sigma}_i, \mathbf{x}_i \Leftarrow$ Steer($\mathbf{x}_i, \mathbf{x}_{new}$)
  if $\boldsymbol{\sigma}_i \in \mathcal{C}_{obs}$ **then**
    continue
  end if
  $c_i = g(\mathbf{x}_i) +$ Cost($\mathbf{x}_i, \mathbf{x}_{new}$)
  if $c_i < g(\mathbf{x}_{min})$ **then**
    $\mathbf{x}_{min} \Leftarrow \mathbf{x}_i,\ \boldsymbol{\sigma}_{min} \Leftarrow \boldsymbol{\sigma}_i, \mathbf{u}_{min} \Leftarrow \mathbf{u}_i$
  end if
end for
τ.AddEdge($\mathbf{x}_{min}, \mathbf{x}_{new}, \boldsymbol{\sigma}_{min}, \mathbf{u}_{min}$)
for $\forall\ \mathbf{x}_j \in \mathcal{X}_{near}$ **do**
  $\mathbf{u}_j, \boldsymbol{\sigma}_j, \mathbf{x}_j \Leftarrow$ Steer($\mathbf{x}_{new}, \mathbf{x}_j$)
  if $\boldsymbol{\sigma}_j \in \mathcal{C}_{obs}$ **then**
    continue
  end if
  $c_j = g(\mathbf{x}_{new}) +$ Cost($\mathbf{x}_{new}, \mathbf{x}_j$)
  if $c_j < g(\mathbf{x}_j)$ **then**
    $\mathbf{x}_{parent} \Leftarrow$ Parent($\mathbf{x}_j$)
    τ.RemoveEdge($\mathbf{x}_{parent}, \mathbf{x}_j$)
    τ.AddEdge($\mathbf{x}_{new}, \mathbf{x}_j, \boldsymbol{\sigma}_j, \mathbf{u}_j$)
  end if
end for
return τ

D being the Hausdorff dimension of the distribution generated by the system dynamics and n_U the number of uniformly distributed samples added to the tree τ.

6.4.3 Steer Function: Augmented POSQ

CLiFF-RRT* draws samples containing not only information regarding hypothetical robot poses but also velocity information. For this reason the steer function needs to be able to generate a trajectory that respects also the desired initial and final velocities of the two samples to connect. We consider wheeled mobile robots with a differential drive kinematic configuration with state $\mathbf{x} = (x, y, \theta, v)$, where $(x, y) \in \mathbb{R}^2$ is the Cartesian position, $\theta \in [-\pi, \pi)$ is the heading orientation and v its translational velocity. After a Cartesian-to-polar coordinate transformation,

the equations of motion are

$$\begin{aligned} \dot{\rho} &= -\cos\alpha\, v \\ \dot{\alpha} &= \frac{\sin\alpha}{\rho} v - \omega \\ \dot{\phi} &= -\omega, \end{aligned} \tag{6.6}$$

where ρ is the Euclidean distance between the Cartesian coordinates of the robot pose (x, y, θ) and of the goal state, ϕ the angle between the x-axis of the robot reference frame $\{X_r\}$ and the x-axis of the goal state frame $\{X_g\}$, α the angle between the y-axis of the robot reference frame and the vector connecting the robot with the goal position, v the translational and ω the angular robot velocity, see Fig. 3.3. Thanks to the polar representation we overcome the obstruction to stabilizability for such system described in the Theorem of Brockett (Brockett et al., 1983). To exactly connect any pairs of states smoothly and efficiently for this description of a wheeled mobile robot, we use and extend the POSQ steer function (see Chapter 3). Each time when the steer function is called in Alg. 7 to connect two sampled states $\mathbf{x}_1 = (x_1, y_1, \theta_1, v_1)$ to $\mathbf{x}_2 = (x_2, y_2, \theta_2, v_2)$, we plan an initial extension by using POSQ as described in Chapter 3 and then modify the velocity profile so that the initial velocity is equal to v_1 and final one is v_2. The velocity profile is generated using an efficient third-order polynomial time-law

6.4.4 Algorithm Properties

RRT* has favourable properties such as probabilistic completeness and asymptotic optimality. In this section we briefly analyze how the alterations of the proposed algorithm impact those properties.

RRT and RRT* are probabilistically complete as their sampling procedure draw samples from a uniform distribution over the state space. This applies also to CLiFF-RRT* which generates, at a given probability, uniformly distributed random samples, none of which are rejected.

Regarding asymptotic optimality, Karaman and Frazzoli have shown that for n uniformly distributed random samples, a steer function that connect two poses exactly, an admissible cost function and a specific constant γ in the selection of the neighboring nodes, RRT* almost surely converges asymptotically to the optimal solution as n goes to infinity (Karaman and Frazzoli, 2013). CLiFF-RRT* uses the same rewiring and neighbor nodes selection procedures of RRT*. It uses a steer function that exactly connects two nodes and its cost function $C(\mathbf{x}_i, \mathbf{x}_j, \mathbf{P}_{\mathcal{D}})$ is an admissible cost function for RRT*: it is monotonic, additive and Lipschitz continuous. Moreover, it generates, at a given probability (1-α), uniformly distributed random samples. Therefore CLiFF-RRT* retains the asymptotic optimality property of RRT*.

6.5 Experiments

The purpose of the experiments is to evaluate the performance of the proposed CLiFF-RRT* algorithm with respect to the baselines of regular RRT and RRT* and an uninformed variant of the algorithm, called All-Mixtures-RRT*, that generates samples from a distribution composed of *all* CLiFF-map mixtures and not on a subset as it the case with CLiFF-RRT*. The latter, as explained in Sec.6.4, draws samples considering a subset of mixtures selected at the beginning of the algorithm via a discrete search (i.e. Dijkstra algorithm). All methods use the steer function described in Sec. 6.4.3 and cost function described in Sec. 6.4.2.

We run the experiments on a single core of an ordinary PC with a 2.80 GHz Intel i7 processor and 32 GB RAM using C++. After a set of informal validation runs we set the parameters α to 0.95 and δR to 4 m, while γ is set in a way to satisfy the requirements of RRT*, see (Karaman and Frazzoli, 2013).

6.5.1 Environments

We study how the planners behave in environments of varying complexity. Given our interest in wheeled mobile service robots, environments are generated by exploiting off-line learned motion models of pedestrian traffic. We have designed four simulated test environments shown in Fig. 6.1 and Fig. 6.3-6.5. In all cases there are different flow dynamics between the start and goal, and different planning solutions (homotopy classes) are possible. The *L* and *P* environments contain a few obstacles. Here the planners have less geometric constraints to better follow the upstream criterion in the free space. The *maze* environment has many different homotopy classes and narrow passages: the environment has many different flows that go against each other. The *intersect* scenario has many flows of pedestrians coming from different corridors intersecting in a junction: also here there are few geometric constraints but several flows are present. All the CLiFF-maps have been generated with the help of the pedestrian simulator Pedsim (Vasquez et al., 2014). The grid cell size for the CLiFF-map is set to 1 m in all the environments.

6.5.2 Metrics

For each planner and environment, we perform 50 runs. For the *L*, *P* and *intersect* scenarios we limit the planning time to 60 s, and for the *maze* scenario to 120 s. We compute the means and standard deviations of the following metrics: planning time T_s (measured in seconds), resulting trajectory length l_p (measured in meters) and final cost C_s. Furthermore, to measure smoothness, we use the roughness metric introduced in Section 2.7 defined as the square of the change in curvature κ of the robot, integrated along the trajectory and normalized by the trajectory

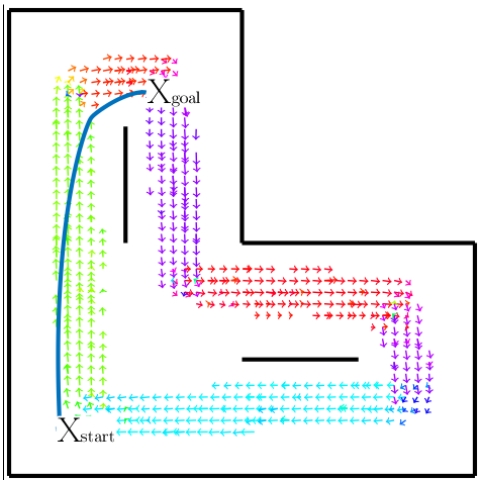

Figure 6.3: An example CLiFF-RRT* path (**in blue**) generated in the *L* scenario. The **arrows** represent the learned mixtures. In the environment just a few obstacles are present. The algorithm finds the best solution that optimizes path length and the upstream criterion: the solutions follow the learned flows.

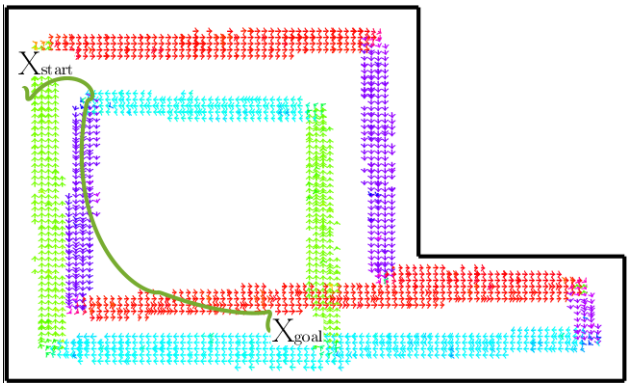

Figure 6.4: An example CLiFF-RRT* path (**in green**) generated in the *P* scenario. The **arrows** represent the learned mixtures. The algorithm finds the best solutions that optimize path length and the upstream criterion.

length L,

$$R = \int_{t_0}^{t_l} \left| \frac{1}{L} \frac{d\kappa}{dt} \right|^2 dt.$$

Smoother trajectories have smaller roughness. We also report the percentage of trajectories found (problems solved) within the planning time limit.

Furthermore, we evaluate the capability of the planners to generate velocity profiles that adhere to the ones that usually flows have into the environments,

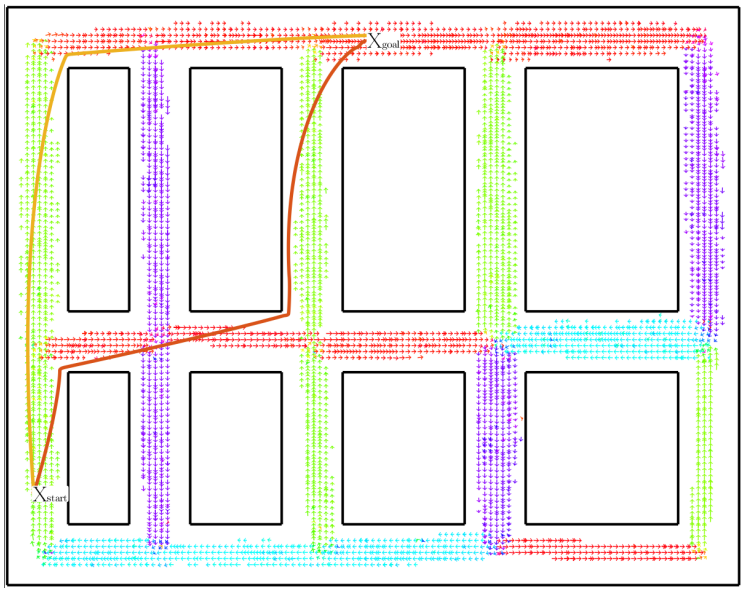

Figure 6.5: An example CLiFF-RRT* path generated in the more complex *maze* scenario. In **red** the RRT* path generated by minimizing only path length. The **arrows** describe the learned mixtures. CLiFF-RRT* computes a path (**in orange**) that better minimizes the upstream criterion, without encountering or crossing flows going in opposing direction.

which have been learned by the CLiFF map. In this respect we compute the average of the mean squared error (V_{MSE}) between the velocity profile generated by the planners and the one contained in the CLiFF map, in the locations where the planner's paths are generated.

6.6 Results And Discussion

The experimental results for CLiFF-RRT* and the three baseline planners are given in Table 6.2-6.5. The best values are highlighted in boldface, smaller values are better for all performance metrics excepts for the percentage of the problems solved.

CLiFF-RRT* outperforms the baselines with respect to all the metrics. We make the following observations:

- CLiFF-RRT* with its focused search finds an initial solution faster than all the baselines (thus also Informed RRT* Gammell et al. (2014) which behaves as RRT* until a first solution is found). RRT and RRT* do not avoid the time-consuming exploration of the entire state space. For this reason the latter more often fails to find an initial solution in the given time. Moreover from Table 6.1, we can see that the planning time of sub-selecting a set of mixtures

with the Dijkstra search does not largely affect the overall planning time of CLiFF-RRT*. Additionally CLiFF-RRT*, in average, converges faster to a lower cost solution than the baselines, see Fig. 6.6. Those results confirm the intuition that having prior knowledge of the environment's dynamics, improves planning efficiency (e.g. in our experiments knowing how people usually move in an environment allows the planner to explore a smaller part of the configuration space).

- CLiFF-RRT* finds less costly solutions if compared to all the baselines. The mixtures selected via the Dijkstra search guide the tree towards areas of the state space where the found trajectory is most likely to offer a good trade-off between length and control-effort against the dynamics of the environment (the *upstream-criterion*). The uniform sampling of RRT and RRT* has not such knowledge thus those planners fail to find a better solution. All-Mixtures-RRT*, without the Dijkstra search biasing, fails to find better solution too in the allowed planning time.

- The CLiFF-RRT* sampling strategy results in smoother trajectories than the baselines. Mainly because the off-line learned mixtures bias the tree towards concatenation of extensions with less velocity discontinuities. Uniform sampling generates velocities without prior knowledge about usual motion in particular portions of the state space, thus producing less correlated velocities.

- As reported in Table 6.5, CLiFF-RRT* generates velocity profiles that better adhere, in average, to the observed flow velocity learned by the CLiFF map.

6.7 Conclusion

In this chapter, we present CLiFF-RRT*, an algorithm that exploits prior knowledge of the environment's dynamics in order to efficiently plan smooth and short paths. Differently from previous approaches that rely on simple vector fields, our method plans considering a novel multi-modal representation of the dynamic obstacles' motion. We evaluated and compared the approach in four different environments to three different baseline planners, namely RRT, RRT* and an uniformed version of RRT* that samples considering all the off-line learned CLiFF distributions. The results indicate that learned CLiFF-map priors help CLiFF-RRT* to find shorter and smoother trajectories faster than all the baselines. Moreover the results show that the approach requires less control effort (smaller cost) to drive a wheeled mobile robot trough a dynamic environment. CLiFF-RRT* retains the probabilistic completeness and the asymptotic optimality of RRT*.

Environments	$\mathbf{T_{Dijkstra}}$ $[ms]$
L	4.36
P	3.36
Intersect	2.83
Maze	121.20

Table 6.1: Planning times of the mixture selection step.

L environment						
Planner	Cost $\mathbf{C_s}$		Plan. time $\mathbf{T_s}$ $[s]$		Traj. length $\mathbf{l_p}$ $[m]$	
CLiFF-RRT*	**111.42**	$\pm$ 5.08	**5.30**	$\pm$ 8.08	**33.59**	$\pm$ 0.78
All-Mixtures-RRT*	130.89	$\pm$ 32.92	14.97	$\pm$ 17.65	35.48	$\pm$ 2.31
RRT	784.82	$\pm$ 618.5	15.83	$\pm$ 17.16	40.59	$\pm$ 7.36
RRT*	212.26	$\pm$ 193.4	28.15	$\pm$ 14.91	37.31	$\pm$ 3.35
Maze environment						
Planner	Cost $\mathbf{C_s}$		Plan. time $\mathbf{T_s}$ $[s]$		Traj. length $\mathbf{l_p}$ $[m]$	
CLiFF-RRT*	**151.51**	$\pm$ 12.62	**31.13**	$\pm$ 32.66	**123.23**	$\pm$ 1.02
All-Mixtures-RRT*	180.52	$\pm$ 54.83	36.74	$\pm$ 38.12	126.43	$\pm$ 3.83
RRT	1260.54	$\pm$ 1278.96	41.74	$\pm$ 17.87	169.94	$\pm$ 25.93
RRT*	560.78	$\pm$ 397.98	64.29	$\pm$ 35.93	176.68	$\pm$ 43.67
P environment						
Planner	Cost $\mathbf{C_s}$		Plan. time $\mathbf{T_s}$ $[s]$		Traj. length $\mathbf{l_p}$ $[m]$	
CLiFF-RRT*	**1125.16**	$\pm$ 659.0	**11.42**	$\pm$ 14.11	**59.13**	$\pm$ 4.48
All-Mixtures-RRT*	1688.09	$\pm$ 27.48	12.38	$\pm$ 7.68	103.54	$\pm$ 31.1
RRT	2532.96	$\pm$ 798.46	21.16	$\pm$ 19.8	82.87	$\pm$ 23.04
RRT*	1128.06	$\pm$ 454.64	25.43	$\pm$ 18.90	122.61	$\pm$ 34.72
Intersect environment						
Planner	Cost $\mathbf{C_s}$		Plan. time $\mathbf{T_s}$ $[s]$		Traj. length $\mathbf{l_p}$ $[m]$	
CLiFF-RRT*	**182.52**	$\pm$ 28.77	**24.96**	$\pm$ 17.29	**34.71**	$\pm$ 1.00
All-Mixtures-RRT*	307.67	$\pm$ 56.25	29.4	$\pm$ 18.70	51.77	$\pm$ 20.76
RRT	722.15	$\pm$ 373.35	27.78	$\pm$ 25.55	41.97	$\pm$ 10.14
RRT*	298.75	$\pm$ 69.42	27.16	$\pm$ 15.77	47.61	$\pm$ 16.39

Table 6.2: Trajectory quality and planning efficiency.

Roughness **R**				
Scenarios	CLiFF-RRT*	All-Mixtures-RRT*	RRT	RRT*
L	**0.00007** $\pm$ 0.00004	0.00009 $\pm$ 0.0001	0.0023 $\pm$ 0.0048	0.00043 $\pm$ 0.0013
Maze	**0.00002** $\pm$ 0.00004	0.000038 $\pm$ 0.00002	0.00058 $\pm$ 0.00064	0.00053 $\pm$ 0.00062
P	**0.000098** $\pm$ 0.00025	0.0007 $\pm$ 0.0024	0.0007 $\pm$ 0.0014	0.00057 $\pm$ 0.00007
Intersect	**0.013** $\pm$ 0.019	0.013 $\pm$ 0.0104	0.0196 $\pm$ 0.012	0.087 $\pm$ 0.0069

Table 6.3: Trajectory roughness.

Problems solved				
Planner	*L* env.	*Maze* env.	*P* env.	*Intersect* env.
CLiFF-RRT*	**100**%	**90**%	**56**%	**76**%
All-Mixtures-RRT*	48%	10%	2%	14%
RRT	36%	10%	16%	10%
RRT*	34%	14%	16%	24%

Table 6.4: Problems solved for the differential drive robot system.

Av. MSE Velocity Profile V_{MSE}				
Scenarios	CLiFF-RRT*	All-Mixtures-RRT*	RRT	RRT*
L	**0.19639**	0.36516	0.35895	0.50673
Maze	**0.18766**	0.28756	0.36056	0.48896
P	**0.3284**	0.44525	0.48278	0.47637
Intersect	**0.28776**	0.32776	0.31372	0.31779

Table 6.5: MSE of the velocity profile generated by the planners respect to the one learned by the CliFF map.

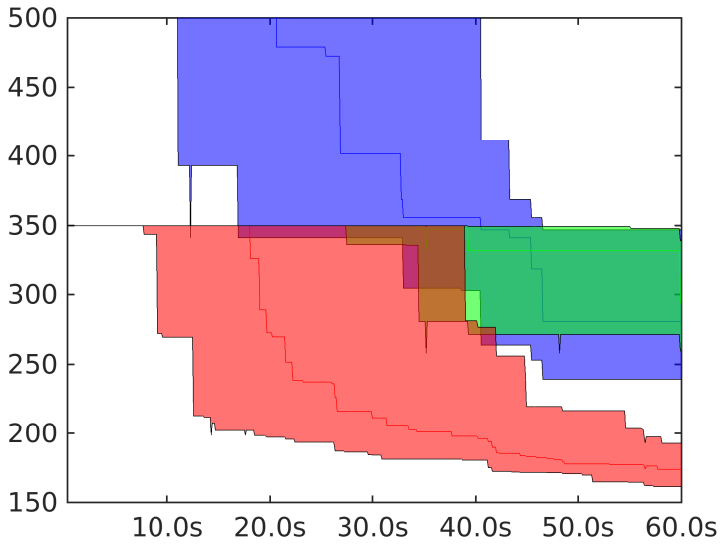

Figure 6.6: Cost convergence plot (median, first and third quartiles computed over 50 runs) respect to the planning time of CLiFF-RRT* (in **red**), All-Mixtures-RRT* (in **green**) and RRT* (in **blue**), for the *Intersect* scenario. The informed sampling allows CLiFF-RRT* to quickly find an initial solution and to converge faster to a lower cost solution than the baselines.

CHAPTER 7

A Fast Random Walk Approach to Find Diverse Paths for Robot Navigation

"One may even say, strictly speaking, that almost all our knowledge is only probable; and in the small number of things that we are able to know with certainty, in the mathematical sciences themselves, the principal means of arriving at the truth - induction and analogy - are based on probabilities, so that the whole system of human knowledge is tied up with the theory set out in this essay"

Pierre-Simon Laplace
Théorie Analytique des Probabilités

The previous chapters introduce several improvements to RRT and RRT, which are single-query sampling-based motion planners, to efficiently find a single solution to the given pair of start and goal poses. Instead of generating a single solution, an appealing navigation strategy for mobile robots is to find a set of diverse paths among dynamic obstacles to qualitatively reason about multiple path hypotheses to the goal. In this chapter we develop an efficient randomized approach based on weighted random walks, that finds K diverse paths on the Voronoi diagram of the environment, where each path represents a distinct homotopy class. We show experimentally that our approach is significantly faster at finding paths of higher diversity in distinct homotopy classes than two state-of-the-art methods. Moreover, we also prove that our method is probabilistically complete.*

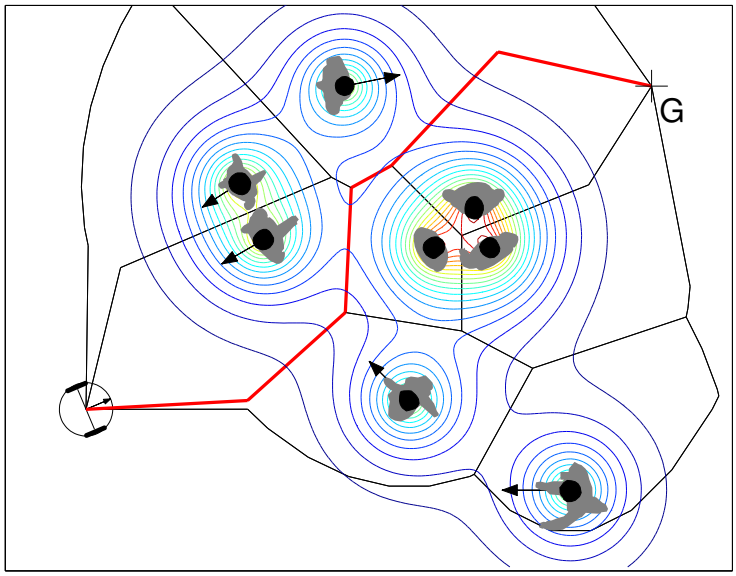

Figure 7.1: An example path selected from the K best homotopy classes in the Voronoi diagram. The robot is enclosed in the **black** circle, in **red** the path selected to reach the goal position **G**. The **black** Voronoi diagram describes the possible ways to go through a crowd by implicitly encoding different homotopy classes.

7.1 Introduction

Single-query robot motion planning generates a single solution from start to goal under a cost function such as shortest path or maximal traversability. In the presence of unmodeled dynamic obstacles a solution may quickly become obsolete, and the path has to be replanned for each change in the environment. Alternatively, one can consider an algorithm that reactively computes and maintains multiple diverse solutions. This approach has benefits in a variety of scenarios: a set of diverse paths, that is continuously checked for validity in the presence of unexpected obstacles, reduces the number of replanning queries in highly dynamic environments (Voss et al., 2015), allows for novel human-robot-interfaces in shared autonomy applications (Kuderer et al., 2014; Liu et al., 2012) and highly efficient, qualitative planning paradigms for social navigation. To ensure robustness of the path set against changes in the environment, paths in the set should be spatially well separated, since similar or nearby paths are more likely to be invalidated together. Also in shared autonomy applications having a set of more diverse paths is obviously a beneficial strategy, since switching between very similar paths is of limited if not questionable usability. Furthermore, when generating a set of paths, a reasonable approach would be to have K paths lying in differ-

ent homotopy classes. For example, trajectory optimization methods, which are limited to finding local minima, may benefit from a set of homotopically distinct paths ((Kuderer et al., 2014), also discussed by Pokorny et al. (2014)).

With the goal of efficiently finding a set of diverse, high quality paths from different homotopy classes for robot navigation, we make the following contributions:

a) We present a fast and easy to implement random walk approach to generate a set of K diverse paths belonging to different homotopy classes. Our method introduced in (Palmieri et al., 2017, 2015) is an alternative to solve the K shortest paths problem with deterministic graph search algorithms (Kuderer et al., 2014; Bhattacharya et al., 2010; Vela et al., 2010). We build a navigation graph (i.e. a road-map, introduced in Sec.7.3.2) from the Voronoi diagram of the environment (see Fig. 7.9-Fig. 7.1), where each path represents a distinct homotopy class, and perform a randomized graph search based on random walks. Additionally, we prove that our approach is probabilistically complete, i.e., it finds all the paths, and therefore all the homotopy classes, in the navigation graph.

b) We conduct an extensive evaluation in terms of planning performance and path quality of our approach, comparing it to two baseline methods that generate diverse and homotopically-distinct paths (Voss et al., 2015; Kuderer et al., 2014), and show that our approach is faster and finds more diverse paths (the robot has a more diversified set from where to choose the path to follow), whilst obtaining a negligible loss in path quality. We use the notion of *robust diversity* of a path set to measure mutual spatial separation of paths. To evaluate the quality of paths in the set, we adopt the *normalized cumulative gain* measure which is used to evaluate web search engine's returned results and ranking quality.

The chapter is structured as follows: a brief discussion of related works in Section 7.2 is followed by the description of our approach and its analysis in Section 7.3. Section 7.4 describes the experiments' settings. We discuss the results in Section 7.5. Section 7.6 concludes the chapter.

7.2 Related Work

Prior research has introduced different methods to generate a set of paths from different homotopy classes.

Demyen and Buro (2006) introduce a method that searches on a graph built using constrained Delaunay triangulations. The obstacles are described via polygonal representation. The paths found in the graph belong to different homotopy classes. In contrast to polygonal representation of the environment, our method works on arbitrary occupancy grid input, which is simpler to handle and currently a *de facto* standard to incorporate data from sensors for real-world operation.

