Sampling-based Algorithms for Optimal Path Planning Problems

by

Sertac Karaman

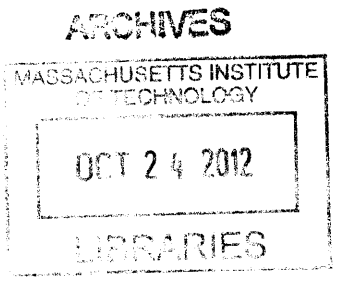
Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2012



Author .....
Department of Electrical Engineering and Computer Science
August 31, 2012

Certified by .....
Emilio Frazzoli
Associate Professor
Thesis Supervisor

Accepted by .....
Leslie A. Kolodziejski
Chairman, Department Committee on Graduate Theses

Sampling-based Algorithms for Optimal Path Planning Problems

by

Sertac Karaman

Submitted to the Department of
Electrical Engineering and Computer Science
on August 31, 2012, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Sampling-based motion planning received increasing attention during the last decade. In particular, some of the leading paradigms, such the Probabilistic RoadMap (PRM) and the Rapidly-exploring Random Tree (RRT) algorithms, have been demonstrated on several robotic platforms, and found applications well outside the robotics domain. However, a large portion of this research effort has been limited to the classical feasible path planning problem, which asks for finding a path that starts from an initial configuration and reaches a goal configuration while avoiding collision with obstacles.

The main contribution of this dissertation is a novel class of algorithms that extend the application domain of sampling-based methods to two new directions: optimal path planning and path planning with complex task specifications.

Regarding the optimal path planning problem, we first show that the existing algorithms either lack asymptotic optimality, *i.e.*, almost-sure convergence to optimal solutions, or they lack computational efficiency: on one hand, neither the RRT nor the k-nearest PRM (for any fixed k) is asymptotically optimal; on the other hand, the simple PRM algorithm, where the connections are sought within fixed radius balls, is not computationally as efficient as the RRT or the efficient PRM variants. Subsequently, we propose two novel algorithms, called PRM* and RRT*, both of which guarantee asymptotic optimality without sacrificing computational efficiency. In fact, the proposed algorithms and the most efficient existing algorithms, such as the RRT, have the same asymptotic computational complexity.

Regarding the path planning problem with complex task specifications, we propose an incremental sampling-based algorithm that is provably correct and probabilistically complete, *i.e.*, it generates a correct-by-design path that satisfies a given deterministic μ-calculus specification, when such a path exists, with probability approaching to one as the number of samples approaches infinity. For this purpose, we develop two key ingredients. First, we propose an incremental sampling-based algorithm, called the RRG, that generates a representative set of paths in the form of a graph, with guaranteed almost-sure convergence towards feasible paths. Second, we propose an incremental local model-checking algorithm for the deterministic μ-calculus. Moreover, with the help of these tools and the ideas behind the RRT*, we construct algorithms that also guarantee almost sure convergence to optimal solutions.

Thesis Supervisor: Emilio Frazzoli
Title: Associate Professor of Aeronautics and Astronautics

Acknowledgments

First and foremost, I would like to express my gratitude to my thesis advisor Emilio Frazzoli. It is impossible not to be truly impressed by his vision, creativity, and enthusiasm even after a five-minute conversation, let alone closely working with him for half a decade. During this time, he has been immensely encouraging and invaluably helpful. He has made many sacrifices so that he could go far beyond his obligations, in order to better teach his students and ensure their success. – Looking back at the day that I was admitted to the PhD program, I now realize I was extremely lucky, that I got the chance to do my thesis work under the supervision of Emilio Frazzoli.

I am indebted also to the other members of my thesis committee. Munther Dahleh, Steven LaValle, and Daniela Rus have provided valuable feedback that has improved this dissertation. I have enjoyed their mentorship and their support for a long time. I am very glad that they agreed to serve on my thesis committee.

There are a number of other faculty and researchers who have been great mentors and valuable collaborators. In particular, I am very thankful to Jonathan How, John Leonard, Seth Teller, and Brian Williams at MIT, Calin Belta at Boston University, Tal Shima at the Israel Institute of Technology, David Hsu at the National University of Singapore, and Siva Banda, Derek Kingston, Steven Rasmussen, and Corey Schumacher at the U.S. Air Force Research Laboratory. They have generously spared their time for inspiring discussions, which contributed to the evolution of the ideas presented in this dissertation and the evolution of my own research interests.

Several friends have made my graduate school experience very joyful and memorable. They are many in number to name here one by one. However, I would like to express my gratitude to two very close friends, Matt Walter and Marek Doniec. They have frequently gone out of their ways to be there for me and our other friends. They are in many memories that makes me remember my time during graduate school as genuinely fun and personally very enriching. Also, special thanks goes to Marco Pavone, *fratello maggiore*, for stimulating discussions and valuable suggestions on almost any possible topic, as he himself once put it, "ranging from distributed algorithms to the Turkish invasion of Sicily *(Mamma li Turchi!)*."

Finally, I would like to thank my parents and my brother. They have provided unconditional support throughout the years that I have spent half a world away from them. Without that support, I could not have completed the PhD program.

The work reported in this dissertation was supported in part by the Michigan/AFRL Collaborative Center on Control Sciences through the AFOSR grant FA 8650-07-2-3744, the National Science Foundation through grant CNS-1016213, the ARO MURI grant W911NF-11-1-0046, the ONR MURI grant N00014-09-1-1051, and an NVIDIA Graduate Fellowship. Their support is gratefully acknowledged.

Contents

List of Figures

Chapter 1

Introduction

1.1 Robot Motion Planning

The robot motion planning problem has received a considerable amount of attention, especially over the last decade, as robots started becoming a vital part of modern industry as well as our daily life (Latombe, 1999; Choset et al., 2005; LaValle, 2006). Even though modern robots may possess significant differences in sensing, actuation, size, workspace, application, etc., the problem of navigating through a complex environment is embedded and essential in almost all robotics applications. Moreover, this problem is relevant to other disciplines such as verification, computational biology, and computer animation (Finn and Kavraki, 1999; Latombe, 1999; Branicky et al., 2001; Liu and Badler, 2003; Bhatia and Frazzoli, 2004; Cortes et al., 2007).

Informally speaking, given a robot with a description of its dynamics, a description of the environment, an initial state, and a set of goal states, the motion planning problem is to find a sequence of control inputs so as the drive the robot from its initial state to one of the goal states while obeying the rules of the environment, e.g., not colliding with the surrounding obstacles. In a widely-studied variant of this problem, the robot is a rigid body that translates and rotates in the d-dimensional Euclidean space populated with fixed obstacles; then, the algorithmic question is to design a procedure to find a sequence of such rigid body transformations that takes the robot from its initial configuration to a goal configuration. Due to the obvious analogy, this problem is called the *mover's problem* (Reif, 1979), or the *piano mover's problem* in some references (Schwartz and Sharir, 1983b,a). A similar class of problems, with a wide range of applications, is also called *spatial planning* by Lozano-Perez (1983). An important generalization, often called the *generalized mover's problem* (Reif, 1979, 1987), features an articulated robot that consists of multiple rigid bodies conjoined with freely-actuated joints. The motivation behind the generalized mover's problem is the robotic manipulator arms commonly found in many modern assembly lines.

An important assumption in the (generalized) mover's problem is the exclusion of complex dynamics, in other words, the absence of differential constraints. In the sequel, such motion planning problems without differential constraints are called *path planning problems*. Path planning problems arise in a number of practical applications

of robotics and beyond (see, *e.g.*, Lozano-Perez, 1983), and a large body of literature is devoted to algorithmic approaches for this problem and the characterization of its complexity, which we review in this section briefly.

Complete Algorithms: An algorithm that addresses a motion planning problem is said to be *complete* if it terminates in finite time, returning a valid solution if one exists, and failure otherwise.

An important step in the design of complete algorithms for motion planning has been the introduction of the notion of *configuration space* by Lozano-Perez and Wesley (1979) (see also Udupa, 1977; Lozano-Perez, 1983), whereby the rigid-body robot is "shrunk" to a point moving in an Euclidean space of dimension equal to the number of degrees of freedom of the robot. This has motivated a number of complete algorithms that represent the obstacle-free portion of the configuration space, called the *free space*, in a computationally convenient form; Then, they evaluate the connectivity of the free space, in particular the connectivity of the initial and the goal configurations, to solve the motion planning problem.

One of the early algorithms in this class was proposed by Schwartz and Sharir (1983a). Their algorithm, often called the *cell decomposition algorithm* (see Sharir, 1997), recursively partitions the configuration space into finite number of cylindrical cells, such that the free space is precisely the union of a subset of these cells. The algorithm, then, constructs a "connectivity graph" in which all adjacent cells are connected by edges. Finally, the motion planning problem is answered by checking whether the two cells that contain the initial and the final configurations are connected to each other through the connectivity graph.

The cell decomposition algorithm is complete. However, being able to guarantee such a strong property comes with a substantial computational cost: the running time of the algorithm is doubly exponential in the number of degrees of freedom of the robot. Later, Canny (1988) proposed the *roadmap algorithm*, which significantly improved the cell decomposition algorithm: the roadmap algorithm runs in time that is only singly exponential in the number of degrees of freedom.

Computational Complexity: It has long been known that, unfortunately, it is very unlikely that the running time of the roadmap algorithm can be improved substantially. As early as 1979, Reif (1979) had shown that the generalized mover's problem is PSPACE-hard in the number of degrees of the robot. However, it is worth noting at this point that both the cell decomposition algorithm and the roadmap algorithm have running times that grow polynomially with both the geometric and the algebraic complexity of the obstacles (see Sharir, 1997). Hence, what makes the (generalized) mover's problem hard is the number of degrees of freedom of the robot, rather than complexity of the environment.

In fact, there are a number of related planning problems, involving several independently actuated parts, with similar computational complexity properties. Arguably, the simplest such problem is the *warehouseman's problem*, which amounts to moving several rectangles on the plane from their initial configurations to their

goal configurations while avoiding collisions with each other. Despite its algebraic, geometric, and conceptual simplicity, the many-degrees-of-freedom nature of the warehouseman's problem renders it PSPACE-hard in the number of rectangles (Hopcroft et al., 1984b). Several other planning problems involving many degrees of freedom, ranging from path planning problems for manipulator arms to many popular puzzles are also known to be PSPACE-hard (Hopcroft et al., 1982; Chazelle et al., 1984; Hopcroft et al., 1984a; Dor and Zwick, 1999; Flake and Baum, 2002; Hearn, 2006).

Practical Approaches: These computational complexity results strongly suggest that any complete algorithm is doomed to suffer from computational intractability in problem instances involving robots with several degrees of freedom. In the light of these results, many practitioners turned to algorithms that relax the completeness requirement one way or another.

Early practical approaches, such as decomposition methods (Brooks and Lozano-Perez, 1983), relaxed the completeness requirement to *resolution completeness*. That is, if properly implemented, these algorithms return a solution, when one exists, if the resolution parameter of the algorithm is set fine enough. Others, such as potential fields (Khatib, 1986; Hwang and Ahuja, 1992), relaxed completeness guarantees altogether.

These planners demonstrated remarkable performance in accomplishing various tasks in complex environments within reasonable time bounds (Ge and Cui, 2002). However, their practical applications were mostly limited to state spaces with up to five dimensions, since decomposition-based methods suffered from large number of cells, and potential field methods from local minima (Koren and Borenstein, 1991). Important contributions towards broader applicability of these methods include navigation functions (Rimon and Koditschek, 1992) and randomization (Barraquand and Latombe, 1991).

Sampling-based Algorithms: The above methods rely on an explicit representation of the obstacles in the configuration space, which is used directly to construct a solution. This may result in an excessive computational burden in high dimensions, and even in environments described by a large number of obstacles. Avoiding such a representation has been the main underlying idea leading to the development of sampling-based algorithms (Kavraki and Latombe, 1994; Kavraki et al., 1996; LaValle and Kuffner, 2001). See (Lindemann and LaValle, 2005) for a historical perspective. These algorithms proved to be very effective for motion planning in high-dimensional spaces, and attracted significant attention over the last decade, including very recent work (see, e.g., Yershova et al., 2005; Stilman et al., 2007; Berenson et al., 2008; Koyuncu et al., 2009; Prentice and Roy, 2009; Tedrake et al., 2010).

Instead of using an explicit representation of the environment, sampling-based algorithms rely on a collision-checking module, providing information about feasibility of candidate trajectories, and connect a set of points sampled from the obstacle-free space in order to build a graph (roadmap) of feasible trajectories. The roadmap is then used to construct the solution to the original motion-planning problem.

Informally speaking, sampling-based methods provide large amounts of computational savings by avoiding explicit construction of obstacles in the configuration space, as opposed to most complete motion planning algorithms. Even though sampling-based algorithms are not complete, they provide *probabilistic completeness* guarantees: roughly speaking, the probability that the planner fails to return a solution, if one exists, decays to zero as the number of samples approaches infinity (Barraquand et al., 1997); see also (Hsu et al., 1997b; Kavraki et al., 1998; Ladd and Kavraki, 2004). Moreover, the rate of decay of the probability of failure is exponential, under the assumption that the environment has good "visibility" properties (Barraquand et al., 1997; Hsu et al., 1997b). More recently, the empirical success of sampling-based algorithms was argued to be strongly tied to the hypothesis that most practical robotic applications, even though involving robots with many degrees of freedom, feature environments with such good visibility properties (Hsu et al., 2006).

Arguably, the most influential sampling-based motion planning algorithms to date include Probabilistic RoadMaps (PRMs) (Kavraki et al., 1996, 1998) and Rapidly-exploring Random Trees (RRTs) (LaValle and Kuffner, 2001). Even though the idea of connecting points sampled randomly from the state space is essential in both approaches, these two algorithms differ in the way that they construct a graph connecting these points.

The PRM algorithm and its variants are multiple-query methods that first construct a graph (the roadmap), which represents a rich set of collision-free trajectories, and then answer queries by computing a shortest path that connects the initial state with a final state through the roadmap. The PRM algorithm has been reported to perform well in high-dimensional state spaces (Kavraki et al., 1996). Furthermore, the PRM algorithm is probabilistically complete, and such that the probability of failure decays to zero exponentially with the number of samples used in the construction of the roadmap (Kavraki et al., 1998). During the last two decades, the PRM algorithm has been a focus of robotics research: several improvements were suggested by many authors and the reasons to why it performs well in many practical cases were better understood (see, e.g., Branicky et al., 2001; Hsu et al., 2006; Ladd and Kavraki, 2004, for some examples).

Even though multiple-query methods are valuable in highly structured environments, such as factory floors, most online planning problems do not require multiple queries, since, for instance, the robot moves from one environment to another, or the environment is not known a priori. Moreover, in some applications, computing a roadmap a priori may be computationally challenging or even infeasible. Tailored mainly for these applications, incremental sampling-based planning algorithms such as RRTs have emerged as an online, single-query counterpart to PRMs (see, e.g., LaValle and Kuffner, 2001; Hsu et al., 2002). The incremental nature of these algorithms avoids the necessity to set the number of samples a priori, and returns a solution as soon as the set of trajectories built by the algorithm is rich enough, enabling on-line implementations. Moreover, tree-based planners do not require connecting two states exactly and more easily handle systems with differential constraints. The RRT algorithm has been shown to be probabilistically complete (LaValle and Kuffner, 2001), with an exponential rate of decay for the probability of failure (LaValle and

Kuffner, 2001; Frazzoli et al., 2002). The basic version of the RRT algorithm has been extended in several directions, and found many applications in the robotics domain and elsewhere (see, *e.g.*, Frazzoli et al., 2002; Bhatia and Frazzoli, 2004; Cortes et al., 2007; Branicky et al., 2006; Zucker et al., 2007). In particular, RRTs have been shown to work effectively for systems with differential constraints and nonlinear dynamics (LaValle and Kuffner, 2001; Frazzoli et al., 2002) as well as purely discrete or hybrid systems (Branicky et al., 2003). Moreover, the RRT algorithm was demonstrated in major robotics events on various experimental robotic platforms (Kuffner et al., 2002; Bruce and Veloso, 2003; Kuwata et al., 2009; Teller et al., 2010).

Other sampling-based planners of note include Expansive Space Trees (EST) (Hsu et al., 1997a) and Sampling-based Roadmap of Trees (SRT) (Plaku et al., 2005). The latter combines the main features of multiple-query algorithms such as PRM with those of single-query algorithms such as RRT and EST.

1.2 Optimal Motion Planning

Although the mover's problem captures a large class of practical problem instances, it imposes no constraint on the quality of the solution returned by the algorithm. In most applications, however, one may be interested in finding paths of minimum cost, with respect to a given cost functional, such as the length of a path, or the time required to execute it. Throughout this dissertation, the class of problems that seek only a feasible solution, such as the mover's problem, will be called feasible motion planning problems. The class of problems that seek a feasible solution with minimal cost, on the other hand, will be called optimal motion planning problems.

Today, there is a rich body of literature devoted to feasible motion planning problems. However, the importance of the optimal motion planning problem has not gone unnoticed. In fact, shortly after complete algorithms were designed for the mover's problem, variants of the optimal motion planning problem were also considered. For instance, Reif and Storer (1985) proposed a complete algorithm with running time that is exponential, in particular in the number of obstacles in the environment.

The computational intractability of the existing complete algorithms comes at no surprise. In fact, the problem of computing optimal motion plans has been proven to be computationally challenging, substantially more so than the feasible motion planning problem. Canny (1988) has shown that computing the shortest path in the three-dimensional Euclidean space populated with obstacles is NP-hard in the number of obstacles. Hence, the optimal motion planning problem is computationally challenging, even when the number of degrees of freedom of the robot is fixed.

More recently, algorithms with resolution completeness properties that address the optimal motion planning problem were also proposed. In particular, graph search algorithms, such as A*, can be applied over a finite discretization (based, e.g., on a grid, or a cell decomposition of the configuration space) that is generated offline. Indeed, recently, these algorithms received a large amount of attention. In particular, they were extended to run in an anytime fashion (Likhachev et al., 2008), deal with dynamic environments (Stentz, 1995; Likhachev et al., 2008), and handle systems

with differential constraints (Likhachev and Ferguson, 2009). These have also been successfully demonstrated on various robotic platforms (Likhachev and Ferguson, 2009; Dolgov et al., 2009). Note, however, that the optimality guarantees of these algorithms are ensured only up to the grid resolution. Moreover, since the number of grid points grows exponentially with the dimensionality of the state space, so does the (worst-case) running time of these algorithms.

In the context of sampling-based motion planning algorithms, the importance of computing optimal solutions has been pointed out in early seminal papers (LaValle and Kuffner, 2001). However, optimality properties of sampling-based motion planning algorithms have not been systematically investigated; Most of the existing relevant work relies on heuristics. For instance, Urmson and Simmons (2003) proposed heuristics to bias the tree growth in RRT towards those regions that result in low-cost solutions. They have also shown experimental results evaluating the performance of different heuristics in terms of the quality of the solution returned. Ferguson and Stentz (2006) considered running the RRT algorithm multiple times in order to progressively improve the quality of the solution. They showed that each run of the algorithm results in a path with smaller cost, even though the procedure is not guaranteed to converge to an optimal solution. A more recent approach is the transition-based RRT (T-RRT) designed to combine rapid exploration properties of the RRT with stochastic optimization methods (Jaillet et al., 2010; Berenson et al., 2011).

1.3 Planning with Complex Task Specifications

In the classical feasible path planning problem, one seeks a path that fulfills a simple task specification: "reach the goal region while avoiding collision with obstacles." An important extension is one that incorporates complex task specifications, which may include, for example, a logical and temporal combination of reachability, safety, ordering, and liveness requirements.

Naturally describing the behavior of complex engineering systems has been a focus of academic research for a long time. Among the most influential publications is the seminal paper by Pnueli (1977), who proposed using temporal logic to describe the behavior of complex computer programs. Pnueli was initially motivated by the *model checking* problem (see Clarke et al., 1999), *i.e.*, checking whether a given complex computer program satisfies a high-level behavior given in temporal logic. However, the problem of *automated synthesis*, *i.e.*, automatic generation of computer programs from their high-level specifications, have become another important research direction very shortly; see for example (Emerson and Clarke, 1982; Manna and Wolper, 1984). Today, model checking and automated synthesis of both computer software and hardware is widely studied in the context of computer science (Holzmann, 1997; Clarke et al., 1999; Peled et al., 2009).

Most early models used in the computer science literature had discrete state spaces. Rigorous introduction of models with continuous state space dates back to early 1990s, for instance, to address problems involving timed systems. However, model checking and especially automated synthesis problems for more complex

continuous-space control systems were considered only very recently.

Motivated by the rich literature in computer science, Tabuada and Pappas (2003) considered model checking of controllable discrete-time linear dynamical systems for Linear Temporal Logic (LTL) specifications. Shortly after, Tabuada and Pappas (2006) also addressed the automated synthesis problem for the same class of dynamical systems. Their work largely relies on generating a discrete abstraction of the original continuous system such that the abstraction exhibits those properties that the original system is endowed with (see Alur et al., 2000). In the recent years, exact and approximate bisimulation relations were introduced as discrete abstractions of a large class of dynamical systems with continuous state spaces (see, *e.g.*, Pappas, 2003; Girard and Pappas, 2007; Fainekos et al., 2009; Girard and Pappas, 2011).