Voss et al. (2015) introduce an algorithm that seeks to find a set of diverse, short paths through a roadmap graph. The algorithm searches the graph for shortest paths avoiding a collection of balls - simulated obstacles in the environment. The obtained paths often belong to different homotopy classes. The authors compare their approach to a K shortest paths algorithm and show that, with tolerable loss in shortness, they produce equally diverse path sets more quickly. Compared to Voss, our approach is much faster, it always returns the requested K paths if they exist in the navigation graph, the returned paths always belong to different homotopy classes and generally are more diverse. Furthermore, our approach has only one parameter and its runtime does not depend on the density of the roadmap graph.

Bhattacharya et al. (2010) propose a method for finding different homotopy classes based on A* search over an augmented graph. The graph encodes topological information via the H-signature, a complex analysis value that characterizes a homotopy class.

Kuderer et al. (2014) select K best homotopy classes by generating K shortest paths in a Voronoi-based navigation graph. During navigation, the paths feed an optimization algorithm used to generate homotopically distinct trajectories. Among those, the best one is selected for navigation. They show that the method is one order of magnitude faster than Bhattacharya's approach (Bhattacharya et al., 2010). Moreover, the authors show that the paths in the Voronoi diagram are safer and better suited for social navigation, as they lie as far as possible from the obstacles among all paths in the same homotopy class. Kuderer's approach employs Katoh's algorithm (Katoh et al., 1982) to find the K best paths in the Voronoi diagram. However, it was shown by Brander and Sinclair (1996) that for small size graphs and paths of small number of vertices, like in the case of Voronoi-based navigation graphs, Yen's algorithm (Yen, 1971) is faster than Katoh's. Therefore, we compare our approach to Yen's algorithm and show that our method is faster and returns more diverse paths. Furthermore, in very complex environments with many homotopy classes, deterministic K best paths are very similar to each other, therefore losing the advantage of having a set of distinct paths. On the contrary, our approach helps to deliver higher diversity in such scenarios, providing high quality paths that explore different regions of the map (see Fig. 7.2).

7.3 A Random Walk Approach to Find Diverse Paths

In this section we detail our approach to find a set of diverse paths lying in distinct homotopy classes. A brief definition of the homotopy class concept and of the navigation graph is followed by the description of the random walk procedure. Next, we prove that our technique is probabilistically complete.

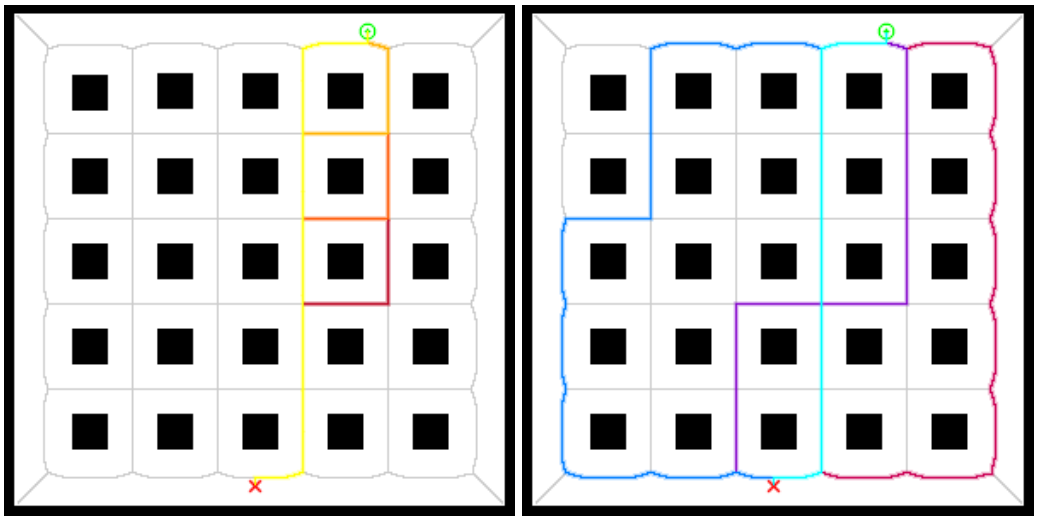

Figure 7.2: Comparison of the paths found by our approach and by Kuderer's approach in the "Cubicles" scenario (best seen in color). The **red** cross represents the robot position, the goal is displayed by the **green** circle. **Left:** Kuderer's 4 best paths in the Voronoi diagram. They all traverse the same region of the space. **Right:** 4 diverse paths found by our approach.

7.3.1 Homotopy Classes

In here we introduce the definition of homotopy and homology classes as also defined by Bhattacharya et al. (2012).

Definition 11 (Homotopy Class)**.** *A homotopy class is defined by the set of paths or trajectories with the same start and goal, in which any two trajectories or paths τ_1 and τ_2 can be continuously deformed into one another without intersecting obstacles, see Fig. 7.3.*

Definition 12 (Homology Class)**.** *An homology class is defined by the set of trajectories with the same start and goal, in which any two trajectories or paths τ_1 and τ_2 (the later with opposite orientation) forms the complete boundary of a 2-dimensional manifold embedded into the configuration space C not containing/intersecting any of the obstacles, see Fig. 7.3.*

Bhattacharya et al. (2012) show that to each homotopy class can be associated a signature $\mathcal{H}(\tau)$ (τ being a trajectory), a general differential 1-form in a given D-dimensional configuration space. The authors proposed a complex-value signature $\mathcal{H}_2(\tau)$ valid for a 2D environment computed based on the *Residue Theorem*, where polygonal obstacles are associated to a single point (or pole).

Vernaza et al. (2012) and Gong et al. (2011) show that finding a signature $\mathcal{H}_2(\tau)$ for a path/trajectory τ is equivalent to find a vector of winding angles

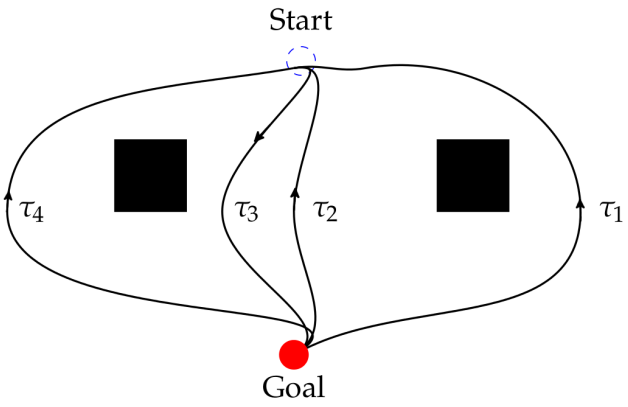

Figure 7.3: Given the position of the obstacles (**black** rectangles) in the scene, different homotopy classes can be identified. Four trajectories (τ_i, $i = 1-4$) connect the same start (**blue** dashed circle) and goal point (**red** circle). τ_2 and τ_3 belong to the same homotopy class since they can be continuously deformed into one another without intersecting any obstacle. They are also homologous, since the area between τ_2 and τ_3 (considering both having opposite direction) does not contain obstacles. τ_4 and τ_1 belong to different homotopy classes since it cannot be continuously deformed into any of the other two.

$(\theta_0, \theta_1, \ldots, \theta_N)$ with respect to all the obstacles in the scene. A winding angle $\theta_i(\tau)$ of a path τ is the sum along τ of the infinitesimal angles $\Delta\theta_i(\tau)$ between the lines starting from two adjacent points of the trajectory and connecting to the obstacle chosen point p_i.

7.3.2 Navigation Graph

To frame the path planning task as a graph search problem, we build the navigation graph of the workspace environment from a Voronoi diagram *VD* generated from the sensor data (see Section 7.4.2 for details on the VD generation). The graph *G(V,E)* consists of a set of nodes (or vertices) V and a set of edges E. Let N be the number of nodes and M the number of edges. $E(v_j)$ denotes the set of incoming and outgoing edges of the vertex v_j. We associate to each edge $e_{ij} = (v_j, v_i)$ a weight or cost c_{ij} (e.g., length of the edge). The adjacency matrix $\boldsymbol{A}$ expresses the topology of the graph G and is defined as

$$[\boldsymbol{A}]_{ij} = \begin{cases} 1 & \text{if } (v_i, v_j) \in E, i \neq j \\ 0 & \textit{otherwise} \end{cases}$$

We compute the set of diverse paths by running our random walk based algorithm on G. A walk w of length $k-1$ in G is a sequence of nodes $v_1, v_2, \ldots, v_k$, where each pair of nodes is connected by an edge, $(v_{i-1}, v_i) \in E$ for $2 < i \leq k$. Henceforward, *walks* are referred to as *paths*. The approach could also be used on road-map graphs generated using different methods (e.g. PRM (Kavraki et al., 1996)), but in our approach we exploit the property, see Sec. 1, of the Voronoi diagram where two different paths with the same start and goal belong to different homotopy classes. In the same situation, road-maps generated by the PRM algorithm may generate different paths belonging to the same homotopy classes: for such a road-map, it would require more iterations (or effort) for an algorithm to find all the homotopy classes in a defined environment.

7.3.3 Randomized Homotopy Classes Finder (RHCF)

To find paths belonging to different homotopy classes we introduce the Randomized Homotopy Classes Finder (**RHCF**), detailed in Alg. 9. Hereinafter all the steps of the while loop in Alg. 9 are referred to as *iteration* of RHCF. We iteratively run the Random Walk procedure (see Alg. 10) on the weighted undirected graph G until K distinct paths are found and stored in the result set $\mathcal{P}$. The walk starts at the initial node $v_s \in G$, where the robot position is mapped to, and aims to find a random path to the goal node v_g. At each step of the random walk we choose a random neighbor of the current node v_j (see `RandomNeighbor`(v_j) in Alg. 10) with probability p_{ij} inversely proportional to the cost c_{ij} associated to the edge e_{ij}

$$p_{ij} = \frac{w_{ij}}{\sum_k w_{kj}} \tag{7.1}$$

with $w_{ij} = \frac{1}{c_{ij}} \boldsymbol{A}_{ij}$ and where $\boldsymbol{A}_{ij}$ is an element of the adjacency matrix $\boldsymbol{A}$. The weights w_{ij} are nonnegative over the entire workspace. The $N \times N$ transition matrix $\boldsymbol{T}$ of the graph G is composed of the elements defined in Eq. 7.1. The node v_j is marked as visited by removing its adjacent edges from the local copy of G_p and the algorithm is not allowed to walk through it again in the current random walk. To bias the search towards a not frequently visited subset of the state space, therefore increasing the probability to generate new paths in the next random walks, we adopt a `DiscountingStrategy`(G, α, v_j, v_i) procedure. Each time we leave a node v_j, the probability p_{ij} associated to the edge e_{ij} is multiplied by a *discounting factor* $\alpha \in (0, 1)$ (for parameter α), therefore, the probability to follow the edge e_{ij} in the next run of the walk decreases

$$p_{ij} := \alpha \; p_{ij}. \tag{7.2}$$

The transition matrix $\boldsymbol{T}$ is then properly normalized. It is worth mentioning that the case of discounting factor $\alpha = 1$ corresponds to an initial version of RHCF

(Palmieri et al., 2015) with no biasing towards unexplored regions. Smaller values of α correspond to heavier bias.

The walk stops when the goal node is found (which means, we have generated a *valid* path) or when we reach a node with all neighbours marked as visited. Each time a valid path $\mathbf{P}$ is generated, we compare it to the ones already found and save it in the result set $\mathcal{P}$ if $\mathbf{P}$ is new, i.e., not generated before. All the visited nodes are then marked unvisited.

Algorithm 9 Randomized Homotopy Classes Finder

function RHCF(v_s, v_g, G, K, α)
$k \leftarrow 0$
$\mathcal{P} \leftarrow \emptyset$
while $k < K$ **do**
  $\mathbf{P} \leftarrow$ `RandomWalk`(v_s, v_g, G, α)
  $P_{new} \leftarrow (\mathbf{P} \notin \mathcal{P}) \wedge$ `isValid`$(\mathbf{P})$
  if P_{new} **then**
    $\mathcal{P} \leftarrow \mathcal{P} \cup \mathbf{P}$
    $k \leftarrow k + 1$
  end if
end while
return $\mathcal{P}$

Algorithm 10 Random Walk(v_s, v_g, G, α)

function RandomWalk(v_s, v_g, G, α)
$v_j \leftarrow v_s$
$G_p \leftarrow G$
$\mathbf{P} \leftarrow v_j$
while $v_j \neq v_g$ **and** $E(v_j) \neq \emptyset$ **do**
  $v_i \leftarrow$ `RandomNeighbor`(v_j)
  $\mathbf{P} \leftarrow \mathbf{P} \cup v_i$
  $G_p \leftarrow G_p \setminus E(v_j)$
  $G \leftarrow$ `DiscountingStrategy`(G, α, v_j, v_i)
  $v_j \leftarrow v_i$
end while
return $\mathbf{P}$

7.3.4 Probabilistic Completeness of RHCF

In Lemma 1 we describe a homotopy-encoding property of the Voronoi diagram. Theorem 1 proves that RHCF finds any arbitrary path in an undirected weighted graph. Finally we prove that RHCF is probabilistically complete in Theorem 2. Here we denote $|\mathcal{P}|_i$ as the size (or cardinality) of the result set $\mathcal{P}$ at iteration i,

Lemma 1. *In the navigation graph $G(V, E)$, built from a Voronoi diagram generated from a 2D environment, two different paths with the same v_s and v_g belong to different homotopy classes.*

Proof. The Voronoi diagram is defined as the set of points in the free space which have equal distance to two or more closest obstacles. The Lemma 1 is derived from the defining property of the Voronoi diagram for 2D environments: only one path between any two obstacles exists. If two paths between two obstacles existed, then they would have different distance to each of those obstacles, which contradicts the definition of the Voronoi diagram. Hence two different paths have at least one obstacle between them, therefore cannot be continuously deformed into one another and belong to different homotopy classes. □

Theorem 1. *Given any arbitrary path in an undirected weighted graph, the* Randomized Homotopy Classes Finder *will find it with probability greater than zero.*

Proof. A random walk is a sequence of transitions (or edges) from a vertex v_j to another v_i where at each step an edge e_{ij} is chosen with a probability higher than zero, $p_{ij} > 0$. We assume that the weights w_{ij} are nonnegative over the entire workspace. Every possible path $\mathbf{P}_k$ in the graph is a concatenation of Z number of edges' transitions,

$$\mathbf{P}_k = \cup_{z=1}^{Z} e_{z,z-1}. \tag{7.3}$$

Given that all the transitions in the graph have a probability greater than zero ($p_{ij} > 0$), every possible concatenation of transitions has a probability greater than zero.

$$Pr(\mathbf{P}_k) = \prod_{z=1}^{Z} p_{z,z-1} = p_{1,s}\ p_{2,1} \dots\ p_{Z,Z-1} > 0, k = 1...K. \tag{7.4}$$

Therefore, any arbitrary path $\mathbf{P}_k$ in the graph has non-zero probability to be found during the random walk. □

Theorem 2. *Consider an undirected graph $G(V, E)$, built from a 2D Voronoi diagram, with nonnegative weights and with encoded K possible paths connecting the start vertex v_s and the goal vertex v_g. Then the probability that RHCF finds all the K paths lying in different homotopy classes, in the graph $G(V, E)$ connecting v_s to v_g, that is the size of the result set $\mathcal{P}$ is equal to K, converges to 1 as the number of iterations n approaches infinity:*

$$\lim_{n\to\infty} Pr\big(|\mathcal{P}|_n = K\big) = 1$$

Proof. By result of Theorem 1, the random walk generates a path $\mathbf{P}_k$ (or walk) from v_s to v_g with a non-zero probability. Each time a valid and new path $\mathbf{P}_k$ is generated, it is added to $\mathcal{P}$ and thus increasing of one the size of $\mathcal{P}$. Given that the

graph $G(V, E)$ is built from a 2D Voronoi diagram, by means of Lemma 1 every new valid path added to $\mathcal{P}$ belongs to a new homotopy class. Given enough time, from Theorem 1, any arbitrary new valid path $\mathbf{P}_k$ can be generated by the random walk and added to $\mathcal{P}$ therefore satisfying $\lim_{n\to\infty} Pr\big(|\mathcal{P}|_n = K\big) = 1$. □

7.4 Experimental Setup

We present a set of experiments to evaluate the performance of our method and show that it outperforms current state-of-the-art algorithms to compute a set of diverse paths presented by Voss et al. (2015) and Kuderer et al. (2014), detailed respectively in Sec. 7.4.5 and Sec. 7.4.6. For our experiments we choose environments of varying complexity and describe appropriate metrics to demonstrate the efficiency of RHCF in terms of planning performance and solution quality.

All the simulated experiments are carried out on a PC with 2.3 GHz Intel Core i5 and 4 GB of RAM. Each reported value is an average of 200 runs. In all the experiments we use path length as the edge costs.

7.4.1 Simulated Environments

We design four simulated environments, shown in Fig. 7.4, to stress different properties of the planner and to study how the algorithm behaves in scenarios of varying complexity. In the *wall of people* scenario, the robot needs to find different ways to the goal through a queue of standing people. This scenario has 36 possible homotopy classes. In the *crowd* scenario, which has 380 possible homotopy classes, the people are placed in a sparser way forming different groups. In the *surrounded* scenario (710 homotopy classes), the robot is placed in the crowd, surrounded by several people. The *corridor* scenario (over 60000 homotopy classes) represents a challenging situation with a crowded corridor and dozens of people, walking alone and in groups. In all the environments we assume that humans poses are provided by a people tracker Linder et al. (2016).

7.4.2 Voronoi Diagram

To generate a Voronoi diagram of the environment, we utilize the open-source C++ package developed by Lau et al. (2013): a computationally efficient approach that is based on incremental updates, applicable in dynamic environments, which was shown to dramatically reduce the computation time needed to build (and update) the Voronoi diagram. Lau's algorithm is in the same complexity class as a simple image passing algorithm, i.e., $O(n^2)$ for an $n \times n$ input map. Lau et al. (2013) also show that in very narrow and cluttered environments Voronoi-based planning (first build a Voronoi diagram and then plan over it) outperforms

single-query sampling-based motion planners (like RRT and KPIECE) in terms of planning efficiency.

7.4.3 Performance Metrics

To quantify the planning performance and quality of our approach, we compute the averages and the standard deviations of the following metrics: T_k time to get K paths, nCG_k normalized cumulative gain, RD_k robust diversity of the result set $\mathcal{P}_K$ of K paths returned by the algorithms. For runtime evaluation we are only interested in measuring the time to generate K paths in the navigation graph, excluding the time to compute the Voronoi diagram of the environment or generate the probabilistic roadmap graph (PRM) that is used by Voss' algorithm. We report the time to build the navigation graph separately.

The *robust diversity* measures how large are of the intra-set distances between pairs of paths in $\mathcal{P}_K$. Let us consider the distance between two paths p_a and p_b to be the discrete Fréchet distance $d_F(p_a, p_b)$. We define RD_k as

$$RD_k = \frac{1}{|\mathcal{P}_K|} \sum_{p_a \in \mathcal{P}_K} \min_{p_b \in \mathcal{P}_K, p_a \neq p_b} d_F(p_a, p_b)$$

Higher diversity value indicates better spatial separation of paths in the set. Paths which are closer to each other (lower robust diversity) intuitively have more geometric similarities, e.g. they may belong to the same homotopy class. The Fréchet distance d_F measures the similarity between two curves. Given two curves $\mathbf{S}_1, \mathbf{S}_2$ where each curve is defined as continuous mapping ($\mathbf{S}_1 : [a, b] \to \mathcal{V}$ and $\mathbf{S}_2 : [c, d] \to \mathcal{V}$ with $a, b, c, d \in \mathcal{R} : a \geq b, c \geq d$ and $(\mathcal{V}, \|.\|_2)$ being a metric space), its continuous representation is defined as:

$$d_F(\mathbf{S}_1, \mathbf{S}_2) = \inf_{\alpha,\beta} \max_{t \in [0,1]} (\|\mathbf{S}_1(\alpha(t)) - \mathbf{S}_2(\beta(t))\|_2)$$

with $\alpha : [0, 1] \to [a, b]$ and $\beta : [0, 1] \to [c, d]$.

Informally, it is the minimum length of a leash required to connect a dog, walking along $\mathbf{S}_1$, and its owner, walking along $\mathbf{S}_2$, as they walk along their respective curves from one endpoint to the other without backtracking. We make use of the discrete d_F computed as described by Voss et al. (2015). Consider the two paths $\mathbf{F}$ and $\mathbf{G}$ formed respectively by the points $f_1, ..., f_n$ and $g_1, ..., g_m$. The discrete

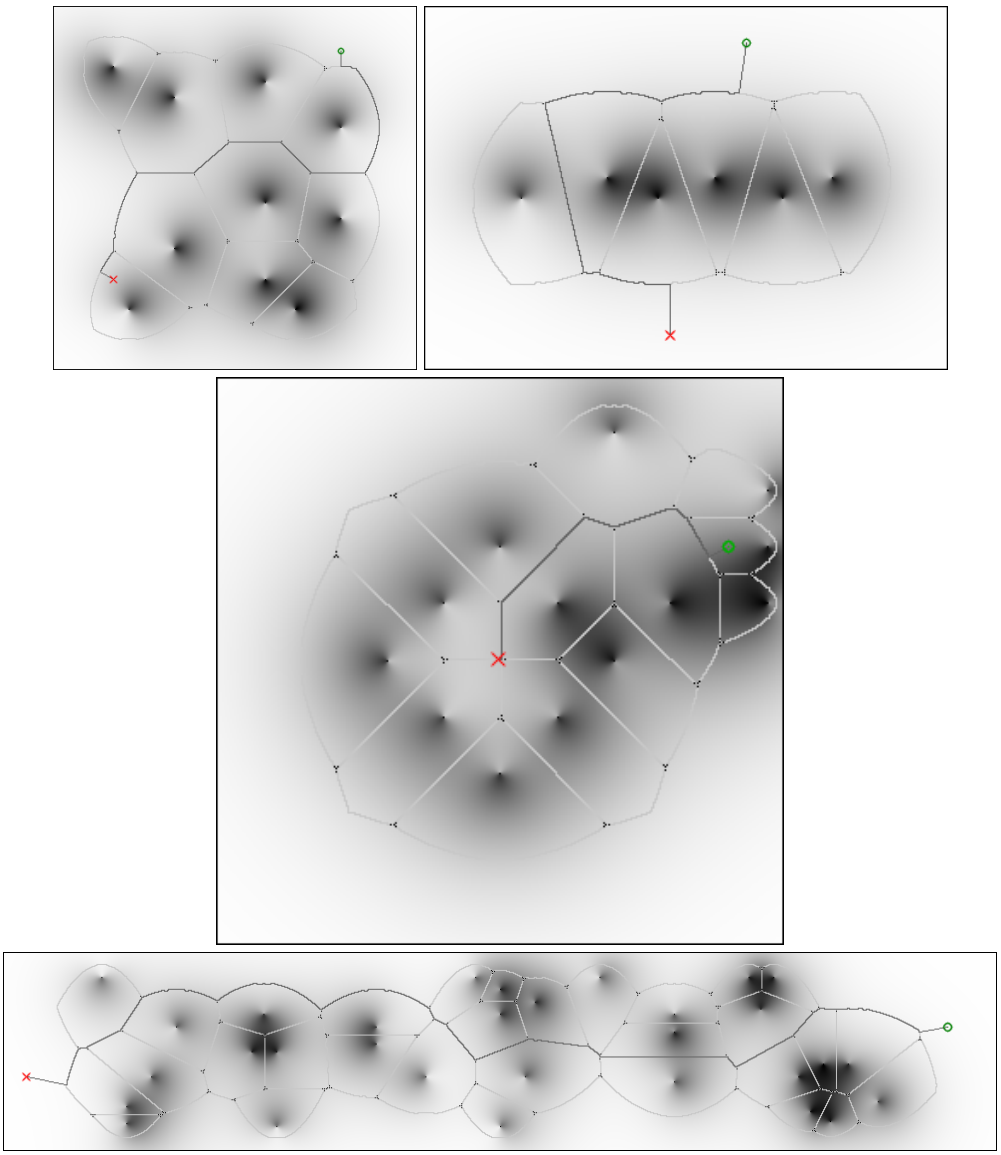

Figure 7.4: **Top left**: *crowd* environment, **top right**: *wall of people* scenario. **Middle**: *surrounded* environment. At the **bottom** we have the *corridor* environment. In all the environments, the grid cells contain obstacles information in terms of cost: the robot is not allowed to go through high cost cells. The grid is generated according to the humans poses and their personal space (Sisbot et al., 2007). Personal space of human agents is displayed in **grey scale**, with **darker** regions corresponding to **higher cost**: its peaks represent the agent positions. The **red** cross represents the robot position, the goal is displayed by the **green** circle. The edges of the Voronoi diagram are in **dark grey** and example paths generated by our approach are displayed with **black** edges.

Fréchet distance d_F is equal to:

$$
\begin{aligned}
d_F(-1,-1) &= 0 \\
d_F(i,-1) = d_F(-1,j) &= \inf \ \forall i,j \geq 0 \\
d_F(i,j) = d_F(-1,j) &= \inf \ \forall i,j \geq 0 \\
d_{Fmin} &= min \begin{cases} d_F(i,j-1) \\ d_F(i-1,j) \\ d_F(i-1,j-1) \end{cases} \\
d_F(i,j) &= \max \begin{cases} \|f_i - g_j\| \\ d_{Fmin} = \min \begin{cases} d_F(i,j-1) \\ d_F(i-1,j) \\ d_F(i-1,j-1) \end{cases} \end{cases}
\end{aligned}
$$

for $0 < i \leq n, 0 < j \leq m$.

The *normalized cumulative gain* nCG_k is used to evaluate ranking performance of web search engine algorithms (in our tests we use a simplified version of the normalized discounted cumulative gain, see Wang et al. (2013)). It computes how far is the candidate ranking set (e.g. a set of K random paths) from the ideal ranking set (a set of K best paths). nCG_k is based on the definition of relevance (*rel*) of a single path. We define the relevance as the inverse of the path cost:

$$rel(p) = \frac{1}{cost(p)}$$

To paths with smaller costs correspond higher *rel* values. The cumulative gain CG_k of a set of paths $\mathcal{P}_K$ is the sum of *rel* values of all paths in the set:

$$CG_k = \sum_{p_i \in \mathcal{P}_K} rel(p_i)$$

The cumulative gain is normalized by the maximum cumulative gain of K best paths in the graph between the start and goal points:

$$nCG_k = \frac{CG_k}{\max(CG_k)}$$

Therefore the nCG_k of the K best paths, e.g. found by Kuderer's algorithm, equals 1 for any K. In general, $nCG_k \in [0,1]$, with higher values corresponding to sets of paths with lower costs.

It is important to note the trade-off between the quality of paths in the set (e.g., their lengths) and diversity: the best K paths are often very similar to each other, contributing to higher cumulative quality of the path set at the cost of very low

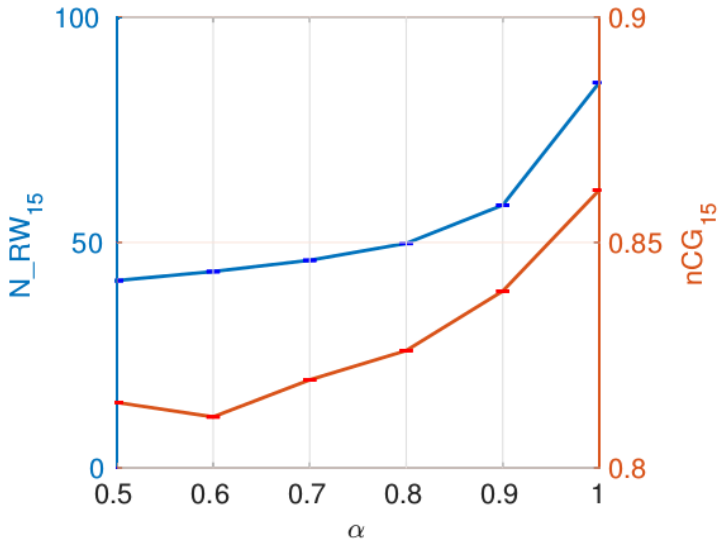

Figure 7.5: Value of the normalized cumulative gain nCG_{15} (in **red**) and number of random walk iterations N_RW_{15} (in **blue**) needed to generate 15 homotopy classes (same trends visible with other values of K) for different values of α in the *crowd* scenario. N_RW_k (consequently also T_k) monotonically decreases with a smaller value of α. The nCG_k has only a slight decrease over the same α range.

diversity (high nCG_k, low RD_k). Adding diverse paths to the set may decrease the nCG_k value. To provide a baseline, in our evaluation we compare the nCG_k value of RHCF to the nCG_k value of a *Uniform Random Path*: a naïve, uninformed algorithm, that samples random K paths uniformly from the set of all possible paths between two nodes.

7.4.4 RHCF Parameters

Prior to the main experiments, we analyze the impact of the single parameter α on the performance of RHCF. After an informal validation, we see that the number of random walks N_RW_k needed to generate K distinct paths, and consequently RHCF runtime, decreases monotonically as α goes from 1 to 0.5 (see Fig. 7.5, where we show the results for the *crowd* scenario, but qualitatively similar trends are visible in the other scenarios too). For our experiments we choose $\alpha = 0.8$, where we achieve a good trade-off between planning time and path quality: a smaller α value yields no considerable improvement to the planning time but causes further decrease of the normalized cumulative gain, therefore compromising the path quality.

7.4.5 Voss's Algorithm

Voss' algorithm seeks to generate a set of diverse paths in the roadmap navigation graph (PRM) of the environment. We evaluate Voss' algorithm on the following metrics: T_k time to generate K paths, and RD_k robust diversity of the result path set. We use a C++ implementation, provided by the authors of the paper. Parameters of Voss' algorithm are set empirically, following the suggestions from their paper: we choose the recommended branching factor $b = 2$, we set the ball radius ρ in each scenario individually to the highest value that still returned an average of 80% of K paths requested, typically $\rho \approx 0.1 - 0.2$. The number of PRM samples is set to 100-500, depending on the complexity of the scenario. We use the C-space distance for simulated obstacles placement and the filtration step of the algorithm accepts all paths.

7.4.6 Kuderer's Algorithm

Both Kuderer's algorithm and RHCF seek to find paths on a navigation graph based on the Voronoi diagram of the environment. Kuderer's approach employs Katoh's algorithm to find the K best paths in the graph. For a fair runtime comparison, we consider three popular algorithms for finding the K best paths (Yen's, Katoh's and Eppstein's) and choose the Yen's algorithm (Yen, 1971) for comparison. Yen's algorithm finds K shortest loopless paths for a given pair of start and goal poses. The algorithm's computational upper bound increases linearly with the value of K: with modern data structures it can be implemented in $O(KN(M + Nlog(N)))$ worst-case time. We use the C++ implementation by Martins and Pascoal (2003), which is reported to have better performance than the straightforward implementation. Yen's loopless K best paths have higher diversity than the paths with loops found by Eppstein's algorithm. Additionally, as shown by Brander and Sinclair (1996), for small size graphs and paths with a small number of vertices (like the graphs generated from a Voronoi diagram), Yen's algorithm is faster than Katoh's.

We evaluate Kuderer's algorithm on the following metrics: T_k, RD_k. We also measure the nCG_k value of our K random paths with respect to the K best paths found in the Voronoi diagram by Kuderer's algorithm.

7.5 Results and Discussion

Tables 7.1-7.5 collect the empirical results generated for all the scenarios. The best values are highlighted in boldface. RHCF significantly outperforms the baselines with respect to all the performance metrics. Moreover, we test the approach in real-world experiments by applying it to socially-aware navigation in dynamic

environments.

$\mathbf{T_k}$ $[ms], K = 10$						
Scenarios	RHCF		Kuderer		Voss	
Crowd	**0.36**	$\pm$ 1.87	1.9	$\pm$ 3.93	183	$\pm$ 135.72
Corridor	**3.86**	$\pm$ 4.90	8.85	$\pm$ 4.50	6474	$\pm$ 1749
Wall of People	**0.36**	$\pm$ 1.88	0.85	$\pm$ 2.78	242	$\pm$ 540
Surrounded	**0.52**	$\pm$ 2.21	1.8	$\pm$ 3.85	69	$\pm$ 66
$\mathbf{T_k}$ $[ms], K = 50$						
Scenarios	RHCF		Kuderer		Voss	
Crowd	**1.8**	$\pm$ 3.8	7.4	$\pm$ 4.7	6541	$\pm$ 13732
Corridor	**9**	$\pm$ 4.4	38.6	$\pm$ 6.2	98570	$\pm$ 89990
Wall of People	-		-		-	
Surrounded	**2.3**	$\pm$ 4.2	6.7	$\pm$ 4.9	2406	$\pm$ 3234

Table 7.1: Planning time results.

Building Time [ms]		
Scenarios	Voronoi graph	PRM graph
Crowd	**8.2**	130.9
Corridor	**30.7**	530
Wall of People	**5.5**	56
Surrounded	**6**	54

Table 7.2: Navigation graph building time.

Surrounded Scenario	
Algorithms	Planning Time [ms]
RRT	1007.2 ± 1.5
KPIECE	1007.3 ± 1.3
STRIDE	1007.1 ± 1.4
EST	1007.1 ± 1.7
Informed RRT*	1006.9 ± 1.4
RRT*	1006.7 ± 1.3

Table 7.3: Comparison to sampling-based motion planners.

7.5.1 Empirical Results

Table 7.1 shows the planning time results for $K = \{10, 50\}$: our approach is at least two times faster than Kuderer's to find a subset of homotopy classes among those present in the navigation graph. In very complex scenarios the difference in runtime becomes significant, e.g. 18.8 ms vs. 143.9 ms in the *corridor* scenario

$\mathbf{nCG_{10}}$				
Scenarios	RHCF		Uniform RP	
Crowd	**0.7839**	± 0.0628	0.4216	± 0.0454
Corridor	**0.5567**	± 0.0435	0.4781	± 0.0363
Wall of People	**0.8884**	± 0.0451	0.5789	± 0.0929
Surrounded	**0.8009**	± 0.0699	0.3538	± 0.0346

Table 7.4: Cumulative gain results.

$\mathbf{RD_{10}}$				
Scenarios	RHCF		Kuderer	Voss
Crowd	**3.01**	± 0.17	2.76	1.34 ± 0.22
Corridor	**3.83**	± 0.26	2.13	2.97 ± 0.30
Wall of People	**2.51**	± 0.10	2.49	1.44 ± 0.13
Surrounded	**2.05**	± 0.19	1.60	1.04 ± 0.18

Table 7.5: Robust diversity results.

for $K = 200$. Moreover, RHCF is faster than Voss's algorithm by several orders of magnitude, also if the graph building times were considered. Table 7.2 reports the building times for the Voronoi-based navigation graphs and PRM graphs (the latter built using the OMPL library (Şucan et al., 2012)) for each scenario.

Table 7.3 details the planning time obtained by state-of-the-art sampling-based motion planners to find a pure geometric solution for the *surrounded* scenario (also these experiments were carried out using the OMPL library). As pointed out in Section 7.4.2 in very complex environments, like the case of the *surrounded* scenario where start and goal poses are surrounded by obstacles, a Voronoi-based path planner outperforms sampling-based motion planners, whose trees (or graphs) fatigue to escape from narrow corridors and complex areas of the environments. In our evaluation for the case of the *surrounded* scenario, sampling-based motion planners are several orders of magnitude slower to find a single solution to the path planning problem while our approach needs a few milliseconds (in average 8.3 ms) to build the navigation graph and to find 50 different paths. In the other scenarios RHCF and the evaluated sampling-based motion planners are equally fast.

Table 7.4 details the results related to the normalized cumulative gain for $K = 10$. RHCF, whilst being faster, also finds solutions with a gain close to the maximum value of 1, which means the solution quality is very high. As a reference, we provide nCG_k results of the *Uniform Random Paths* (Uniform RP) algorithm that draws samples from the set of all paths between two nodes with uniform distribution. RHCF reveals much higher nCG_k results, indicating its bias towards high quality solutions. Only in one scenario, *corridor*, we have a lower nCG_k: this is due to the higher number of total homotopy classes of the scen-

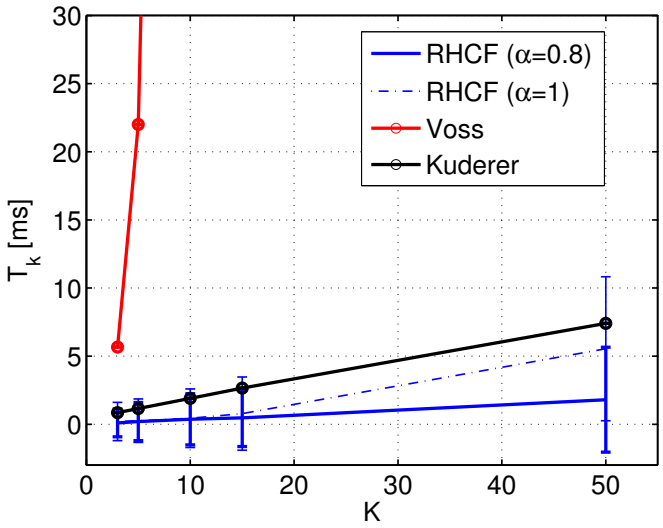

Figure 7.6: Planning time $\mathbf{T}_k$ for different values of K in the *crowd* scenario. In **blue** our approach considering two values of the discounting factor α, in **black** Kuderer's algorithm and in **red** Voss's method. Our approach is faster than all the baselines. The introduction of the discounting strategy ($\alpha = 0.8$) further improved the planning performance of our previous RHCF implementation ($\alpha = 1$).

ario. In this case the above mentioned quality-diversity trade-off is prominent: the best K paths of over 60000 present in the *corridor* scenario are very similar, so introducing a certain degree of diversity into the path set inevitably leads to lower normalized cumulative gain.

Table 7.5 details the diversity of the paths generated by the approaches for $K = 10$: RHCF outperforms Kuderer's and Voss's algorithms in all the environments, delivering more diverse path sets.

Fig. 7.6-7.8 show the metrics trends for different values of K in the *crowd* scenario (same trends are visible in other scenarios). In all figures the standard deviation is depicted with vertical lines. RHCF is significantly faster than the baselines, as Fig. 7.6 indicates. The introduction of the discounting strategy further improved the planning performance of our previous RHCF implementation (Palmieri et al., 2015). Our approach has noticeably higher robust diversity RD_k, see Fig. 7.7: the paths produced are more diverse than the ones generated by the baselines for small values of K. RHCF quickly converges to the optimal value of the normalized cumulative gain as K increases, consequently generating high quality paths (see Fig. 7.8, where the nCG_k trends are reported for all the scenarios).

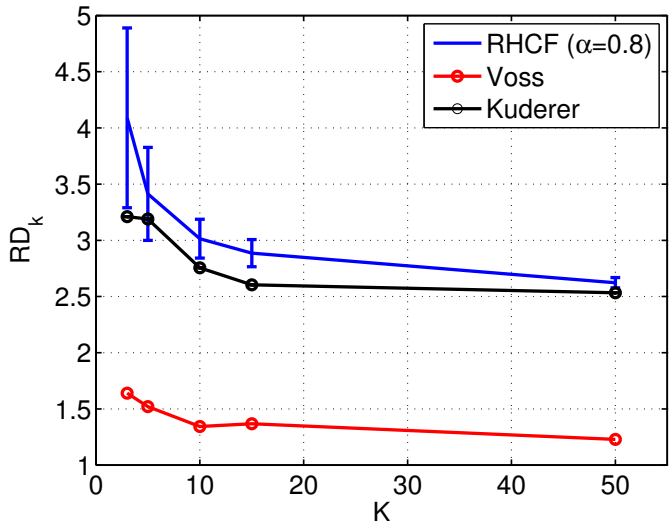

Figure 7.7: Robust diversity $\mathbf{RD}_k$ obtained by varying K in the *crowd* scenario. The paths produced by our approach are more diverse than the ones generated by the baselines for small values of K.

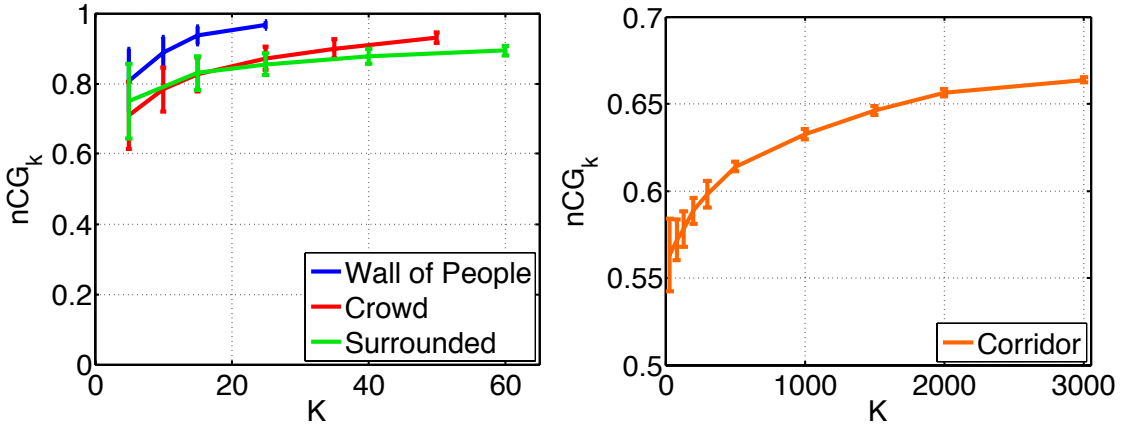

Figure 7.8: Normalized Cumulative Gain $\mathbf{nCG}_k$ obtained by varying K in the *wall of people*, *crowd*, *surrounded* and *corridor* scenarios. As K increases, our approach converges to the optimal value of the normalized cumulative gain $\mathbf{nCG}_k$.

7.5.2 Application to Social Navigation

We conduct further experiments in real-world settings. RHCF allows us to address the task of social robot navigation as a qualitative planning problem that enables a robot to evaluate several diverse paths (see Fig. 7.9, Fig. 7.10) with respect to social costs, more rapidly than the other baselines. More specifically, we

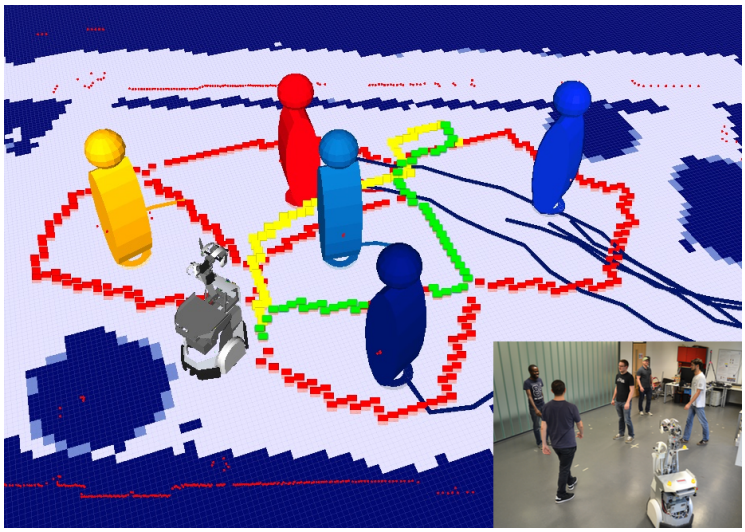

Figure 7.9: We test our approach in real-world experiments by applying it to social robot navigation in dynamic environments. The **red** Voronoi diagram, which implicitly encodes homotopy classes, describes the possible ways to go through a crowd in a room. Our approach rapidly selected two possible paths (in **yellow** and **green**).

integrate the approach in a hierarchical motion planning framework that re-plans at a given frequency ($6Hz$) [1]: firstly a set of diverse paths lying in different homotopy classes is generated from a Voronoi diagram using RHCF, subsequently among those we choose the best path according to a social cost based on the social force model by Helbing and Molnar (1995). Finally, a smooth trajectory with a velocity profile that respects the dynamic constraints of the robot is generated in the chosen homotopy class by using an nonholonomic RRT* based algorithm (Palmieri and Arras, 2014a). Throughout several experiments where the movements of the people were uncertain, our method promptly reacts to the environment changes while generating high quality solutions and respecting the social constraints of the scene. As Section 7.5.1 points out, for this task (where the robot is trapped by a group of people moving around it) using sampling-based motion planners, to plan a path considering the entire environment, would result in higher planning times.

[1] A `ROS` package is available at `https://github.com/srl-freiburg/srl_rhcf_planner`

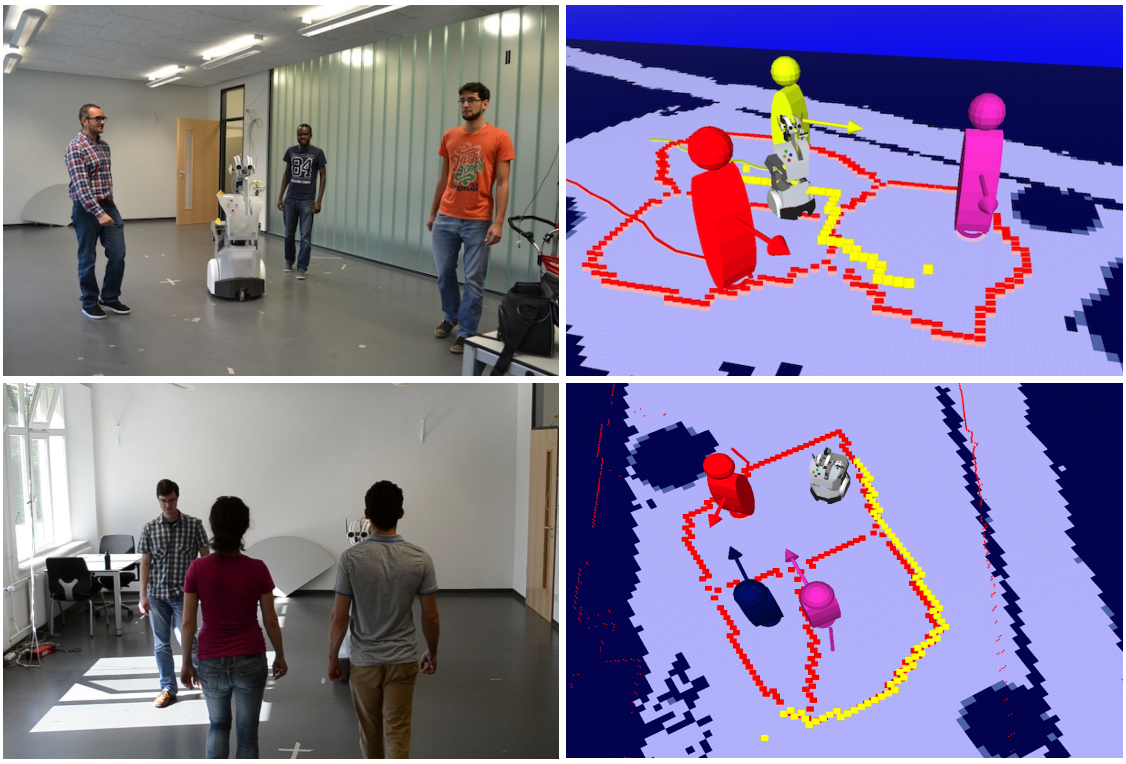

Figure 7.10: Application in social settings. **Upper Left:** dynamic scene with three people walking, the robot has to reach a point in front of them, respecting a social cost. **Upper Right:** The robot reaches its goal by following the **yellow** path selected by **RHCF** from the Voronoi diagram (in **red**). People, tracked by a multi-modal tracker (Linder et al., 2016), are represented by **colored cylinder-shaped objects**. **Bottom row:** three people that interact with each other and robot following the rapidly generated socially acceptable path.

7.6 Conclusions

In this chapter we introduce the Randomized Homotopy Classes Finder that finds a set of K diverse paths belonging to distinct homotopy classes in an undirected weighted graph built from a Voronoi diagram. We prove that our approach is probabilistically complete, i.e., it finds all the paths, and therefore all the homotopy classes. Our experimental evaluation shows that RHCF finds diverse paths significantly faster than two state-of-the-art methods. Moreover, as the cumulative gain results show, the paths produced by our approach are of similar quality to Kuderer's true K best paths. A key property of our method is that it computes a set of more diverse paths with respect to the baselines: usually in dynamic environments spatially separated paths are more robust to invalidation due to

unexpected obstacles, than similar or adjacent paths. Furthermore we test our approach in real-world experiments by applying it to social navigation settings in dynamic environments.

CHAPTER 8

A Socially-Aware Motion Planner for Highly Crowded Environments

"Il semble que la perfection soit atteinte non quand il n'y a plus rien à ajouter, mais quand il n'y a plus rien à retrancher"

Terre des Hommes
Antoine de Saint Exupéry

This chapter presents the socially-aware motion planner for nonholonomic wheeled mobile robots developed based on the findings and analyses carried on in the previous chapters. It describes the techniques to generate motion for the fully autonomous and particularly fast passenger guidance robot of the EU-funded project SPENCER in un-controlled and densely crowded environments. Different path planning algorithms and a novel multi-hypothesis planning technique, have been adapted and combined into a system that generates smooth robot motion among people, while respecting social norms and being efficient in accomplishing the guidance tasks. The approach has been successfully and extensively tested during the final deployment of the Spencer robot at Schiphol Airport in Amsterdam, The Netherlands.

8.1 Introduction

One of the goal of this thesis is the development of a motion planning system that generates smooth and efficient robot motion in crowded environments. As detailed in Chapter 1, for robots navigating in dynamic and crowded environments, a motion planner needs to be *efficient* and to generate *smooth* robot movements. *Planning efficiency* would allow the planner to promptly counteract unpredictable environment changes. *Smoothness of robot movements* helps to generate natural behaviors for a robot navigating among humans; Kruse et al. (2013) write: "one aspect of natural motion is smoothness. This refers to both the geometry of the path taken as well as the velocity profile". Smoothness relates to *legibility*: Kruse

et al. (2014); Lichtenthäler and Kirsch (2013) report and show that legible robot motion can be generated by adjusting the velocity profile, thus slowing down, when humans are closer or intersect the robot.

Generating motion by balancing efficiency with smoothness is not straightforward. Kruse et al. (2013) describe the contradiction between *efficiency* and *smoothness* in context of human-aware navigation: "In Human Robot Interaction, the assumption is that the shortest or most energy efficient path is not necessarily the most desirable. Instead in HRI the intention is to find a path that is also sufficiently safe, comfortable, natural, legible, etc. to persons in the area". If robot motion are smooth enough a human operator could easily infer the robot intentions, thus being legible from a human point of view.

With the insights gained in the previous chapters, where we have understood limitations and advantages of several single-query sampling-based motion planners and of homotopy-based path planners, we now deploy a socially-aware multi-hypothesis planning architecture on a real robot in a real-world application. The approach quickly generates smooth solutions among people while respecting social norms and efficiently solving robot-guidance tasks. The system presents a hierarchical combination of two planners, a global and a local one, that interact with each other by choosing different behaviors (being efficient or polite) for the robot. Based on a multi-hypothesis test, the planners choose the behavior to apply and efficiently generate kinematically feasible motion. The approach has been successfully and extensively tested during the final demo of the SPENCER project at Schiphol airport in Amsterdam, The Netherlands, on circa 50 km of fully autonomous navigation.

The chapter is structured as follows: the introduction of the Spencer project and its robot in Section 8.2 is followed by the description of the socially-aware motion planner in 8.3.

8.2 The SPENCER Project

SPENCER is an EU-funded FP7 research project[1] in the area of robotics, formed by a consortium of six universities, namely Albert-Ludwigs-Universität Freiburg (coordinator), Technische Universität München, Universiteit Twente, Örebro University, Centre National de la Recherche Scientifique (CNRS), Rheinisch-Westfälische Technische Hochschule Aachen and two industrial partners, the Dutch airline company KLM and the Swiss robotics company BlueBotics SA.

The SPENCER project addresses a very interesting business case, highly motivated by actual challenges in the aviation industry, specifically by the growing

[1] **Name of the project**: Social situation-aware perception and action for cognitive robots. **Date**: from 1 April 2013 to 31 March 2016. **Call** FP7-ICT-2011-9.

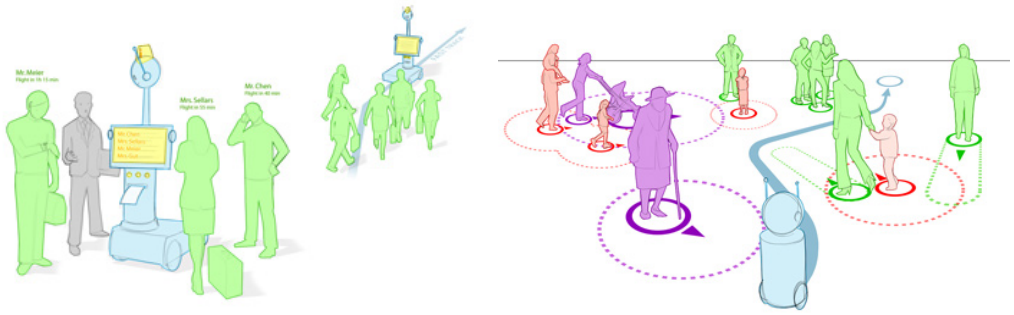

Figure 8.1: SPENCER concepts. **Left:** the robot interacts with the passengers which need to find their gate of departure. **Right:** To every tracked passengers and group of passengers in the scene, a cost that describes their social behavior is associated; the robot moves smoothly among the people by respecting their implicit social spaces.

needs of airlines to reduce the number of missed flights and delays for transfer passengers at hub airports. Particularly the Dutch airline company KLM, end-user in the SPENCER consortium, is interested in reducing the number of missed flights or delays of the oversea transfer passengers. For several types of transfer passengers, finding an efficient way from the arrival to the departure gate, can be challenging, especially for first-time air travelers, people that have a limited knowledge of foreign languages (and alphabet) and are not used to airport's signs and processes. Moreover, in order to catch a connecting flight to Schengen countries, once transfer passengers from a non-Schengen country arrive at Schiphol, they have to go through a passport control (the Schengen filter). For KLM, missed or delayed connecting flights result in additional costs due to flights re-booking, hotel bookings, baggage reloading, while for the passengers they mean further delays and exasperations.