However, it was quickly recognized that constructing bisimulation relations is computationally challenging. Thus, the more recent literature has focused on algorithms that achieve computational efficiency, often by sacrificing the completeness guarantees. For instance, Kloetzer and Belta (2008) considered the problem of control synthesis from LTL specifications for continuous-time affine dynamical systems. More recently, Wongpiromsarn et al. (2009) proposed an automated control synthesis algorithm based on a receding-horizon control strategy for a larger class of dynamical systems. Although computationally effective, these approaches fail to provide any completeness guarantees.

Today, there is a large and growing literature devoted to the application formal methods in the context of control theory and robotics. In addition to those referred to above, see also (Belta et al., 2007; Kress-Gazit et al., 2007, 2009). The literature on automated synthesis of optimal control strategies is relatively narrow, although the problem has been considered more recently (see, *e.g.*, Smith et al., 2010; Karaman and Frazzoli, 2011a). However, to the best of the author's knowledge, with the exception of the work we present in this dissertation and that given in (Bhatia et al., 2010), sampling-based approaches to address feasible or optimal motion planning problems with complex task specifications have not yet been considered.

1.4 Statement of Contributions

The main contribution of this thesis is a novel class of sampling-based algorithms that collectively address the optimal path planning problem as well as the path planning problem with complex task specifications, while providing provable guarantees on correctness, completeness, computational complexity, and convergence towards globally optimal solutions. While introducing these algorithms, we also present a thorough analysis of existing sampling-based algorithms in terms of these properties.

More formally, the contributions of this thesis can be listed as follows. We first analyze the existing algorithms, the PRM and the RRT, in terms of probabilistic completeness, computational complexity, as well as asymptotic optimality, *i.e.*, almost-sure convergence to optimal solutions. We show that all existing algorithms either fail to converge to optimal solutions or lack computational efficiency. We then propose novel algorithms, called the PRM* and the RRT*, which are shown to be asymptot-

ically optimal while preserving the computational effectiveness of, for instance, the RRT algorithm. Next, we consider the path planning problems with complex task specifications. We propose another novel algorithm, called the Rapidly-exploring Random Graph (RRG), that operates in an incremental way, like the RRT, and at the same time provides suitable guarantees on convergence to paths of a certain class. We also propose novel incremental model-checking algorithms that decide whether or not a task specification, given in the form of deterministic μ-calculus, is satisfied as the graph maintained by the RRG gets richer. A careful combination of this incremental model-checking algorithm with the RRG yields incremental sampling-based algorithms that guarantee convergence to paths that satisfy the given task specification. We also propose a sampling-based algorithm that guarantees almost-sure convergence towards an optimal path that satisfies the given task specification.

An important property of the proposed sampling-based algorithms is their *anytime* flavor; they provide a feasible solution almost as fast the existing computationally-effective sampling-based algorithms, such as the RRT, and improve this solution towards a globally optimal one if allowed more computation time. In fact, in many field implementations of sampling-based planning algorithms (see, e.g., Kuwata et al., 2009), it is often the case that, since a feasible path is found quickly, additional available computation time is devoted to improving the solution using heuristics until the solution is executed, without any performance guarantees.

The broader aim of this thesis is to outline a program to extend the application domain of anytime sampling-based algorithms beyond feasible motion planning problems, such as differential games, stochastic optimal control, and partially-observable optimal control problems. In all these applications, we seek sampling-based algorithms with suitable convergence and computational complexity guarantees, leading to an suitably-defined anytime flavor as in the case of RRT* and its variants. We denote a chapter to outline some of our recent work in this direction.

This material presented in this thesis is largely based on two recent publications by Karaman and Frazzoli (2011b, 2012). The material presented in Chapter ?? is taken from a number of publications co-authored by the author of this dissertation; these publications are referred to therein whenever appropriate.

1.5 Organization

This dissertation is organized as follows. First, some preliminary material is provided in Chapter 2. In particular, the notation used throughout the dissertation is introduced in Section 2.1; the theory of random geometric graphs and the μ-calculus are briefly introduced in Sections 2.2 and 2.3, respectively.

Chapter 3 is devoted to the formulation of the path planning problems that this dissertation is concerned with. The feasible path planning problem is introduced in Section 3.1, followed by the optimal path planning problem in Section 3.2. These problems are extended with complex task specifications in Section 3.3, where feasible and optimal path planning planning with deterministic μ-calculus are formulated.

Chapter 4 is devoted the introduction of the sampling-based algorithms analyzed

in this thesis. First, two important paradigms in sampling-based motion planning, namely the probabilistic roadmaps and the rapidly-exploring random trees, are introduced. Subsequently, novel algorithms that extend PRMs and RRTs to deal with optimal path planning problems and path planning problems with deterministic μ-calculus specifications are given. Finally, in Chapter 4.4, a simulation study is presented to demonstrate the proposed incremental optimal path planning algorithms in a number of illustrative examples.

In Chapter 5, both the existing and the proposed algorithms are analyzed in terms of probabilistic completeness, asymptotic optimality, and computational complexity. First, the results are reviewed, without proofs, in Section 5.1. Then, the proofs of all major results regarding completeness, optimality, and complexity are presented in Sections 5.2, 5.3, and 5.4, respectively.

The dissertation is concluded with a summary and some remarks in Chapter 6. This chapter also provides a review of a number of recent results that extend the algorithms presented in this thesis in a number of novel directions. In particular, some partial results for extending the proposed algorithms to handle motion planning problems involving non-trivial differential constraints is discussed. Moreover, recently-proposed incremental sampling-based algorithms for a number of control problems, namely differential games, stochastic optimal control, and optimal filtering, are reviewed. Along the way, a number of open problems are pointed out.

Chapter 2

Preliminaries

2.1 Notation

This section formalizes the notation used in this dissertation. When in doubt, the user is referred to this section to clarify notation. Unfortunately, clash of notation has been impossible to avoid. Under such circumstances, however, we routinely remind any unusual definition. Throughout the dissertation, we make every effort to follow the standard notation adopted in standard text books and widely-known manuscripts.

Let $\mathbb{N}$ denote the set of positive integers and $\mathbb{R}$ denote the set of reals. In a number of places, we use ω to denote the set of all non-negative integers, since this notation is commonly used in the mathematical logic literature. Let $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$, and $\mathbb{R}_{>0}$, $\mathbb{R}_{\geq 0}$ denote the sets of positive and non-negative reals, respectively. The cardinality of a set is denoted by $\text{card}\,(\cdot)$. A sequence on a set A is a mapping from $\mathbb{N}$ to A, denoted as $\{a_i\}_{i\in\mathbb{N}}$, where $a_i \in A$ is the element that $i \in \mathbb{N}$ is mapped to. Given $a, b \in \mathbb{R}$, closed and open intervals between a and b are denoted by $[a, b]$ and (a, b), respectively. The Euclidean norm is denoted by $|\cdot|$. We reserve the notation $\|\cdot\|$ for function norms, which we introduce later in the text. Given a set $\mathcal{X} \subset \mathbb{R}^d$, the closure of $\mathcal{X}$ is denoted by $\text{cl}(\mathcal{X})$. The closed ball of radius $r > 0$ centered at $x \in \mathbb{R}^d$, *i.e.*, $\{y \in \mathbb{R}^d : |y - x| \leq r\}$, is denoted by $\mathcal{B}_{x,r}$, also called the r-ball centered at x. Given a set $\mathcal{X} \subseteq \mathbb{R}^d$, the Lebesgue measure of $\mathcal{X}$ is denoted by $\mu(\mathcal{X})$. The Lebesgue measure of such a set is also referred to as its volume. The volume of the unit ball in $\mathbb{R}^d$, is denoted by ζ_d, *i.e.*, $\zeta_d = \mu(\mathcal{B}_{0,1})$. The letter e is used to denote the base of the natural logarithm, also called the Euler's number.

Given a probability space $(\Omega, \mathcal{F}, \mathbb{P})$, where Ω is a sample space, $\mathcal{F} \subseteq 2^\Omega$ is a σ-algebra, and $\mathbb{P}$ is a probability measure, an event A is an element of $\mathcal{F}$. The complement of an event A is denoted by A^c. Given a sequence of events $\{A_n\}_{n\in\mathbb{N}}$, the event $\cap_{n=1}^{\infty} \cup_{i=n}^{\infty} A_i$ is denoted by $\limsup_{n\to\infty} A_n$; In this case, A_n is said to occur infinitely often. The event $\cup_{n=1}^{\infty} \cap_{i=n}^{\infty} A_i$ is denoted by $\liminf_{n\to\infty} A_n$; In this particular case, A_n is said to occur for all large n. A (real) random variable is a measurable function that maps Ω into $\mathbb{R}$. An extended random variable can also take the values $\pm\infty$ with non-zero probability. If $\varphi(\omega)$ is a property that is either true or false for any given $\omega \in \Omega$, the event that denotes the set of all ω for which $\varphi(\omega)$

holds, *i.e.*, $\{\omega \in \Omega : \varphi(\omega)\}$, is simply written as $\{\varphi\}$, for notational convenience. For instance, $\{\omega \in \Omega : \lim_{n\to\infty} Y_n(\omega) = Y(\omega)\}$ is written as $\{\lim_{n\to\infty} Y_n = Y\}$. Expected value of a random variable Y is defined as $\mathbb{E}[Y] = \int_\Omega Y \, d\mathbb{P}$. A sequence of random variables $\{Y_n\}_{n\in\mathbb{N}}$ is said to converge surely to a random variable Y if $\lim_{n\to\infty} Y_n(\omega) = Y(\omega)$ for all $\omega \in \Omega$; the sequence is said to converge almost surely if $\mathbb{P}(\{\lim_{n\to\infty} Y_n = Y\}) = 1$. The Poisson random variable with parameter λ is denoted by Poisson(λ). The binomial random variable with parameters n and p is denoted by Binomial(n, p).

Let $f(n)$ and $g(n)$ be two functions with domain and range $\mathbb{N}$ or $\mathbb{R}$. The function $f(n)$ is said to be $O(g(n))$, denoted as $f(n) \in O(g(n))$, if there exists two constants M and n_0 such that $f(n) \leq Mg(n)$ for all $n \geq n_0$. The function $f(n)$ is said to be $\Omega(g(n))$, denoted as $f(n) \in \Omega(g(n))$, if there exists constants M and n_0 such that $f(n) \geq Mg(n)$ for all $n \geq n_0$. The function $f(n)$ is said to be $\Theta(g(n))$, denoted as $f(n) \in \Theta(g(n))$, if $f(n) \in O(g(n))$ and $f(n) \in \Omega(g(n))$.

Let $\mathcal{X}$ be a subset of $\mathbb{R}^d$. A (directed) graph $G = (V, E)$ on $\mathcal{X}$ is composed of a vertex set V and an edge set E, such that V is a finite subset of $\mathcal{X}$, and E is a subset of $V \times V$. A directed path on G is a sequence $(v_1, v_2, \ldots, v_n)$ of vertices such that $(v_i, v_{i+1}) \in E$ for all $1 \leq i \leq n-1$. Given a vertex $v \in V$, the sets $\{u \in V : (u, v) \in E\}$ and $\{u \in V : (v, u) \in E\}$ are said to be its incoming neighbors and outgoing neighbors, respectively. A (directed) tree is a directed graph, in which each vertex but one has a unique incoming neighbor. The vertex with no incoming neighbor is called the root vertex, and any vertex with no outgoing neighbors is called a leaf vertex.

2.2 Random Geometric Graphs

The objective of this section is to summarize some of the results on random geometric graphs that are available in the literature, and are relevant to the analysis of sampling-based path planning algorithms. In the remainder of this thesis, several connections are made between the theory of random geometric graphs and path-planning algorithms in robotics, providing insight on a number of issues, including, *e.g.*, probabilistic completeness and asymptotic optimality, as well as technical tools to analyze the algorithms and establish their properties. In fact, the data structures constructed by most sampling-based motion planning algorithms in the literature coincide, in the absence of obstacles, with the standard models of random geometric graphs.

Random geometric graphs are in general defined as stochastic collections of points in a metric space, connected pairwise by edges if certain conditions (*e.g.*, on the distance between the points) are satisfied. Such objects have been studied since their introduction by Gilbert (1961); See, *e.g.*, (Penrose, 2003; Bollobás and Riordan, 2006; Balister et al., 2008) for an overview of recent results.

A large amount of the literature on random geometric graphs is devoted to infinite graphs defined on unbounded domains, with vertices generated by a homogeneous Poisson point process. Recall that a Poisson random variable of parameter $\lambda \in \mathbb{R}_{>0}$ is an integer-valued random variable Poisson(λ) : $\Omega \to \mathbb{N}_0$ such that

$\mathbb{P}(\{\text{Poisson}(\lambda) = k\}) = e^{-\lambda}\lambda^k/k!$. A homogeneous Poisson point process of intensity λ on $\mathbb{R}^d$ is a random countable set of points $\mathcal{P}_\lambda^d \subset \mathbb{R}^d$ such that, for any disjoint measurable sets $\mathcal{S}_1, \mathcal{S}_2 \subset \mathbb{R}^d$, $\mathcal{S}_1 \cap \mathcal{S}_2 = \emptyset$, the numbers of points of $\mathcal{P}_\lambda^d$ in each set are independent Poisson variables, *i.e.*, $\text{card}\left(\mathcal{P}_\lambda^d \cap \mathcal{S}_1\right) = \text{Poisson}(\lambda\,\mu(\mathcal{S}_1))$ and $\text{card}\left(\mathcal{P}_\lambda^d \cap \mathcal{S}_2\right) = \text{Poisson}(\lambda\,\mu(\mathcal{S}_2))$. In particular, the intensity of a homogeneous Poisson point process can be interpreted as the expected number of points generated in the unit cube, *i.e.*, $\mathbb{E}(\text{card}\left(\mathcal{P}_\lambda^d \cap (0,1)^d\right)) = \mathbb{E}(\text{Poisson}(\lambda)) = \lambda$.

One of the most widely studied models of infinite random geometric graphs is the following, introduced by Gilbert (1961), and often called Gilbert's disc model, or the Boolean model:

Definition 2.1 (Infinite random r-disc graph) *Let $\lambda, r \in \mathbb{R}_{>0}$, and $d \in \mathbb{N}$. An infinite random r-disc graph $G_\infty^{\text{disc}}(\lambda, r)$ in d dimensions is an infinite graph with vertices $\{X_i\}_{i\in\mathbb{N}} = \mathcal{P}_\lambda^d$, and such that (X_i, X_j), $i, j \in \mathbb{N}$, is an edge if and only if $|X_i - X_j| < r$.*

A fundamental issue in infinite random graphs is whether the graph contains an infinite connected component, with non-zero probability. If it does, the random graph is said to *percolate*. Percolation is an important paradigm in statistical physics, with many applications in disparate fields such as material science, epidemiology, and microchip manufacturing, just to name a few (Sahimi, 1994).

Consider the infinite random r-disc graph, for $r = 1$, *i.e.*, $G_\infty^{\text{disc}}(\lambda, 1)$, and assume that the origin is one of the vertices of this graph. Let $p_k(\lambda)$ denote the probability that the connected component of $G_\infty^{\text{disc}}(\lambda, 1)$ containing the origin contains k vertices, and define $p_\infty(\lambda)$ as $p_\infty(\lambda) = 1 - \sum_{k=1}^{\infty} p_k(\lambda)$. The function $p_\infty : \lambda \to p_\infty(\lambda)$ is monotone, and $p_\infty(0) = 0$ and $\lim_{\lambda\to\infty} p_\infty(\lambda) = 1$ (Penrose, 2003). A key result in percolation theory is that there exists a *critical intensity* $\lambda_c := \sup\{\lambda : p_\infty(\lambda) = 0\}$, that is non-zero and finite. In other words, for all $\lambda > \lambda_c$, there is a non-zero probability that the origin is in an infinite connected component of $G_\infty^{\text{disc}}(\lambda, 1)$; moreover, under these conditions, the graph has precisely one infinite connected component, almost surely (Meester and Roy, 1996). The function p_∞ is continuous for all $\lambda \neq \lambda_c$: in other words, the graph undergoes a phase transition at the critical density λ_c, often also called the continuum percolation threshold Penrose (2003). The exact value of λ_c is not known; Meester and Roy (1996) provide $0.696 < \lambda_c < 3.372$ for $d = 2$, and simulations studies by Quintanilla and Torquato (2000) suggest that $\lambda_c \approx 1.44$.

For many applications, including those studied in this dissertation, models of finite graphs on a bounded domain are more relevant. An widely-studied such model (see Penrose, 2003) is defined below:

Definition 2.2 (Random r-disc graph) *Let $r \in \mathbb{R}_{>0}$, and $n, d \in \mathbb{N}$. A random r-disc graph $G^{\text{disc}}(n, r)$ in d dimensions is a graph whose n vertices, $\{X_1, X_2, \ldots, X_n\}$, are independent, uniformly distributed random variables in $(0,1)^d$, and such that (X_i, X_j), $i, j \in \{1, \ldots, n\}$, $i \neq j$, is an edge if and only if $|X_i - X_j| < r$.*

For finite random geometric graph models, one is typically interested in whether a random geometric graph possesses certain properties asymptotically as n increases.

Since the number of vertices is finite in random graphs, percolation can not be defined easily. In this case, percolation is studied in terms of the scaling of the number of vertices in the largest connected component with respect to the total number of vertices; in particular, a finite random geometric graph is said to percolate if it contains a "giant" connected component containing at least a constant fraction of all the nodes. As in the infinite case, percolation in finite random geometric graphs exhibits a suitably-defined phase transition. The following result is available for random r-disc graphs:

Theorem 2.3 (Percolation of random r-disc graphs (Penrose, 2003)) *Let $G^{\text{disc}}(n,r)$ be a random r-disc graph in $d \geq 2$ dimensions, and let $N_{\max}(G^{\text{disc}}(n,r))$ be the number of vertices in its largest connected component. Then, almost surely,*

$$\lim_{n\to\infty} \frac{N_{\max}(G^{\text{disc}}(n,r_n))}{n} = 0, \qquad \text{if } r_n < (\lambda_c/n)^{1/d},$$

and

$$\lim_{n\to\infty} \frac{N_{\max}(G^{\text{disc}}(n,r))}{n} > 0, \qquad \text{if } r_n > (\lambda_c/n)^{1/d},$$

where λ_c is the continuum percolation threshold.

A random r-disc graph with $\lim_{n\to\infty} n\, r_n^d = \lambda \in (0,\infty)$ is said to operate in the *thermodynamic limit*. The random r-disc graph is said to be in the *subcritical* regime when $\lambda < \lambda_c$ and in the *supercritical* regime when $\lambda > \lambda_c$.

Another property of interest is connectivity. Clearly, connectivity implies percolation. Interestingly, similar to percolation, emergence of connectivity in random geometric graphs also exhibits a phase transition.

Theorem 2.4 (Connectivity of random r-disc graphs (Penrose, 2003)) *Let $G^{\text{disc}}(n,r)$ be a random r-disc graph in d dimensions. Then,*

$$\lim_{n\to\infty} \mathbb{P}\left(\{G^{\text{disc}}(n,r) \text{ is connected }\}\right) = \begin{cases} 1, & \text{if } \zeta_d r^d > \log(n)/n, \\ 0, & \text{if } \zeta_d r^d < \log(n)/n, \end{cases}$$

where ζ_d is the volume of the unit ball in d dimensions.

Another model of random geometric graphs considers edges between k nearest neighbors. Both infinite and finite models are considered, as follows.

Definition 2.5 (Infinite random k-nearest neighbor graph) *Let $\lambda \in \mathbb{R}_{>0}$, and $d, k \in \mathbb{N}$. An infinite random k-nearest neighbor graph $G_\infty^{\text{near}}(\lambda,k)$ in d dimensions is an infinite graph with vertices $\{X_i\}_{i\in\mathbb{N}} = \mathcal{P}_\lambda^d$, and such that (X_i, X_j), $i,j \in \mathbb{N}$, is an edge if X_j is among the k nearest neighbors of X_i, or if X_i is among the k nearest neighbors of X_j.*

Definition 2.6 (Random k-nearest neighbor graph) *Let $d, k, n \in \mathbb{N}$. A random k-nearest neighbor graph $G^{\text{near}}(n,k)$ in d dimensions is a graph whose n vertices,*

$\{X_1, X_2, \ldots, X_n\}$, *are independent, uniformly distributed random variables in* $(0,1)^d$, *and such that* (X_i, X_j), $i, j \in \{1, \ldots, n\}$, $i \neq j$, *is an edge if* X_j *is among the* k *nearest neighbors of* X_i, *or if* X_i *is among the* k *nearest neighbors of* X_j.

Percolation and connectivity for random k-nearest neighbor graphs exhibit phase transition phenomena, as in the random r-disc case. However, the results available in the literature are more limited. In particular, the following result characterizes, to some degree, the percolation phenomenon in infinite k-nearest graphs:

Theorem 2.7 (Percolation in infinite random k-nearest graphs (Balister et al., 2008)) *Let* $G_\infty^{\text{near}}(\lambda, k)$ *be an infinite random* k*-nearest neighbor graph in* $d \geq 2$ *dimensions. Then, there exists a constant* $k_d^{\text{p}} > 0$ *such that*

$$\mathbb{P}\left(\{G_\infty^{\text{near}}(1,k) \text{ has an infinite component }\}\right) = \begin{cases} 1, & \text{if } k \geq k_d^{\text{p}}, \\ 0, & \text{if } k < k_d^{\text{p}}. \end{cases}$$

The value of k_d^{p} is not known. However, it is believed that $k_2^{\text{p}} = 3$, and $k_d^{\text{p}} = 2$ for all $d \geq 3$ (Balister et al., 2008). It is known that percolation does not occur for $k = 1$ (Balister et al., 2008).