To solve these challenges, the SPENCER project has developed a prototype robot which efficiently guides transfer passengers from their gate of arrival to their gate of departure.

The SPENCER consortium aimed to build a wheeled mobile robot that differently from earlier tour-guide robot systems (e.g. Burgard et al. (1999) and Siegwart et al. (2003)), perceives, learns and models human social behaviors and uses the latter to generate appropriate real-time actions in highly dynamic crowded environments. The consortium contributed on different fields:

- robust detection, tracking and multi-person analysis of individuals and groups of people;

- recognition of human social relations, social hierarchies and social activities;

Figure 8.2: The Spencer robot at Schiphol. The robot has a screen to interact with the passengers. A boarding pass scanner, located under the screen reads the passengers' gate information. Once the robot has read this information, it guides the passengers to their gate of departure.

- normative human behavior learning and modeling;
- socially-aware motion, task and interaction planning;
- learning socially-aware annotated maps in highly dynamic environments;
- empirically evaluating socio-psychological effects of normative robot behaviors.

After several integration events, the Spencer robot was deployed and successfully tested at Schiphol airport, a very challenging environment for socially-aware mobile robots.

8.2.1 The Demo Environment And Its Challenges

The Amsterdam Airport Schiphol is one of the busiest airports in Europe: according to the statistics[2] published by the airport operator, the number of passengers travelling through Amsterdam Airport Schiphol amounted to 58.2 million passengers (excluding transit direct) in 2015. In the same year the number of people that visited the Louvre, the most visited museum in the world, amounts

[2]For recent statistics please refer to official Schiphol airport web-site http://www.schiphol.nl/

to 8.6 million visitors[3]. This simple comparison intuitively shows that airports are highly crowded and dynamic environments.

Several difficulties due to the airport environment were considered while developing the motion planning system:

- Safety for passengers and robot is very important in terms of harms for the passengers and for the public image of the airport, consortium partners and the robotics scientific community.
- Human movements and velocities are hard to track and predict in crowded environments where usually a lot of occlusions happen, their results' quality therefore should not always be trusted.
- Illumination conditions that could influence the behavior of the robot exteroceptive sensors (e.g. stereo cameras, rgb-d sensors).
- The robot risks to be "freezed", see Fig. 8.4, to be trapped into the crowd due to the lack of free space through the people surrounding it (Trautman et al., 2015).
- Unexpected dynamic objects are disturbing the robot's operation: e.g. hand-luggage carried by humans (see Fig. 8.5), cars to assist people with disabilities or elderly people, several types of vehicles for floor cleaning services at the airports, hand-luggage carts.
- Airports corridors have shops that dynamically change the shape of the environment: e.g. stands of the shops can be moved during the day; chairs and tables of the bars and restaurants can be moved according to the density of people.
- The airport floor is not a flat surface: e.g. moving walkways are used to facilitate passengers movements; some gates are in a upper or lower floor thus requiring stairs to access them.
- 3D obstacles and difficult surfaces for the perception system (e.g. glass doors).

For the final demo of the SPENCER project, the robot was mainly operating into pier B and C (see Fig. 8.3). Pier C is narrower than pier B. The latter is longer and has more shops. Both piers have in the center very long moving walkways. The two piers connect to a junction were a shopping mall is located. The Schengen filter, the passport control, is located at one side of the shopping mall. A large

[3]Refer to http://presse.louvre.fr/ for additional information

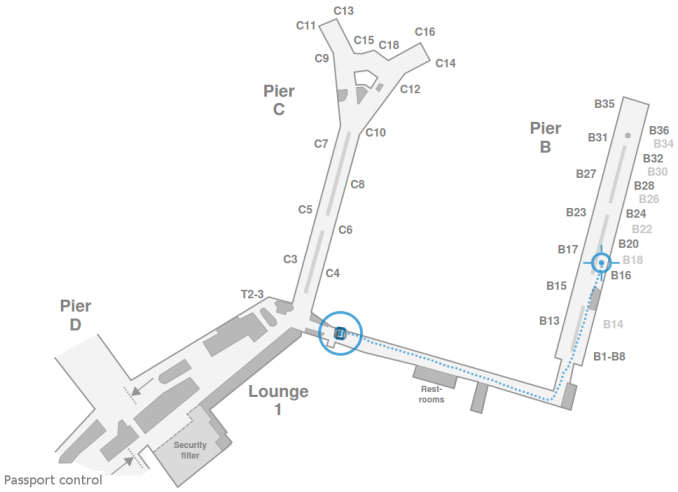

Figure 8.3: Partial representative Schiphol airport map. Main part of the robot operation and the final demo have been carried on in the pier C and pier B of the airport. The robot (in a **blue** circle) follows the planned path (dashed **blue line**) to reach gate B18.

Figure 8.4: The robot faces a crowd at Schiphol, the environment is densely populated and highly dynamic, it has no free ways to the goal. The robot risks to be "freezed", to be trapped into the crowd.

part of the experiments had the robot guiding people from the passport control area to pier B.

During the project and integration events, all these challenges influenced the development of the robot, the motion planning architecture and all the other software components.

Figure 8.5: Passengers with hand luggage interacting with the Spencer robot. Many people were always trying to interrupt the robot during its tasks, by blocking its way or by hindering its movements with their hand luggages.

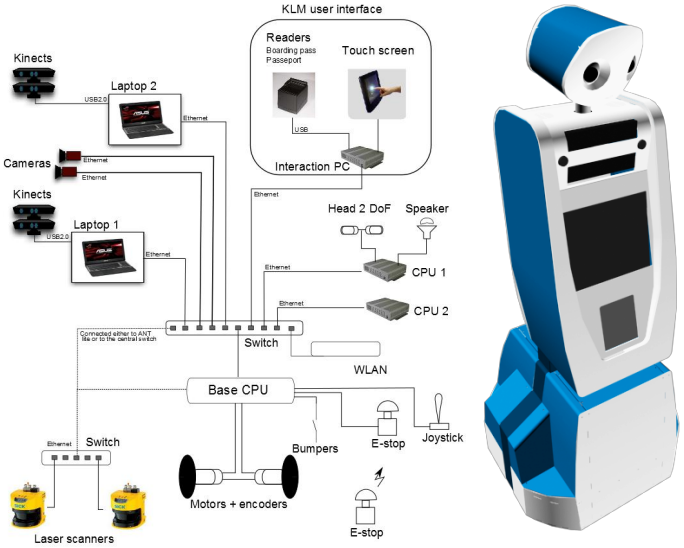

Figure 8.6: SPENCER architecture. **Right:** A 3D view of the Spencer robot CAD model. **Left:** Overview of the hardware configuration running on the Spencer robot.

8.2.2 The Spencer Robot And Its Hardware

The Spencer robot has an abstract human-like appearance, see Fig. 8.2 and Fig. 8.6. Its appearance was designed to increase the overall acceptance of the entire robot by the humans: Its friendly look conveys trustfulness and calmness.

The head is used for a simplified non-verbal communication (e.g. nodding or orientating towards the intended motion direction). The platform has a touch-screen and a boarding pass reader, to resemble an operating information kiosk. During the demo, those allowed the passengers to interact with the robot: by scanning their boarding passes they could start the guidance task of the robot.

The robot has 5 degrees of freedom: 2 for the mobile base with a differential-drive kinematics, 2 for the head (pan and tilt) and 1 for the eyes (only horizontal movements). The height of the robot is of 1926 mm, its base platform is 800×810 mm and its weight (batteries included) is 250 kg. The robot can achieve a maximum velocity of 1.8 $^{m}/_{s}$.

A schematic view of the hardware architecture is given in Fig. 8.6. The robot is equipped with 3 industrial (2 Intel Core i7 and one i5) PCs and 2 gaming laptops with nVidia graphics cards. The latter mainly used by the perception components which require powerful GPUs, while the remaining robot tasks (e.g. planning, localization etc.) are performed on the industrial PCs. The robot drive motors are interfaced to the BlueBotics ANT system, used only as gateway between the motors and the rest of the ROS-based robot software architecture. The sensory setup consists of:

- two SICK LMS 500 2D laser scanners mounted at a height of 0.70 m,
- two front and two rear RGB-D Asus Xtion Pro live cameras,
- two AVT cameras with 4.5 mm lens as stereo camera system mounted at shoulder height,
- Velodyne VLP-16 3D lidar mounted on one of the robot's shoulders (not planned at the beginning of the project but it turned out to be necessary to cope with localization difficulties).

8.2.3 Software Architecture Of The Spencer Robot

All software components developed during the SPENCER project are using the Robot Operating System (ROS) [4]: this middleware is a *de facto* standard for robotics systems, used not only in university laboratories but also by research department units of private companies. The Spencer consortium made extensive use and contributed to the development of its main packages (e.g ros-navigation,

[4]For further information see www.ros.org

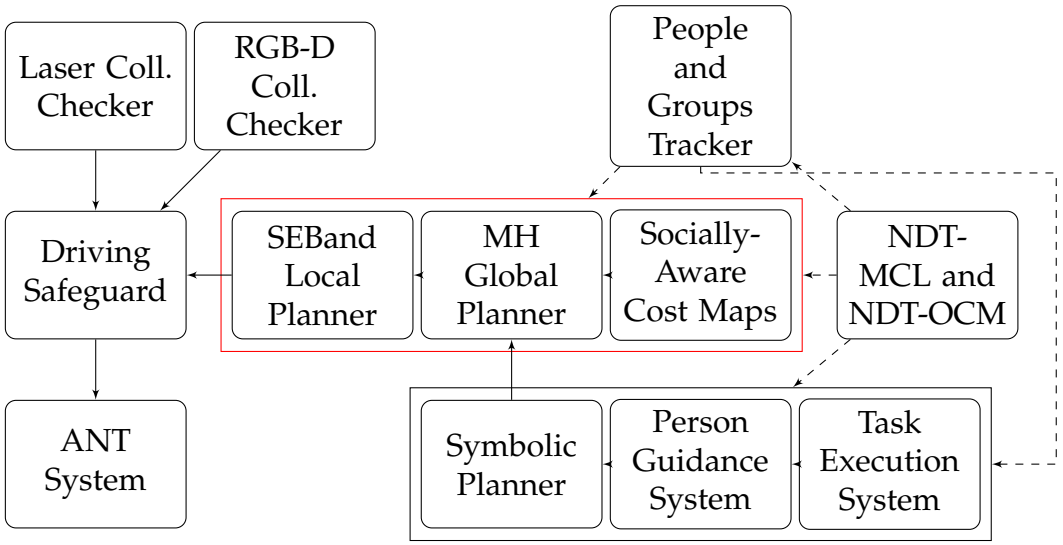

Figure 8.7: Outline of the Spencer navigation architecture. The **red** block includes the components of the motion planning architecture: the socially-aware elastic band (**SEBand Local Planner**), the multi-hypothesis global planner (**MH Global Planner**) and the **Socially-Aware Cost Maps**.

ros-core). Here a brief list of the software components (see Fig. 8.7) developed for the Spencer robot:

- Mapping and localization in large and dynamic environments are very challenging tasks, to solve them the Örebro University group adopted the Normal Distribution Transform Monte Carlo Localisation (NDT-MCL) (Kucner et al., 2015; Saarinen et al., 2013a) and Normal Distributions Transform Occupancy Map (NDT-OCM) (Saarinen et al., 2013b) frameworks, both having as input the 3D point cloud provided by the Velodyne sensor.

- To safely navigate among humans, the University of Freiburg group proposed and used a state-of-the-art real-time multi-modal people and groups tracker (Linder et al., 2016) based on 2D laser and RGB-D data detectors.

- A supervision system (Fiore et al., 2016) to guide people through the airport via a socially-aware POMDP action plan developed by the CNRS group. The method interacts with the motion planning component by sending subgoals. The latter are generated via a symbolic planner that plans over a manually annotated graph, covering the airport's areas where the robot is allowed to move. The user interface management and its connection with the supervision system were performed by a task execution system.

- 2D cost maps with different layers which model the environment around the robot. They are used by the motion planner to generate robots motion. Cost maps encode obstacles (static and dynamic) positions and learned social behaviors (Okal and Arras, 2016).

- Learning approaches to acquire navigation behaviors from demonstrations of socially normative human behavior. The group of the University of Freiburg proposed a flexible graph-based representation (Okal and Arras, 2016) to capture relevant task structure and extend Bayesian inverse reinforcement learning to use sampled trajectories from this representation. The approach enables a robot to learn complex navigation behaviors of varying degrees of social normativeness using the same set of simple features. In Spencer we learned and used three different behaviors as reported in (Okal and Arras, 2016): *polite*, where the robot always avoids breaking pair-wise social relations and intruding into people personal spaces; *sociable*, where the robot gets as close as possible to the people by avoiding to break pair-wise social relations; *rude*, where the robot gets to the goal as fast as possible.

- To further improve the overall safety of the robot platform, a low level collision checker (see Fig. 8.8) runs on the robot at a frequency of 20 Hz. It checks for collisions in two areas (rectangular zones) around the robot: the warning (in **yellow**) and error zones (in **red**). When a segment of laser (or RGB-D) points is in the warning zone, which starts at 60 cm in front of the robot (adapted to the robot speed) and 20 cm to the sides, the translational velocity of the robot is limited to 0.3 $^{m}/_{s}$, the angular velocity is scaled down to keep the initially planned curvature. If the segment is found in the error zone (starting at 35 cm in front and 3 cm to the sides of the robot), the robot is stopped and only motion in the opposite direction side of the virtual collision are allowed. The collisions' status is provided to the driving safeguard node that will act accordingly.

- A driving safeguard node that monitors the collisions status or velocity commands timeouts: in these cases the safeguard node stops immediately the robot.

8.3 Socially-Aware Motion Planner: Combining Efficiency and Social Norms

We now describe a socially-aware multi-hypothesis planning architecture, that was deployed on the Spencer robot for passengers guidance at Schiphol airport. The approach quickly generates smooth solutions among people while respecting

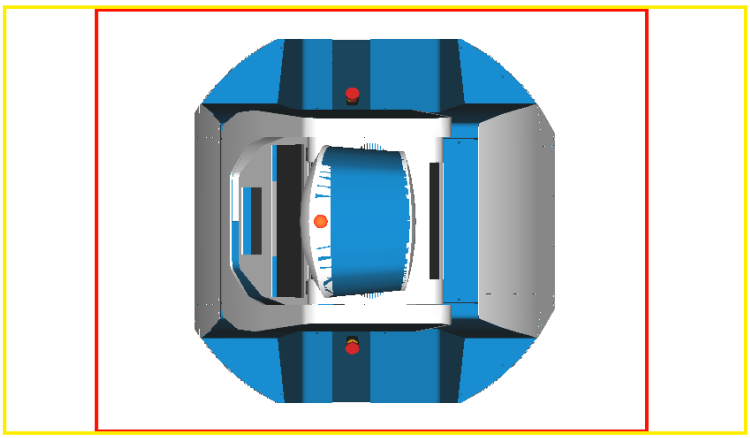

Figure 8.8: The Spencer robot has a low level collision checker that slows down or stops the platform according to the sensors data. If points of a laser segment (or RGB-D point clouds) are inside the warning rectangle (in **yellow**), the maximum robot speed is lowered. If the segment is found in the error zone (in **red**), the robot is stopped and only motion in the opposite side of the virtual collision are allowed.

social norms and efficiently solving robot-guidance tasks. Our approach consists of:

- a global planner that chooses between being socially-aware or efficient according to geometric properties of two path hypotheses and to the presence of a "cluster of people" in front (referred to its moving direction) of the robot;
- a socially-aware elastic band as local planner that while providing obstacle and passengers avoidance, it also generates legible motion;
- a learned “rude behavior” in which the robot ignores many social norms, by reducing the cost inflations associated to the people tracks.

We anticipate that the approach was able to balance task efficiency with social-awareness, significantly reducing erroneous robot behaviors (i.e. freezing or dancing robot (Trautman et al., 2015)) when navigating through a crowd.

8.3.1 The Architecture

The socially-aware motion planner, deployed during the final demonstration at Schiphol airport consists of four components: a multi-hypothesis global path planner, a socially-aware elastic band as local planner, socially-aware cost maps

and the package `move_base` [5]. The latter is a ROS package that defines proper interfaces to interact with sensors and actuators of a mobile base.

Concretely the two planners (local and global) have been implemented as plug-ins for `move_base`. The global planner runs in the same `move_base` thread, while the local planner is in a different thread. Each planner has a different re-planning frequency. The global planner is also asked to re-plan whenever the local planner cannot generate velocity commands.

The multi-hypothesis global path planner plans a path from the current robot position to the provided goal considering the social norms, encoded into the socially-aware cost maps, without neglecting efficiency.

The socially-aware elastic band tracks the geometric path generated by the global path planner while respecting the kinematic constraints of the Spencer robot and being legible for the humans around the robot. Moreover it provides obstacle avoidance functionalities when needed.

8.3.2 Modeling The Environment With Cost Maps

For each planner we use two different cost maps: one provided by `move_base` which encodes the off-line learned social behaviors and all the static and dynamic obstacles of the scene (hereafter referred as *dynamic cost map*) and one that maps only the static obstacles (*static cost map*). The dynamic cost maps have four layers: static, obstacle, social behaviors and inflation. The static cost maps only two: static and inflation. The layers are organized as follows:

- *Obstacle layer:* all the 2D laser points that are not part of person detections predicted by the people tracker and the RGB-D point clouds are considered dynamic obstacles and provided as input to the *obstacle layer* of the dynamic cost maps. The RGB-D point clouds are mainly used to detect obstacles at or below the height of laser, e.g small hand luggage.

- *Inflation layer:* for all the cost maps we inflate the obstacles with the radius of the inscribed circle drawn into the footprint of the robot.

- *Static layer:* it contains the occupancy grid map generated off-line by the NDT-OCM mapping software.

- *Social behaviors layer:* During the deployment, we tested three different off-line learned behaviors: *polite*, *sociable*, *rude*. With the polite behavior the robot always avoids breaking pair-wise social relations and overrunning over the people personal spaces (as in proxemics theory (Hall, 1966)). In the sociable behavior, the robot does not break pair-wise social relations but it gets as close to people as possible. In the rude case, the robot tries to get to

[5] Further information can be found here: http://wiki.ros.org/move_base.

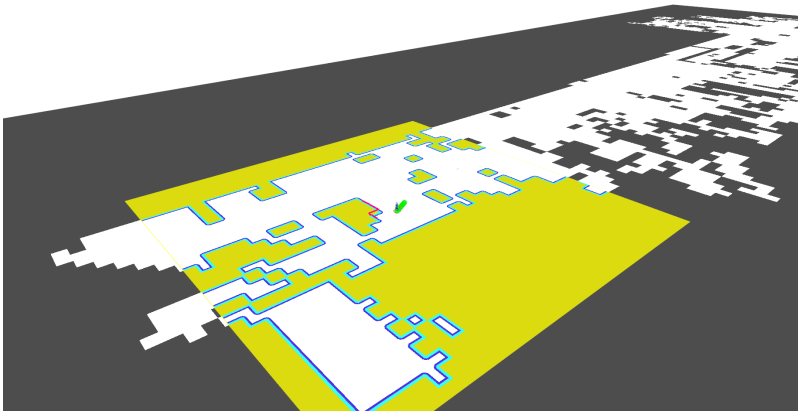

Figure 8.9: The cost maps (here showed mainly in **yellow**) are rolling-windows: they move together with the Spencer robot (in **white** and **blue**). The NDT-OCM (in **black**) provides information to the static layer.

the goal as fast as possible ignoring all social relations and people personal spaces. A more detailed description of the learning algorithm is detailed by Okal and Arras (2016). The *social layer*, by reading the people and groups tracks, represents the desired behaviors in the form of costs. Which social cost (or behavior) to run is highly dependent on the environment, the motion planner architecture and the type of requested efficiency.

All the cost maps are kept as *rolling windows* (see Fig. 8.9), centered on the robot, following its motion through the map. Their *static layer* receives as input the occupancy grid map generated off-line by NDT-OCM and it sub-selects and copies from it the area where the rolling window moves. Costs of the cells are integers and take a value between 0 and 255: occupied cells are marked with a cost equal to 253 or 254; the social behaviors cost map layer maps its cost between 0 and 255.

Global cost maps provided to the global path planner, the dynamic and static one, have a size of 40 m $\times$ 40 m, while both local cost maps provided to the local planner (also in this case one dynamic and one static) have a size of 13 m $\times$ 13 m. The cost maps are updated with a frequency of 4 Hz and have a cell resolution of 0.1 m.

8.3.3 Planning with Multi-Hypothesis

In this section we detail the global path planner deployed on the Spencer robot, the approach aims to balance efficiency (fast accomplishment of the guidance tasks) with social-awareness (navigating through humans without generating hindrances to them). The planner reasons about the difficulties of the crowded

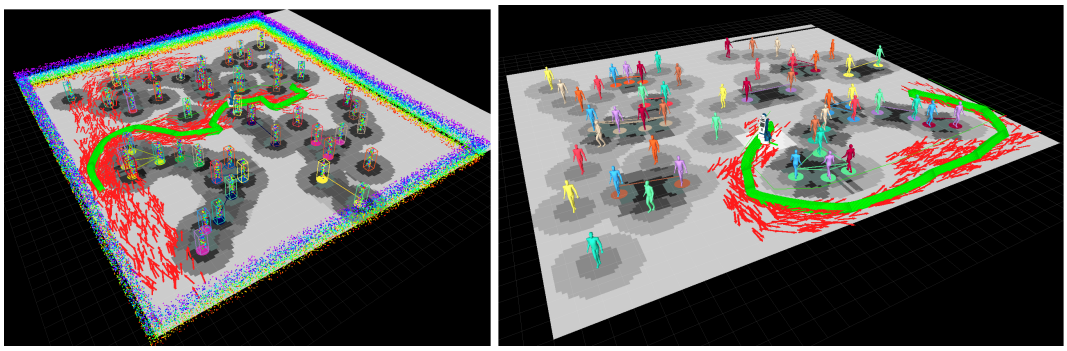

Figure 8.10: Example paths **in green** generated by Theta*-RRT in a pedestrian simulator. In **red** the samples drawn by the algorithm among the simulated pedestrians. The social costs are displayed in **dark grey**.

environments, where a robot often generates erroneous behaviors. As a matter of fact, if the robot is too polite, it may risk to be freezed by the crowd or to generate large detours (as described by Trautman et al. (2015)). Moreover a robot can be too reactive in terms of directions and acceleration, generating unexpected movement oscillations, therefore reducing its motion smoothness and legibility. Reactiveness of the robot in terms of motion direction is highly visible when using a single-query planner. In this case, we may observe direction oscillations due to re-planning while the robot is driving through a crowd. Each time that there is a re-plan, the planner is considering new obstacles and costs thus creating a path which is distant from the previously generated one. The path planner is essentially generating oscillations due to the highly dynamic environment. Also such oscillations may generate detours of the robot e.g. trying to avoid a crowd as one big obstacle.

During the Spencer project, in order to reduce the detailed robot erroneous movements and oscillations, we initially used the rude behavior encoded by the social behaviors cost maps (as detailed in the above subsection) which helped, *to some extent*, by making the platform to be more efficient than socially-aware.

To additionally counteract the described issues, we introduce a novel multi-hypothesis path planning approach (**MH global planner**). The planner reasons on the environment by considering it in two different conditions: first, modeled only by static obstacles and second modeled by the static obstacles and all humans/dynamic obstacles. For both conditions, a path is planned and then compared to each other. The planner may generate two different behaviors (paths) as described in Fig. 8.17:

- **Static-World Path (P_{hsw}):** in this case the planner provides a path generated considering only the static world. The robot keeps going on the static path while locally respecting social-norms, performing obstacle avoidance and a

Human-Robot-Interaction (HRI) action when needed.

- **Socially-Aware Path ($\mathbf{P}_{hsa}$):** the path is generated considering dynamic and static obstacles. The robot can follow the new socially-aware path without encountering large detours or oscillations, also for this case performing locally obstacle avoidance.

The two paths (*static-world* path $\mathbf{P}_{hsw}$ and *socially-aware* path $\mathbf{P}_{hsa}$) are generated considering a different cost map (hypothesis) for each case: $\mathbf{P}_{hsw}$ is generated considering only the static cost map while $\mathbf{P}_{hsa}$ considers the dynamic cost map.

The first path $\mathbf{P}_{hsw}$ can be useful to solve the case in which there is a large crowd in front of the robot, and instead of being blocked by the crowd or avoiding it with a very large and inefficient detour, the robot keeps going on the path that goes through the crowd, while performing a Human-Robot-Interaction (HRI) action. The second path $\mathbf{P}_{hsa}$ is generated when the space in front of the robot is not fully obstructed by the crowd and the robot can follow the new socially-aware path without encountering large detours or oscillations.

The generation of the two paths is performed with the Dijkstra algorithm (Dijkstra, 1959), one of the best choices to generate a low cost solution given a cost function over a 8-connected 2D costmap (grid). Moreover the latter has a known complexity that does not scale with the complexity of the environment (shape and number of the obstacles, like the case of sampling-based motion planners) but rather with the size of the grid: in our implementation without a Fibonacci heap, the complexity is $\mathcal{O}(E \log V)$, with E number of the edges and V number of vertices, edges and vertices of the graph generated from the 2D cost map.

Sampling-based motion planners, although their interesting properties (like the innate respect of the nonholonomic constraints), are still not the best choice for complex, dynamic and large environments like the airports: as seen before, sampling-based motion planners' planning time scale with the complexity of the environment (as previously described in Chapter 3-5, see Fig. 8.10), moreover their results have high variance (i.e for distinct runs, they may generate different paths for the same pair of start and goal).

8.3.3.1 Deciding From The Hypotheses

In order to evaluate the hypotheses, the global path planner chooses the final path by considering both geometric differences between the two path hypotheses and perception information. Specifically for each path we evaluate and compare path length l_p, discrete Fréchet distance d_F, homotopy class, number of people tracks forming a cluster against robot's direction of motion. Concretely the metrics are defined as follow:

- **Path length (l_p):** the path length l_p of a path $\mathbf{P}$, composed of N_p Cartesian points $\mathbf{P}_i$, is computed as:

$$l_p(\mathbf{P}) = \sum_{i=0}^{N_p-1} \|\mathbf{P}_{i+1} - \mathbf{P}_i\|_2 \,. \tag{8.1}$$

  The path length of $\mathbf{P}_{hsw}$ is denoted as l_{phsw}, while the path length for the socially-aware path $\mathbf{P}_{hsa}$ is denoted as l_{phsa}.

- **Fréchet distance (d_F):** The Fréchet distance d_F measures the similarity between two curves. Given two curves $\mathbf{S}_1, \mathbf{S}_2$ where each curve is defined as continuous mapping ($\mathbf{S}_1 : [a,b] \rightarrow \mathcal{V}$ and $\mathbf{S}_2 : [c,d] \rightarrow \mathcal{V}$ with $a,b,c,d \in \mathcal{R} : a \geq b, c \geq d$ and $(\mathcal{V}, \|.\|_2)$ being a metric space), its continuous representation is defined as:

$$d_F(\mathbf{S}_1, \mathbf{S}_2) = \inf_{\alpha,\beta} \max_{t\in[0,1]} (\|\mathbf{S}_1(\alpha(t)) - \mathbf{S}_2(\beta(t))\|_2)$$

  with $\alpha : [0,1] \rightarrow [a,b]$ and $\beta : [0,1] \rightarrow [c,d]$.

  Informally, it is the minimum length of a leash required to connect a dog, walking along $\mathbf{S}_1$, and its owner, walking along $\mathbf{S}_2$, as they walk along their respective curves from one endpoint to the other without backtracking. We make use of the discrete d_F computed as described by Voss et al. (2015). Consider the two paths $\mathbf{F}$ and $\mathbf{G}$ formed respectively by the points $f_1, ..., f_n$ and $g_1, ..., g_m$. The discrete Fréchet distance d_F is equal to:

$$\begin{aligned}
d_F(-1,-1) &= 0 \\
d_F(i,-1) &= d_F(-1,j) = \text{inf } \forall i,j \geq 0 \\
d_F(i,j) &= d_F(-1,j) = \text{inf } \forall i,j \geq 0 \\
d_{Fmin} &= min \begin{cases} d_F(i,j-1) \\ d_F(i-1,j) \\ d_F(i-1,j-1) \end{cases} \\
d_F(i,j) &= \max \begin{cases} \|f_i - g_j\| \\ d_{Fmin} = \min \begin{cases} d_F(i,j-1) \\ d_F(i-1,j) \\ d_F(i-1,j-1) \end{cases} \end{cases}
\end{aligned}$$

  for $0 < i \leq n, 0 < j \leq m$.

- **Homotopy Class Check (Hc):** As described in Chapter. 7, two trajectories or paths τ_1 and τ_2 belong to the same homotopy class (see Fig. 8.3.3.1) if

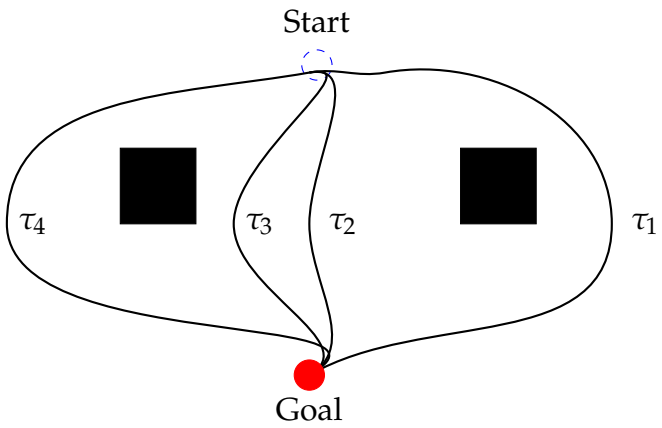

Figure 8.11: Given the position of the obstacles (**black** rectangles) in the scene, different homotopy classes can be identified. Four trajectories (τ_i, $i = 1 - 4$) connect the same start (**blue** dashed circle) and goal point (**red** circle). τ_2 and τ_3 belong to the same homotopy class since they can be continuously deformed into the other without intersecting any obstacle. τ_4 and τ_1 belong to different homotopy classes since it cannot be continuously deformed into any of the other two.

they can be continuously deformed into the other without intersecting any obstacle. Bhattacharya et al. (2012) show that to each homotopy class can be associated a signature $\mathcal{H}(\tau)$ (τ being a trajectory), a general differential 1-form in a given D-dimensional configuration space. The authors proposed a complex-value signature $\mathcal{H}_2(\tau)$ valid for a 2D environment computed based on the *Residue Theorem*, where polygonal obstacles are associated to a single point (or pole). Vernaza et al. (2012) and Gong et al. (2011) show that finding a signature $\mathcal{H}_2(\tau)$ for a path/trajectory τ is equivalent to find a vector of winding angles $(\theta_0, \theta_1, \ldots, \theta_N)$ with respect to all the obstacles in the scene. A winding angle $\theta_i(\tau)$ of a path τ is the sum along τ of the infinitesimal angles $\Delta\theta_i(\tau)$ between the lines starting from two adjacent points of the trajectory and connecting to the obstacle chosen point p_i.

To check if two paths τ_1 and τ_2 belong to the same homotopy class we compare the sum of the winding angles for each path with respect to the obstacles in the global static map. The obstacles centroids' positions p_i are computed by using the border following algorithm introduced by Suzuki et al. (1985). For efficiency reasons, we evaluate the infinitesimal angles $\Delta\theta_i(\tau)$ of the winding angle only on a subset of points of the path τ. **Hc** equals to true if the two generated paths belong to the same homotopy class, thus having the same vector of winding angles, and false otherwise.

Parameter	Value
α	1.17 m
d_{Fmin}	1.9 m
r_{hw}	2.5 m
θ_{hw}	1.27 rad
N_h	3

Table 8.1: Multi-hypothesis path planner parameters

- **Cluster (or wall) of people tracks (Hw):** to check if the robot is in front of a cluster of pedestrians blocking its way, we exploit the information provided by the people tracker. We count all the tracks that fall into a sector of circle of radius r_{hw} and angle θ_{hw} (r_{hw} and θ_{hw} as parameters), centered around the vector describing the robot current direction. After an informal validation r_{hw} and θ_{hw} have been set respectively to 2.5 m and 1.27 rad. If at least N_h (in our case N_h=3) tracks are into the sector, we declare this as a *human wall* (**Hw** equal to true).

Based on these four criteria, the planner chooses to follow the socially-aware path hypothesis $\mathbf{P}_{hsa}$ when the following two conditions are false:

- We detect a human wall in front of the robot and the socially-aware path $\mathbf{P}_{hsa}$ is much longer (α times, as parameter) than $\mathbf{P}_{hsw}$. In addition their Fréchet distance $d_F(\mathbf{P}_{hsa}, \mathbf{P}_{hsw})$ is higher than a defined threshold d_{Fmin} (as parameter):

$$\mathbf{Hw} \wedge (\alpha l_{phsw} <= l_{phsa}) \wedge (d_{Fmin} < d_F(\mathbf{P}_{hsa}, \mathbf{P}_{hsw})) \tag{8.2}$$

- We detect that the paths do not belong to the same homotopy class ($\neg$**Hc** is true) and the socially-aware path length l_{phsa} is α times bigger than l_{phsw}

$$\neg Hc \wedge (\alpha l_{phsw} <= l_{phsa}) \tag{8.3}$$

The parameters of the global path planner, as used on the robot Spencer, are reported in Tab.8.1.

Once the global path planner has chosen one of the two hypotheses, it forwards the path to the local planner together with its type, so that the socially-aware Elastic Band (the current local planner, see below) can select the proper cost map and activate the proper HRI action.

The multi-hypothesis planner runs with a frequency of 0.1 Hz moreover replanned is invoked whenever the elastic band cannot generate velocity commands.

8.3.4 Socially-Aware Elastic Band

The multi-hypothesis path planner is coupled with a modified version of the Elastic Band (Quinlan and Khatib, 1993) path planner. The elastic band is a dynamic path optimization algorithm that locally optimizes (adapts) the global plan based on the obstacles' positions and in general on environmental changes. Each obstacle pushes the band away by means of an elastic force. We modified the elastic band to better couple it with our multi-hypothesis global path planner and to generate legible and socially-aware motion.

We adapt the band by using cost maps that encode obstacle positions and social behaviors of the robot (Okal and Arras, 2016), differently from other previous approaches (as in (Philippsen, 2004)) which used raw sensor information to read only obstacle positions. Moreover velocity commands are generated based not only on the geometry properties of the band and the kinematic constraints of the robot platform, but we adaptively change them to account for legibility (as proposed by Kruse et al. (2014)): the authors show that legible robot motion can be generated by adjusting the robot velocity profile, thus slowing down, when humans are closer or intersect the robot movements.

8.3.4.1 Following The Path Hypothesis

We describe now the proposed algorithm. The socially-aware elastic band planner (**SEband planner**) generates velocity commands such that the robot does not collide with obstacles and humans, it follows the path generated by the global path planner and activates the HRI action accordingly; moreover its velocities are legible by the humans and do not violate the dynamic constraints of the actuators (see Fig. 8.13 for an overview of its architecture).

Each time that the global path planner provides a new path, our socially-aware elastic band planner selects the proper local cost map (*static* or *dynamic* one) based on the hypothesis that the robot is following. In case that the *static* path $\mathbf{P_{hsw}}$ is selected the SEband planner activates its HRI module: every time that N_h (as parameter, see Tab.8.1) people tracks are in front of the robot direction (and stopping the robot), the HRI action is triggered to avoid that the robot freezes. During the SPENCER project final demonstration, the HRI action consisted in the robot asking for permission to pass, by saying "Excuse me". When the *socially-aware* path hypothesis $\mathbf{P_{hsa}}$ is selected, the robot avoids unexpected obstacles-pedestrians by adapting the path through the elastic band algorithm.

8.3.4.2 Adapting The Path To The Dynamic Environment

Here we detail the elastic band method and our modifications to take in account the learned social behaviors. The elastic band approach (Quinlan and Khatib, 1993), instead of representing the entire free configuration space of the robot,

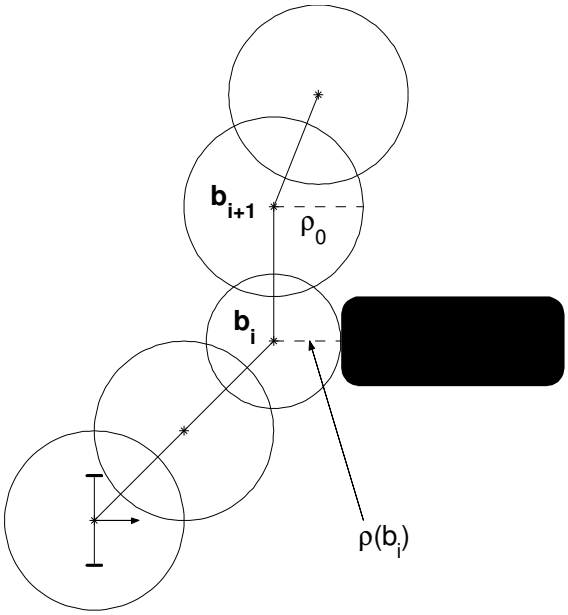

Figure 8.12: Elastic band approach. The path is covered with a set of bubbles (**black** circles). Their initial radius is equal to the mask distance ρ_0. Their volume (area in the specific case of a 2D environment) is squeezed according to the distance, $\rho(\mathbf{b}_i)$, to the nearest obstacle (**black** rectangle).

models the free environment only locally as a union of free space subsets. Each subset is a bubble centered on a global path point $\mathbf{b}$ and expanded according the local freedom of the robot at that path point (see Fig. 8.12). The free space subsets are determined based on the relative distance $\rho(\mathbf{b})$ of the robot at configuration $\mathbf{b}$ (considering its shape and kinematics) to the closest obstacle in the scene: in such a situation the robot can move freely without collisions in all directions up to the distance $\rho(\mathbf{b})$. Differently to previous approaches, we compute the distance $\rho(\mathbf{b})$ based on the social behaviors costs encoded into the cost maps.

A free space subset $\mathcal{B}(\mathbf{b})$ (called bubble) at configuration $\mathbf{x}'$ is defined as the set of all robot configurations $\mathbf{x}$ that can be reached from $\mathbf{x}'$ before colliding with an obstacle,

$$\mathcal{B}(\mathbf{x}') = \Big\{\mathbf{x} : D(\mathbf{x}' - \mathbf{x}) < \rho(\mathbf{x}')\Big\},$$

where $D(\mathbf{x}' - \mathbf{x})$ represents the distance between two configurations based on the kinematic constraints and shape of the robot. For a differential drive robot navigating in a 2D environment, where robot and obstacles can be geometrically modeled as 2D polygons, the distance $D(\mathbf{x}' - \mathbf{x})$ between the configurations $\mathbf{x} =$

(x, y, θ) and $\mathbf{x}' = (x', y', \theta')$ can be defined as:

$$D(\mathbf{x}' - \mathbf{x}) = \sqrt{(\Delta x^2 + \Delta y^2)} + r_{circ}$$

with $\Delta x = x - x'$, $\Delta y = y - y'$ and r_{circ} being the radius of the circumscribed circle of the robot footprint.

The distance $\sqrt{(\Delta x^2 + \Delta y^2)}$ in our approach is computed with the cost maps information: it is the distance to the closest high cost cell (to account for the social behaviors, we assume that a cost is high if it is higher or equal to 128).

Once a global path is provided, the elastic band is built: it is formed by a finite series of bubbles $(\mathcal{B}(\mathbf{b}_0), \mathcal{B}(\mathbf{b}_1), \ldots, \mathcal{B}(\mathbf{b}_{N-1}))$, centered on the configurations $(\mathbf{b}_0, \ldots, \mathbf{b}_{N-1})$. To insure that collision free motion can be generated between the bubbles, we impose the condition that they overlap at consecutive via points.

The initial bubble of the band represents the current robot pose, while the last one is centered at the final point of the local path, a portion of the global path inside the local cost map. The initial radius of the bubbles is equal to the mask distance ρ_0 (as parameter, set to 1 m for the Spencer robot): obstacles above that distance are not considered during the band adaptation. At every re-planning cycle of the local planner the elastic band is optimized by moving the bubbles positions (i.e $\mathbf{b}_i, i = 0, \ldots, N-1$) considering a set of internal and external forces.

$$\mathbf{b}_{i(new)} = \mathbf{b}_{i(old)} + f_{tot}(\mathbf{b}_{i(old)})$$

where f_{tot} is sum of internal forces f_{int} and external forces f_{ext}:

$$\begin{aligned}
f_{tot}(\mathbf{b}_i) &= f_{int}(\mathbf{b}_i) + f_{ext}(\mathbf{b}_i), \\
f_{int}(\mathbf{b}_i) &= k_{int}\left(\frac{\mathbf{b}_{i-1} - \mathbf{b}_i}{\|\mathbf{b}_{i-1} - \mathbf{b}_i\|} + \frac{\mathbf{b}_{i+1} - \mathbf{b}_i}{\|\mathbf{b}_{i+1} - \mathbf{b}_i\|}\right) \\
f_{ext}(\mathbf{b}_i) &= -\frac{k_{ext}}{2b_{exp}}\begin{bmatrix} \rho(\mathbf{b}_i - b_{exp}\hat{\mathbf{x}}) - \rho(\mathbf{b}_i + b_{exp}\hat{\mathbf{x}}) \\ \rho(\mathbf{b}_i - b_{exp}\hat{\mathbf{y}}) - \rho(\mathbf{b}_i + b_{exp}\hat{\mathbf{y}}) \end{bmatrix}
\end{aligned}$$

with b_{exp} being $\rho(\mathbf{b}_i)$ (k_{int}, k_{ext} as parameters). As described by Quinlan and Khatib (1993) to mitigate the movement of bubbles along the elastic band, we remove the tangential component from the total force applied to the elastic band. The number of bubbles is not constant, according to the obstacles movements (in this case cost maps costs changes) bubbles are inserted when the overlap between adjacent bubbles is not sufficient, and superfluous bubbles are removed from the band in order to be efficient during its optimization. The parameters of the elastic band optimization algorithm are reported in Tab. 8.2

Parameter	Value
k_{int}	1.5
k_{ext}	0.5
ρ_0	1 m

Table 8.2: Elastic band optimization parameters

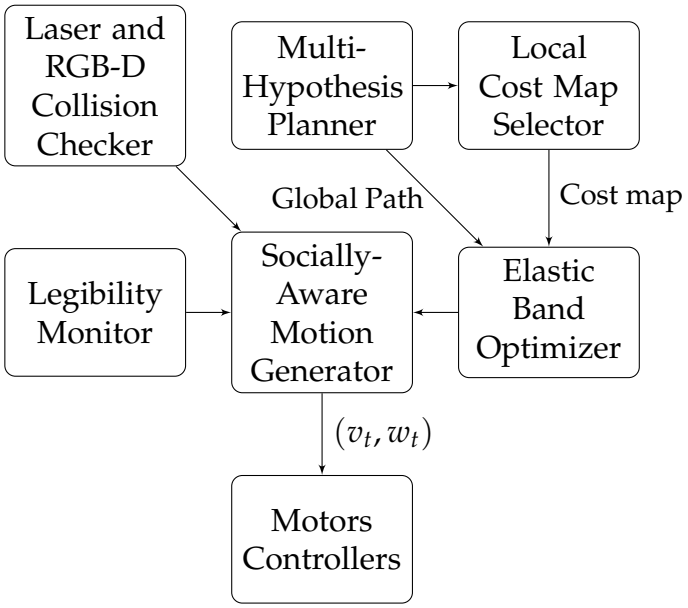

Figure 8.13: Outline of the socially-aware elastic band planner architecture. In the socially-aware elastic band (**SEBand Local Planner**) many components interact with each other. The multi-hypothesis path planner generates a path and selects a cost map which are both provided to the elastic band optimizer. The socially-aware motion generator will generate velocities according to the optimized elastic band, the legibility, the collision status and the given kino-dynamic constraints.

8.3.4.3 Socially-Aware Motion Generation

Following the motivation detailed in (Kruse et al., 2014), we compute the pair of velocities (v_t, w_t), provided to the robot motors, by considering robot motion *legibility* (i.e. adjusting the velocity profile, thus slowing down, when humans are closer or intersect the robot). Moreover the velocities are generated by taking in account the bubbles forming the band and respecting the kinematic constraints of the platform.

Although different control strategies can be applied (e.g. path tracking or trajectory tracking (De Luca et al., 2001)), we make use of a simpler pure pursuit con-

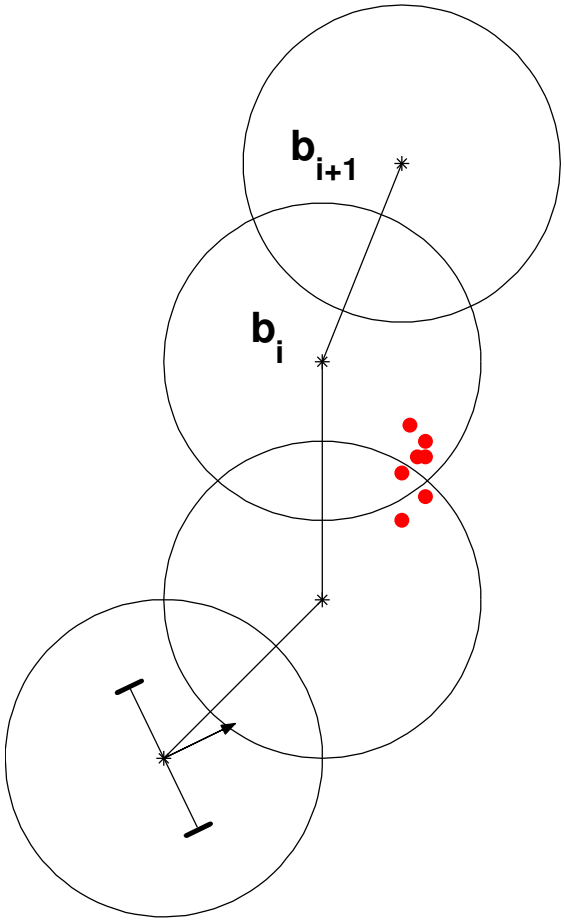

Figure 8.14: The elastic band covers the path that the robot has to follow. While generating the velocity commands the socially-aware elastic band slows down in case of un-expected dynamic obstacles (in the image represented with **red** laser scan dots) that suddenly appear and jump on the virtual elastic band. In this case the robot slows down and optimize the band so to avoid the un-expected obstacle or pedestrian.

troller (Coulter, 1992) to better cope with the high dynamic environment, where at each time step we generate the velocities by looking only at one step ahead (in terms of bubbles). This approach leads to an efficient generation of smooth and continuous motion. Given that the robot at time t is mapped to the initial bubble $\mathcal{B}(\mathbf{b}_0)$, we make use of a pure pursuit strategy that operates on the error between the current robot heading and the heading to the next bubble point $\mathbf{b}_0$ and the Euclidean distance between the two bubbles. The feasibility of the velocities is checked against the desired accelerations (see Tab.8.3 for the acceleration and velocity limits used during the SPENCER demo). To improve legibility of

Parameter	Value
v_{max}	1.3 m/s
0.65 v_{leg}	0.65 m/s
w_{max}	1.57 rad/s
max trans. acc.	0.125 m/s^2
max ang. acc.	0.1 rad/s^2

Table 8.3: Socially-aware elastic band motion generation parameters

the robot motion, the velocities of the robot, generated by the SEband planner are scaled based on the distance of the laser scans readings to the current band (see Fig. 8.14): when some laser points (at least 4) are found inside the elastic band, the maximum translational velocity v_{max} is limited to v_{leg} (see Tab.8.3). To reduce the oscillations in terms of translational velocity (thus improving legibility), we did not adopt people tracking information, given that the tracker provides the correct information only within a given confidence. To improve legibility "a priori" we use low accelerations for translational and rotational movements too. This planner runs with a frequency of 6 Hz.

8.4 Final Demo At Schiphol

During the final integration weeks of the SPENCER project in Schiphol (one took place from November 30 to December 5, 2015 and an other one from March 10 to 24, 2016), we found that an airport is not only a very crowded environment, it is also a place where humans are very goal-oriented, partially in a rush, and less attentive of what is happening around them. For those passengers, even an intelligent robot that usually attracts a lot of attention, becomes an imperceptible object to which they do not pay particular attention. Moreover passengers often, during their navigation in the airport, do not cooperate with the robot and/or other passengers: therefore in such cases, a method that would start a tentative cooperation between the robot and the humans, as described by Trautman et al. (2015), would not improve the overall robot navigation quality. Our tests in the airport led to the conclusion that under such conditions a robot that fully respects all learned social norms (e.g. in the learned “polite behavior”), ends up in being overly reactive and/or partially freeze in highly dynamic crowds of fast moving people.

To reduce the number of such freeze-events, we adopted the motion planner (described in the previous sections) to achieve an overall "socially-aware and efficient" behavior of the robot.

Our approach significantly improved the robot efficiency (time to reach the

Figure 8.15: The Spencer robot while testing its navigation capabilities at Schiphol.

Date	Dist. Auton.	Dist. Not Auton.	Dist. per day	Day's event
09/03/16	0	4210	4210	Mapping session
10/03/16	589	1055	1644	
11/03/16	2168	2701	4868	
12/03/16	371	1555	1926	Mapping session
13/03/16	3475	1099	4574	
14/03/16	3932	2176	6107	
15/03/16	3768	882	4650	
16/03/16	2248	1739	3986	
17/03/16	3830	2249	6079	
18/03/16	7060	1877	8937	Film crew
19/03/16	2235	1096	3331	
20/03/16	4397	627	5024	User study (morning)
21/03/16	4813	1084	5897	
22/03/16	3562	1379	4941	Review
23/03/16	3912	1980	5891	Film crew and user study
24/03/16	0	1051	1051	Data Recording and Packing
Total	46360	26759	73118	

Table 8.4: The table details the distances (in meters) traveled every day (**Dist. per day**) during the final demo in Schiphol. The total distance is split into two: the distance traveled with the robot driving autonomously (**Dist. Auton.**) and not autonomously (**Dist. Not Auton.**). Only on particular days like the one of the data recording and the mapping sessions, the robot did not drive autonomously, in the remaining days the robot was mainly operated autonomously.

gate of destination) and reduced the number of missions aborted for planning problems to zero, compared to an initial motion planning strategy adopted in the Spencer project. The initial strategy was merely reactive: global planner and local planner were communicating only via a single path (hypothesis), the local planner (based on the dynamic window approach (Fox et al., 1997)) was very

Figure 8.16: Scenes from the final SPENCER integration week and deployment. During the integration week, the robot drove autonomously for circa 46 km. Tests and user studies were carried on along all days, maps were recorded during two nights, when fewer people were passing through the terminals.

reactive and often generating the freezing behavior or unexpected direction oscillations. Clearly the multi-hypothesis strategy allowed our planner to better balance social-awareness with efficiency, as depicted also in Fig. 8.17: the planner

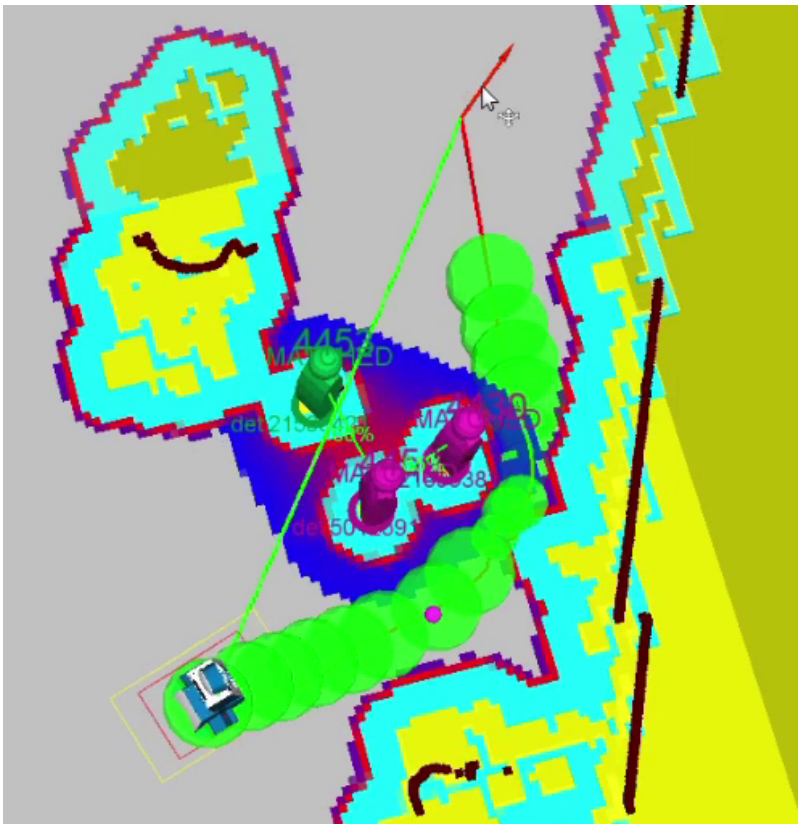

Figure 8.17: Socially aware motion planning. This figures shows two possible ways to pass a crowd or a group of people (here a group of three people where one takes a picture of the other two). The **blue area** on the floor shows the social link between the group members, the **red line** is the socially-aware path computed and selected by our system, and the **green line** is the static (direct) path. **Green spheres** represent the elastic band.

generated the socially-aware path hypothesis $\mathbf{P}_{hsa}$, each time that in a crowded situation this choice would not lead to large detour respect to the static world path hypothesis $\mathbf{P}_{hsw}$.