Regarding connectivity of random k-nearest neighbor graphs, the only available results in the literature are not stated in terms of a given number of vertices, *i.e.*, in terms of the Binomial point process. Rather, the results are stated in terms of the restriction of a homogeneous Poisson point process to the unit cube. In other words, the vertices of the graph are obtained as $\{X_1, X_2, \ldots\} = \mathcal{P}_\lambda^d \cap (0,1)^d$. This is equivalent to setting the number of vertices as a Poisson random variable of parameter n, and then sampling the Poisson(n) vertices independently and uniformly in $(0,1)^d$:

Lemma 2.8 ((Stoyan et al., 1995)) *Let* $\{X_i\}_{i \in \mathbb{N}}$ *be a sequence of points drawn independently and uniformly from* $\mathcal{S} \subseteq \mathcal{X}$. *Let* Poisson$(n)$ *be a Poisson random variable with parameter* n. *Then,* $\{X_1, X_2, \ldots, X_{\text{Poisson}(n)}\}$ *is the restriction to* $\mathcal{S}$ *of a homogeneous Poisson point process with intensity* $n/\mu(\mathcal{S})$.

The main advantage in using such a model to generate the vertices of a random geometric graph is independence: in the Poisson case, the numbers of points in any two disjoint measurable regions $\mathcal{S}_1, \mathcal{S}_2 \subset [0,1]^d$, $\mathcal{S}_1 \cap \mathcal{S}_2 = \emptyset$, are independent Poisson random variables, with mean $\mu(\mathcal{S}_1)\lambda$ and $\mu(\mathcal{S}_2)\lambda$, respectively. These two random variables would not be independent if the total number of vertices were fixed a priori (also called a binomial point process). With some abuse of notation, such a random geometric graph model will be indicated as $G^{\text{near}}(\text{Poisson}(n), k)$.

Theorem 2.9 (Connectivity of random k-nearest graphs (Xue and Kumar, 2004)) *Let* $G^{\text{near}}(\text{Poisson}(n), k)$ *indicate a* k*-nearest neighbor graph model in* $d = 2$ *dimensions, such that its vertices are generated using a Poisson point process of intensity* n. *Then, there exists a constant* $k_2^{\text{c}} > 0$ *such that*

$$\lim_{n \to \infty} \mathbb{P}\left(\{G^{\text{near}}(\text{Poisson}(n), \lfloor k \log(n) \rfloor) \text{ is connected }\}\right) = \begin{cases} 1, & \text{if } k \geq k_2^{\text{c}}, \\ 0, & \text{if } k < k_2^{\text{c}}. \end{cases}$$

The value of k_2^c is not known; the current best estimate is $0.3043 \leq k_2^c \leq 0.5139$ (Balister et al., 2005).

Finally, the last model of random geometric graph that will be relevant for the analysis of the algorithms in this paper is the following:

Definition 2.10 (Online nearest neighbor graph) *Let $d, n \in \mathbb{N}$. An online nearest neighbor graph $G^{\mathrm{ONN}}(n)$ in d dimensions is a graph whose n vertices, $(X_1, X_2, \ldots, X_n)$, are independent, uniformly distributed random variables in $(0,1)^d$, and such that (X_i, X_j), $i, j \in \{1, \ldots, n\}$, $j > 1$, is an edge if and only if $|X_i - X_j| = \min_{1 \leq k < j} |X_k - X_j|$.*

Clearly, the online nearest neighbor graph is connected by construction. Recent results for this random geometric graph model include estimates of the total power-weighted edge length and an analysis of the vertex degree distribution (see, *e.g.*, Wade, 2009).

2.3 μ-calculus, Infinite Games, and Tree Automata

In this section, we introduce the necessary machinery to describe and study complex task specifications in a formal manner. First, we introduce a fixed-point logic called the μ-calculus. Subsequently, we focus on the deterministic fragment of μ-calculus and discuss its expressive power. Next, we introduce infinite parity games, a widely-studied class of two-person zero-sum infinite games played on a finite graph. We point out an important connection between a given μ-calculus formula and a corresponding infinite parity game, using alternating tree automata, which we also introduce in the text. Our notation closely follows that of Gradel et al. (2002).

Kripke Structures and the μ-calculus

In the computer science literature, Kripke structures are widely used as models of computer software and hardware, especially for the purposes of verification and automated synthesis. In what follows, we first define this model, and subsequently we introduce a fixed-point logic called the μ-calculus.

Let Π be a finite set. An element of Π is called *atomic proposition.*

Definition 2.11 (Kripke Structure) *A* Kripke structure *$\mathcal{K}$ defined on a set Π of atomic propositions is a tuple $\mathcal{K} = (\mathcal{S}, s_{\mathrm{init}}, \mathcal{R}, \mathcal{L})$, where $\mathcal{S}$ is a finite set of states, $s_{\mathrm{init}} \in \mathcal{S}$ is the initial state, $\mathcal{R} \subseteq \mathcal{S} \times \mathcal{S}$ is a transition relation, and $\mathcal{L} : \mathcal{S} \to 2^{\Pi}$ is a set of atomic propositions.*

The syntax of μ-calculus is defined as follows. Let `Var` be a finite set of variables.

Definition 2.12 (Syntax of μ-calculus) *The syntax of μ-calculus is given in the Backus-Naur form as follows:*

$$\phi ::= \bot \mid \top \mid p \mid \neg p \mid x \mid \phi \wedge \phi \mid \phi \vee \phi \mid \Diamond \phi \mid \Box \phi \mid \mu x.\phi \mid \nu x.\phi,$$

where $x \in \mathtt{Var}$ *and* $p \in \Pi$. *The set of all* μ*-calculus formulas is denoted by* L_μ.

Let $\mathcal{K} = (\mathcal{S}, s_{\text{init}}, \mathcal{R}, \mathcal{L})$ be a Kripke structure defined on a set Π of atomic propositions, and ϕ be a μ-calculus formula. We denote by $[\![\phi]\!]_\mathcal{K}$ the set of all states in $\mathcal{S}$ that satisfy ϕ. Below, we provide the semantics of the μ-calculus, which aims to define the set $[\![\phi]\!]_\mathcal{K}$ precisely for any given $\mathcal{K}$ and ϕ.

Given a set $S \subset \mathcal{S}$ of states and a variable $x \in \mathtt{Var}$, let $\mathcal{K}_x^S$ denote the Kripke structure $\mathcal{K}_x^S := (\mathcal{S}, s_{\text{init}}, \mathcal{R}, \mathcal{L}')$, defined on an augmented set $\Pi \cup \{x\}$ of atomic positions, where

$$\mathcal{L}' := \begin{cases} \mathcal{L}(s) \cup \{x\}, & \text{for all } s \in S, \\ \mathcal{L}(s), & \text{for all } s \notin S. \end{cases}$$

Then, the semantics of the μ-calculus is given as follows.

Definition 2.13 (Semantics of μ-calculus) *Let* $p \in \Pi$, $x \in \mathtt{Var}$, $\phi, \psi \in L_\mu$, *and* $\mathcal{K} = (\mathcal{S}, s_{\text{init}}, \mathcal{R}, \mathcal{L})$. *Then, the set* $[\![\phi]\!]_\mathcal{K}$ *of states that satisfy* ϕ *is defined recursively as follows:*

- $[\![\bot]\!]_\mathcal{K} = \emptyset$;
- $[\![\top]\!]_\mathcal{K} = \mathcal{S}$;
- $[\![p]\!]_\mathcal{K} = \{s \in \mathcal{S} : p \in \mathcal{L}(s)\}$;
- $[\![\neg p]\!]_\mathcal{K} = \{s \in \mathcal{S} : p \notin \mathcal{L}(s)\}$;
- $[\![\phi \wedge \psi]\!]_\mathcal{K} = [\![\phi]\!]_\mathcal{K} \cap [\![\psi]\!]_\mathcal{K}$;
- $[\![\phi \vee \psi]\!]_\mathcal{K} = [\![\phi]\!]_\mathcal{K} \cup [\![\psi]\!]_\mathcal{K}$;
- $[\![\Diamond\phi]\!]_\mathcal{K} = \{s \in \mathcal{S} :$ *there exists* $s' \in \mathcal{S}$ *such that* $(s, s') \in \mathcal{R}$ *and* $s' \in [\![\phi]\!]_\mathcal{K}\}$;
- $[\![\Box\phi]\!]_\mathcal{K} = \{s \in \mathcal{S} :$ *for all* $s' \in \mathcal{S}$ *with* $(s, s') \in \mathcal{R}$ *there holds* $s' \in [\![\phi]\!]_\mathcal{K}\}$;
- $[\![\mu x.\phi]\!]_\mathcal{K}$ *is the set-inclusion-wise least fixed point of the set-valued function* $f(S) = [\![\phi]\!]_{\mathcal{K}_x^S}$, i.e., $[\![\mu x.\phi]\!]_\mathcal{K}$ *is such that (i)* $[\![\mu x.\phi]\!]_\mathcal{K} = [\![\phi]\!]_{\mathcal{K}_x^{[\![\mu x.\phi]\!]_\mathcal{K}}}$ *and (ii) for any* $S \subseteq \mathcal{S}$ *with* $S = [\![\phi]\!]_{\mathcal{K}_x^S}$ *we have* $[\![\mu x.\phi]\!]_\mathcal{K} \subseteq S$;
- $[\![\nu x.\phi]\!]_\mathcal{K}$ *is the set-inclusion-wise greatest fixed point of the set-valued function* $f(S) = [\![\phi]\!]_{\mathcal{K}_x^S}$, i.e., $[\![\nu x.\phi]\!]_\mathcal{K}$ *is such that (i)* $[\![\nu x.\phi]\!]_\mathcal{K} = [\![\phi]\!]_{\mathcal{K}_x^{[\![\nu x.\phi]\!]_\mathcal{K}}}$ *and (ii) for any* $S \subseteq \mathcal{S}$ *with* $S = [\![\phi]\!]_{\mathcal{K}_x^S}$ *we have* $S \subseteq [\![\nu x.\phi]\!]_\mathcal{K}$.

For any L_μ formula ϕ and Kripke structure $\mathcal{K}$, semantics of the μ-calculus describes a recursive procedure to compute $[\![\phi]\!]_\mathcal{K}$, the set of all states in $\mathcal{K}$ satisfying ϕ. Note that this procedure requires computing least and greatest fixed points of set functions. These fixed points can be computed using a simple algorithm described in the Knaster-Tarski fixed point theorem provided below.

Theorem 2.14 (Knaster-Tarski Theorem (see *e.g.*, (Schneider, 2004))) *Let $\mathcal{K} = (\mathcal{S}, s_{\text{init}}, \mathcal{R}, \mathcal{L})$ be a Kripke structure and ϕ be an L_μ formula. Define Q_i recursively as follows: $Q_i := [\![\phi]\!]_{\mathcal{K}_x^{Q_{i-1}}}$, where $Q_0 := \emptyset$.*

Then, $Q_i \subseteq Q_{i-1}$ for all $i \in \mathbb{N}$, and there exists a number $n \in \mathbb{N}$ such that $Q_n = Q_{n-1} = [\![\mu x.\phi]\!]_{\mathcal{K}}$. Furthermore, if $Q_0 := \mathcal{S}$ in the above definition, then $Q_{i-1} \subseteq Q_i$ for all $i \in \mathbb{N}$, and there exists a number $m \in \mathbb{N}$ such that $Q_m = Q_{m-1} = [\![\nu x.\phi]\!]_{\mathcal{K}}$.

The *deterministic μ-calculus* is a fragment of the μ-calculus. The formulas of the deterministic μ-calculus is given by the following syntax. Recall that Π and `Var` denote the finite sets of atomic propositions and variables, respectively.

Definition 2.15 (Syntax of deterministic μ-calculus) *The syntax of the deterministic fragment of μ-calculus is given in the Backus-Naur form as follows:*

$$\phi ::= \bot \mid \top \mid p \mid \neg p \mid x \mid \phi \wedge p \mid \phi \wedge \neg p \mid \phi \vee \phi \mid \mu x.\phi \mid \nu x.\phi,$$

where $p \in \Pi$ and $x \in$ `Var`. The set of all deterministic μ-calculus formulas is denoted by L_1.

The deterministic fragment of μ-calculus differs from the full μ-calculus in two ways. Firstly, conjunctions of two arbitrary formulas are not allowed. More precisely, one of the formulas in a conjunction must be a literal, *i.e.*, either an atomic proposition or its negation. Secondly, the "$\Box$" operator is not present in the deterministic fragment of the μ-calculus.

In Section 3.3, we provide some examples of deterministic μ-calculus formulas in the context of robot motion planning. We also discuss the expressive power of the deterministic μ-calculus in the same section.

Infinite Parity Games

An infinite parity game is a zero-sum two-person infinite game played on a finite graph. Infinite parity games are essential in the definition of alternating tree automata, which we discuss in the next section.

Let $\mathcal{V}$ be a finite set vertices, and let $\mathcal{V}_1$ and $\mathcal{V}_2$ be a partition of $\mathcal{V}$, *i.e.*, $\mathcal{V}_1 \cap \mathcal{V}_2 = \emptyset$ and $\mathcal{V}_1 \cup \mathcal{V}_2 = \mathcal{V}$. An *arena* is a tuple $A = (\mathcal{V}_1, \mathcal{V}_2, \mathcal{E})$, where $\mathcal{V}_1$ and $\mathcal{V}_2$ are the set of vertices of players one and two, respectively, and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is an edge relation. Notice that there are no restrictions on the number of outgoing edges of a vertex, nor we assume that the partition $(\mathcal{V}_1, \mathcal{V}_2)$ makes the graph bipartite.

An infinite game played on a finite graph can be explained informally as follows. Initially, the players place a token on one of the vertices, usually called the initial vertex. In each stage, Player 1 (respectively Player 2) gets to push the token to a new vertex along the edges in $\mathcal{E}$, if the token is on one of the vertices in $\mathcal{V}_1$ (respectively in $\mathcal{V}_2$). Repeating this for as many stages as possible, the token travels on the vertices of the graph $(\mathcal{V}, \mathcal{E})$. Possibly, the game is played for infinitely many stages, if the token never gets stuck, *i.e.*, never reaches a vertex with no outgoing edges; otherwise, the game stops when the token gets stuck. This path of the token is called a play of

the game on arena $A = (\mathcal{V}_1, \mathcal{V}_2, \mathcal{E})$. More precisely, denoted by π, a *play* is a finite or an infinite sequence of vertices that constitutes path through the graph $(\mathcal{V}, \mathcal{E})$, *i.e.*, consecutive vertices that appear in π respect the edge relation. When the play is finite, *i.e.*, in the form $\pi = (v_0, v_1, \ldots, v_k)$, the final vertex v_k has no outgoing edges.

The winner of a given play is determined as follows. If the play π is finite, then the player that gets stuck loses the game. That is, Player 1 wins all plays of the form $\pi = (v_0, v_1, \ldots, v_k)$, where $v_k \in \mathcal{V}_2$; otherwise Player 2 wins. If the play π is infinite, the winner is decided in a more elaborate way. Let C be a finite set of non-negative integers. Define the *coloring function* as $\chi : \mathcal{V} \to C$, which assigns each vertex in $\mathcal{V}$ a color from C. Player 1 is said to win play $\pi = (v_0, v_1, \ldots)$, if the maximum color appearing infinitely often in π is even, *i.e.*, $\max\{\chi(v) : v \text{ is repeated infinitely often in } \pi\}$ is even; otherwise, Player 2 wins π. For future reference, we formalize the definition of an infinite parity game below.

Definition 2.16 (Infinite Parity Game) *An* infinite parity game *is a tuple* $(A = (\mathcal{V}_1, \mathcal{V}_2, \mathcal{E}), \chi)$*, where* A *is an arena and* χ *is a coloring function.*

A (memoryless) *strategy* for Player 1 is a function $h_1 : \mathcal{V}_1 \to \mathcal{V}$ such that $(v, h_1(v)) \in \mathcal{E}$ for all $v \in \mathcal{V}_1$. A play $\pi = (v_0, v_1, \ldots)$ is said to be *consistent* with strategy h_1, if $v_i \in \mathcal{V}_1$ implies that $v_{i+1} = h_1(v_i)$ for all $i \in \omega$. Roughly speaking, when Player 1 plays according to the strategy h_1, she moves the token to vertex $h_1(v)$ whenever the token is in vertex $v \in \mathcal{V}_1$. Since her strategy does not involve at all the history of the play, but only the current vertex, such a strategy is called memoryless.

A strategy h_1 is said to be a *winning strategy* for Player 1, if she wins all plays consistent with h_1. A winning strategy for Player 2 is defined similarly. An important result in the theory of infinite games is the fact that infinite parity games are memorylessly determined: in any infinite parity game, there exists a memoryless winning strategy either for Player 1 or for Player 2. Thus, it is enough to consider only the (finite) set of memoryless strategies when searching for a winning strategy.

Alternating Tree Automata

The usual finite state automata (see, *e.g.*, Sipser, 2006) accept or reject finite strings. Alternating tree automata are finite-state abstract computational devices that accept or reject Kripke structures. In fact, we will see in the next section that, for any μ-calculus formula ϕ, there exists an alternating tree automata that accepts exactly those Kripke structures $\mathcal{K} = (\mathcal{S}, s_{\text{init}}, \mathcal{R}, \mathcal{L})$ for which $s_{\text{init}} \in [\![\phi]\!]_{\mathcal{K}}$.

Let us note the following preliminary definitions before formalizing the behavior of an alternating tree automaton. Let $\mathcal{Q}$ be a finite set of automaton states. We define the set of *transition conditions*, denoted by $\text{TC}^{\mathcal{Q}}$, as follows: *(i)* 0 and 1 are in $\text{TC}^{\mathcal{Q}}$, *(ii)* $p, \neg p \in \text{TC}^{\mathcal{Q}}$ for all $p \in \Pi$, and *(iii)* $\Diamond q$, $\Box q$, $q_1 \wedge q_2$, and $q_1 \vee q_2$ are in $\text{TC}^{\mathcal{Q}}$ for all $q, q_1, q_2 \in \mathcal{Q}$. Then, an alternating tree automaton is defined as follows.

Definition 2.17 (Alternating tree automaton) *An* alternating tree automaton *is a tuple* $\mathcal{A} = (\mathcal{Q}, q_{\text{init}}, \delta, \Omega)$*, where* $\mathcal{Q}$ *is a finite set of states,* $q_{\text{init}} \in \mathcal{Q}$ *is an initial state,* $\delta : \mathcal{Q} \to \text{TC}^{\mathcal{Q}}$ *is a transition function, and* $\Omega : \mathcal{Q} \to \omega$ *is the priority function.*

The precise behavior of an alternating tree automaton is defined using infinite parity games. Let $\mathcal{K} = (\mathcal{S}, s_{\text{init}}, \mathcal{R}, \mathcal{L})$ be a Kripke structure defined on set Π of atomic propositions, and let $\mathcal{A} = (\mathcal{Q}, q_{\text{init}}, \delta, \Omega)$ be an alternating tree automaton. We define an infinite parity game $(A = (\mathcal{V}_1, \mathcal{V}_2, \mathcal{E}), \chi)$, where $\mathcal{V}_1 \cup \mathcal{V}_2 \subseteq (\{0,1\} \cup \Pi \cup \mathcal{Q}) \times \mathcal{S}$, as follows. Firstly, the sets $\mathcal{V}$ and $\mathcal{E}$ of vertices and edges are defined as follows.

- if $\delta(q) = 0$, then $(0, s) \in \mathcal{V}$;
- if $\delta(q) = 1$, then $(1, s) \in \mathcal{V}$;
- if $\delta(q) = p$ for some $p \in \Pi$, then $(p, s) \in \mathcal{V}$;
- if $\delta(q) = \neg p$ for some $p \in \Pi$, then $(\neg p, s) \in \mathcal{V}$;
- if $\delta(q) = q'$ for some $q \in \mathcal{Q}$, then $(q', s) \in \mathcal{V}$ and $\big((q, s), (q', s)\big) \in \mathcal{E}$;
- if $\delta(q) = q_1 \wedge q_2$ or $\delta(q) = q_2 \vee q_2$ for some $q_1, q_2 \in \mathcal{Q}$, then $(q_1, s), (q_2, s) \in \mathcal{V}$ and $\big((q, s), (q_1, s)\big), \big((q, s), (q_2, s)\big) \in \mathcal{E}$;
- if $\delta(q) = \Diamond q'$ or $\delta(q) = \Box q'$ for some $q' \in \mathcal{Q}$, then $(q', s') \in \mathcal{V}$ and $\big((q, s), (q', s')\big) \in \mathcal{E}$ for all $s' \in \mathcal{S}$ with $(s, s') \in \mathcal{R}$.

Second, the vertex sets $\mathcal{V}_1$ and $\mathcal{V}_2$ that partition $\mathcal{V}$ are defined as follows. A vertex $v = (q, s) \in \mathcal{V}$ belongs to Player 1, *i.e.*, $v \in \mathcal{V}_1$, if one of the following holds: *(i)* $\delta(q) = 0$, *(ii)* $\delta(q) = p$ and $p \notin \mathcal{L}(s)$, *(iii)* $\delta(q) = \neg p$ and $p \in \mathcal{L}(s)$, *(iv)* $\delta(q) = q'$ for some $q' \in \mathcal{Q}$, *(v)* $\delta(q) = q_1 \vee q_2$ for some $q_1, q_2 \in \mathcal{Q}$, or *(vi)* $\delta(q) = \Diamond q'$ for some $q' \in \mathcal{Q}$. Vertex v belongs to Player 2 otherwise, *i.e.*, when one of the following conditions are satisfied: *(i)* $\delta(q) = 1$, *(ii)* $\delta(q) = p$ and $p \in \mathcal{L}(s)$, *(iii)* $\delta(q) = q_1 \wedge q_2$ for some $q_1, q_2 \in \mathcal{Q}$, or *(iv)* $\delta(q) = \Box q'$ for some $q' \in \mathcal{Q}$.