The legibility aspects helped in improving the robot behaviors: respect to the initial planning strategy, overall no passengers complained about unexpected robot behaviors (i.e. oscillation in direction and acceleration); during the robot operation, they significantly reduced the risk of collisions with rushing crowd of passengers crossing or moving against the robot.

During the final deployment of the Spencer robot, see Fig. 8.16 and Fig. 8.18, before to start the guidance experiments, we have further fine-tuned the motion planner parameters to take the particular dynamics of Schiphol Airport into ac-

Figure 8.18: Part of the Spencer project team at Schiphol during the final deployment.

count. All tests and user studies had to be done under uncontrollable passenger dynamics. The robot was operating in daylight conditions when the majority of flights were arriving, only twice the robot and some of the developers were operating in the late evening to record a map of the environment. We developers had to work on the field, without having the facilities of a research laboratory.

Overall the robot drove autonomously circa 46 kilometers in 15 days. Table 8.4 details the distances traveled every day, which is reported as sum of the distances traveled by the robot autonomously and not autonomously. Only on particular days like the one of the data recording and the mapping sessions, the robot did not drive autonomously.

8.5 Conclusions

During the SPENCER final demo at Schiphol we thoroughly tested the developed socially-aware and efficient motion planner for nonholonomic wheeled mobile robots. The planner by means of a multi-hypothesis reasoning balances efficiency with socially-awareness and generates locally legible motion that safely and smartly reacts to the dynamics of the environment.

Overall the robot met the requirements of safe and reliable operation in a highly dynamic and uncontrolled environment, which is very different from a laboratory setup, extremely well: the robot drove, gently and faultlessly (without collisions with and hindrances to the pedestrians), autonomously circa 46 kilometers in 15 days indicating that this it is a valuable system to generate smooth and efficient motion in very crowded environments.

CHAPTER 9

Conclusions

"Und jedem Anfang wohnt ein Zauber inne, der uns beschützt und der uns hilft, zu leben."

Herman Hesse,
Stufen

9.1 Summary

In this thesis, we developed several new path and motion planning techniques for wheeled mobile robots. With the goal of smooth and natural real-time motion generation and by leveraging different methods from machine learning, artificial intelligence and control theory, we proposed diverse approaches that extend state-of-the-art motion planners in several aspects. In summary, we have addressed the following questions:

1. How can we improve the efficiency of state-of-the-art motion planners for nonholonomic wheeled mobile robots in complex environments?

2. How can we exploit place-dependent learned motion priors of flows of dynamic obstacles (i.e. pedestrians), specifically multi-modal Gaussian mixtures models, into kino-dynamic sampling-based motion planners?

3. How can we quickly produce, evaluate and select several path hypotheses in dynamic crowded environments? And in particular, how can we generate a small representation of the search space that would allow a wheeled mobile robot to quickly reason about several possible ways to the goal?

4. How can we generate smooth, efficient and socially-aware motion for a fully autonomous and particularly fast robot (1.3 $^{m}/_{s}$), operating in not-controlled and densely crowded environments (i.e. airports)?

Chapters 3, 4 and 5 answer the first question in the frame of sampling-based motion planners. In Chapter 3, we show how the performance of a sampling based

motion planner can be improved by using a novel stabilizer for wheeled mobile robots instead of random control propagations (or motion primitives): this allows the planner to exploit as much knowledge of the nonholonomic constraints of the system as possible to ensure both high planning efficiency and high trajectory quality. This combination was shown to produce smoother paths in less time with smaller trees than a set of state-of-the-art baseline methods. Moreover when combined with an optimal sampling-based motion planner, the approach produces the shortest paths and achieves the lowest cost solutions when given more planning time.

In Chapter 4, we present an on-line machine learning approach to compute the distance pseudo-metric for nonholonomic sampling base-motion planners. The approach is shown to be faster in planning time by several factors with respect to different baselines at a negligible loss of path quality. The use of machine learning – we use a fast constant-time regression approach – helps to efficiently compute the distance pseudo-metric by avoiding to solve a computationally expensive 2P-BVP problem.

In Chapter 5, we introduce a hierarchical combination of (discrete) any-angle search with (continuous) sampling-based motion planning for nonholonomic wheeled mobile robots. The combination generates a trajectory by expanding a tree of geodesics toward sampled states whose distribution summarizes geometric information of the any-angle path. We evaluate the algorithm by considering the cases of a differential drive robot and of a high-dimensional truck-and-trailer system. The approach, that is probabilistically complete, finds smooth and shorter trajectories significantly faster than four baseline planners without loss of smoothness.

In Chapter 6 we give an answer to the second question by introducing a motion planning approach (for wheeled mobile robots) under kinodynamic constraints that generates paths by trading off classical path quality metrics with the compliance to off-line learned model of the environment dynamics. Specifically it exploits learned perception priors of dynamic environments in the form of continuous Gaussian mixture fields. The approach, which is asymptotically optimal, efficiently generates high-quality solutions in terms of path smoothness, path length as well as minimum control effort through multi-modal representations of Gaussian mixture fields.

In Chapter 7 we propose a solution for the problem detailed in the third question by presenting an efficient randomized approach based on weighted random walks, that finds K diverse path hypotheses on the Voronoi diagram of the environment, where each path represents a distinct homotopy class. The approach is significantly faster at finding paths of higher diversity in distinct homotopy classes than two state-of-the-art methods.

Finally, in Chapter 8, we detail a socially-aware and efficient motion planner for nonholonomic wheeled mobile robots navigating at high speed in densely

crowded environments. Instead of a mere reactive planner, we propose a motion planning approach, that by means of a multi-hypothesis reasoning, balances efficiency with social-awareness and generates locally legible motion that safely and smartly reacts to the dynamics of the environment. The planning system was developed for the EU-funded project SPENCER and deployed at Schiphol Amsterdam Airport (a densely crowded environment): overall the robot gently and faultlessly drove autonomously without collisions with and hindrances to the pedestrians, with an average velocity of $1.3^{m}/_{s}$, circa 46 kilometers in 15 days, indicating that this system is a valuable answer to the aforementioned question number four.

All the introduced methods were tested not only in simulated environments but also in real-world experiments on two robotic platforms of the Social Robotics Laboratory, University of Freiburg: Spencer and Daryl. Many of the methods are open-source available.

9.2 Discussion

As detailed in the previous chapters, there are many valuable works in the field of path and motion planning from different scientific communities.

Nevertheless, we believe that more work needs to be done in general in the context of sampling-based motion planners. Although being the choice for solving very complex tasks, their typical planning times are still high in big and cluttered environments, which makes them feasible to be used for environments of small size, but not for big ones (i.e. airports) as detailed in Chapter 8. We also believe more research needs to be done to reduce their results' variance (i.e for the first returned solution to a problem) and to improve their efficiency in dynamic environments.

This thesis shows that by better coupling low-level control techniques with high-level motion planning (i.e. search) algorithms the overall performance of the robotic system can be improved (see the before mentioned results for Theta*-RRT, RRT/RRT*-POSQ, multi-hypothesis path planner and socially-aware elastic band). An example that goes in this direction is the combination of high level motion planning with model predictive control techniques, which can be informed about future possible state transitions, while still solving their optimization problem on a shorter horizon. Following this line of research, the overall goal of our community should be the development of a single algorithm, that by combining planning and control units, can solve complex tasks.

Moreover, the deployment of the robot Spencer at Schiphol, showed us the importance of planning considering multiple path hypotheses and robot behaviors

(i.e. social-awareness and efficiency). While the robot Spencer was operating at Schiphol airport, we used only geometric features and people track information to select two hypotheses: for future applications more priors about the environment (i.e. motion prediction and uncertainty) should be considered thus reducing the cases of erroneous behaviors selection.
Lastly, nowadays planners can handle events which are foreseen to happen (i.e. most likely occupied cells, predicted motion for a dynamic object). Planning considering rare events (Hertwig et al., 2004; Weinan and Vanden-Eijnden, 2010) is an interesting line of research to increase the robustness of robot operations: by studying rare events, such as near collision with unpredictable obstacles, the robots can develop better planning behaviors for inexperienced or unforeseen situations.

9.3 Recommendations For Future Work

This thesis introduces many methods for wheeled mobile robot motion planning which may inspire new ideas and developments.
Here listed interesting follow-up future research for different chapters:

- As discussed in Chapter 3, an initial extension to this work could be a systematic analysis of how different cost functions impact the overall robot motion behavior: with different cost functions the final trajectories have different geometric properties and velocity profiles. Moreover the coupling of POSQ with RRT/RRT* could be extended to a dynamical representation of the system: in Appendix B, we detail a dynamic version of POSQ, this could be used into RRT* to generate trajectories which adhere also to acceleration limits. POSQ can be also used in the framework of feedback-based motion planning (Yershov and LaValle, 2011; LaValle, 2006): in this case it may be used to generate a vector field of feasible controls into the entire configuration space to guide the robot to the final goal. Furthermore, the approach could be modified in a way to become "obstacle-aware": its guiding Lyapunov function could be aware of the obstacles and thus generating control policies that avoid obstacles.

- Based on the ideas presented in Chapter 4, an interesting follow-up work is the learning of the distance pseudo-metric for higher dimensional configuration spaces. We believe that for complex systems such as humanoids and mobile manipulators this approach will lead to an even more dramatic improvement in planning time. Clearly, new features and a 2P-BVP solver will be required. Moreover new methods could be implemented by using machine learning techniques to improve the efficiency of sampling-based

motion planners (see also Arslan (2015) and Pan et al. (2013)). A learned distance metric can be used also to approximate the landmark heuristic (Paden et al., 2017) for incremental sampling based motion planners. Although existing works suggest to use deep neural networks for solving motion planning tasks in end-to-end trajectory generation methods (Levine et al., 2016), these methods could also be used into the framework of sampling-based motion planners. One example could be to substitute the sampling unit, the distance metric and the collision checker, with a generative model. The latter, based on the current tree and the environment, generates the samples that may extend the tree by using a steer function towards the goal region without encountering a collision. This would definitely improve the planning efficiency.

- Chapter 5 shows us how, by using information of a discrete global search approach, a sampling based motion planner could improve planning efficiency without compromising path quality. In the same way, we could introduce new methods to combine global and local search methods (an algorithm that goes in this direction is described by Choudhury et al. (2016)): a steer function that is aware of the obstacles could reduce dramatically for example the number of collision checks, by reducing the number of failed attempts to connect samples lying nearby obstacles. Moreover the approach can be easily extended to use different any-angle path planning algorithms: instead of Theta* (Daniel et al., 2010), one could use *Incremental Phi** (Nash et al., 2009) or *ANYA* (Harabor and Grastien, 2013), or *Lazy Theta** (Nash et al., 2010). Theta* and all the previously cited any-angle algorithms can generate a path also in a 3D workspace, thus meaning that the approach could be used for systems that move into the 3D workspace (i.e. manipulators or quadrotor) and not only into the 2D Cartesian plane. The any-angle path biasing technique may also be used in optimal-sampling based motion planners like RRT* (thus introducing the algorithm Theta*-RRT*).

- Regarding Chapter 6, that introduces a kinodynamic approach to compute trajectories based on a Gaussian mixture field description of dynamic objects and humans, a possible extension of this work is to study the behavior of the algorithm in different types of dynamic environments (e.g. UAVs in robot olfaction scenarios, for example, may plan paths that help to estimate gas distributions or localize gas sources). To further improve planning efficiency and reduce the variance of the results, the approach could be extended to include the use of deterministic Janson et al. (2018), as opposed to random sampling sequences. Moreover the approach can be extended to enable the selection of multiple behaviors: by adjusting the bias towards the CLiFF-map based sampling one may choose if the robot has to follow a flow

of dynamic obstacles or to move in opposite direction. The planner could also generate different behaviors based on the time of the planning request: the CLiFF maps may represent different behaviors learned during different hours of the day, thus the planner may select which behavior to represent by choosing the CLiFF distributions related to the time of planning request.

- Both Chapters 5-6 show us the importance of the sampling strategy in sampling-based motion planners. We believe our community should invest more on finding optimal sampling sequences that are obstacle and graph aware (see for example the works by Burns and Brock (2005); Kiesel (2016); Hsu et al. (2007)): generating states that are easier (in terms of reducing collision checks and computational efforts) to connect to the current graph (road-map or tree) would allow the planner to more efficiently search the configuration space.

- Chapter 7 introduces a method to quickly compute a set of diverse and homotopically distinct paths. To reduce the number of re-planning events (which includes generation of the Voronoi diagram and the iterations of the RHCF algorithm), future methods could profit from a local optimizer (e.g. Elastic Band, CHOMP (Zucker et al., 2013)) that elastically deform the initial homotopically distinct paths to respond to the dynamic changes in the environment (in a way similar to elastic roadmaps (Yang and Brock, 2006) and reactive deformation roadmaps (Gayle et al., 2007)). Moreover, future work may also include the analysis of space and time complexity of the algorithm, as well as extend it to generate even more robust path sets by considering sensing and motion uncertainty.

- Regarding Chapter 8, we believe that to build more efficient navigation systems in crowded environments we need a more robust way to compute human motion predictions and an intelligent way to consider them during the planning routines: recently we have been working on this line of work, see (Rudenko et al., 2017) for preliminary results. Human motion predictions could be exploited, for example, into the local trajectory optimization methods (e.g. Elastic Band, CHOMP Zucker et al. (2013), TrajOpt Schulman et al. (2014)), to reduce the need of unexpected re-planning for those methods. We believe also that a more intelligent local planner that has more knowledge about the global behavior of the system (like the case of the global planner) and that is better coupled with the controls generation, could reduce sub-optimal behaviors (i.e. unexpected turns or decelerations), see a recent work by Tamar et al. (2017) that shows an approach on how this goal could be achieved by combining model predictive control techniques with machine learning.

Overall we believe that by better coupling the efforts of communities with different backgrounds (e.g. AI, machine learning, control theory), existing motion planning methods could further be improved and new interesting ideas could be developed to efficiently and smoothly solve the motion planning problem.

Appendices

APPENDIX A

Notions on Nonholonomic systems

In this appendix we illustrate general properties for nonholonomic systems following (Laumond et al., 1998) and (Jean, 2014). We do not go into the details of the Lie algebra.

A.1 Nonholonomic control systems

Let $\mathcal{C} \subset \mathbb{R}^d$ be the state space, $\mathcal{U} \subset \mathbb{R}^m$ the control space (with $m < d$), and $\mathcal{C}_{obs} \subset \mathcal{C}$ and $\mathcal{C}_{free} = \mathcal{C} \setminus \mathcal{C}_{obs}$ the obstacle and free spaces, respectively. A (control) system Σ for nonholonomic systems on the state space $\mathcal{C}$ is a differential system such that

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t))\, u + g(\mathbf{x}(t)) \qquad \mathbf{x}(0) = \mathbf{x}_{start}, \tag{A.1}$$

where $\mathbf{x}_{start} \in \mathcal{C}$ and for all t, $\mathbf{x}(t) \in \mathcal{C}$ and $\mathbf{u}(t) \in \mathcal{U}$. g describes the drift, and f describes the system dynamics.
The nonholonomic system Σ in Equ. A.1 can be rewritten as the sum of contributions of many control inputs.

$$\dot{\mathbf{x}}(t) = \sum_{i=1}^{m} u_i X_i(x) + g(\mathbf{x}(t)) \qquad \mathbf{x}(0) = \mathbf{x}_{start}, \tag{A.2}$$

where the configuration space $\mathcal{C}$ of the system is a C^∞ manifold, $X_1, \ldots, X_m$ are C^∞ vector fields on $\mathcal{C}$ and the control functions $u(t) = (u_1(t), \ldots, u_m(t))$ are defined in $\mathcal{U}$.
The system Σ determines a family of vector spaces,

$$\Delta\mathbf{x} = span\{X_1(x), \ldots, X_m(x)\} \subset T_x\mathcal{C},\, x \in \mathcal{C}$$

whose dimensions depends on $\mathbf{x}$ and it can be constant, where T_x is the tangent space of $\mathcal{C}$ at x .

A.1.1 Controllability and Reachable Sets

Controllability is the ability to steer a system from a point of its space to an other point of the same space. Controllability is often coupled with the Reachability property.

Definition 13 (Reachable set from $\mathbf{x} \in \mathcal{C}$ (Jean, 2014))**.** *The reachable set from* $\mathbf{x} \in \mathcal{C}$ *is defined to be the set* $\mathcal{A}_\mathbf{x}$ *of points reached by a trajectory of* Σ *issued from* $\mathbf{x}$*.*

The system Σ is said to be **controllable** when the reachable set from any point in $\mathcal{C}$ is equal to the whole manifold $\mathcal{C}$. The controllability of the system Σ is characterized by the properties of the Lie algebra generated by the vector fields $X_1, \ldots, X_m$.

Hereinafter we describe some fundamental results to solve the nonholonomic motion planning problem. Mainly they characterize the conditions under which a nonholonomic system can be defined controllable and the conditions under which a trajectory between two states in $\mathcal{C}$ exists.

Definition 14 (Chow's Condition or Lie algebra rank condition (LARC), (Jean, 2014))**.** Σ *or the vector fields* $(X_1, \ldots, X_m)$ *satisfies Chow's condition if*

$$Lie(X_1, \ldots, X_m)(\mathbf{x}) = T_\mathbf{x}\mathcal{C} \tag{A.3}$$

or equivalently, for any $\mathbf{x} \in \mathcal{C}$*, there exists an integer* $r = r(\mathbf{x})$ *such that* $|\Delta^r(\mathbf{x})| = d$*, where* d *is the dimension of* $\mathcal{C}$*.*

$Lie(X_1, \ldots, X_m)(\mathbf{x})$ are the Lie brackets associated to the vector fields $(X_1, \ldots, X_m)$ at $\mathbf{x}$. In this case Δ is said to be *bracket generating*.

Lemma 1 (Chow's condition and reachability, (Jean, 2014))**.** *If* Σ *satisfies Chow's condition, then for every* $\mathbf{x} \in \mathcal{C}$*, the reachable set* $\mathcal{A}_\mathbf{x}$ *is a neighborhood of* $\mathbf{x}$*.*

Theorem 2 (Chow-Rashevsky' theorem (Chow, 1940))**.** *If* $\mathcal{C}$ *is connected and if* Σ *satisfies Chow's condition, then any two points of* $\mathcal{C}$ *can be joined by a trajectory of* Σ*.*

If two states that we want to connect are lying into the same connected set, Chow's theorem assures us the existence of a trajectory (between the two states) that adheres to the nonholonomic constraints of the system Σ.
An other interesting property for nonholonomic systems, that we use in Theta*-RRT (see Chapter 5) is the small-time controllability.

Definition 15 (Small-time controllability (Laumond et al., 1998))**.** *The system* Σ *is locally controllable from* $\mathcal{C}$ *if the set of states reachable from* $\mathcal{C}$ *by an admissible trajectory contains a neighborhood of* $\mathcal{C}$*. It is small-time controllable from* $\mathcal{C}$ *if, for any time* T*, the set of states reachable from* $\mathcal{C}$ *before time* T *contains a neighborhood of* $\mathcal{C}$*.*

Checking if a system is small-time controllable is easy for symmetric systems (with $d = |\mathcal{C}|$).

Theorem 3 (Small-time controllability for symmetric systems (Laumond et al., 1998), Lie algebra rank condition)**.** *A symmetric system without drift* $(g(\mathbf{x}) = 0)$ *is small-time controllable from* $\mathcal{C}$ *if and only if the rank of the vector space spanned by the family of vector fields together with all their brackets is* d *at* $\mathcal{C}$*.*

A.1.2 Reachability

An important theorem in the context of nonholonomic motion planning, also used in RRT* (Karaman and Frazzoli, 2013), is the Ball-Box Theorem that gives an estimate of the reachable set for a nonholonomic system. To define it, we need to detail the sub-Riemannian ball and analyze a nonholonomic system from a set of privileged coordinates.

A.1.2.1 Sub-Riemannian distance and ball

The arc length of a path $\sigma(t)$, generated by the system Σ, is defined as $l(\sigma(t)) = \int_0^T \sqrt{\langle \dot{\sigma}(t), \dot{\sigma}(t) \rangle}$, where $\langle ., . \rangle$ denotes the Euclidean inner product in $\mathcal{C}$. This function induces a sub-Riemannian distance d on $\mathcal{C}$, defined for $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{C}$ as $d(\mathbf{x}_1, \mathbf{x}_2) = \inf_{\mathbf{x}} l(\mathbf{x})$. The sub-Riemannian ball is defined as $B(\mathbf{x}, \epsilon) = \{\forall \mathbf{y} \in \mathcal{C} : d(\mathbf{x}, \mathbf{y}) \leq \epsilon\}$, which is a subset of the Euclidean Ball $B(\mathbf{x}, \epsilon) \subset B^{Eucl}(\mathbf{x}, \epsilon)$. Clearly the sub-Riemannian ball of a nonholonomic system Σ describes the reachable sets of Σ. Computing analytically the sub-Riemannian distance and ball is difficult and expensive.

A.1.2.2 Regular and singular points

Chow's condition assumes that $\forall \mathbf{x} \in \mathcal{C}$ there exists a smallest integer $r = r(\mathbf{x})$ such that $\Delta^{r(\mathcal{C})} = T_{\mathbf{x}}\mathcal{C}$. The value of the set Δ^s generated by Σ forms a *flag of subspaces* of $T_{\mathbf{x}}\mathcal{C}$: $\Delta^1(\mathbf{x}) \subset \Delta^2(\mathbf{x}) \subset \ldots \subset \Delta^r(\mathbf{x}) = T_{\mathbf{x}}\mathcal{C}$, where $r = r(\mathbf{x})$ is called the *degree of nonholonomy* at $\mathbf{x}$. The r-tuple of integers $(n_1(\mathbf{x}), \ldots, n_r(\mathbf{x}))$, with $n_i(\mathbf{x}) = |\Delta^i(\mathbf{x})|$, is called the growth vector of $\mathbf{x}$.

A state (or point) $\mathbf{x}$ is a *regular point* if there exists an open neighborhood of $\mathcal{C}$ around $\mathbf{x}$ such that the growth vector is constant, otherwise is a *singular point*.

A.1.2.3 Privileged coordinates

The structure of the flag can be described by using a sequence of *weights* at $\mathbf{x}$, $w_i = w_i(\mathbf{x}), i = 1, \ldots, n$ by setting $w_j = s$ if $n_{s-1}(\mathbf{x}) < j \leq n_s(\mathbf{x})$ where $n_0 = 0$. If $\mathbf{x}$ is a regular point, the weights at $\mathbf{x}$ form an increasing sequence $w_1(\mathbf{x}) \leq \ldots \leq w_n(\mathbf{x})$.

Let us call $X_i X_j f, X_i X_j X_k f, \ldots$ the nonholonomic derivatives of order 2, 3 and so on of f. If the nonholonomic derivatives of order lower than $s-1$ of f vanish at $\mathbf{x}$, we say that f is of order greater and equal to s at $\mathbf{x}$. The function f is said to be of order s at $\mathbf{x}$ if its order is greater and equal to s but not of order greater and equal to $s+1$.

A set of local coordinates $(z_1, \ldots, z_n)$ centered at $\mathbf{x}$ are privileged coordinates at $\mathbf{x}$ if the order of z_i at $\mathbf{x}$ is equal to w_i, for $i = 1, \ldots, n$.

Given a set of privileged coordinates z_i, we can define the *pseudo-norm* at $\mathbf{x}_0$ as

$$\|z\|_{\mathbf{x}_0} = \max\{|z_1|^{1/w_1}, \ldots, |z_n|^{1/w_n}\}$$

A.1.2.4 Ball-box theorem

A *w-weighted box* of size ϵ at $\mathbf{x}_0$, given a set of privileged coordinates, can be defined as $Box^w(\epsilon) = \{z \in \mathbb{R}^n : \|z\|_{\mathbf{x}_0} \leq \epsilon\}$.

Theorem 4 (Ball-box theorem (Bellaiche et al., 1998))**.** *The following estimate holds if and only if $(z_1, \ldots, z_n)$ form a system of privileged coordinates at $\mathbf{x}$: there exist constants $c_\mathbf{x}$, $C_\mathbf{x}$ and $\epsilon_\mathbf{x} > 0$ such that, for $\mathbf{y} \in \mathcal{C}$ with $d(\mathbf{x}, \mathbf{y}) < \epsilon_\mathbf{x}$,*

$$c_\mathbf{x}(|z_1|^{1/w_1}, \ldots, |z_n|^{1/w_n}) \leq d(0, (z_1, \ldots, z_n)) \leq C_\mathbf{x}(|z_1|^{1/w_1}, \ldots, |z_n|^{1/w_n}) \quad \text{(A.4)}$$

The estimate A.4 of the sub-Riemannian distance induces a geometric interpretation:

Corollary 1. *Expressed in a given system of privileged coordinates, the sub-Riemannian balls $B(\mathbf{x}, \epsilon)$ satisfy for $\epsilon < \epsilon_\mathbf{x}$,*

$$Box^w(c_\mathbf{x}\epsilon) \subset B(\mathbf{x}, \epsilon) \subset Box^w(C_\mathbf{x}\epsilon) \quad \text{(A.5)}$$

where $Box^w(\epsilon) = [-\epsilon^{w_1}, \epsilon^{w_1}] \times \ldots \times [-\epsilon^{w_n}, \epsilon^{w_n}]$.

A.2 Topological property

In here we describe the topological property and its properties as introduced by Sekhavat and Laumond (1998), a key property requested by RRT* for nonholonomic systems. Let us denote: with $B(\mathbf{x}, \rho)$ the ball centred around the configuration $\mathbf{x}$ and radius ρ, with $\mathbf{x}$ belonging to the configuration space $\mathcal{C}$; $l_{a,b}$ a path connecting the configuration a and b generated by the local planner or steer function l.

Definition 16 (Local Planner or Steer Function, (Sekhavat and Laumond, 1998))**.** *Given a nonholonomic system Σ and its configuration space $\mathcal{C}$, a local planner or steer function is a map*

$$l : \mathcal{C} \times \mathcal{C} \to \mathcal{C}^{[0,1]}$$
$$(\mathbf{x}_0, \mathbf{x}_1) \to l_{\mathbf{x}_0, \mathbf{x}_1}$$

such that $l_{\mathbf{x}_0,\, \mathbf{x}_1}(0) = \mathbf{x}_0$, $l_{\mathbf{x}_0,\, \mathbf{x}_1}(1) = \mathbf{x}_1$ and $l_{\mathbf{x}_0,\, \mathbf{x}_1}(t)$ with $t \in [0, 1]$, is a path respecting the kinematic constraints of Σ.

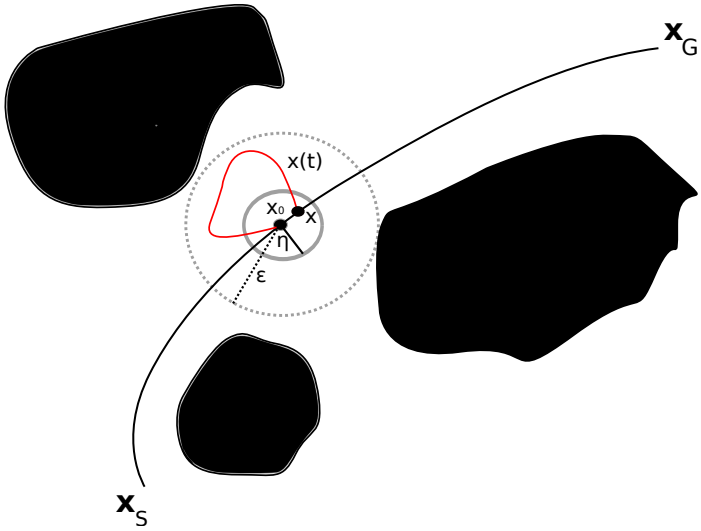

Figure A.1: Example describing the topological property. The **red** path $l_{\mathbf{x}_0,\,\mathbf{x}}(t)$, generated by a steer function to connect $\mathbf{x}_0$ to $\mathbf{x} \in B(\mathbf{x}_0, \eta)$, is contained always in a ball of radius ϵ, $\forall t \in [0,1], l_{\mathbf{x}_0,\,\mathbf{x}}(t) \in B(\mathbf{x}_0, \epsilon)$.

Definition 17 (Topological Property, (Sekhavat and Laumond, 1998))**.** *A local planner l respects the topological property if:*

$$\forall \epsilon > 0, \exists \eta > 0 \mid \forall \mathbf{x}_0 \in \mathcal{C}, \forall \mathbf{x} \in B(\mathbf{x}_0, \eta)$$
$$\forall t \in [0,1], l_{\mathbf{x}_0,\,\mathbf{x}}(t) \in B(\mathbf{x}_0, \epsilon)$$

Namely the topological property, indicates that a local planner l generates paths from a configuration $\mathbf{x}_0$ to $\mathbf{x}$ (whose distance is lower than a certain $\eta > 0$) that do not escape from a ball B of radius ϵ centered on the initial configuration $\mathbf{x}_0$, see Fig. A.1.

A.2.1 Sufficient condition to respect the Topological Property

Let us define a metric d_{max} between any two paths $\boldsymbol{\sigma}_1(t)$ and $\boldsymbol{\sigma}_2(t)$, with parameter $t \in [0,1]$:

$$d_{max} = \max_{t=[0\ 1]} d\left(\boldsymbol{\sigma}_1(t), \boldsymbol{\sigma}_2(t)\right) \tag{A.6}$$

where d is an Euclidean metric on the configuration space $\mathcal{C}$. A sufficient condition guaranteeing that a local planner l respects the topological property:

Definition 18 (Sufficient condition to respect the topological property, (Sekhavat and Laumond, 1998))**.** *Given a local planner l. If l verifies the two following conditions:*

1. *l is continuous in the topology associated with d_{max};*
2. *$\forall a \in \mathcal{C}, \forall t \in [0,1], l_{a,a}(t) = a$;*

then l respects the topological property on any compact set.

APPENDIX B

Dynamic POSQ

In the following we detail an extension of the POSQ steer function, presented in Chapter 3, to generate commands to physical systems in which the torques are the control inputs.

B.1 The Control Law

As in Sec.3.3.1, a control law is developed so that the closed loop system is asymptotically stable: also in this case the discontinuity in $\dot{\alpha}$ is asymptotically canceled, following the approach in Astolfi (1999).
Let ρ be the Euclidean distance between the initial pose and the goal pose ($\mathbf{x}_{near}$ and $\mathbf{x}_{rand}$ in an RRT notation), ϕ the angle between the x-axis of the robot reference frame $\{X_R\}$ and the x-axis of the goal pose frame $\{X_G\}$, α the angle between the y-axis of the robot reference frame and the vector Z connecting the robot with the goal position, v the translational and ω the angular robot velocity, see Fig. 3.3.

We consider the dynamic extended model of a differential drive robot in polar coordinates, with the assumption that the motor dynamics have been compensated,

$$\begin{aligned} \dot{v} &= T_1 \\ \dot{\omega} &= T_2 \\ \dot{\rho} &= -\cos\alpha\, v \\ \dot{\alpha} &= \frac{\sin\alpha}{\rho} v - \omega \\ \dot{\phi} &= -\omega \end{aligned} \tag{B.1}$$

Lemma 2 (Boundness of the internal variable ϵ)**.** *Consider the system*

$$\begin{aligned} \dot{v} &= T_1 \\ \dot{\rho} &= -\cos\alpha\, v \end{aligned} \tag{B.2}$$

where $\alpha(t) \in D_1 =]-\frac{\pi}{2}, \frac{\pi}{2}]$, $\rho(0) \neq 0$.
Let us apply the control law

$$T_1 = -K_\rho K_v cos\alpha \frac{v}{\cosh(K_v\,\rho)^2} - \lambda_1 \left(v - K_\rho \tanh(K_v \rho)\right) \tag{B.3}$$

with $\lambda_1 > 0$, $K_\rho > 0$, $K_v > 0$. *The ratio* $\epsilon(t) = v(t)/\rho(t)$ *is defined and bounded for every* $t \geq 0$, *furthermore we have:* $\lim_{t\to+\infty} \epsilon(t) = K_v K_\rho$, $\lim_{t\to+\infty} v(t) = 0$ *and* $\lim_{t\to+\infty} \rho(t) = 0$.

Proof. Locally the first derivative of the variable ϵ is approximated to:

$$\dot{\epsilon} = (\epsilon\, cos\alpha - \lambda_1)\left(\epsilon - K_\rho\, K_v\right) \tag{B.4}$$

It follows that the equilibrium point $\epsilon = K_\rho\, K_v$ is locally exponentially stable, and $\epsilon = 0$ belongs to its basin of attraction. To show the global stability of the system B.2 we use the Lyapunov method. The candidate Lyapunov function is the following:

$$W(\rho, v) = 2\lambda_1\, K_\rho\, \rho^2 + \lambda_1 \left(v - K_\rho \tanh(K_v \rho)\right)^2 \tag{B.5}$$

with its first derivative equal to

$$\dot{W} = -4\, K_\rho\, \rho\, \lambda_1\, v\, \cos\alpha - 2\lambda_1 \left(v - K_\rho \tanh(K_v \rho)\right)^2 \tag{B.6}$$

For $\lambda_1 > 0, K_\rho > 0$ we have that $\dot{W} \leq 0$, and $S = \left\{(v, \rho) : \dot{W}(v, \rho) = 0\right\}$ contains only the trivial trajectory $(v, \rho) = \mathbf{0}$. For the LaSalle invariance principle then we have that the origin of the system in (B.2) is globally asymptotically stable. □

Proposition 4. *Considering now the following choice of the torque control inputs:*

$$\begin{aligned} T_1 &= -K_\rho K_v cos\alpha \frac{v}{\cosh(K_v\, \rho)^2} - \lambda_1 \left(v - K_\rho \tanh(K_v \rho)\right) \\ T_2 &= K_\alpha \left(\frac{v}{\rho}\, \sin\alpha - w\right) - K_\phi\, w - \lambda_2 \left(\omega - K_\alpha\, \alpha - K_\phi\, \phi\right) \end{aligned} \tag{B.7}$$

Substituting (B.7) in (B.1), we have that the closed loop system is locally exponentially stable iff

$$\begin{aligned} \lambda_1 &> 0 \\ \lambda_2 &> 0 \\ K_v &> 0 \\ K_\rho &> 0 \\ K_\phi &< 0 \\ K_\alpha + K_\phi - K_\rho K_v &> 0\,. \end{aligned} \tag{B.8}$$

Proof. Showing this proposition is trivial. The closed loop system can be locally approximated by a linear system. If and only if the conditions in (B.8) are valid then the eigenvalues of the matrix describing the linear approximation of the model has all negative real parts so the system is locally exponentially stable. □

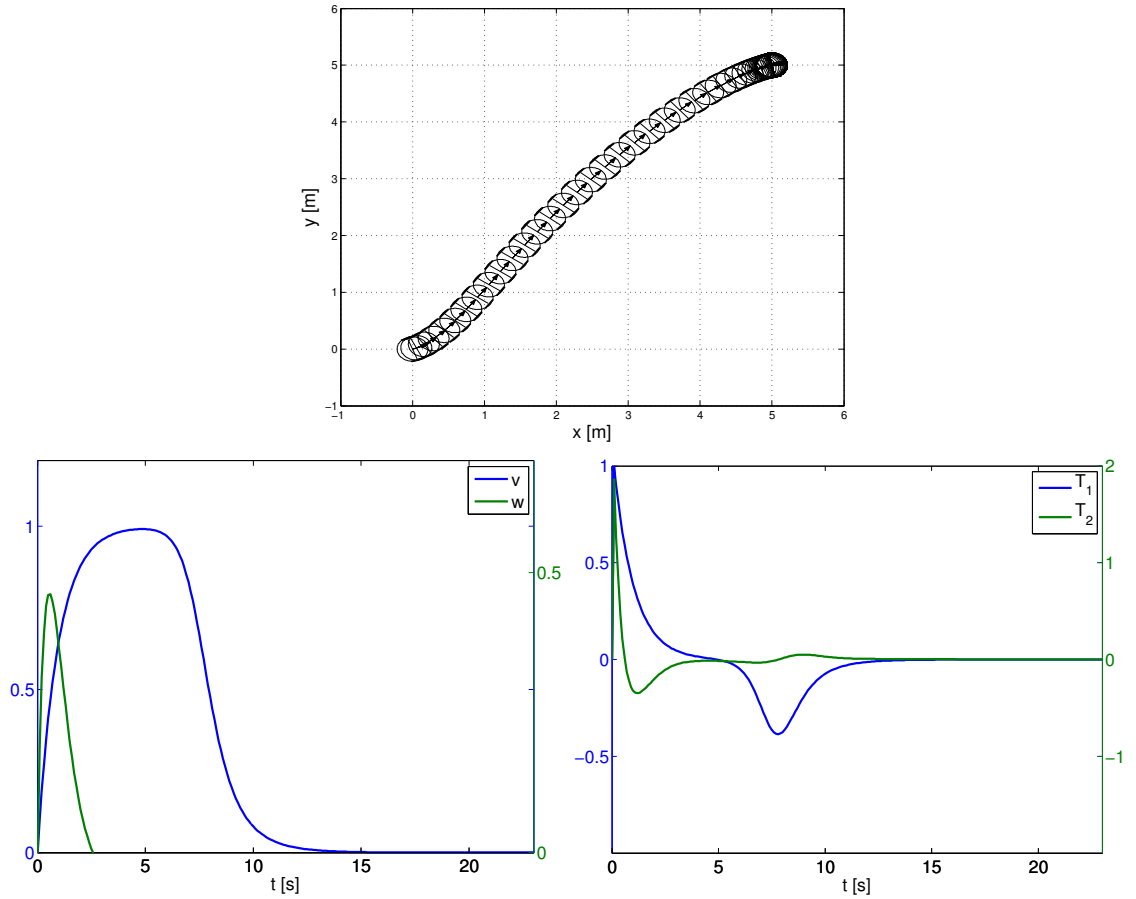

Figure B.1: Example path (on the **top**), velocities (**bottom left**) and torques (**bottom right**) profiles generated with control law of Eq.B.7, when connecting $\mathbf{x}_1 = [0, 0, 0.36]$ to $\mathbf{x}_2 = [5, 5, 0.2]$. The gains are set as follows: $Kv = 1$, $K\rho = 1$, $K\alpha = 3$, $K\phi = -1$, $\lambda_1 = 1$, $\lambda_2 = 1$.

Proposition 5. *Consider the system in (B.1) with the control laws in (B.7) and assuming* $\alpha(0) \in D_1 =]-\frac{\pi}{2}, \frac{\pi}{2}]$, $\rho(0) \neq 0$.

If the relations in (B.8) and Lemma 2 are valid then for every initial condition in the region $\Omega_{dyn} = \left\{(v, w, \rho, \alpha, \phi) \in R^5 \mid v \in R, w \in R, \rho \geq 0, \alpha \in D_1,\ \phi \in (-\pi, \pi]\right\}$, *the system converges asymptotically to the origin.*

Proof. From the following candidate Lyapunov function

$$V_{dyn}(v, \omega, \rho, \alpha, \phi) \;=\; \lambda_2 \;\left(v - K_\rho \;\tanh(K_v \rho)\right)^2 + \lambda_2 \;\left(\omega - K_\alpha \;\alpha - K_\phi \;\phi\right)^2 \quad \text{(B.9)}$$

with simple calculations we can show

$$\dot{V}_{dyn} \;\;=\;\; -\lambda_2 \;\; \lambda_1 \;\; \left(v - K_\rho \;\tanh(K_v \rho)\right)^2 \;-\; \lambda_2^2 \;\; \left(\omega - K_\alpha \;\alpha - K_\phi \;\phi\right)^2 \quad \text{(B.10)}$$

V_{dyn} is positive and $\dot{V}_{dyn}$ is non-positive in Ω_{dyn}. Let us define $S = \{(v, w, \rho, \alpha, \phi) \in \Omega_1 \mid \dot{V}_{dyn} = 0\}$. For the LaSalle invariance principle, the trajectory of the system converges to the positively invariant set $M = \{(v, w, \rho, \alpha, \phi) \in \Omega_{dyn} \mid \dot{V}_{dyn} = 0, t \to \infty\} \in S$, which contains the point $(0, 0, 0, 0, 0)$, the unique w-limit point of any trajectory starting in the region of attraction Ω_{dyn}. □

APPENDIX C

Robot Platforms

In this appendix we describe the robot platforms used to run several experiments during the thesis.

C.1 Spencer

Spencer is the robot designed during the EU project SPENCER, see Section 8.2, and built by the Swiss robotics company BlueBotics SA. The robot has an abstract human-like appearance, see Fig. C.1. Its appearance was designed to increase the overall acceptance of the entire robot by the humans: Its friendly look conveys trustfulness and calmness. The head is used for a simplified non-verbal commu-

Figure C.1: Pictures of the Spencer robot at Schiphol.

nication (e.g. nodding or orientating towards the intended motion direction). The platform has a touchscreen and a boarding pass reader, to resemble an operating information kiosk. During the demo, those allowed the passengers to interact with the robot: by scanning their boarding passes they could start the guidance task of the robot.

The robot has 5 degrees of freedom: 2 for the mobile base with a differential-drive kinematics, 2 for the head (pan and tilt) and 1 for the eyes (only horizontal movements). The height of the robot is of 1926 mm, its base platform is 800×810 mm and its weight (batteries included) is 250 kg. The robot can achieve a maximum velocity of 1.8 m/s.

A schematic view of the hardware architecture is given in Fig. 8.6. The robot is equipped with 3 industrial PCs (2 Intel Core i7 and one i5) and 2 gaming laptops with nVidia graphics cards. The latter mainly used by the perception components which require powerful GPUs, while the remaining robot tasks (e.g. planning, localization etc.) are performed on the industrial PCs. The robot drive motors are interfaced to the BlueBotics ANT system, used only as gateway between the motors and the rest of the ROS-based robot software architecture. The sensory setup consists of:

- two SICK LMS 500 2D laser scanners mounted at a height of 0.70m,
- two front and two rear RGB-D Asus Xtion Pro live cameras,
- two AVT cameras with 4.5mm lens as stereo camera system mounted at shoulder height,
- Velodyne VLP-16 3D lidar mounted on one of the robot's shoulders (not planned at beginning of the project but it turned out to be necessary to cope with localization difficulties).

C.2 Daryl

The robot Daryl is a custom-designed 10 degrees of freedom (dof) wheeled mobile robot platform. It was designed by the Social Robotics Laboratory of the University of Freiburg and realized in collaboration with two Swiss companies: Robonaut and Formfabrik.

Compared to Spencer, Daryl has more degrees of freedom: two dof for the differential drive base, two dof for a laser pointing device mounted on one of its shoulders, two dof for the ears and four dof for the neck mechanism. The robot sensory setup consists of: two SICK laser scanners, two cameras, bumpers and wheel encoders, an industry standard embedded system with an RTOS and a C++ API. It was often used to study how robots may better interact with humans, see Fig. C.2-C.3. In fact, differently from Spencer, that had simply the head to express intentions, Daryl instead has:

- ears, which can rotate and may be used to show different expression-emotions (happiness, sadness etc.);

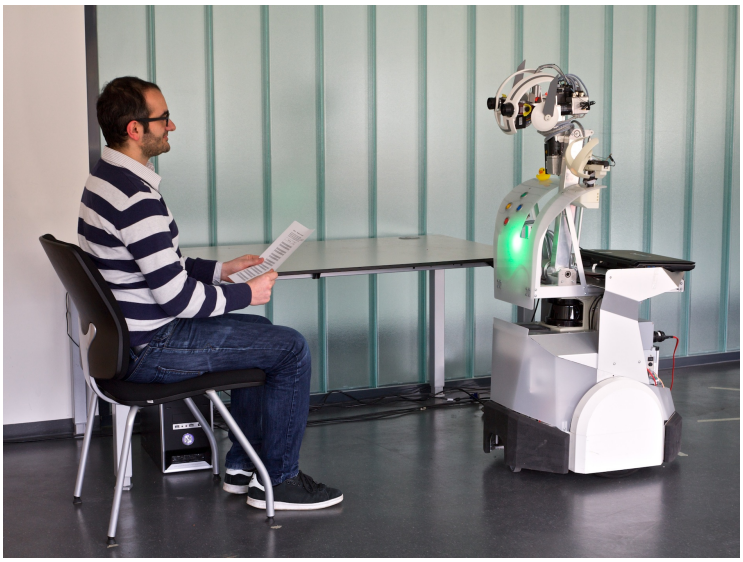

Figure C.2: Daryl is communicating with a human during an experiment.

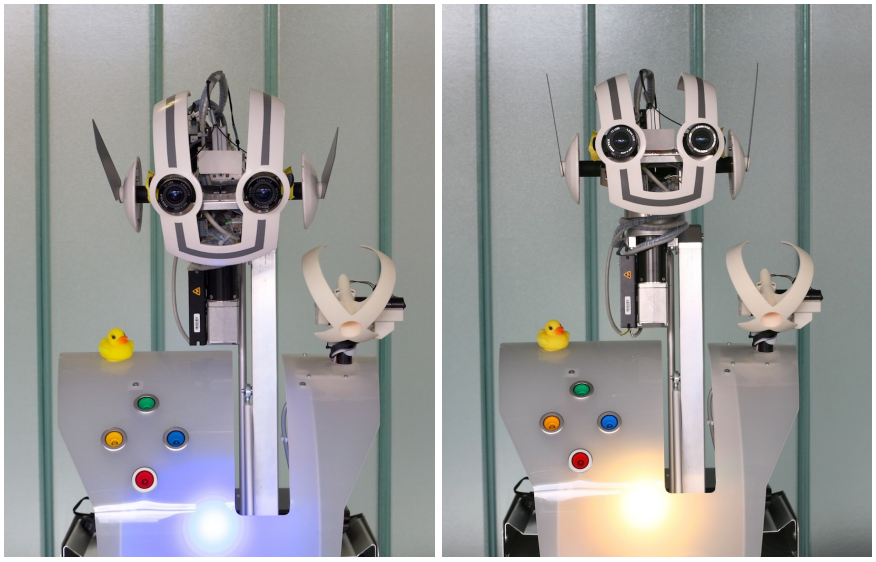

Figure C.3: Examples of Daryl possible expressions. **Left:** Daryl has a sad expression. **Right:** Daryl is thinking about a possible answer to a question.

- a body that can move towards or away from the interacting humans, to show interest (or disappointment);
- colors, showed at the center of the robot body;
- R2-D2 robot speech like.

By combining the robot movements, with these additional expressive modality, Daryl can generate different behaviors and interaction with humans.

List of Figures

List of Tables

List of Algorithms

Bibliography

R. E. Allen, A. A. Clark, J. A. Starek, and M. Pavone. A machine learning approach for real-time reachability analysis. In *Int. Conf. on Intelligent Robots and Systems (IROS)*, Chicago, USA, 2014.

N. Amato, O. Bayazit, L. Dale, C. Jones, and D. Vallejo. Choosing good distance metrics and local planners for probabilistic roadmap methods. *IEEE Trans. on Robotics and Automation (TRO)*, 16(4):442–447, Aug 2000.

O. Arslan. *Machine learning and dynamic programming algorithms for motion planning and control*. PhD thesis, Georgia Institute of Technology, 2015.

O. Arslan and P. Tsiotras. Use of relaxation methods in sampling-based algorithms for optimal motion planning. In *Int. Conf. on Robotics and Automation (ICRA)*, Karlsruhe, Germany, 2013.

A. Astolfi. Exponential stabilization of a wheeled mobile robot via discontinuous control. *Journal of Dynamic Systems, Measurement, and Control*, 121(1), 1999.

R. Bajcsy. *Position Statement: Robotics Science*, pages 583–585. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

S. Balasubramanian, A. Melendez-Calderon, and E. Burdet. A robust and sensitive metric for quantifying movement smoothness. *IEEE Transactions on Biomedical Engineering*, 59(8), 2012.

K. Bekris and L. Kavraki. Informed and probabilistically complete search for motion planning under differential constraints. In *First International Symposium on Search Techniques in Artificial Intelligence and Robotics (STAIR), Chicago, IL*, 2008.

A. Bellaiche, F. Jean, and J.-J. Risler. Geometry of nonholonomic systems. In *Robot Motion Planning and Control*, pages 55–91. Springer, 1998.

M. Bennewitz, W. Burgard, and S. Thrun. Adapting navigation strategies using motions patterns of people. In *Int. Conf. on Robotics and Automation (ICRA)*, Taipei, China, 2003.

M. Bharatheesha, W. Caarls, W. Wolfslag, and M. Wisse. Distance metric approximation for state-space RRTs using supervised learning. In *Int. Conf. on Intelligent Robots and Systems (IROS)*, Chicago, USA, 2014.

S. Bhattacharya, V. Kumar, and M. Likhachev. Search-based path planning with homotopy class constraints. In *Third Annual Symposium on Combinatorial Search, AAAI*, Atlanta, Georgia, USA, 2010.

S. Bhattacharya, M. Likhachev, and V. Kumar. Topological constraints in search-based robot path planning. *Autonomous Robots*, 33(3):273–290, 2012.

A. Botea, M. Mueller, and J. Schaeffer. Near optimal hierarchical path-finding. *Journal of Game Development*, 1:7–28, 2004.

M. Brady. *Robotics science*, volume 1. MIT press, 1989.

A. W. Brander and M. C. Sinclair. A comparative study of k-shortest path algorithms. *Proc. of 11th UK Performance Engineering Workshop*, 1996.

L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

O. Brock and O. Khatib. Elastic strips: A framework for motion generation in human environments. *Int. Journal of Robotics Research (IJRR)*, 21(12):1031–1052, 2002.

R. W. Brockett et al. Asymptotic stability and feedback stabilization. *Differential geometric control theory*, 27(1), 1983.

R. Brooks. A robust layered control system for a mobile robot. *IEEE journal on robotics and automation*, 2(1):14–23, 1986.

M. Brunner, B. Bruggemann, and D. Schulz. Hierarchical rough terrain motion planning using an optimal sampling-based method. In *Int. Conf. on Robotics and Automation (ICRA)*, Karlsruhe, Germany, 2013.

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

B. Burns and O. Brock. Toward optimal configuration space sampling. In *Proc. of the Robotics: Science and Systems (RSS)*, MIT, Cambridge, MA, USA, 2005.

S. Calderara, A. Prati, and R. Cucchiara. Mixtures of von Mises distributions for people trajectory shape analysis. *IEEE Transactions on Circuits and Systems for Video Technology*, 21(4), 2011.

S. Caron, Q.-C. Pham, and Y. Nakamura. Completeness of randomized kinodynamic planners with state-based steering. In *Int. Conf. on Robotics and Automation (ICRA)*, Hong Kong, China, 2014.

P. Cheng and S. LaValle. Reducing metric sensitivity in randomized trajectory design. In *Int. Conf. on Intelligent Robots and Systems (IROS)*, San Francisco, USA, 2001.

H. M. Choset. *Principles of robot motion: theory, algorithms, and implementation*. MIT press, 2005.

S. Choudhury, J. D. Gammell, T. D. Barfoot, S. S. Srinivasa, and S. Scherer. Regionally accelerated batch informed trees (RABIT*): A framework to integrate local information into optimal path planning. In *Int. Conf. on Robotics and Automation (ICRA)*, Stockholm, Sweden, 2016.

W. Chow. Ueber systeme von linearen partiellen differentialgleichungen erster ordnung. *Mathematische Annalen*, 117:98–105, 1940.

B. Cohen, M. Phillips, and M. Likhachev. Planning single-arm manipulations with n-arm robots. In *Proc. of the Robotics: Science and Systems (RSS)*, UC Berkeley, Berkeley, CA, USA, 2014.

B. J. Cohen, S. Chitta, and M. Likhachev. Search-based planning for manipulation with motion primitives. In *Int. Conf. on Robotics and Automation (ICRA)*, Anchorage, AK USA, 2010.

R. C. Coulter. Implementation of the pure pursuit path tracking algorithm. Technical report, DTIC Document, 1992.

R. V. Cowlagi and P. Tsiotras. Hierarchical motion planning with dynamical feasibility guarantees for mobile robotic vehicles. *IEEE Trans. on Robotics and Automation (TRO)*, 28(2), 2012.

K. Daniel, A. Nash, S. Koenig, and A. Felner. Theta*: Any-angle path planning on grids. *Artificial Intelligence Research, Journal of*, 39(1), 2010.

M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational Geometry, Algorithms and Applications*. Springer-Verlag Berlin Heidelberg, 2008.

A. De Luca, G. Oriolo, and M. Vendittelli. Control of wheeled mobile robots: An experimental overview. In *Ramsete*, pages 181–226. Springer, 2001.