Third, the coloring function χ of the infinite parity game is defined as $\chi((q, s)) = \Omega(q)$. In the sequel, we denote the resulting game $(A = (\mathcal{V}_1, \mathcal{V}_2, \mathcal{E}), \chi)$ by $\mathcal{G}_{\mathcal{A},\mathcal{K}}$.

Finally, we are ready to present the behavior of the automaton $\mathcal{A}$ on a Kripke structure $\mathcal{K}$ as follows: The automaton $\mathcal{A}$ accepts the Kripke structure $\mathcal{K}$, if Player 1 has a winning strategy in the game $\mathcal{G}_{\mathcal{A},\mathcal{K}}$. The *language* of an alternating tree automaton $\mathcal{A}$ is the set of all Kripke structures accepted by $\mathcal{A}$.

From μ-calculus to Alternating Tree Automata

The μ-calculus and the theory of alternating tree automata are tightly connected. For any μ-calculus formula ϕ, there exists an alternating tree automaton $\mathcal{A}_\phi$ such that the following holds: for any Kripke structure $\mathcal{K} = (\mathcal{S}, s_{\text{init}}, \mathcal{R}, \mathcal{L})$, we have $s_{\text{init}} \in [\![\phi]\!]_{\mathcal{K}}$ if and only if $\mathcal{A}_\phi$ accepts $\mathcal{K}$. This section is devoted to the construction of $\mathcal{A}_\phi$. The correctness of the construction is proven in (Gradel et al., 2002).

First let us note the following preliminary definitions. Let ϕ be a μ-calculus formula. Then, ψ is said to be a *subformula* of ϕ, denoted by $\psi \leq \phi$, if ϕ contains ψ as a substring. Roughly speaking, a variable $x \in \texttt{Var}$ is said to be a *free variable* in ϕ if it is not bound with a fixed point operator. More precisely, the set of all free variables in ϕ is defined recursively as follows: *(i)* $\texttt{FreeVar}_\phi(\bot) = \texttt{FreeVar}_\phi(\top) = \emptyset$, *(ii)* $\texttt{FreeVar}_\phi(p) = \texttt{FreeVar}_\phi(p) = \emptyset$ for all $p \in \Pi$, *(iii)* $\texttt{FreeVar}_\phi(x) = \{x\}$ for all $x \in \texttt{Var}$, *(iv)* $\texttt{FreeVar}_\phi(\psi_1 \wedge \psi_2) = \texttt{FreeVar}_\phi(\psi_1 \vee \psi_2) = \texttt{FreeVar}_\phi(\psi_1) \cap \texttt{FreeVar}_\phi(\psi_2)$ for $\psi_1, \psi_2 \in L_\mu$, *(v)* $\texttt{FreeVar}_\phi(\Diamond\psi) = \texttt{FreeVar}_\phi(\Box\psi) = \texttt{FreeVar}_\phi(\psi)$ for all $\psi \in L_\mu$,

and *(vi)* $\texttt{FreeVar}_\phi(\mu x.\psi) = \texttt{FreeVar}_\phi(\nu x.\psi) = \texttt{FreeVar}_\phi(\psi) \setminus \{x\}$. A variable x is said to be *bound* in ϕ, if x is a subformula of ϕ but it is not a free variable in ϕ. Finally, ϕ is said to be in *normal formal* if every bound variable is bound exactly once, *i.e.*, for all $x \in \texttt{Var}$ with $x \leq \phi$, there exists a unique subformula $\psi \leq \phi$ of the form $\psi = \mu x.\psi'$ or $\psi = \nu x.\psi'$. This unique subformula is called the *binding formula* of x, and it is denoted by $\texttt{BindingFormula}_\phi(x)$. Let us note that any μ-calculus formula can be put into normal form simply by renaming those variables that appear more than once. In the sequel, we tacitly assume that any μ-calculus under consideration is in normal form.

The *alternation depth* of a μ-calculus formula ϕ is an integer that counts the number of alternations of least and greatest fixed points in ϕ. More precisely, the alternation depth of ϕ, denoted by $\alpha(\phi)$, is defined recursively as follows: *(i)* $\alpha(\bot) = \alpha(\top) = 0$, *(ii)* $\alpha(p) = \alpha(\neg p) = 0$ for all $p \in \Pi$, *(iii)* $\alpha(\Box\psi) = \alpha(\Diamond\psi) = \alpha(\psi)$, *(iv)* $\alpha(\mu x.\psi) = \max\{\{1.\alpha(\psi)\} \cup \{\alpha(\nu x'.\psi') + 1 : \nu x'.\psi' \leq \psi \text{ and } x \in \texttt{FreeVar}_\phi(\nu x'.\psi')\}\}$, and *(v)* $\alpha(\nu x.\psi) = \max\{\{1.\alpha(\psi)\} \cup \{\alpha(\mu x'.\psi') + 1 : \mu x'.\psi' \leq \psi \text{ and } x \in \texttt{FreeVar}_\phi(\mu x'.\psi')\}\}$.

For notational convenience, let us denote the set of all μ-calculus formulas of the form $\mu.x.\psi$ and $\nu x.\psi$ by F_μ and F_ν, respectively, and let us define $F_\eta := F_\mu \cup F_\nu$. Without loss of any generality, assume that all atomic propositions in Π and all variables in $\texttt{Var}$ appear in the formula ϕ. For any subformula $\psi \leq \phi$, we define a state, denoted by $\langle\psi\rangle$. Finally, the automata $\mathcal{A}_\phi = (\mathcal{Q}, q_{\text{init}}, \delta, \Omega)$ is defined as follows:

- $\mathcal{Q} = \{\langle\psi\rangle : \psi \leq \phi\}$;
- $q_{\text{init}} = \langle\phi\rangle$;
- $\delta : \mathcal{Q} \to \text{TC}^{\mathcal{Q}}$ is defined recursively as follows: *(i)* $\delta\langle\bot\rangle = 0$ and $\delta\langle\top\rangle = 1$, *(ii)* $\delta\langle p\rangle = \langle p\rangle$ and $\delta\langle\neg p\rangle = \langle\neg p\rangle$ for all $p \in \Pi$, *(iii)* $\delta\langle x\rangle = \langle\texttt{BindingFormula}_\phi(x)\rangle$ for all $x \in \texttt{Var}$, *(iv)* $\delta\langle\psi_1 \wedge \psi_2\rangle = \langle\psi_1\rangle \wedge \langle\psi_2\rangle$ and $\delta\langle\psi_1 \vee \psi_2\rangle = \langle\psi_1\rangle \vee \langle\psi_2\rangle$ for all $\psi_1 \wedge \psi_2, \psi_1 \vee \psi_2 \leq \phi$, *(v)* $\delta\langle\Diamond\psi\rangle = \Diamond\langle\psi\rangle$ and $\delta\langle\Box\psi\rangle = \Box\langle\psi\rangle$ for all $\Diamond\psi, \Box\psi \leq \phi$, and *(vi)* $\delta\langle\mu x.\psi\rangle = \delta\langle\nu x.\psi\rangle = \langle\psi\rangle$ for all $\mu x.\psi, \nu x.\psi \leq \phi$;
- $\Omega : \mathcal{Q} \to \omega$ is defined as follows: *(i)* $\Omega(\langle\psi\rangle)$ is the smallest odd number greater than or equal to $\alpha(\psi) - 1$ if $\psi \in F_\mu$, *(ii)* $\Omega(\langle\psi\rangle)$ is the smallest even number greater than or equal to $\alpha(\psi) - 1$ if $\psi \in F_\nu$, and *(iii)* $\Omega(\langle\psi\rangle) = 0$ if $\langle\psi\rangle \notin F_\eta$.

The μ-calculus and Infinite Games

The automaton $\mathcal{A}_\phi$ gives rise to an infinite parity game called the L_μ game.

Definition 2.18 (L_μ Game) *Let $\mathcal{K} = (\mathcal{S}, s_{\text{init}}, \mathcal{R}, \mathcal{L})$ be a Kripke structure and ϕ be a μ-calculus formula. The L_μ game for $\mathcal{K}$ and ϕ is the infinite parity game $(A = (\mathcal{V}_1, \mathcal{V}_2, \mathcal{E}), \chi)$ defined as follows. For all $s \in \mathcal{S}$,*

- *$(s, p) \in \mathcal{V}_1$ for all $p \in \Pi$ with $p \notin \mathcal{L}(s)$;*
- *$(s, \neg p) \in \mathcal{V}_1$ for all $p \in \Pi$ with $p \in \mathcal{L}(s)$;*
- *$(s, x) \in \mathcal{V}_1$ and $((s, x), (s, \texttt{BindingFormula}_\phi(x))) \in \mathcal{E}$ for all $x \in \texttt{Var}$;*
- *$(s, \psi_1 \vee \psi_2) \in \mathcal{V}_1$ and $\big((s, \psi_1 \vee \psi_2), (s, \psi_1)\big), \big((s, \psi_1 \vee \psi_2), (s, \psi_2)\big) \in \mathcal{E}$ for all $\psi_1 \vee \psi_2 \leq \phi$;*

- $(s, \Diamond\psi) \in \mathcal{V}_1$ *and* $\big((s, \Diamond\psi), (s', \psi)\big) \in \mathcal{E}$ *for all* $\Diamond\psi \leq \phi$ *and all* $(s, s') \in \mathcal{R}$;
- $(s, \eta x.\psi) \in \mathcal{V}_1$ *and* $\big((s, \eta x.\psi)\big) \in \mathcal{E}$ *for all* $\eta x.\psi \leq \phi$, *where* $\eta \in \{\mu, \nu\}$;
- $(s, p) \in \mathcal{V}_2$ *for all* $p \in \Pi$ *with* $p \in \mathcal{L}(s)$;
- $(s, \neg p) \in \mathcal{V}_2$ *for all* $p \in \Pi$ *with* $p \notin \mathcal{L}(s)$;
- $(s, \psi_1 \wedge \psi_2) \in \mathcal{V}_2$ *and* $\big((s, \psi_1 \wedge \psi_2), (s, \psi_1)\big), \big((s, \psi_1 \wedge \psi_2), (s, \psi_2)\big) \in \mathcal{E}$ *for all* $\psi_1 \wedge \psi_2 \leq \phi$;
- $(s, \Box\psi) \in \mathcal{V}_2$ *and* $\big((s, \Box\psi), (s', \psi)\big) \in \mathcal{E}$ *for all* $\Box\psi \leq \phi$ *and all* $(s, s') \in \mathcal{R}$.

The coloring function $\chi : \mathcal{V} \to \{0, 1, \ldots, \alpha(\phi) + 1\}$ *satisfies: for all* $s \in \mathcal{S}$ *and* $\psi \leq \phi$,

- *if* ψ *is of the form* $\psi = \mu x.\psi'$, *then* $\chi(\psi)$ *is equal to the smallest odd number greater than or equal to* $\alpha(\psi)$;
- *if* ψ *is of the form* $\psi = \nu x.\psi'$, *then* $\chi(\psi)$ *is equal to the smallest even number greater than or equal to* $\alpha(\psi)$;
- *otherwise,* $\chi(\psi) = 0$.

The following theorem can be deduced easily from the constructions presented in the previous sections.

Theorem 2.19 *Let* $\mathcal{K} = (\mathcal{S}, s_{\text{init}}, \mathcal{R}, \mathcal{L})$ *be a Kripke structure and* ϕ *be a* μ*-calculus formula. Then,* ϕ *is satisfied at* s_{init} *of* $\mathcal{K}$, i.e., $s_{\text{init}} \in [\![\phi]\!]_{\mathcal{K}}$ *if and only if Player 1 has a (memoryless) winning strategy in the* L_μ *game for* $\mathcal{K}$ *and* ϕ.

An important class of infinite parity games captures exactly those that are induced by a deterministic μ-calculus formula. These games are endowed with a number of important properties which we point out in the rest of this section.

The L_1 game of a deterministic μ-calculus formula ϕ and a Kripke structure $\mathcal{K}$ is defined as in Defintion 2.18. Theorem 2.19 holds also for L_1 games. Notice that in the L_1 game for $\mathcal{K}$ and ϕ, the actions of Player 2 are very limited. Firstly, Player 2 does not have any vertices of the form $(s, \Box\psi)$, since the "$\Box$" operator is not present in the deterministic fragment of μ-calculus. Second, any vertex of Player 2 of the form $(s, \psi \wedge \psi')$ satisfies either $\psi' = p$ or $\psi' = \neg p$ for some $p \in \Pi$. In fact, these limitations are severe enough that, roughly speaking, Player 2 can not take any actions that influences the game for more than one stage. This "deterministic" nature of the L_1 game is tightly connected to the expressive power of the deterministic μ-calculus being limited only to linear-time properties.

L_1 games are important for the purposes of this thesis for two reasons. Firstly, L_1 games will be used to provide a rigorous definition of optimal path planning problems with complex (linear-time) task specifications given in the form of deterministic μ-calculus (see Section 3.3). Secondly, the memoryless determinacy of L_1 games and their simple structure will be the key ingredient in the analysis of a computationally-efficient incremental model-checking algorithm (see Section 4.3.4).

Chapter 3

Path Planning Problems

This chapter is devoted to a formal definition of the path planning problems that this thesis is concerned with. The feasible and optimal path planning problems are formalized in Sections 3.1 and 3.2, respectively. In Section 3.3, these problems are extended with complex task specifications given in the form of deterministic μ-calculus.

3.1 Feasible Path Planning

Roughly speaking, the feasible path problem is to find a collision-free path between an initial and a final configuration. We formalize this problem as follows. The *configuration space*, *i.e.*, the set of all configurations of the robot, is defined as $\mathcal{X} := (0,1)^d$. The *obstacle set* and the *goal set* are denoted by $\mathcal{X}_{\text{obs}}$ and $\mathcal{X}_{\text{goal}}$, respectively, and the *initial configuration* is denoted by x_{init}. We assume that $\mathcal{X}_{\text{obs}}$ and $\mathcal{X}_{\text{goal}}$ are open sets. The *free space* is defined as $\mathcal{X}_{\text{free}} := \text{cl}(\mathcal{X} \setminus \mathcal{X}_{\text{obs}})$, where $\text{cl}(\cdot)$ is the closure operator,

Given a function $\sigma : [0,1] \to \mathbb{R}^d$, its *total variation* is defined as follows:

$$\text{TV}(\sigma) := \sup_{n \in \mathbb{N},\, 0=\tau_0<\tau_1<\cdots<\tau_n=s} \sum_{i=1}^{n} |\sigma(\tau_i) - \sigma(\tau_{i-1})|$$

A function σ with $\text{TV}(\sigma) < \infty$ is said to have *bounded variation*. A continuous function $\sigma : [0,1] \to \mathbb{R}^d$ is called a *path* if it has bounded variation. Note that the total variation of a path is essentially its length, in other words the Euclidean distance traversed by the path in $\mathbb{R}^d$. Thus, a path has finite length by definition.

A path is said to be *collision free* if it avoids the obstacle set, *i.e.*, $\sigma(\tau) \in \mathcal{X}_{\text{free}}$ for all $\tau \in [0,1]$. A collision-free path is said to be *feasible*, if it starts form the initial configuration and reaches a goal configuration, *i.e.*, $\sigma(0) = x_{\text{init}}$ and $\sigma(1) \in \mathcal{X}_{\text{goal}}$. The feasible path planning problem is to find a feasible path if one exists, and return failure otherwise. An instance of this problem is denoted by $(\mathcal{X}_{\text{free}}, x_{\text{init}}, \mathcal{X}_{\text{goal}})$.

Problem 3.1 (Feasible Path Planning) *Given a feasible path planning problem instance* $(\mathcal{X}_{\text{free}}, x_{\text{init}}, \mathcal{X}_{\text{goal}})$, *find a path* $\sigma : [0,1] \to \mathbb{R}^d$ *that:*

- *starts from the initial configuration,* i.e., $\sigma(0) = x_{\text{init}}$,
- *reaches the goal set,* i.e., $\sigma(1) \in \mathcal{X}_{\text{goal}}$, *and*
- *avoids collision with obstacles,* i.e., $\sigma(\tau) \in \mathcal{X}_{\text{free}}$ *for all* $\tau \in [0,1]$.

3.2 Optimal Path Planning

Let Σ denote the set of all paths. A function $c : \Sigma \to \mathbb{R}_{\geq 0}$ is said to be a *cost function* if it assigns a non-negative cost to all non-trivial paths, *i.e.*, $c(\sigma) = 0$ if and only if $\sigma(\tau) = \sigma(0)$ for all $\tau \in (0, 1]$. The *optimal path planning problem* asks for finding a feasible path with minimum cost. We denote an instance of the optimal path planning problem by the tuple $(\mathcal{X}_{\text{free}}, x_{\text{init}}, \mathcal{X}_{\text{goal}}, c)$.

Problem 3.2 (Optimal Path Plannig) *Given an optimal path planning problem instance* $(\mathcal{X}_{\text{free}}, x_{\text{init}}, \mathcal{X}_{\text{goal}}, c)$, *find a feasible path* σ^* *such that* $c(\sigma^*) = \min\{c(\sigma) : \sigma \text{ is feasible}\}$, *and return failure if no such path exists.*

3.3 Path Planning Problems with Deterministic μ-calculus Specifications

In this section, we introduce a novel class of path planning problems. As opposed to the classical path planning problem of finding a path that "reaches the goal region while avoiding collision with the obstacle region," this new class of problems asks for finding a path that satisfies a given complex task specification given in the form of deterministic μ-calculus. In the rest of this section, we first formalize this class of problems and subsequently discuss its generality.

Recall that a path is defined as a continuous function with bounded variation, of the form $\sigma : [0, 1] \to \mathcal{X}$. During execution, the robot implements this path starting from configuration $\sigma(0)$ and eventually terminating at configuration $\sigma(1)$. In this section, we will consider infinite-horizon paths, *i.e.*, motions that do not necessarily terminate. These paths capture, in particular, tasks that are repeated infinitely often, such as persistent surveillance. For all practical purposes, however, we are only interested in infinite-horizon paths that can be finitely parametrized.In this dissertation, we consider paths that are represented by a finite prefix together with an infinitely-repeated suffix.[1] More precisely, an *infinite-horizon path* is parametrized by a prefix and a suffix, denoted by σ_p and σ_s, respectively, such that $\sigma_p(1) = \sigma_s(0) = \sigma_s(1)$, *i.e.*, the suffix starts and ends at the configuration where the prefix ends; during execution time, the robot first implements σ_p followed by an infinitely-repeated implementation of σ_s. By setting the suffix path to a constant value, *i.e.*, $\sigma_s(\tau) = \sigma_p(1)$ for all $\tau \in [0, 1]$, this definition also captures motions that terminate. In the sequel, we denote infinite-horizon paths parameterized in this way by (σ_p, σ_s).

Recall that the configuration space is denoted by $\mathcal{X} \subset \mathbb{R}^d$. Let $R_1, R_2, \ldots, R_k \subseteq \mathcal{X}$ be open sets. Define the corresponding set of atomic propositions as $\Pi = \{p_1, p_2, \ldots, p_k\}$. Given a configuration $x \in \mathcal{X}$, the *set of atomic propositions satisfied by* x is a set-valued function defined as follows: $\lambda(x) := \{p_k : x \in R_k\}$.

[1]Note that this parametrization is not restrictive when working with the deterministic μ-calculus. Since any deterministic μ-calculus specification can be represented by a non-deterministic Büchi automaton (see, *e.g.*, Gradel et al., 2002), by the definition, any satisfying path is a finite prefix followed by an infinitely-repeated suffix. That is, any deterministic μ-calculus specification can be satisfied by executing such a path, if the specification can be satisfied at all.

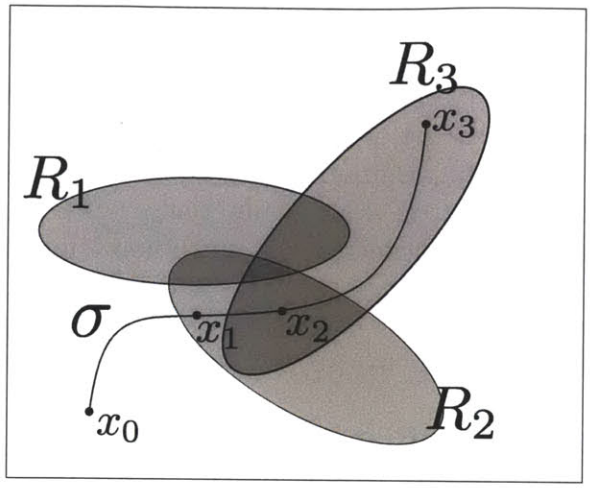


Figure 3-1: The trace (x_0, x_1, x_2, x_3) of a path σ is illustrated. The regions R_1, R_2, R_3 are also shown.