D. Demyen and M. Buro. Efficient triangulation-based pathfinding. In *Proc. of the AAAI Conf. on Artificial Intelligence (AAAI)*, Boston, MA, USA, 2006.

M. Diehl, H. J. Ferreau, and N. Haverbeke. Efficient numerical methods for nonlinear MPC and moving horizon estimation. In *Nonlinear model predictive control*, pages 391–417. Springer, 2009.

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

D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel. Path planning for autonomous vehicles in unknown semi-structured environments. *Int. Journal of Robotics Research (IJRR)*, 29(5):485–501, 2010.

B. Donald, P. Xavier, J. Canny, and J. Reif. Kinodynamic motion planning. *Journal of the ACM (JACM)*, 40(5):1048–1066, 1993.

A. D. Dragan, K. C. Lee, and S. S. Srinivasa. Legibility and predictability of robot motion. In *Human-Robot Interaction (HRI), 2013 8th ACM/IEEE International Conference on*, Tokyo, Japan, 2013.

F. Dyson. A meeting with Enrico Fermi. *Nature*, 427(6972):297–297, 2004.

D. Ferguson and A. Stentz. Field D*: An interpolation-based path planner and replanner. In *Robotics research*, pages 239–253. Springer, 2007.

M. Fiore, A. Clodic, and R. Alami. On planning and task achievement modalities for human-robot collaboration. In *Experimental Robotics*, pages 293–306. Springer, 2016.

D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1):23–33, 1997.

E. Frazzoli, M. A. Dahleh, and E. Feron. Real-time motion planning for agile autonomous vehicles. In *American Control Conference*, Virginia, USA, 2001.

E. Frazzoli, M. A. Dahleh, and E. Feron. Maneuver-based motion planning for nonlinear systems with symmetries. *IEEE Trans. on Robotics and Automation (TRO)*, 21(6), 2005.

C. Fulgenzi, A. Spalanzani, C. Laugier, and C. Tay. Risk based motion planning and navigation in uncertain dynamic environment. *INRIA Research Report*, 2010.

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 *Int. Conf. on Intelligent Robots and Systems (IROS)*, Chicago, USA, 2014.

J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot. Batch informed trees (BIT*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs. In *Int. Conf. on Robotics and Automation (ICRA)*, Seattle, WA, USA, 2015.

R. Gayle, A. Sud, M. C. Lin, and D. Manocha. Reactive deformation roadmaps: motion planning of multiple robots in dynamic environments. In *Int. Conf. on Intelligent Robots and Systems (IROS)*, San Diego, CA, USA, 2007.

F. Ghilardelli, G. Lini, and A. Piazzi. Path generation using η^4-splines for a truck and trailer vehicle. *Automation Science and Engineering, IEEE Transactions on*, 11 (1), 2014.

E. G. Gilbert, D. W. Johnson, and S. S. Keerthi. A fast procedure for computing the distance between complex objects in three-dimensional space. *IEEE Journal on Robotics and Automation*, 4(2):193–203, 1988.

E. Glassman and R. Tedrake. A quadratic regulator-based heuristic for rapidly exploring state space. In *Int. Conf. on Robotics and Automation (ICRA)*, Anchorage, USA, 2010.

H. Gong, J. Sim, M. Likhachev, and J. Shi. Multi-hypothesis motion planning for visual object tracking. In *2011 International Conference on Computer Vision*, Barcelona, Spain, 2011.

E. T. Hall. *The hidden dimension*. Doubleday Garden City, N.Y, [1st ed.] edition, 1966.

D. D. Harabor and A. Grastien. An optimal any-angle pathfinding algorithm. In *Twenty-Third International Conference on Automated Planning and Scheduling*, Rome, Italy, 2013.

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

D. Helbing and P. Molnar. Social force model for pedestrian dynamics. *Physical review E*, 51(5), 1995.

R. Hertwig, G. Barron, E. U. Weber, and I. Erev. Decisions from experience and the effect of rare events in risky choice. *Psychological science*, 15(8):534–539, 2004.

D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock. Randomized kinodynamic motion planning with moving obstacles. *Int. Journal of Robotics Research (IJRR)*, 21, 2002.

D. Hsu, J.-C. Latombe, and H. Kurniawati. On the probabilistic foundations of probabilistic roadmap planning. In *Robotics Research*, pages 83–97. Springer, 2007.

J. Hwan J, S. Karaman, and E. Frazzoli. Anytime computation of time-optimal off-road vehicle maneuvers using the RRT*. In *Decision and Control and European Control Conference (CDC-ECC), 2011 50th IEEE Conference on*, Orlando, FL, USA, 2011.

L. Janson, E. Schmerling, A. Clark, and M. Pavone. Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions. *Int. Journal of Robotics Research (IJRR)*, 34(7):883–921, 2015.

L. Janson, B. Ichter, and M. Pavone. Deterministic sampling-based motion planning: Optimality, complexity, and performance. *Int. Journal of Robotics Research (IJRR)*, 37(1):46–61, 2018.

F. Jean. *Control of nonholonomic systems: from sub-Riemannian geometry to motion planning*. Springer, 2014.

M. Kalisiak and M. van de Panne. RRT-blossom: RRT with a local flood-fill behavior. In *Int. Conf. on Robotics and Automation (ICRA)*, Orlando, FL, USA, 2006.

M. Kalisiak and M. van de Panne. Faster motion planning using learned local viability models. In *Int. Conf. on Robotics and Automation (ICRA)*, Rome, Italy, 2007.

S. Karaman. Sampling-based Motion Planning (SMP) Template Library. `http://www.mit.edu/~sertac/smp_doc`, 2011.

S. Karaman and E. Frazzoli. Optimal kinodynamic motion planning using incremental sampling-based methods. In *49th IEEE Conference on Decision and Control (CDC)*, Atlanta, GA, USA, 2010a.

S. Karaman and E. Frazzoli. Incremental sampling-based algorithms for optimal motion planning. In *Proc. of the Robotics: Science and Systems (RSS)*, Universidad de Zaragoza, Zaragoza, Spain, 2010b.

S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *Int. Journal of Robotics Research (IJRR)*, 30(7):846–894, 2011.

S. Karaman and E. Frazzoli. Sampling-based optimal motion planning for nonholonomic dynamical systems. In *Int. Conf. on Robotics and Automation (ICRA)*, Karlsruhe, Germany, 2013.

N. Katoh, T. Ibaraki, and H. Mine. An efficient algorithm for k shortest simple paths. *Networks*, 12(4), 1982.

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

S. Kiesel. *Robotics needs non-classical planning*. PhD thesis, University of New Hampshire, 2016.

I. Ko, B. Kim, and F. C. Park. Randomized path planning on vector fields. *Int. Journal of Robotics Research (IJRR)*, 33(13), 2014.

S. Koenig and M. Likhachev. D* lite. In *AAAI/IAAI*, pages 476–483, 2002.

S. Koenig, M. Likhachev, and D. Furcy. Lifelong planning A*. *Artificial Intelligence*, 155(1):93–146, 2004.

A. J. Krener. A generalization of Chow's theorem and the bang-bang theorem to nonlinear control problems. *SIAM Journal on Control*, 12(1):43–52, 1974.

T. Kruse, A. K. Pandey, R. Alami, and A. Kirsch. Human-aware robot navigation: A survey. *Robotics and Autonomous Systems*, 61(12):1726–1743, 2013.

T. Kruse, A. Kirsch, H. Khambhaita, and R. Alami. Evaluating directional cost models in navigation. In *Proceedings of the 2014 ACM/IEEE international conference on Human-robot interaction*. ACM, 2014.

T. P. Kucner, M. Magnusson, and A. J. Lilienthal. Where am I? an NDT-based prior for MCL. In *Proc. of the European Conference on Mobile Robots (ECMR)*. IEEE, 2015.

T. P. Kucner, M. Magnusson, E. Schaffernicht, V. H. Bennetts, and A. J. Lilienthal. Tell me about dynamics! Mapping velocity fields from sparse samples with Semi-Wrapped Gaussian Mixture Models. In *RSS 2016 Workshop: Geometry and Beyond - Representations, Physics, and Scene Understanding for Robotics*, University of Michigan, Ann Arbor, MI, USA, 2016.

T. P. Kucner, M. Magnusson, E. Schaffernicht, V. H. Bennetts, and A. J. Lilienthal. Enabling flow awareness for mobile robots in partially observable environments. *IEEE Robotics and Automation Letters*, 2(2):1093–1100, 2017.

M. Kuderer, C. Sprunk, H. Kretzschmar, and W. Burgard. Online generation of homotopically distinct navigation paths. In *Int. Conf. on Robotics and Automation (ICRA)*, Hong Kong, China, 2014.

J. J. Kuffner. Effective sampling and distance metrics for 3D rigid body path planning. In *Int. Conf. on Robotics and Automation (ICRA)*, New Orleans, USA, 2004.

D. Kularatne, S. Bhattacharya, and M. A. Hsieh. Time and energy optimal path planning in general flows. *Proc. of the Robotics: Science and Systems (RSS)*, 2016.

T. Kunz and M. Stilman. Kinodynamic RRTs with fixed time step and best-input extension are not probabilistically complete. *Int. Workshop on the Algorithmic Foundations of Robotics (WAFR)*, 2014.

Y. Kuwata, S. Karaman, J. Teo, E. Frazzoli, J. How, and G. Fiore. Real-time motion planning with applications to autonomous urban driving. *IEEE Transactions on Control Systems Technology*, 17(5), 2009.

A. M. Ladd and L. E. Kavraki. Measure theoretic analysis of probabilistic path planning. *IEEE Trans. on Robotics and Automation (TRO)*, 20(2), 2004.

F. Lamiraux, D. Bonnafous, and O. Lefebvre. Reactive path deformation for nonholonomic mobile robots. *IEEE Trans. on Robotics and Automation (TRO)*, 20(6): 967–977, 2004.

J.-C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

B. Lau, C. Sprunk, and W. Burgard. Kinodynamic motion planning for mobile robots using splines. In *Int. Conf. on Intelligent Robots and Systems (IROS)*, St. Louis, MO, USA, 2009. IEEE.

B. Lau, C. Sprunk, and W. Burgard. Efficient grid-based spatial representations for robot navigation in dynamic environments. *Robotics and Autonomous Systems*, 61(10):1116–1130, 2013.

J.-P. Laumond. Feasible trajectories for mobile robots with kinematic and environment constraints. In *Intelligent Autonomous Systems, An International Conference*, Amsterdam, The Netherlands, The Netherlands, 1987. North-Holland Publishing Co.

J.-P. Laumond, S. Sekhavat, and F. Lamiraux. *Guidelines in nonholonomic motion planning for mobile robots*. Springer, 1998.

S. LaValle and J. Kuffner. Randomized kinodynamic planning. In *Int. Conf. on Robotics and Automation (ICRA)*, Detroit, USA, 1999.

S. M. LaValle. *Planning algorithms*. Cambridge university press, 2006.

S. M. LaValle and J. J. Kuffner Jr. Randomized kinodynamic planning. *Int. Journal of Robotics Research (IJRR)*, 20(5):378–400, 2001.

S. M. LaValle, M. S. Branicky, and S. R. Lindemann. On the relationship between classical grid search and probabilistic roadmaps. *Int. Journal of Robotics Research (IJRR)*, 23(7-8):673–692, 2004.

S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research*, 17(39):1–40, 2016.

Y. Li and K. Bekris. Learning approximate cost-to-go metrics to improve sampling-based motion planning. In *Int. Conf. on Robotics and Automation (ICRA)*, Shanghai, China, 2011.

C. Lichtenthäler and A. Kirsch. Towards legible robot navigation-how to increase the intend expressiveness of robot navigation behavior. In *Int. Conf. Soc. Robot. Embodied Commun. Goals Intentions*, 2013.

M. Likhachev and D. Ferguson. Planning long dynamically feasible maneuvers for autonomous vehicles. *Int. Journal of Robotics Research (IJRR)*, 28(8):933–945, 2009.

T. Linder, S. Breuers, B. Leibe, and K. O. Arras. On multi-modal people tracking from mobile platforms in very crowded and dynamic environments. In *Int. Conf. on Robotics and Automation (ICRA)*, Stockholm, Sweden, 2016.

P. Liu, G. Xiong, H. Zhang, Y. Jiang, J. Gong, and H. Chen. A multi-path selecting navigation framework with human supervision. In *Social Robotics: 4th Int. Conf., ICSR 2012, Chengdu, China, October 29-31, 2012. Proceedings*, 2012.

T. Lolla, P. J. Haley, and P. F. Lermusiaux. Time-optimal path planning in dynamic flows using level set equations: realistic applications. *Ocean Dynamics*, 64(10): 1399–1417, 2014.

T. Lozano-Pérez. Spatial planning: A configuration space approach. *IEEE Transactions on Computers*, 100(2):108–120, 1983.

T. Lozano-Pérez and M. A. Wesley. An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM*, 22(10):560–570, 1979.

E. Q. V. Martins and M. M. B. Pascoal. A new implementation of Yen's ranking loopless paths algorithm. *Quarterly Journal of the Belgian, French and Italian Operations Research Societies*, 1(2), 2003.

R. Montgomery. *A tour of subriemannian geometries, their geodesics and applications*. Number 91. American Mathematical Soc., 2006.

A. Nash, K. Daniel, S. Koenig, and A. Felner. Theta*: Any-angle path planning on grids. In *Proc. of the AAAI Conf. on Artificial Intelligence (AAAI)*, Vancouver, Canada, 2007.

A. Nash, S. Koenig, and M. Likhachev. Incremental Phi*: Incremental any-angle path planning on grids. Pasadena, CA, USA, 2009.

A. Nash, S. Koenig, and C. Tovey. Lazy Theta*: Any-angle path planning and path length analysis in 3D. In *Proc. of the AAAI Conf. on Artificial Intelligence (AAAI)*, Atlanta, GA, USA, 2010.

N. Nilsson. A mobile automaton: An application of AI techniques. In *Proceedings of the 1st International Joint Conference on Artificial Intelligence*, pages 509–520, 1969.

J. Nocedal and S. Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

S. T. O'Callaghan, S. P. Singh, A. Alempijevic, and F. T. Ramos. Learning navigational maps by observing human motion patterns. In *Int. Conf. on Robotics and Automation (ICRA)*, Shanghai, China, 2011.

B. Okal and K. O. Arras. Learning socially normative robot navigation behaviors with bayesian inverse reinforcement learning. In *Int. Conf. on Robotics and Automation (ICRA)*, Stockholm, Sweden, 2016.

M. Otte and E. Frazzoli. RRTX: Asymptotically optimal single-query sampling-based motion planning with quick replanning. *Int. Journal of Robotics Research (IJRR)*, 35(7):797–822, 2016.

M. Otte, W. Silva, and E. Frew. Any-time path-planning: Time-varying wind field+ moving obstacles. In *Int. Conf. on Robotics and Automation (ICRA)*, Stockholm, Sweden, 2016.

B. Paden, Y. Nager, and E. Frazzoli. Landmark guided probabilistic roadmap queries. In *Int. Conf. on Intelligent Robots and Systems (IROS)*, Vancouver, Canada, 2017.

L. Palmieri and K. O. Arras. A novel RRT extend function for efficient and smooth mobile robot motion planning. In *Int. Conf. on Intelligent Robots and Systems (IROS)*, Chicago, USA, 2014a.

L. Palmieri and K. O. Arras. Distance metric learning for RRT-based motion planning for wheeled mobile robots. In *Proc. of Machine Learning in Planning and Control of Robot Motion Workshop, IROS*, Chicago, USA, 2014b.

L. Palmieri and K. O. Arras. Distance metric learning for RRT-based motion planning with constant-time inference. In *Int. Conf. on Robotics and Automation (ICRA)*, Seattle, WA, USA, 2015.

L. Palmieri, A. Rudenko, and K. O. Arras. A fast randomized method to find homotopy classes for socially-aware navigation. In *IROS 2015 Workshop on Assistance and Service Robotics in a Human Environment Workshop*, Hamburg, Germany, 2015.

L. Palmieri, S. Koenig, and K. O. Arras. RRT-based nonholonomic motion planning using any-angle path biasing. In *Int. Conf. on Robotics and Automation (ICRA)*, Stockholm, Sweden, 2016.

L. Palmieri, A. Rudenko, and K. O. Arras. A fast random walk approach to find diverse paths for robot navigation. *IEEE Robotics and Automation Letters*, 2(1): 269–276, 2017.

J. Pan, S. Chitta, and D. Manocha. Faster sample-based motion planning using instance-based learning. In *Int. Workshop on the Algorithmic Foundations of Robotics (WAFR)*, Boston, MA, USA, 2013.

J. J. Park. *Graceful Navigation for Mobile Robots in Dynamic and Uncertain Environments*. PhD thesis, The University of Michigan, 2016.

M. Penrose. *Random geometric graphs*. Number 5. Oxford University Press, 2003.

A. Perez, R. Platt, G. Konidaris, L. Kaelbling, and T. Lozano-Pérez. LQR-RRT*: Optimal sampling-based motion planning with automatically derived extension heuristics. In *Int. Conf. on Robotics and Automation (ICRA)*, St. Paul, USA, 2012.

R. Philippsen. *Motion planning and obstacle avoidance for mobile robots in highly cluttered dynamic environments*. PhD thesis, EPFL, 2004.

A. Piazzi, C. Bianco, and M. Romano. η^3-splines for the smooth path generation of wheeled mobile robots. *IEEE Trans. on Robotics and Automation (TRO)*, 23(5), 2007.

M. Pivtoraiko and A. Kelly. Efficient constrained path planning via search in state lattices. In *International Symposium on Artificial Intelligence, Robotics, and Automation in Space*, Munich, Germany, 2005.

M. Pivtoraiko, R. A. Knepper, and A. Kelly. Differentially constrained mobile robot motion planning in state lattices. *Journal of Field Robotics*, 26(3):308–333, 2009.

E. Plaku, L. E. Kavraki, and M. Y. Vardi. Discrete search leading continuous exploration for kinodynamic motion planning. In *Proc. of the Robotics: Science and Systems (RSS)*, Georgia Tech, Georgia, AT, USA, 2007.

E. Plaku, L. Kavraki, and M. Y. Vardi. Motion planning with dynamics by a synergistic combination of layers of planning. *IEEE Trans. on Robotics and Automation (TRO)*, 26(3), 2010.

F. T. Pokorny, M. Hawasly, and S. Ramamoorthy. Multiscale topological trajectory classification with persistent homology. In *Proc. of the Robotics: Science and Systems (RSS)*, UC Berkeley, Berkeley, CA, USA, 2014.

S. Quinlan and O. Khatib. Elastic bands: Connecting path planning and control. In *Int. Conf. on Robotics and Automation (ICRA)*, Atlanta, GA, USA, 1993.

M. Rickert, A. Sieverling, and O. Brock. Balancing exploration and exploitation in sampling-based motion planning. *Robotics, IEEE Transactions on*, 30(6), 2014.

M. Riedmiller and H. Braun. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In *Neural Networks, 1993., IEEE International Conference On*, pages 586–591, 1993.

J. Rios-Martinez, A. Spalanzani, and C. Laugier. Understanding human interaction for probabilistic autonomous navigation using Risk-RRT approach. In *Int. Conf. on Intelligent Robots and Systems (IROS)*, San Francisco, CA, USA, 2011.

A. Roy, S. K. Parui, and U. Roy. A Mixture Model of Circular-Linear Distributions for Color Image Segmentation. *International Journal of Computer Applications*, 58 (9), 2012.

A. Roy, S. K. Parui, and U. Roy. SWGMM: a semi-wrapped Gaussian mixture model for clustering of circular-linear data. *Pattern Analysis and Applications*, 2014.

A. Rudenko, L. Palmieri, and K. O. Arras. Predictive planning for a mobile robot in human environments. In *Workshop on AI Planning and Robotics: Challenges and Methods (at ICRA 2017)*, Singapore, 2017.

J. Saarinen, H. Andreasson, T. Stoyanov, and A. J. Lilienthal. Normal distributions transform monte-carlo localization (NDT-MCL). In *Int. Conf. on Intelligent Robots and Systems (IROS)*, Tokyo, Japan, 2013a. IEEE.

J. Saarinen, T. Stoyanov, H. Andreasson, and A. J. Lilienthal. Fast 3D mapping in highly dynamic environments using normal distributions transform occupancy maps. In *Int. Conf. on Intelligent Robots and Systems (IROS)*, Tokyo, Japan, 2013b. IEEE.

B. Schölkopf, A. J. Smola, R. C. Williamson, and P. L. Bartlett. New support vector algorithms. *Neural Computation*, 12(5):1207–1245, 2000.

J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel. Motion planning with sequential convex optimization and convex collision checking. *Int. Journal of Robotics Research (IJRR)*, 33(9):1251–1270, 2014.

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*, 36(3):345–398, 1983.

S. Sekhavat and J. P. Laumond. Topological property for collision-free nonholonomic motion planning: the case of sinusoidal inputs for chained form systems. *IEEE Trans. on Robotics and Automation (TRO)*, 14(5):671–680, 1998.

S. Sekhavat, F. Lamiraux, J. P. Laumond, G. Bauzil, and A. Ferrand. Motion planning and control for hilare pulling a trailer: experimental issues. In *Int. Conf. on Robotics and Automation (ICRA)*, Albuquerque, NM USA, 1997.

S. Sekhavat, P. Svestka, J.-P. Laumond, and M. H. Overmars. Multilevel path planning for nonholonomic robots using semiholonomic subsystems. *The international journal of robotics research*, 17(8):840–857, 1998.

B. Siciliano and O. Khatib. *Springer Handbook of Robotics*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007. ISBN 354023957X.

R. Siegwart, K. O. Arras, S. Bouabdallah, D. Burnier, G. Froidevaux, X. Greppin, B. Jensen, A. Lorotte, L. Mayor, M. Meisser, et al. Robox at Expo. 02: A large-scale installation of personal robots. *Robotics and Autonomous Systems*, 42(3): 203–222, 2003.

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

C. Spearman. The proof and measurement of association between two things. *Int. Journal of Epidemiology*, 39(5):1137–1150, 2010.

I. A. Şucan and L. E. Kavraki. On the implementation of single-query sampling-based motion planners. In *Int. Conf. on Robotics and Automation (ICRA)*, Anchorage, AK, USA, 2010. IEEE.

I. A. Şucan, M. Moll, and L. E. Kavraki. The Open Motion Planning Library. *IEEE Robotics & Automation Magazine*, 19(4):72–82, December 2012. `http://ompl.kavrakilab.org`.

S. Suzuki et al. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 30(1):32–46, 1985.

A. Tamar, G. Thomas, T. Zhang, S. Levine, and P. Abbeel. Learning from the hindsight plan–episodic mpc improvement. In *Int. Conf. on Robotics and Automation (ICRA)*, Singapore, 2017.

E. Todorov and W. Li. A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems. In *Proceedings of the 2005, American Control Conference*, Portland, OR, USA, 2005.

M. Toussaint. A tutorial on Newton methods for constrained trajectory optimization and relations to SLAM, Gaussian Process smoothing, optimal control, and probabilistic inference. In J.-P. Laumond, editor, *Geometric and Numerical Foundations of Movements*. Springer, 2016.

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. *Int. Journal of Robotics Research (IJRR)*, 34(3):335–356, 2015.

J. van den Berg. Iterated LQR smoothing for locally-optimal feedback control of systems with non-linear dynamics and non-quadratic cost. In *2014 American Control Conference*, Portland, OR, USA, 2014.

V. Varricchio, B. Paden, D. Yershov, and E. Frazzoli. Efficient nearest-neighbor search for dynamical systems with nonholonomic constraints. In *Int. Workshop on the Algorithmic Foundations of Robotics (WAFR)*, San Francisco, CA, USA, 2016.

D. Vasquez, B. Okal, and K. O. Arras. Inverse reinforcement learning algorithms and features for robot navigation in crowds: An experimental comparison. In *Int. Conf. on Intelligent Robots and Systems (IROS)*, Chicago, IL, USA, 2014.

P. Vela, A. Vela, and G. Ogunmakin. Topologically based decision support tools for aircraft routing. In *Digital Avionics Systems Conference (DASC)*, Salt Lake City, USA, 2010.

P. Vernaza, V. Narayanan, and M. Likhachev. Efficiently finding optimal winding-constrained loops in the plane. In *Proc. of the Robotics: Science and Systems (RSS)*, University of Sydney, Sydney, NSW, Australia, July 2012.

S. Vijayakumar and S. Schaal. Locally weighted projection regression: Incremental real time learning in high dimensional space. In *Proceedings of the Seventeenth International Conference on Machine Learning*. Morgan Kaufmann Publishers Inc., 2000.

C. Voss, M. Moll, and L. E. Kavraki. A heuristic approach to finding diverse short paths. In *Int. Conf. on Robotics and Automation (ICRA)*, Seattle, USA, 2015.

Y. Wada, M. Trincavelli, Y. Fukazawa, and H. Ishida. Collecting a Database for Studying Gas Distribution Mapping and Gas Source Localization with Mobile Robots. In *Int. Conf. Adv. Mechatronics*, Osaka, Japan, 2010.

Y. Wang, L. Wang, Y. Li, D. He, and T.-Y. Liu. A theoretical analysis of ndcg type ranking measures. In *Conference on Learning Theory*, Princeton, NJ, USA, 2013.

D. Webb and J. van den Berg. Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics. In *Int. Conf. on Robotics and Automation (ICRA)*, Karlsruhe, Germany, 2013.

E. Weinan and E. Vanden-Eijnden. Transition-path theory and path-finding algorithms for the study of rare events. *Annual review of physical chemistry*, 61, 2010.

P. J. Werbos. Backpropagation and neurocontrol: A review and prospectus. In *International Joint Conference on Neural Networks (IJCNN)*, Washington, DC, USA, 1989.

K. Yang, S. Moon, S. Yoo, J. Kang, N. Doh, H. Kim, and S. Joo. Spline-based RRT path planner for non-holonomic robots. *Journal of Intelligent and Robotic Systems*, 73(1-4), 2014.

Y. Yang and O. Brock. Elastic roadmaps: Globally task-consistent motion for autonomous mobile manipulation in dynamic environments. In *Proc. of the Robotics: Science and Systems (RSS)*, Philadelphia, PA, USA, 2006.

P. Yap. Grid-based path-finding. In *Conference of the Canadian Society for Computational Studies of Intelligence*, Calgary, Canada, 2002.

J. Y. Yen. Finding the k shortest loopless paths in a network. *Management Science*, 17(11), 1971.

D. S. Yershov and S. M. LaValle. Simplicial Dijkstra and A* algorithms for optimal feedback planning. In *Int. Conf. on Intelligent Robots and Systems (IROS)*, San Francisco, CA, USA, 2011.

M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa. CHOMP: Covariant hamiltonian optimization for motion planning. *Int. Journal of Robotics Research (IJRR)*, 32(9-10):1164–1193, 2013.