Let $\sigma : [0,1] \to \mathcal{X}$ be a (finite-horizon) path. The *trace* of σ is a sequence $(x_0, x_1, \ldots, x_l)$ of configurations, for which there exists non-negative real numbers $\tau_0 < \tau_1 < \cdots < \tau_l$ such that *(i)* $x_i = \sigma(\tau_i)$ for all $i \in \{0, 1, \ldots, l\}$, *(ii)* $\tau_0 = 0$ and $\tau_l = 1$, *(iii)* $\lambda(\sigma(\tau_i)) \neq \lambda(\sigma(\tau_{i+1}))$ for all $i \in \{0, 1, \ldots, l-1\}$, and *(iv)* $\lambda(\sigma(\tau)) \in \{\lambda(\sigma(\tau_i)), \lambda(\sigma(\tau_{i+1}))\}$ for all $\tau \in (\tau_i, \tau_{i+1})$ and all $i \in \{0, 1, \ldots, l-1\}$. That is, the trace of a path is a minimal sequence of configurations along the path such that no change in the labels $\lambda(\sigma(\tau))$ is skipped as τ varies. See Figure 3-1. It is trivial to show that the trace corresponding to a given path is always unique.

Let (σ_p, σ_s) be an infinite-horizon path. The *Kripke structure induced by* (σ_p, σ_s) is denoted by $\mathcal{K}_{(\sigma_p,\sigma_s)} = (\mathcal{S}, s_{\text{init}}, \mathcal{R}, \mathcal{L})$ and defined as follows:

- $\mathcal{S} = \{s_0, s_2, \ldots, s_l, s_{l+1}, \ldots s_{l+m}\}$;
- $s_{\text{init}} = s_0$;
- (s_i, s_i) for all $i \in \{0, 1, \ldots, l+m\}$, $(s_i, s_{i+1}) \in \mathcal{R}$ for all $i \in \{0, 1, \ldots, l+m-1\}$, and $(s_{l+m-1}, s_l) \in \mathcal{R}$;
- $\mathcal{L}(s_i) = \lambda(x_i^p)$ for all $i \in \{0, 1, \ldots, l\}$, and $\mathcal{L}(s_{l+i}) = \lambda(x_i^s)$ for all $i \in \{1, 2, \ldots, m\}$, where $(x_1^p, x_2^p, \ldots, x_l^p)$ and $(x_1^s, x_2^s, \ldots, x_m^s)$ are the traces of σ_p and σ_s.

A natural extension of the feasible path planning problem (see Problem 3.1) is the problem of feasible path planning with deterministic μ-calculus specifications. The latter problem is formalized as follows. We denote an instance of the feasible path planning problem with deterministic μ-calculus specifications by the tuple $(x_{\text{init}}, R_1, R_2, \ldots, R_k, \phi_{\text{spec}})$, where ϕ_{spec} is a deterministic μ-calculus formula.

Problem 3.3 (Feasible Path Planning with Deterministic μ-calculus Specifications) *Given a problem instance* $(x_{\text{init}}, R_1, R_2, \ldots, R_k, \phi_{\text{spec}})$, *find an infinite-horizon path* (σ_p, σ_s) *such that the Kripke structure* $\mathcal{K}_{(\sigma_p,\sigma_s)} = (\mathcal{S}, s_{\text{init}}, \mathcal{R}, \mathcal{L})$ *induced by* (σ_p, σ_s) *satisfies the following:* $s_{\text{init}} \in [\![\phi_{\text{spec}}]\!]_{\mathcal{K}_{(\sigma_p,\sigma_s)}}$.

An infinite-horizon path is said to be *feasible* if it solves Problem 3.3.

The optimal path planning with deterministic μ-calculus specification is an extension of the optimal path planning problem presented in Problem 3.2 in a similar way. Recall that Σ denotes the set of all (finite-horizon) paths. The set of all infinite-horizon paths (that are finitely parametrized by a prefix and a suffix) is a well-defined subset of the Cartesian product $\Sigma \times \Sigma$. A function $c : \Sigma \times \Sigma \to \mathbb{R}_{\geq 0}$ is said to be a *cost function* if it assigns a non-negative cost to all non-trivial infinite-horizon paths, *i.e.*, $c\big((\sigma_p, \sigma_s)\big) = 0$ if and only if $\sigma_p(\tau) = \sigma_p(0)$ and $\sigma_s(\tau) = \sigma_s(0)$.

An instance of the optimal path planning problem with deterministic μ-calculus specifications is denoted by $(x_{\text{init}}, R_1, R_2, \ldots, R_k, \phi_{\text{spec}}, c)$.

Problem 3.4 (Optimal Path Planning with Deterministic μ-calculus Specifications) *Given a problem instance* $(x_{\text{init}}, R_1, R_2, \ldots, R_k, \phi_{\text{spec}}, c)$, *find a feasible path* (σ_p^*, σ_s^*) *such that* $c\big((\sigma_p^*, \sigma_s^*)\big) = \min\{c\big((\sigma_p, \sigma_s)\big) : (\sigma_p, \sigma_s)$ *is feasible*$\}$.

Deterministic μ-calculus is the fragment of modal μ-calculus in which no branching property can be expressed. Rather than a limitation, this is a desirable feature for motion planning problems, since the motion plan in the end is itself a "trajectory" respecting the linear flow of the time. Hence, by employing the deterministic μ-calculus we rule out all the branching-time specifications and focus only on those specifications for which a linear time trajectory can be generated. In terms of its expressive power, the deterministic μ-calculus is strictly more expressive than the Linear Temporal Logic (LTL); See, *e.g.*, (Emerson et al., 2001; Henzinger et al., 2005). More precisely, L_1 is known to be equally expressive as the set of all $\omega-$regular properties. That is, any temporal property that can be expressed using, for instance, non-deterministic Büchi automata (Thomas, 1991), can be expressed in the deterministic μ-calculus (see Emerson et al., 2001; Henzinger et al., 2005, for constructive proofs). Hence, deterministic μ-calculus is indeed the most expressive regular language that can be used for the specification of linear time properties, which makes it the most expressive temporal logic for motion planning applications.

Despite its raw expressive power, the μ-calculus is not well accepted for direct use in practical applications due to its unnatural semantics. That is, unlike, for instance, the LTL, long μ-calculus specifications are found to be quite hard to understand by inspection, and expressing temporal properties using μ-calculus, even though possible, is hard for humans. However, there are algorithms which convert a given temporal logic specification, *e.g.*, in LTL, into a deterministic μ-calculus specification automatically (see, *e.g.*, Emerson et al., 2001). To further introduce the deterministic fragment of μ-calculus, we present a few example formulas below.

Reachability Specifications: Consider the μ-calculus formula $\phi_{\text{spec}} = \mu x.(p \vee \Diamond x)$. In words, ϕ_{spec} is satisfied by the smallest set of states, which, if labeled with x, would satisfy $p \vee \Diamond x$. Notice that such set is the set Q of all states that either satisfy p or can reach a state that satisfies p. One way to see this is to carry out the iteration in the Tarski-Knaster theorem (see Theorem 2.14): Q_1 is the set of states that satisfy p, Q_2 is the set of states that either satisfy p (that are in Q_1) or that have an outgoing

edge to a state that satisfies p, Q_3 is the set of states that are either in Q_2 or have a transition to a state in Q_2, etc. This iteration converges to the set of all states from which there is a trajectory leading to a state that satisfies p. Another, perhaps more intuitive look at ϕ_{spec} is the following. First, note that such set of states is indeed a fixed point, *i.e.*, if Q is labeled with x, then the set of states that satisfy $p \vee \Diamond x$ would be Q itself. That is, all the states in Q satisfy $p \vee \Diamond x$ and no other state outside Q satisfy $p \vee \Diamond x$. The former statement is true, since each state in Q either satisfy p or has a transition to a state that satisfies x. The latter one is also correct, since if there is any state s' that is not in Q but it satisfies $p \vee \Diamond x$, then it either has to satisfy p or it has to have a transition to a state which is labeled with x. In any case, s' would have a path that reaches Q and thus reaches a state that satisfy p. Hence, ϕ_{spec} defines a *reachability property*, ensuring reaching to a state that satisfies p.

Safety Specifications: Next, consider $\phi_{\text{spec}} = \nu x.(q \wedge \Diamond x)$. In words, this formula is satisfied by the largest set Q of states that both satisfy q and has a transition to a state with the same property. Hence, for any state in Q, there must be a cycle of states, all of which satisfy q.

Safely Reaching a Region: The standard motion planning objective is to avoid obstacles and reach a goal state. Let us label the goal states with p and the obstacles with q. Then, the specification $\phi_{\text{spec}} = \mu x.(\neg q \wedge (p \vee \Diamond x))$ is the smallest set of states for which there exists a trajectory reaching a state that satisfies p (*i.e.*, a goal state) and along the way never goes through a state that satisfies q (*i.e.*, an obstacle state).

Reaching a Safe Region: Another example is to eventually reach a point where a property p can be retained forever. Essentially, this can be done easily by merging the first two examples together as in $\phi_{\text{spec}} = \mu x.((\nu y.(p \wedge \Diamond y)) \vee \Diamond x)$.

Ordering Specifications: A common specification is, for instance, to ensure some property p until another property q is attained. In this case, a corresponding μ-calculus specification is $\phi_{\text{spec}} = \mu x.(p \vee (q \wedge \Diamond x))$, which intuitively states that either q is satisfied or there is a next state which satisfies p and the property $p \vee (q \wedge \Diamond x)$.

Liveness Specifications: Consider a final example, where it is desired to satisfy a property p infinitely often. That is, at all points along the path, p is satisfied in the future. One way to specify such a behavior is to use $\phi_{\text{spec}} = \nu y.\mu x.\Diamond((p \wedge y) \vee x)$, which intuitively states that in the next states either the property p is satisfied, or there is a path to a state which satisfies p (stated via the disjunction and the $\mu x.$ operators). Moreover, this statement is true at all times (stated via the conjunction and the $\nu y.$ operators).

Chapter 4

Algorithms

This section is devoted to a detailed presentation of the sampling-based path planning algorithms that this dissertation is concerned with. First, the PRM and the RRT algorithms are outlined, as they are representatives of the major paradigms for sampling-based path planning algorithms in the literature. Subsequently, a number of novel sampling-based algorithms, namely the PRM*, RRT*, and the RRG algorithms, are introduced as extensions of PRM and RRT to address the optimal path planning problem as well as path planning problems with deterministic μ-calculus specifications. To address the latter class of problems, an incremental model-checking algorithm, used in conjunction with the RRG, is also provided. Finally, the chapter is concluded with a simulation study involving an illustrative example.

4.1 Primitive Procedures

Before introducing the PRM and the RRT algorithms, let us introduce the primitive procedures that these algorithms rely on.

Sampling: The `Sample` procedures returns a sequence of independent and identically distributed (i.i.d.) samples from the configuration space $\mathcal{X}$. For notational convenience, the ith sample in this sequence is denoted by `Sample`(i). For simplicity, we assume throughout that the samples are drawn from a uniform distribution, although our results naturally extend to any absolutely continuous sampling distribution with density bounded away from zero on $\mathcal{X}$. For convenience, we define also the `SampleFree` procedure, which returns i.i.d. samples from the free space $\mathcal{X}_{\text{free}}$.

Nearest and Near Neighbors: Given a finite set $V \subset \mathcal{X}$ of configurations and a configuration $x \in \mathcal{X}$, the `Nearest` procedure returns the vertex in V that is 'closest' to x in terms of a given distance function. Throughout this thesis, we use the Euclidean distance for simplicity. We refer the interested reader to (LaValle and Kuffner, 2001) for alternative choices. Hence,

$$\texttt{Nearest}(V, x) := \arg\min_{v \in V} |x - v|,$$

where $|\cdot|$ is the usual Euclidean norm in the d-dimensional Euclidean space.

We also define two set-valued versions of this procedure. The $\texttt{kNearest}(V, x, k)$ procedure returns the k vertices that are closest to x in V in terms of the same distance procedure; by convention the procedure returns V if the cardinality of V is smaller than k. For the sake of mathematical rigor, this procedure can be defined as follows: $\texttt{kNearest}(V, x) := \arg\min_{V' \subseteq V, \text{card}(V') = \bar{k}} \sum_{v \in V'} |x - v|$, where $\bar{k} = \min\{k, \text{card}\,(V)\}$. The $\texttt{Near}(V, x, r)$ procedure returns the set of all configurations in V that are within a distance r from x in terms of the same distance function, *i.e.*,

$$\texttt{Near}(V, x, r) := \{v \in V : |x - v| \leq r\}.$$

Steering: Given two configurations $x, x' \in \mathcal{X}$, the `Steer` procedure returns a configuration x'' such that x'' is closer to x' than is x. Unless stated otherwise, we define the `Steer` procedure such that

$$\texttt{Steer}(x, x') := \operatorname*{arg\,min}_{x'' \in \mathcal{X},\ |x'' - x| \leq \eta} |x'' - x'|,$$

where $\eta > 0$ is a pre-specified parameter.

Collision Queries: Given two points $x, x' \in \mathcal{X}$, the $\texttt{CollisionFree}(x, x')$ procedure returns `True` if the line segment between x and x' lies entirely in $\mathcal{X}_{\text{free}}$; it returns `False` otherwise.

4.2 Existing Algorithms

In this section, we recall from the literature two of the important paradigms in sampling-based motion planing, namely the PRM and the RRT algorithms. The inputs and the outputs of the algorithms are as follows. These algorithms take as input a feasible path planning problem instance $(\mathcal{X}_{\text{free}}, x_{\text{init}}, \mathcal{X}_{\text{goal}})$ and an integer $n \in \mathbb{N}$. These inputs are shared with the procedures called within the algorithms. That is, all primitive procedures have access to these inputs. All algorithms return a graph $G = (V, E)$, where $V \subset \mathcal{X}_{\text{free}}$ and $E \subseteq V \times V$. The solution to the feasible path planning problem can be easily deduced from such a graph, for example, using standard graph search algorithms (Schrijver, 2003). Similarly, given a cost function c, the solution to the optimal path planning problem can be deduced from the same graph via standard shortest-path algorithms (Schrijver, 2003).

4.2.1 Probabilistic Roadmaps

The PRM algorithm is primarily aimed at multi-query applications. In its basic form, it consists of a pre-processing phase, in which a roadmap is constructed by attempting connections among n configurations randomly sampled from the free space, and a query phase, in which paths connecting the initial and the final configurations through the roadmap are sought. The PRM algorithm has been a focus of robotics research

for many years. A number of extensions and variations have been proposed. In this section, we describe a number of important variations of the algorithm. In particular, we describe the algorithm proposed by Kavraki et al. (1996) and the algorithm analysed by Kavraki et al. (1998). Although a number of "expansion" heuristics are available in the literature (see Kavraki et al., 1996), they do not have any impact on the analysis. Thus, these heuristics will not be discussed here.

The pre-processing phase, given in Algorithm 1, begins with an empty graph. At each iteration, a configuration $x_{\text{rand}} \in \mathcal{X}_{\text{free}}$ is sampled, and added to the vertex set V. Then, connections are attempted between x_{rand} and other vertices in V within a ball of radius r centered at x_{rand}, in the order of increasing distance from x_{rand}, using a simple local planner, for example a straight-line connection. Successful, *i.e.*, collision-free, connections result in the addition of a new edge to the edge set E. Since the focus of the algorithm is establishing connectivity (to solve the feasible path planning problem), connections between x_{rand} and vertices in the same connected component are avoided. Hence, the roadmap constructed by PRM is a forest, *i.e.*, a collection of trees.

In the literature, mathematically rigorous analysis of, for example, probabilistic completeness, is only available for a "simplified" version of the PRM algorithm (see Kavraki et al., 1998), which we call the sPRM algorithm in this text. See Algorithm 2. The simplified PRM algorithm initializes the vertex set with the initial configuration, samples n configurations from the free space, and attempts to connect the sampled configuration to all configurations within a distance r. All sampled configurations are added to the vertex set, and all successful connections result in the addition of a new edge to the edge set. The connection logic for the sPRM algorithm is similar to that of the PRM algorithm, with the difference that connections between vertices in the same connected component are allowed. Note that in the absence of obstacles, *i.e.*, when $\mathcal{X}_{\text{free}} = \mathcal{X}$, the roadmap constructed in this way is a random r-disc graph (see Section 2.2).

In many practical implementations of (s)PRM algorithms, alternative choices have been considered for the computation of the set U of vertices, with which the connections are attempted (see Line 4 in Algorithm 1 and Line 3 in Algorithm 2). The following criteria are of particular interest for our purposes:

- **k-nearest (s)PRM:** Choose the nearest k neighbors to the sample, for some prespecified k, a typical value for which was reported as $k = 15$ (see LaValle, 2006). That is, $U \leftarrow \texttt{kNearest}(G = (V, E), x_{\text{rand}}, k)$ in Line 4 of Algorithm 1 and $U \leftarrow \texttt{kNearest}(G = (V, E), v, k) \setminus \{v\}$ in Line 3 of Algorithm 2. Notice that the graph constructed in this way in an obstacle-free environment is a random k-nearest graph.
- **Bounded-degree (s)PRM:** For any fixed r, the average number of connections attempted for each sample is proportional to the number of vertices in V, and can result in excessive computational burden for large n. To address this issue, an upper bound k can be imposed on the cardinality of the set U. A typical value for this upper bound was reported as $k = 20$ (see LaValle, 2006). More precisely, $U \leftarrow \texttt{Near}(G, x_{\text{rand}}, r) \cap \texttt{kNearest}(G, x_{\text{rand}}, k)$ in Line 4 of Algorithm 1

and $U \leftarrow \big(\texttt{Near}(G, v, r) \cap \texttt{kNearest}(G, v, k)\big) \setminus \{v\}$ Line 3 of Algorithm 2.

- **Variable-radius (s)PRM:** Another option to maintain the degree of the vertices in the roadmap computationally manageable is to scale the connection radius r appropriately as a function of the number of samples n. However, there are no clear indications in the literature on the appropriate functional relationship between r and n.

Algorithm 1: PRM (pre-processing phase)

```
1  V ← ∅; E ← ∅;
2  for i= 1,2, ..., n do
3      x_rand ← SampleFree(i);
4      U ← Near(G = (V, E), x_rand, r);
5      V ← V ∪ {x_rand};
6      for all u ∈ U in increasing order of ||u − x_rand|| do
7          if x_rand and u are in different connected components of G = (V, E) then
8              if CollisionFree(x_rand, u) then
9                  E ← E ∪ {(x_rand, u), (u, x_rand)};
10     return G = (V, E);
```

Algorithm 2: sPRM

```
1  V ← {x_init} ∪ {SampleFree(i) : i = 1, 2, ..., n}; E ← ∅;
2  for all v ∈ V do
3      U ← Near(G = (V, E), v, r) \ {v};
4      for all u ∈ U do
5          if CollisionFree(v, u) then
6              E ← E ∪ {(v, u), (u, v)};
```

4.2.2 Rapidly-exploring Random Trees

The RRT algorithm is primarily aimed at single-query applications. In its basic form, the algorithm incrementally builds a tree of collision-free trajectories, rooted at the initial configuration. See Algorithm 3. The algorithm is initialized with a graph that includes the initial configuration as its single vertex, and no edges. At each iteration, the algorithm samples a configuration $x_{\text{rand}} \in \mathcal{X}_{\text{free}}$, and attempts to connect the nearest vertex v_{nearest} to a new vertex, denoted by x_{new}, that is close to x_{rand}. If the connection is successful, the algorithm adds x_{new} to the graph along with an edge connecting v_{nearest} and x_{new}.

In the original version of the RRT the iteration is stopped as soon a feasible trajectory is found, *i.e.*, the vertex set contains a configuration that lies in the goal set. However, for the purposes of consistency, in this thesis, the iteration is performed

n times regardless. Notice that in the absence of obstacles, *i.e.*, when $\mathcal{X}_{\text{free}} = \mathcal{X}$, the tree constructed in this manner is an online nearest neighbor graph (see Section 2.2).

Algorithm 3: RRT

```
V ← {x_init}; E ← ∅;
for i = 1, 2, ..., n do
    x_rand ← SampleFree(i);
    v_nearest ← Nearest(G = (V, E), x_rand);
    x_new ← Steer(v_nearest, x_rand);
    if CollisionFree(v_nearest, x_new) then
        V ← V ∪ {x_new};
        E ← E ∪ {(v_nearest, x_new)};
```

4.3 Proposed Algorithms

The PRM and the RRT algorithms were originally designed to solve the feasible path planning problem. In this section, we introduce a number of algorithms that extend PRMs and RRTs to solve optimal path planning problem as well as path planning problems with complex task specifications. First, we introduce the PRM* and RRT* algorithms as computationally-efficient and asymptotically-optimal counterparts of the PRM and the RRT algorithms. Along the way, we also introduce the RRG algorithm which builds a graph of trajectories like the PRM, but in an incremental way like the RRT. The RRG algorithm and a novel incremental model-checking procedure are the two key building blocks for sampling-based algorithms that we propose to address path planning problems with deterministic μ-calculus specifications.

4.3.1 Optimal Probabilistic Roadmap (PRM*)

In the standard PRM algorithm, as well as in the simplified "batch" version which we call sPRM, connections are attempted between roadmap vertices that are within a fixed radius r from one another. The constant r is a parameter to the (s)PRM algorithm. The proposed PRM* algorithm, given in Algorithm 4, is similar to sPRM, except that the connection radius is chosen as a function of the number of samples n as $r(n) := \gamma_{\text{PRM}}(\log(n)/n)^{1/d}$, where $\gamma_{\text{PRM}} > \gamma_{\text{PRM}}{}^* = 2\,(1 + 1/d)^{1/d}(\mu(\mathcal{X}_{\text{free}})/\zeta_d)$, d is the dimensionality of the space $\mathcal{X}$, and $\mu(\cdot)$ and ζ_d denote the usual Lebesgue measure and the volume of the unit ball in the d-dimensional Euclidean space. The connection radius decreases with the number of samples. The rate of decay is chosen such that the expected number of connections attempted for each roadmap vertex is proportional to $\log(n)$.

Let us note at this point that, in the discussion of variable-radius PRM, LaValle (2006) suggests that radius is chosen as a function of the sample dispersion.[1] Indeed,

[1]Recall that the dispersion of a point set contained in a bounded set $S \subset \mathbb{R}^d$ is the radius of the

the dispersion of a set of n random configurations sampled uniformly and independently from the bounded set $\mathcal{X}_{\text{free}}$ is proportional to $(\log(n)/n)^{1/d}$ (see Niederreiter, 1992), which is precisely the rate at which the connection radius is scaled in PRM*.

Algorithm 4: PRM*

```
1 V ← {x_init} ∪ {SampleFree(i)}; E ← ∅;
2 for all v ∈ V do
3     U ← Near(G = (V, E), v, γ_PRM (log(n)/n)^{1/d});
4     for all u ∈ U do
5         if CollisionFree(v, u) then
6             E ← E ∪ {(v, u), (u, v)};
7 return G = (V, E);
```

Another version of the PRM* algorithm is called the k-nearest PRM*, where the connections are sought within k-nearest neighbors like the k-nearest PRM algorithm, with the exception that the number of k is chosen as a function of the number of samples n, rather than a fixed constant. More precisely, we set $k(n) := k_{\text{PRM}} \log(n)$, where $k_{\text{PRM}} > k^*_{\text{PRM}} = e(1 + 1/d)$, and $U \leftarrow \texttt{kNearest}(G = (V, E), v, k_{\text{PRM}} \log(n)) \setminus \{v\}$ in Line 3 of Algorithm 4.

4.3.2 Rapidly-exploring Random Graph (RRG)

The Rapidly-exploring Random Graph (RRG) algorithm is introduced as an *incremental* algorithm to build a *connected* roadmap, possibly containing cycles. See Algorithm 5. The RRG algorithm is similar to the RRT in that it first attempts to connect the nearest node to the new sample. If the connection attempt is successful, the new configuration is added to the vertex set. However, the RRG differs from the RRT as follows. Each time a new configuration x_{new} is added to the vertex set V, connections are attempted from all other vertices in V that are within a ball of radius $r(\text{card}\,(V)) = \min\{\gamma_{\text{RRG}}(\log(\text{card}\,(V))/\,\text{card}\,(V))^{1/d}, \eta\}$, where η is the constant that appears in the definition of the `Steer` procedure and $\gamma_{\text{RRG}} > \gamma^*_{\text{RRG}} = 2(1 + 1/d)^{1/d}(\mu(\mathcal{X}_{\text{free}})/\zeta_d)^{1/d}$. For each successful connection, a new edge is added to the edge set E.

It is clear that, when run with the same sample sequence, the RRT graph, as a directed tree, is a subset of the RRG graph. More precisely, the two graphs share the same vertex set, and the edge set of the RRG graph contains that of the RRT graph.

Another version of the RRG algorithm is called the k-nearest RRG, in which the connections are sought within k-nearest neighbors with $k(\text{card}\,(V)) := k_{\text{RRG}} \log(\text{card}\,(V))$, where $k_{\text{RRG}} > k^*_{\text{RRG}} = e(1+1/d)^{1/d}$, and $U \leftarrow \texttt{kNearest}(G, x_{\text{new}}, \gamma_{\text{RRG}} \log(\text{card}\,(V)))$ in Line 7 of Algorithm 5.

largest ball empty ball centered at some point inside S.

Algorithm 5: RRG

1 $V \leftarrow \{x_{\text{init}}\}$; $E \leftarrow \emptyset$;
2 **for** $i = 1, 2, \ldots, n$ **do**
3 $\quad x_{\text{rand}} \leftarrow \texttt{SampleFree}(i)$;
4 $\quad v_{\text{nearest}} \leftarrow \texttt{Nearest}(G = (V, E), x_{\text{rand}})$;
5 $\quad x_{\text{new}} \leftarrow \texttt{Steer}(v_{\text{nearest}}, x_{\text{rand}})$;
6 $\quad$ **if** $\texttt{CollisionFree}(v_{\text{nearest}}, x_{\text{new}})$ **then**
7 $\quad\quad U \leftarrow \texttt{Near}\big(G = (V, E), x_{\text{new}}, \min\{\gamma_{\text{RRG}}(\log(\text{card}\,(V))/\,\text{card}\,(V))^{1/d}, \eta\}\big)$;
8 $\quad\quad V \leftarrow V \cup \{x_{\text{new}}\}$; $E \leftarrow E \cup \{(v_{\text{nearest}}, x_{\text{new}}), (x_{\text{new}}, v_{\text{nearest}})\}$;
9 $\quad\quad$ **for** *all* $u \in U$ **do**
10 $\quad\quad\quad$ **if** $\texttt{CollisionFree}(u, x_{\text{new}})$ **then**
11 $\quad\quad\quad\quad E \leftarrow E \cup \{(u, x_{\text{new}}), (x_{\text{new}}, u)\}$;
12 **return** $G = (V, E)$;

4.3.3 Optimal Rapidly-exploring Random Tree (RRT*)

Maintaining a tree structure rather than a graph is not only economical in terms of memory requirements, but may also be advantageous in some applications, due to, for example, relatively easy extensions to motion planning problems with differential constraints, or to cope with modeling errors. The RRT* algorithm is obtained by modifying the RRG in such a way that the formation of cycles is avoided, by removing "redundant" edges. Since the RRT and the RRT* graphs are directed trees with the same root and vertex set, and the edge sets that are subsets of that of the RRG, this amounts to a "rewiring" of the RRT tree, ensuring that vertices are reached through paths converging to those with minimal cost.

Before presenting the algorithm in detail, let us introduce the following two data structures. First, the $\texttt{Parent} : V \to V$ function maps a vertex $v \in V$ to the unique vertex $u \in V$ with $(u, v) \in V$. By convention, $\texttt{Parent}(v_0) = v_0$, where v_0 is the root vertex. Second, $\texttt{Cost} : V \to \mathbb{R}_{\geq 0}$ maps each vertex v to the cost of the unique path from v_0 to v. By convention, $\texttt{Cost}(v_0) = 0$. For simplicity, we assume that the cost function c is additive so that $\texttt{Cost}(v) = \texttt{Cost}(\texttt{Parent}(v)) + c(\sigma_v)$, where σ_v is the straight-line path that connects $\texttt{Parent}(v)$ and v. In the algorithm description, we denote the straight path between two configurations $x, x' \in \mathcal{X}$ by $\texttt{Path}(x, x')$.

The RRT* algorithm is presented in Algorithm 6. The algorithm adds new configurations to the vertex set V in the same way as the RRT and the RRG. It also considers connections from the new vertex x_{new} to vertices set U of vertices within a distance $r(\text{card}\,(V)) := \min\{\gamma_{\text{RRT}^*}\,(\log(\text{card}\,(V))/\,\text{card}\,(V))^{1/d}, \eta\}$ to x_{new} (see Line 7 of Algorithm 6). However, not all feasible connections result in new edges being inserted into the edge set E. In particular, *(i)* an edge is created from the vertex in U that can be connected to x_{new} along a path with minimal cost, and *(ii)* new edges are created from x_{new} to vertices in U, if the path through x_{new} has lower cost than the path through the current parent; in the latter case, the edge linking the vertex to its current parent is deleted, to maintain the tree structure.

Algorithm 6: RRT*

1 $V \leftarrow \{x_{\text{init}}\}$; $E \leftarrow \emptyset$;
2 **for** $i = 1, 2, \ldots, n$ **do**
3 $\quad x_{\text{rand}} \leftarrow \texttt{SampleFree}(i)$;
4 $\quad v_{\text{nearest}} \leftarrow \texttt{Nearest}(G = (V, E), x_{\text{rand}})$;
5 $\quad x_{\text{new}} \leftarrow \texttt{Steer}(v_{\text{nearest}}, x_{\text{new}})$;
6 $\quad$ **if** $\texttt{CollisionFree}(v_{\text{nearest}}, x_{\text{new}})$ **then**
7 $\quad\quad U \leftarrow \texttt{Near}(G = (V, E), x_{\text{new}}, \min\{\gamma_{\text{RRT}^*} (\log(\text{card}\,(V))/\,\text{card}\,(V))^{1/d}, \eta\})$;
8 $\quad\quad V \leftarrow V \cup \{x_{\text{new}}\}$;
$\quad\quad$ `// Connect along a minimum-cost path`
9 $\quad\quad v_{\min} \leftarrow v_{\text{nearest}}$; $c_{\min} \leftarrow \texttt{Cost}(v_{\text{nearest}}) + c(\texttt{Path}(v_{\text{nearest}}, x_{\text{new}}))$;
10 $\quad\quad$ **for** *all* $u \in U$ **do**
11 $\quad\quad\quad$ **if** $\texttt{CollisionFree}(u, x_{\text{new}})$ *and* $\texttt{Cost}(u) + c(\texttt{Path}(u, x_{\text{new}})) < c_{\min}$ **then**
12 $\quad\quad\quad\quad v_{\min} \leftarrow u$; $c_{\min} \leftarrow \texttt{Cost}(u) + c(\texttt{Path}(u, x_{\text{new}}))$;
13 $\quad\quad E \leftarrow E \cup \{(v_{\min}, x_{\text{new}})\}$;
$\quad\quad$ `// Rewire vertices`
14 $\quad\quad$ **for** *all* $u \in U$ **do**
15 $\quad\quad\quad$ **if** $\texttt{CollisionFree}(x_{\text{new}}, u)$ *and* $\texttt{Cost}(x_{\text{new}}) + c(\texttt{Path}(x_{\text{new}}, u)) < \texttt{Cost}(u)$ **then**
16 $\quad\quad\quad\quad v_{\text{parent}} \leftarrow \texttt{Parent}(u)$;
17 $\quad\quad\quad\quad E \leftarrow \big(E \setminus \{(v_{\text{parent}}, u)\}\big) \cup \{(x_{\text{new}}, u)\}$;
18 **return** $G = (V, E)$;

Another version of the algorithm is called the k-nearest RRT*, in which connections are sought to $k(\text{card}\,(V)) := k_{\text{RRT}^*} \log(\text{card}\,(V))$ nearest neighbors and we set $U \leftarrow \texttt{kNearest}(G = (V, E), x_{\text{new}}, k_{\text{RRT}^*} \log(\text{card}\,(V)))$ in Line 7 of Algorithm 6.

4.3.4 Incremental Model Checking for the Deterministic μ-calculus

Recall the definition of a Kripke structure and that of a μ-calculus formula from Section 2.3. Roughly speaking, given a Kripke structure $\mathcal{K} = (\mathcal{S}, s_{\text{init}}, \mathcal{R}, \mathcal{L})$ and a μ-calculus formula ϕ, a *global model checking algorithm* (for the μ-calculus) is a procedure that computes the set $[\![\phi]\!]_{\mathcal{K}}$, *i.e.*, the set of all states that satisfies the given formula. A *local model checking algorithm* is a procedure that decides whether or not $s_{\text{init}} \in [\![\phi]\!]_{\mathcal{K}}$ (see, *e.g.*, Clarke et al., 1999; Schneider, 2004).

A *incremental model checking algorithm* is one that maintains suitable data structures to answer the model checking query on the fly, as new states and transitions are being inserted into the Kripke structure (see, *e.g.* Sokolsky and Smolka, 1994). An algorithm that decides the model checking question while states or transitions are being deleted is usually called *decremental*, whereas one that can handle both insertions and deletions is called *dynamic*.

In this section, we provide an incremental local model checking algorithm for the deterministic μ-calculus. In Section 4.3.5, this algorithm is used in conjunction with the RRG to solve path planning problems with deterministic μ-calculus specifications. Clearly, in the context of sampling-based motion planning, a decremental model-checking algorithm is not necessary, since edges and vertices are never deleted once they are inserted into the graph. Note that (non-incremental) local or global model checking algorithms for the deterministic μ-calculus are widely known. The reader is referred to (Gradel et al., 2002; Schneider, 2004) for detailed surveys.

The algorithm we present in this section heavily relies on the preliminary material presented in Section 2.3. In particular, the incremental model-checking algorithm maintains a subset of the arena of the L_1 game for $\mathcal{K}$ and ϕ (see Definition 2.18 in Section 2.3 and following discussion). Loosely speaking, this subset contains the set of all vertices that Player 1 can move the token to without loosing the game. As new states and transitions are added to the $\mathcal{K}$, the corresponding arena is populated in a computationally efficient manner.

Global Variables: Together with the corresponding arena, the algorithm maintains five types of global variables, denoted by `RV`, `Pr`, `Cs`, `Cs`, `Nv`, and `Lv`. The global variable `RV`, defined for all game arena vertices $(s, \psi) \in \mathcal{V}$, is a subset of the the arena vertices, *i.e.*, $\texttt{RV}(s, \psi) \subseteq \mathcal{V}$, such that, for all $(s', \psi') \in \texttt{RV}(s, \psi)$, Player 1 can move the token from (s', ψ') to (s, ψ) without Player 2 being able to push the token to an arena vertex in which Player 1 loses the game by getting stuck. The global variable `Pr`, defined for all pairs of arena vertices $(s, \psi), (s', \psi') \in \mathcal{V}$ with $(s', \psi') \in \texttt{RV}(s, \psi)$, returns an arena vertex (s'', ψ''). The arena vertex (s'', ψ'') is the parent to (s, ψ) in the token path starting from (s', ψ') such that the token passes through (s'', ψ'') right before reaching (s, ψ). This variable is used to recover the (optimal) winning strategy for Player 1 in a computationally efficient manner. The global variable `Cs`, similarly defined for all pairs of arena vertices $(s, \psi), (s', \psi') \in \mathcal{V}$ with $(s', \psi') \in \texttt{RV}(s, \psi)$, maintains the cost for the path of the token to reach (s, ψ) starting from (s', ψ'). The global variable `Nv` is the set of some select arena vertices (s, ψ), where ψ is of the form $\psi = \nu x.\psi'$ for some $\psi' \in L_1$. In the sequel, such arena vertices are called ν-vertices. Finally, the global variable `Lv` is the set of all arena vertices (s, ψ), where ψ is of the form ψ is of the form $\psi = p$ or $\psi = \neg p$ for some atomic proposition $p \in \Pi$. Such arena vertices are called literal vertices.

Interfacing Functions: The interface to the incremental model-checking algorithm is through two procedures, namely the `AddState` and the `AddTransition` procedures. The `AddState` procedure takes as input a new state s along with the pair $(\mathcal{K}, A)$, where $\mathcal{K}$ is some Kripke structure and A is a subset of the game arena for the L_1 game for $\mathcal{K}$ and formula ϕ. The procedure returns the updated Kripke structure which includes the new state s, and the updated game arena. The `AddTransition` procedure takes as input a new transition (s, s') along with a pair $(\mathcal{K}, A)$, and returns the updated Kripke structure (with the new transition) and the updated game arena.

The `AddState` procedure is presented in Algorithm 7. The procedure simply adds

the new state to the set $\mathcal{S}$ of states, while keeping the game arena unchanged. The `AddTransition` procedure, given in Algorithm 8, first adds the new transition into the transition relation, and calls the `UpdateArena` and the `UpdateGlobalDataStructures` procedures. The `UpdateArena` procedure is presented in Algorithm 9. The procedure takes as input the pair $(\mathcal{K}, A)$ along with the two arena vertices $(s_1, \psi_1), (s_2, \psi_2)$. First, it ensures that the latter vertex (s_2, ψ_2) is added to the vertex set of the appropriate player (Lines 13-16). Second, it adds $\big((s_1, \psi_1), (s_2, \psi_2)\big)$ to the edge set of the arena (Line 17). Finally, it recursively calls itself to expand new vertices and edges that are connected to the vertex (s_2, ψ_2) (Lines 18-32).

To maintain computational efficiency, the algorithm does not expand the arena to include vertices starting from which Player 1 is doomed to lose the game by getting stuck (Lines 2-8). It also updates the set of ν-vertices, whenever `AcceptVertex`$(A, (s_2, \psi_2))$ returns true (Lines 9-10).[2] The `UpdateArena` procedure also updates the set of all literal vertices when it encounters a literal vertex (Lines 11-12). The procedure, then, adds the new arena vertex to the appropriate player's vertex set (Lines 13-16), and updates the edge relation of the game arena (Line 17). Finally, the algorithm recursively calls itself to ensure that the game arena is fully populated (Lines 18-32).

The `UpdateGlobalDataStructures` procedure is given in Algorithm 10. Recall that the function χ is the coloring function in the L_1 game for the Kripke structure $\mathcal{K}$ and the deterministic μ-calculus formula ϕ (see Definition 2.18). This procedure first transfers all vertices in $\texttt{RV}((s_1, \psi_1))$ with color number less than that of (s_2, ψ_2) to the set $\texttt{RV}(s_2, \psi_2)$ (Lines 2-7). Second, if (s_1, ψ_1) is a ν-vertex, then (s_1, ψ_1) is placed into the set $\texttt{RV}(s_2, \psi_2)$, if it satisfies the same color condition (Lines 8-13). Along the way, the parent vertices and the costs are updated. If any additions to $\texttt{RV}(s_2, \psi_2)$, then updates are propagated (Lines 14-16). With a slight abuse of notation, the cost of the straight path between configurations s_1 and s_2 is denoted by $\texttt{Cost}(s_1, s_2)$; by convention, $\texttt{Cost}(s_1, s_2) = 0$ when $s_1 = s_2$.

Algorithm 7: `AddState`$\big(\mathcal{K} = (\mathcal{S}, s_{\text{init}}, \mathcal{R}, \mathcal{L}), A = (\mathcal{V}_1, \mathcal{V}_2, \mathcal{E}), s\big)$

```
1 S ← S ∪ {s};
2 return (K = (S, s_init, R, L), A = (V_1, V_2, E));
```

Algorithm 8: `AddTransition`$\big(\mathcal{K} = (\mathcal{S}, s_{\text{init}}, \mathcal{R}, \mathcal{L}), A = (\mathcal{V}_1, \mathcal{V}_2, \mathcal{E}), (s, s')\big)$

```
1 R ← R ∪ {(s, s')};
2 for all ◇ψ ≤ φ do
3 |   A ← UpdateArena(K, A, (s, ◇ψ), (s', ψ));
4 |   UpdateGlobalDataStructures(K, A, (s, ◇ψ), (s', ψ));
5 return (K = (S, s_init, R, L), A = (V_1, V_2, E));
```

[2]Different implementations of the `AcceptVertex` procedure leads to different algorithmic properties. In the next section, we discuss a particular implementation that leads to computationally-efficient sampling-based algorithm.

Algorithm 9: `UpdateArena`$\big(\mathcal{K} = (\mathcal{S}, s_{\text{init}}, \mathcal{R}, \mathcal{L}), A = (\mathcal{V}_1, \mathcal{V}_2, \mathcal{E}), (s_1, \psi_1), (s_2, \psi_2)\big)$

if $\big((s_1, \psi_1), (s_2, \psi_2)\big) \notin \mathcal{E}$ **then**

  `// Remove arena vertex if there is no winning strategy for Player 1`

  if $(\psi_2 = p \text{ and } p \notin \mathcal{L}(s_2))$ *or* $(\psi_2 = \neg p \text{ and } p \in \mathcal{L}(s_2))$ *or*
  $(\psi_2 = \psi' \wedge p \text{ and } p \notin \mathcal{L}(s_2))$ *or* $(\psi_2 = \psi' \wedge p \text{ and } p \in \mathcal{L}(s_2))$ **then**

    $(s, \psi) \leftarrow (s_2, \psi_2)$;

    while $\psi \neq \psi' \vee \psi''$ *and* $\psi \neq \Diamond\psi'$ **do**

      $\mathcal{V}_1 \leftarrow \mathcal{V}_1 \setminus \{(s, \psi)\}$; $\mathcal{V}_2 \leftarrow \mathcal{V}_2 \setminus \{(s, \psi)\}$;

      $\texttt{Lv} \leftarrow \texttt{Lv} \setminus \{(s, \psi)\}$; $\texttt{Nv} \leftarrow \texttt{Nv} \setminus \{(s, \psi)\}$;

      $(s, \psi) \leftarrow \texttt{Pr}\big((s, \psi)\big)$;

    return $A = (\mathcal{V}_1, \mathcal{V}_2, \mathcal{E})$;

  `// Update the set of` ν`-vertices`

  if $(s_1, \psi_1) \notin \texttt{Nv}$ *and* $\psi_1 = \nu x.\psi'$ *and* `AcceptVertex`$\big(A, (s_1, \psi_1)\big)$ **then**

    $\texttt{Nv} \leftarrow \texttt{Nv} \cup \{(s_1, \psi_1\}$;

  `// Update the set of literal vertices`

  if $(s_1, \psi_1) \notin \texttt{Lv}$ *and* $(\psi_1 = p$ *or* $\psi_2 = \neg p)$ **then**

    $\texttt{Lv} \leftarrow \texttt{Lv} \cup \{(s_1, \psi_1)\}$;

  `// Update the vertex set`

  if $(\psi_2 = \psi' \wedge p \text{ and } p \in \mathcal{L}(s))$ *or* $(\psi_2 = \psi' \wedge \neg p \text{ and } p \notin \mathcal{L}(s))$ **then**

    $\mathcal{V}_2 \leftarrow \mathcal{V}_2 \cup \{(s_2, \psi_2)\}$;

  else

    $\mathcal{V}_1 \leftarrow \mathcal{V}_1 \cup \{(s_2, \psi_2)\}$;

  `// Update the edge relation`

  $\mathcal{E} \leftarrow \mathcal{E} \cup \big\{\big((s_1, \psi_1), (s_2, \psi_2)\big)\big\}$;

  `// Recursively add new vertices/edges to the arena`

  case $\psi_2 = \psi' \wedge p \text{ and } p \in \mathcal{L}(s_2)$

    $A \leftarrow \texttt{UpdateArena}(A, (s_2, \psi_2), (s_2, \psi'))$;

  case $\psi_2 = \psi' \wedge \neg p \text{ and } p \notin \mathcal{L}(s_2)$

    $A \leftarrow \texttt{UpdateArena}(A, (s_2, \psi_2), (s_2, \psi'))$

  case $\psi_2 = \psi' \vee \psi''$

    $A \leftarrow \texttt{UpdateArena}(A, (s_2, \psi_2), (s_2, \psi'))$;

    $A \leftarrow \texttt{UpdateArena}(A, (s_2, \psi_2), (s_2, \psi''))$;

  case $\psi_2 = \Diamond\psi'$

    for *all* $s' \in \mathcal{S}$ *with* $(s_2, s') \in \mathcal{R}$ **do**

      $A \leftarrow \texttt{UpdateArena}(A, (s_2, \psi_2), (s', \psi'))$;

  case $\psi_2 = x$

    $\psi' \leftarrow \texttt{BindingFormula}_{\phi}(x)$;

    $A \leftarrow \texttt{UpdateArena}(A, (s_2, \psi_2), (s_2, \psi'))$;

  case $\psi_2 = \mu x.\psi'$ *or* $\psi_2 = \nu x.\psi'$

    $A \leftarrow \texttt{UpdateArena}(A, (s_2, \psi_2), (s_2, \psi'))$;

  return $A = (\mathcal{V}_1, \mathcal{V}_2, \mathcal{E})$;

Algorithm 10: `UpdateGlobalDataStructures`$(\mathcal{K}, A, (s_1, \psi_1), (s_2, \psi_2))$

```
1  UpdatedVertexSet ← False;
2  for all (s', ψ') ∈ RV(s_1, ψ_1) do
3      if (s', ψ') ∉ RV(s_2, ψ_2) and χ((s_2, ψ_2)) ≤ χ((s', ψ')) then
4          RV(s_2, ψ_2) ← RV(s_2, ψ_2) ∪ {(s', ψ')};
5          Pr((s', ψ'), (s_2, ψ_2)) ← (s_1, ψ_1);
6          Cs((s', ψ'), (s_2, ψ_2)) ← Cs((s', ψ'), (s_1, ψ_1)) + Cost(s_1, s_2);
7          UpdatedVertexSet ← True;
8  if (s_1, ψ_1) ∈ Nv then
9      if (s_1, ψ_1) ∉ RV(s_2, ψ_2) and χ((s_2, ψ_2)) ≤ χ((s_1, ψ_1)) then
10         RV(s_2, ψ_2) ← RV(s_2, ψ_2) ∪ {(s_1, ψ_1)};
11         Pr((s_1, ψ_1), (s_2, ψ_2)) ← (s_1, ψ_1);
12         Cs((s_1, ψ_1), (s_2, ψ_2)) ← Cost(s_1, s_2);
13         UpdatedVertexSet ← True;
14 if UpdatedVertexSet = True then
15     for all (s', ψ') ∈ V with ((s_2, ψ_2), (s', φ')) ∈ E do
16         UpdateGlobalDataStructures(V, (s_2, ψ_2), (s', ψ'));
```

Termination Criteria and the Solution Obtained From the Algorithm: Player 1 has a winning strategy in the L_1 game of $\mathcal{K}$ and ϕ, when either one of the following is satisfied: *(i)* there exists a $(s, \psi) \in \texttt{Lv}$, or *(ii)* there exists $(s, \psi) \in \texttt{Nv}$ such that $(s, \psi) \in \texttt{RV}(s, \psi)$. In the former case, it is guaranteed that Player 1 can move the token to vertex (s, ψ), where Player 2 gets stuck and loses the game. In the latter case, Player 1 can move the token to (s, ψ) and then it can cycle the token in a path that later visits (s, ψ). In both cases, Player 1 does not give Player 2 the chance to win the game while moving the token, because all the vertices starting from which are deleted from the arena immediately after they are discovered (see Lines 2-8). This argument establishes the soundness of the incremental model-checking algorithm.[3]

4.3.5 Algorithms for Problems with Complex Task Specifications

In this section, we introduce two novel incremental sampling-based algorithms, called the μ-RRT and μ-RRT*, that solve, respectively, the feasible and the optimal path planning problems introduced in Section 3.3. These algorithms extend the RRT

[3] A more elaborate analysis would carefully investigate Algorithm 9 case by case, for each operator. This analysis would first establish that a vertex is deleted from the arena if and only if Player 2 win starting from that vertex. This can be established by carefully examining Lines 2-8 of Algorithm 9. This implies that the as long as there is a path in the arena to some vertex $(s, \psi) \in \texttt{Lv}$, then Player 1 can move the token to that vertex, without leaving Player 2 a move to win the game during the process. The same argument applies to moving the token to a vertex $(s, \psi) \in \texttt{Nv}$ first and then repeating a cycle that starts and ends at this vertex. Notice that in this cycle the maximum appearing color is the color assigned to (s, ψ).

and the RRT* algorithms to handle complex task specifications given in the form of deterministic μ-calculus, using the incremental model-checking algorithm presented in Section 4.3.4. In particular, the algorithms presented in this section have access to the global data structures maintained by the incremental model-checking algorithm.

The μ-RRT algorithm, presented in Algorithm 11, simply extends RRG algorithm with appropriate calls to the incremental model-checking algorithm. In this manner, while the graph representing a rich set of paths is generated, the model checking algorithm is invoked to decide whether the resulting graph contains a feasible solution. Instead of the `CollisionFree` procedure, a new procedure, called `SingleLabel`, is employed in the μ-RRT algorithm. Let σ denote the straight path connecting the two configurations x and x'. Then, the `SingleLabel`(x, x') procedure checks whether the set of atomic propositions satisfied along σ switches at most once, *i.e.*, set set-valued function $\lambda(\sigma(\tau))$ switches value at most once as τ varies from zero to one.

The μ-RRT* algorithm, given in Algorithm 12, extends the μ-RRT algorithm with an RRT*-like strategy to keep track of optimal paths.

Algorithm 11: μ-RRT

```
   // Initialize the Kripke structure and the game arena
1  S ← {x_init}; s_init ← x_init; R ← ∅; L(s_init) ← λ(x_init);
2  V_1 ← {(s_init, φ_spec)}; V_2 ← ∅; E ← ∅;
3  K ← (V, s_init, R, L); A ← (V_1, V_2, E);

   // Initialize the global data structures
4  Nv ← ∅; Lv ← ∅; RV((x_init, φ_spec)) ← (x_init, φ_spec);
5  Pr((x_init, φ_spec), (x_init, φ_spec)) ← NULL; Cs((x_init, φ_spec), (x_init, φ_spec)) ← 0;

   // Incremental search
6  while i = 1, 2, ..., n do
7      x_rand ← Sample(i);
8      v_nearest ← Nearest(x_rand);
9      x_new ← Steer(v_nearest, x_new);
10     if SingleLabel(v_nearest, x_new) then
           // Update the graph maintained by the algorithm
11         V ← V ∪ {x_new}; E ← E ∪ {(v_nearest, x_new)};
           // Expand the Kripke structure and the game arena
12         (K, A) ← AddState((K, A), x_new);
13         (K, A) ← AddTransition((K, A), v_nearest, x_new);
           // Make connections with near vertices
14         U ← Near(V, x_new, min{γ_RRG(log(card(V))/ card(V))^{1/d}, η});
15         for all u ∈ U do
16             if SingleLabel(x_new, u) then
17                 (K, A) ← AddTransition((K, A), (x_new, u));
18                 (K, A) ← AddTransition((K, A), (u, x_new));
```

Algorithm 12: μ-RRT*

// Initialize the Kripke structure and the game arena
$\mathcal{S} \leftarrow \{x_{\text{init}}\}$; $s_{\text{init}} \leftarrow x_{\text{init}}$; $\mathcal{R} \leftarrow \emptyset$; $\mathcal{L}(s_{\text{init}}) \leftarrow \lambda(x_{\text{init}})$;
$\mathcal{V}_1 \leftarrow \{(s_{\text{init}}, \phi_{\text{spec}})\}$; $\mathcal{V}_2 \leftarrow \emptyset$; $\mathcal{E} \leftarrow \emptyset$;
$\mathcal{K} \leftarrow (V, s_{\text{init}}, \mathcal{R}, \mathcal{L})$; $A \leftarrow (\mathcal{V}_1, \mathcal{V}_2, \mathcal{E})$;

// Initialize the global data structures
Nv $\leftarrow \emptyset$; Lv $\leftarrow \emptyset$; RV$((x_{\text{init}}, \phi_{\text{spec}})) \leftarrow (x_{\text{init}}, \phi_{\text{spec}})$;
Pr$((x_{\text{init}}, \phi_{\text{spec}}), (x_{\text{init}}, \phi_{\text{spec}})) \leftarrow$ NULL; Cs$((x_{\text{init}}, \phi_{\text{spec}}), (x_{\text{init}}, \phi_{\text{spec}})) \leftarrow 0$;

// Incremental search
while $i = 1, 2, \ldots, n$ **do**
  $x_{\text{rand}} \leftarrow$ Sample(i);
  $v_{\text{nearest}} \leftarrow$ Nearest(x_{rand});
  $x_{\text{new}} \leftarrow$ Steer$(v_{\text{nearest}}, x_{\text{new}})$;
  if SingleLabel$(v_{\text{nearest}}, x_{\text{new}})$ **then**
    // Update the graph maintained by the algorithm
    $V \leftarrow V \cup \{x_{\text{new}}\}$; $E \leftarrow E \cup \{(v_{\text{nearest}}, x_{\text{new}})\}$;
    // Expand the Kripke structure and the game arena
    $(\mathcal{K}, A) \leftarrow$ AddState$((\mathcal{K}, A), x_{\text{new}})$;
    $(\mathcal{K}, A) \leftarrow$ AddTransition$((\mathcal{K}, A), v_{\text{nearest}}, x_{\text{new}})$;
    // Make connections with near vertices
    $U \leftarrow$ Near$(V, x_{\text{new}}, \min\{\gamma_{\text{RRG}}(\log(\text{card}\,(V))/\,\text{card}\,(V))^{1/d}, \eta\})$;
    for *all* $u \in U$ **do**
      if SingleLabel(x_{new}, u) **then**
        $(\mathcal{K}, A) \leftarrow$ AddTransition$((\mathcal{K}, A), (x_{\text{new}}, u))$;
        $(\mathcal{K}, A) \leftarrow$ AddTransition$((\mathcal{K}, A), (u, x_{\text{new}}))$;
    // Find the best parent
    for *all* $(x_{\text{new}}, \psi) \in \mathcal{V}$ *with* $\Diamond\psi \leq \phi_{\text{spec}}$ **do**
      for *all* $(s', \psi') \in$ RV$((x_{\text{new}}, \psi))$ **do**
        for *all* $(s, \Diamond\psi) \in \mathcal{V}$ *with* $((s, \Diamond\psi), (x_{\text{new}}, \psi)) \in \mathcal{E}$ **do**
          if Cs$((s, \Diamond\psi))$ + Cost$(x_{\text{new}}, s) <$ Cs$((x_{\text{new}}, \psi))$ **then**
            Pr$((s', \psi'), (x_{\text{new}}, \psi)) \leftarrow (s, \Diamond\psi)$;
            Cost$((s', \psi'), (x_{\text{new}}, \psi)) \leftarrow$ Cs$((s, \Diamond\psi))$ + Cost(x_{new}, s);
    // Rewire vertices
    for *all* $(x_{\text{new}}, \Diamond\psi) \in \mathcal{V}$ **do**
      for *all* $(s', \psi') \in$ RV$((x_{\text{new}}, \Diamond\psi))$ **do**
        for *all* $(s, \psi) \in \mathcal{V}$ *with* $((x_{\text{new}}, \Diamond\psi), (s, \psi)) \in \mathcal{E}$ **do**
          if Cs$((x_{\text{new}}, \Diamond\psi))$ + Cost$(x_{\text{new}}, s) <$ Cs(s, ψ) **then**
            Pr$((s', \psi'), (s, \psi)) \leftarrow (x_{\text{new}}, \Diamond\psi)$;
            Cs$((s', \psi'), (s, \psi)) \leftarrow$ Cs$((x_{\text{new}}, \Diamond\psi))$ + Cost(x_{new}, s);

4.4 Computational Experiments

This section is devoted to computational experiments. The computational experiments presented in this section have previously appeared in (Karaman and Frazzoli, 2011b), and focus on demonstrating the performance of the RRT* algorithm in an illustrative example. More simulation studies demonstrating the performance of the proposed algorithms in challenging practical examples can be found in (Karaman and Frazzoli, 2010a; Karaman et al., 2011; Perez et al., 2011; Jeon et al., 2011).

The RRT and the RRT* algorithms were implemented using the C language and executed on a computer with 2.66 GHz processor and 4GB RAM running the Linux operating system. Unless stated otherwise, the cost of a path is its total variation.

The first scenario includes no obstacles. Both algorithms are run in a square environment. The trees maintained by the algorithms are shown in Figure 4-1 at several stages. The figure illustrates that, in this case, the RRT algorithm does not improve the solution towards an optimum solution. On the other hand, running the RRT* algorithm further improves the paths in the tree to lower cost ones. The convergence properties of the two algorithms are also investigated in Monte-Carlo runs. Both algorithms were run for 20,000 iterations 500 times and the cost of the best path in the trees were averaged for each iteration. The results are shown in Figure 4-2, which shows that in the limit the RRT algorithm has cost very close to a $\sqrt{2}$ factor the optimal solution, whereas the RRT* converges to the optimal solution. Moreover, the variance over different RRT runs approaches 2.5, while that of the RRT* approaches zero. Hence, almost all RRT* runs have the property of convergence to an optimal solution, as expected.

In the second scenario, both algorithms are run in an environment in presence of obstacles. In Figure 4-3, the trees maintained by the algorithms are shown after 20,000 iterations. The tree maintained by the RRT* algorithm is also shown in Figure 4-4 in different stages. It can be observed that the RRT* first rapidly explores the state space just like the RRT. Moreover, as the number of samples increase, the RRT* improves its tree to include paths with smaller cost and eventually discovers a path in a different homotopy class, which reduces the cost of reaching the target considerably. Results of a Monte-Carlo study for this scenario is presented in Figure 4-5. Both algorithms were run alongside up until 20,000 iterations 500 times and cost of the best path in the trees were averaged for each iteration. The figures illustrate that all runs of the RRT* algorithm converges to the optimum, whereas the RRT algorithm is about 1.5 of the optimal solution on average. The high variance in solutions returned by the RRT algorithm stems from the fact that there are two different homotopy classes of paths that reach the goal. If the RRT luckily converges to a path of the homotopy class that contains an optimum solution, then the resulting path is relatively closer to the optimum than it is on average. If, on the other hand, the RRT first explores a path of the second homotopy class, which is often the case for this particular scenario, then the solution that RRT converges to is generally around twice the optimum.

Finally, in the third scenario, where no obstacles are present, the cost function is selected to be the line integral of a function, which evaluates to 2 in the high cost region, 1/2 in the low cost region, and 1 everywhere else. The tree maintained by the

RRT* algorithm is shown after 20,000 iterations in Figure 4-6. Notice that the tree either avoids the high cost region or crosses it quickly, and vice-versa for the low-cost region.

To compare the running time, both algorithms were run alongside in an environment with no obstacles for up to one million iterations. Figure 4-7, shows the ratio of the running time of RRT* and that of RRT versus the number of iterations averaged over 50 runs. As expected from the computational complexity analysis, this ratio converges to a constant value. A similar figure is produced for the second scenario and provided in Figure 4-8.

The RRT* algorithm was also run in a 5-dimensional state space. The number of iterations versus the cost of the best path averaged over 100 trials is shown in Figure 4-9. A comparison with the RRT algorithm is provided in the same figure. The ratio of the running times of the RRT* and the RRT algorithms is provided in Figure 4-10. The same experiment is carried out for a 10-dimensional configuration space. The results are shown in Figure 4-11.

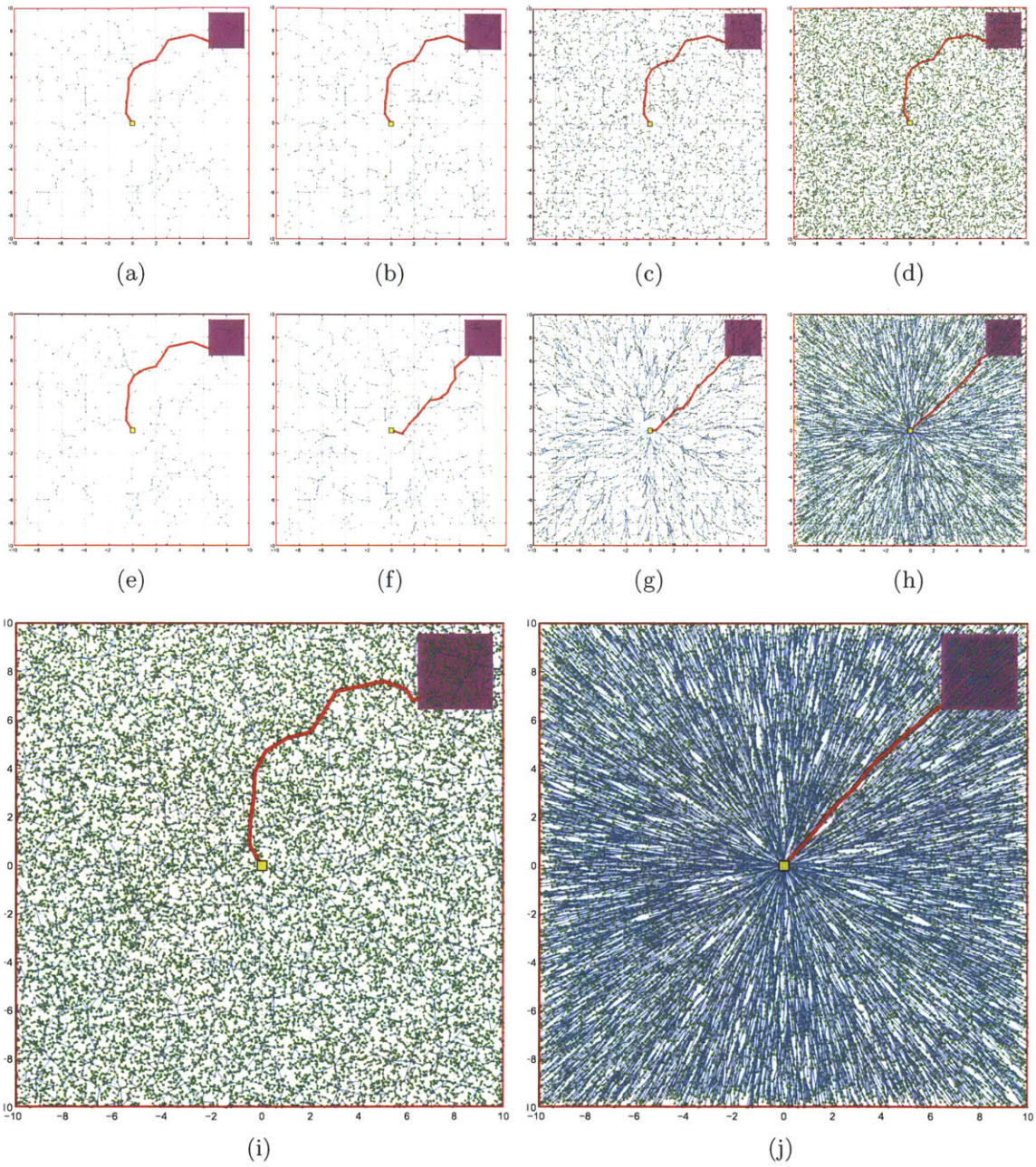
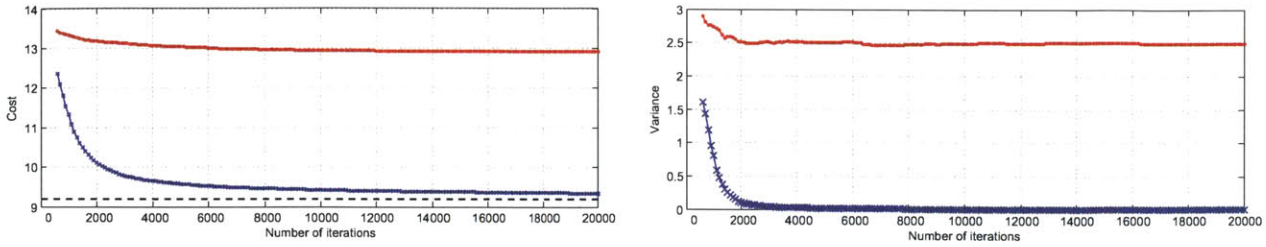
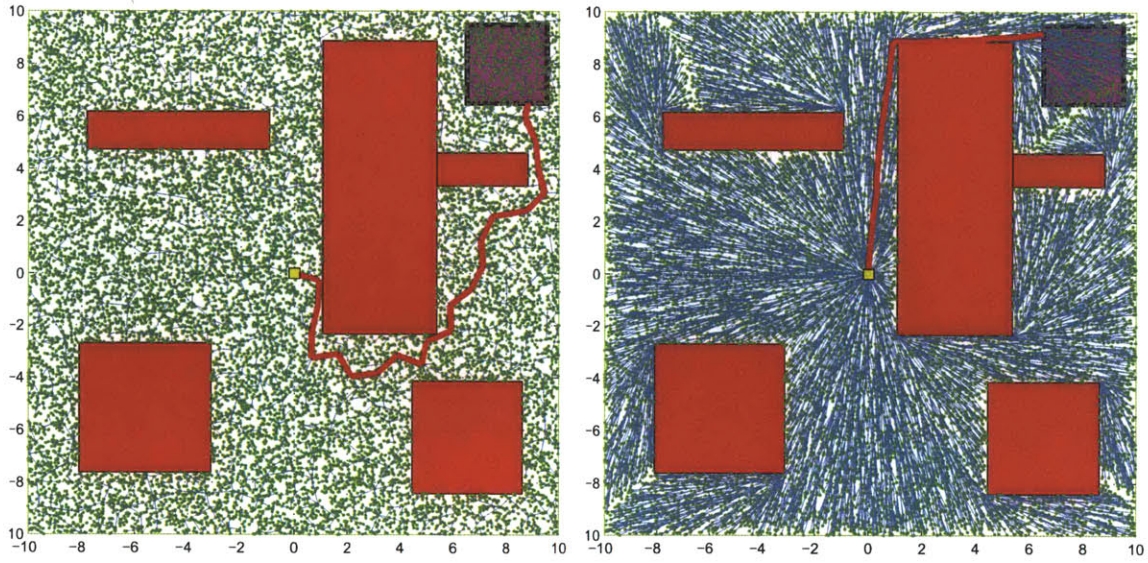


Figure 4-1: A Comparison of the RRT* and RRT algorithms on a simulation example with no obstacles. Both algorithms were run with the same sample sequence. Consequently, in this case, the vertices of the trees at a given iteration number are the same for both of the algorithms; only the edges differ. The edges formed by the RRT algorithm are shown in (a)-(d) and (i), whereas those formed by the RRT* algorithm are shown in (e)-(h) and (j). The tree snapshots (a), (e) contain 250 vertices, (b), (f) 500 vertices, (c), (g) 2500 vertices, (d), (h) 10,000 vertices and (i), (j) 20,000 vertices. The goal regions are shown in magenta (in upper right). The best paths that reach the target in all the trees are highlighted with red.



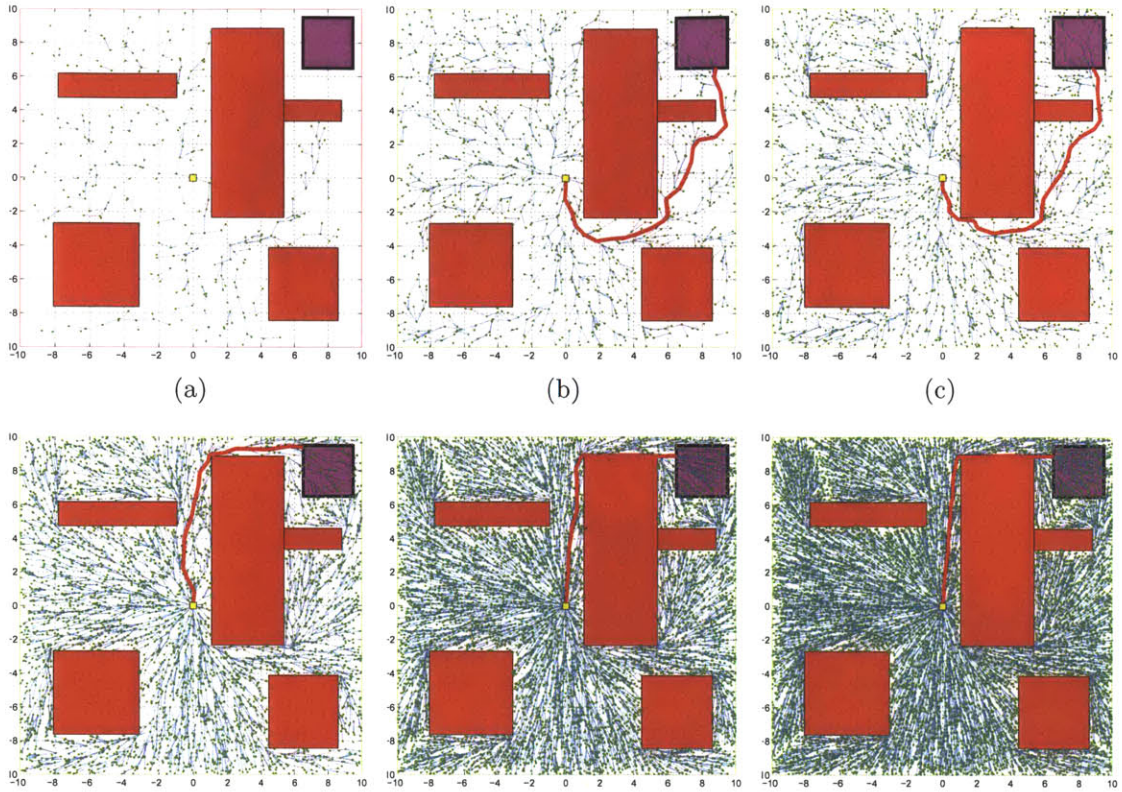
(a) (b)

Figure 4-2: The cost of the best paths in the RRT (shown in red) and the RRT* (shown in blue) plotted against iterations averaged over 500 trials in (a). The optimal cost is shown in black. The variance of the trials is shown in (b).



(a) (b)

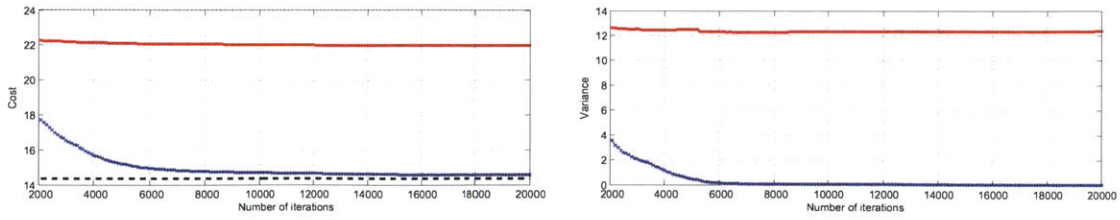
Figure 4-3: A Comparison of the RRT (shown in (a)) and RRT* (shown in (b)) algorithms on a simulation example with obstacles. Both algorithms were run with the same sample sequence for 20,000 samples. The cost of best path in the RRT and the RRG were 21.02 and 14.51, respectively.



(a) (b) (c)

(d) (e) (f)

Figure 4-4: RRT* algorithm shown after 500 (a), 1,500 (b), 2,500 (c), 5,000 (d), 10,000 (e), 15,000 (f) iterations.



(a) (b)

Figure 4-5: An environment cluttered with obstacles is considered. The cost of the best paths in the RRT (shown in red) and the RRT* (shown in blue) plotted against iterations averaged over 500 trials in (a). The optimal cost is shown in black. The variance of the trials is shown in (b).

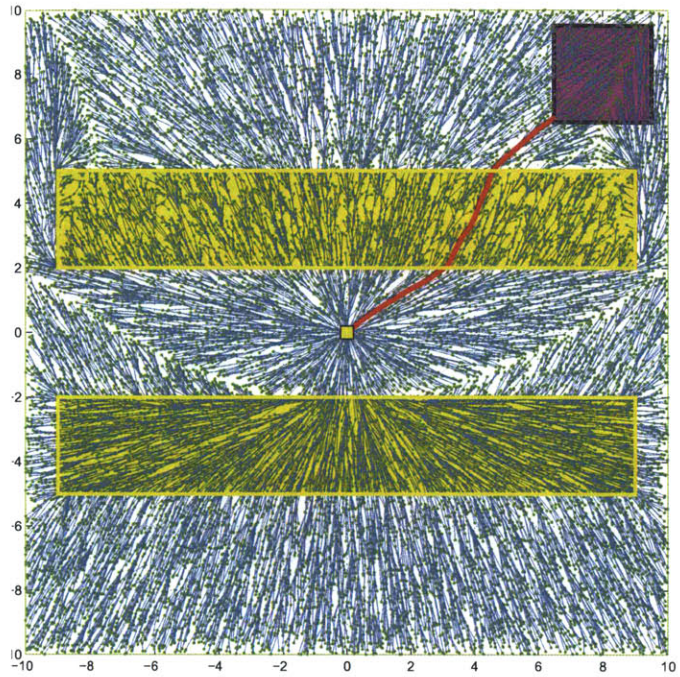


Figure 4-6: RRT* algorithm at the end of iteration 20,000 in an environment with no obstacles. The upper yellow region is the high-cost region, whereas the lower yellow region is low-cost.

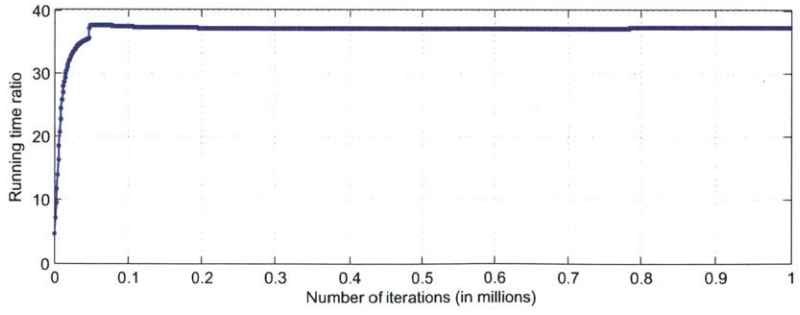


Figure 4-7: A comparison of the running time of the RRT* and the RRT algorithms. The ratio of the running time of the RRT* over that of the RRT up until each iteration is plotted versus the number of iterations.

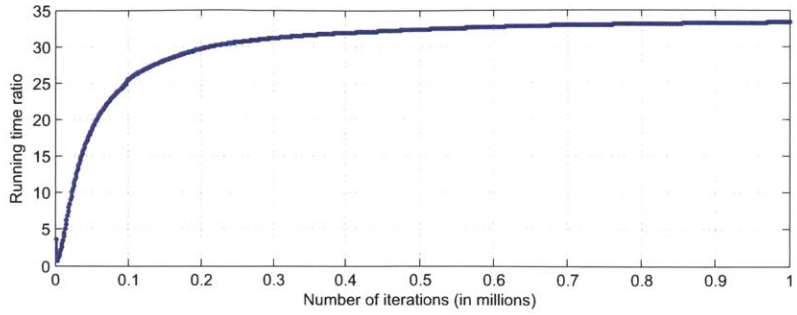
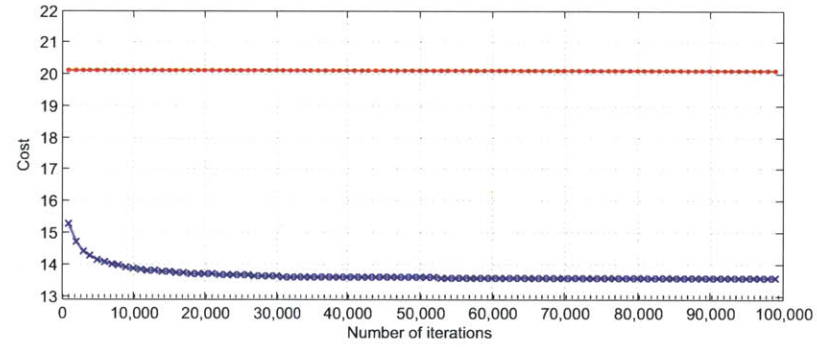
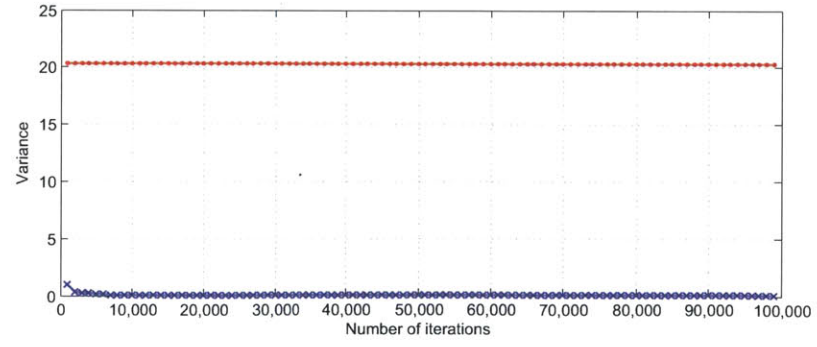


Figure 4-8: A comparison of the running time of the RRT* and the RRT algorithms in an environment with obstacles. The ratio of the running time of the RRT* over that of the RRT up until each iteration is plotted versus the number of iterations.

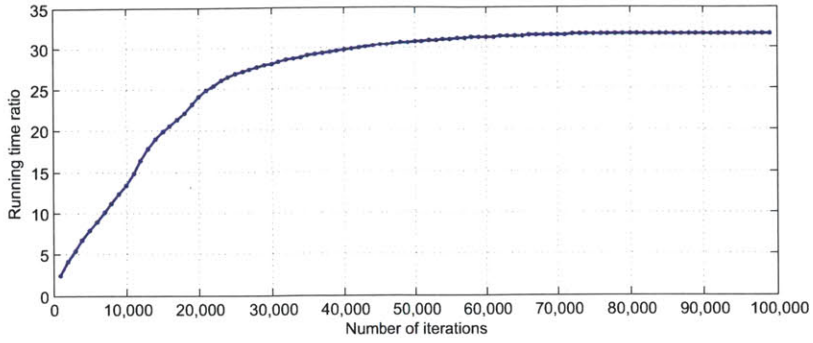


(a)



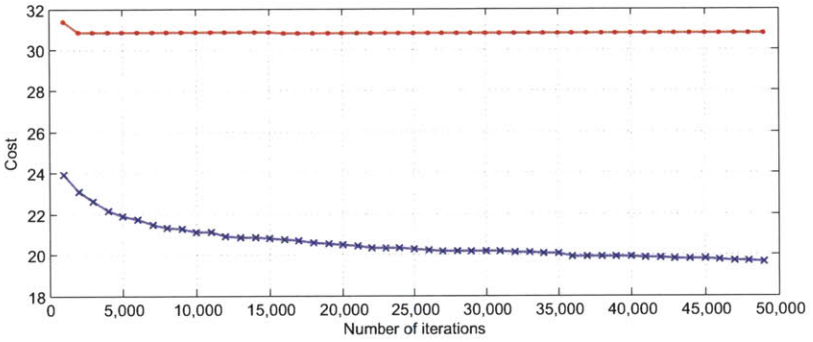
(b)

Figure 4-9: The cost of the best paths in the RRT (shown in red) and the RRT* (shown in blue) run in a 5 dimensional obstacle-free configuration space plotted against iterations averaged over 100 trials in (a). The optimal cost is shown in black. The variance of the trials is shown in (b).

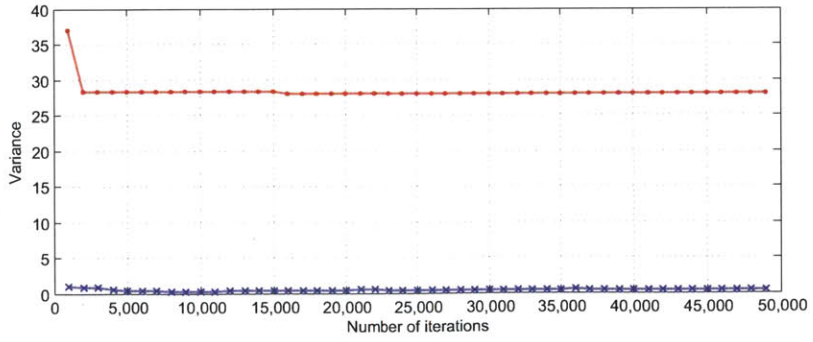


(a)

Figure 4-10: The ratio of the running time of the RRT and the RRT* algorithms is shown versus the number of iterations.



(a)



(b)

Figure 4-11: The cost of the best paths in the RRT (shown in red) and the RRT* (shown in blue) run in a 10 dimensional configuration space involving obstacles plotted against iterations averaged over 25 trials in (a). The variance of the trials is shown in (b).

Chapter 5

Analysis

This chapter is devoted to the theoretical contributions of the thesis. The algorithms introduced in Chapter 4 are analyzed in terms of probabilistic completeness, asymptotic optimality, and computational complexity. First, in Section 5.1, we state all our major results without the proofs. We discuss important implications of ours results in this section. Then, in Sections 5.2, 5.3, and 5.4, we provide detailed proofs for the novel results stated in Section 5.1.

5.1 Statement of Results

5.1.1 Probabilistic Completeness

In this section, we list a number of results regarding the probabilistic completeness of the algorithms introduced in Chapter 4. Some of the results that we present in this section can be found in the literature. For those, we point out the suitable references. The proofs of novel results are given in Section 5.2.

A rigorous definition of probabilistic completeness: Consider a feasible path planning problem instance $(\mathcal{X}_{\text{free}}, x_{\text{init}}, \mathcal{X}_{\text{goal}})$, where $\mathcal{X}_{\text{free}}$ is the free space, x_{init} is the initial configuration, and $\mathcal{X}_{\text{goal}}$ is the goal set. Let $\delta > 0$ be a real number. A configuration $x \in \mathcal{X}_{\text{free}}$ is said to be a *δ-interior configuration*, if the closed Euclidean ball of radius δ lies entirely inside $\mathcal{X}_{\text{free}}$. The δ-interior of $\mathcal{X}_{\text{free}}$ is defined as the collection of all δ-interior configurations, *i.e.*, $\text{int}_\delta(\mathcal{X}_{\text{free}}) := \{x \in \mathcal{X}_{\text{free}} : \mathcal{B}_\delta(x) \subset \mathcal{X}_{\text{free}}\}$, where $\mathcal{B}_\delta(x)$ is the Euclidean ball of radius δ centered at x. See Figure 5-1. A collision-free path $\sigma : [0,1] \to \mathcal{X}_{\text{free}}$ is said to have *strong δ-clearance*, if σ lies entirely inside the δ-interior of the free space, *i.e.*, $\sigma(\tau) \in \text{int}_\delta(\mathcal{X}_{\text{free}})$ for all $\tau \in [0,1]$. A feasible path planning problem instance is said to be *robustly feasible instance*, if there exists a path σ with strong δ-clearance, for some $\delta > 0$, such that σ is feasible for the problem instance at hand. In the sequel, such a path σ is called a *robustly feasible path.*

Finally, we define probabilistic completeness as follows. Let $G = (V, E)$ be a graph, where $V \subset \mathbb{R}^d$ and $E \subseteq V \times V$. The set of all paths through $G = (V, E)$ contains exactly those paths that visit a finite number of vertices from V in an order that respects the edge relation E, where consecutive vertices are connected via

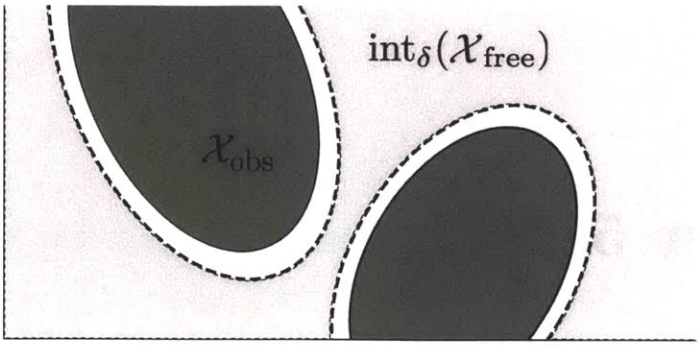


Figure 5-1: An illustration of the δ-interior of $\mathcal{X}_{\text{free}}$. The obstacle set $\mathcal{X}_{\text{obs}}$ is shown in dark gray and the δ-interior of $\mathcal{X}_{\text{free}}$ is shown in light gray. The distance between the dashed boundary of $\text{int}_\delta(\mathcal{X}_{\text{free}})$ and the solid boundary of $\mathcal{X}_{\text{free}}$ is precisely δ.

straight paths. Strictly speaking, a path $\sigma : [0, 1] \to \mathbb{R}^d$ is said to be a path through $G = (V, E)$, if there exists a sequence $(v_0, v_1, \ldots, v_k)$ of vertices and a sequence $(\tau_0, \tau_1, \ldots, \tau_k)$ of real numbers, such that *(i)* $(v_{i-1}, v_i) \in E$, *(ii)* $\tau_0 = 0$, $\tau_k = 1$, and $\tau_{i-1} < \tau_i$ for all i, *(iii)* $\sigma(\tau_i) = v_i$ for all i, and *(iv)* $\sigma(\tau) = (\tau - \tau_{i-1})/(\tau_i - \tau_{i-1})v_i + (\tau_i - \tau)/(\tau_i - \tau_{i-1})v_{i-1}$ for all $\tau \in (\tau_i, \tau_{i+1})$ and all i.

Definition 5.1 (Probabilistic Completeness) *An algorithm ALG is said to be* probabilistically complete, *if, for any robustly feasible path planning problem instance,*[1]

$$\liminf_{n\to\infty} \mathbb{P}(\{\textit{There exists a feasible path } \sigma \textit{ through } G_n^{ALG} = (V_n^{ALG}, E_n^{ALG})\}) = 1.$$

If an algorithm is probabilistically complete, and the problem instance at hand is robustly feasible, then the limit $\lim_{n\to\infty} \mathbb{P}(\{\text{There exists a feasible path } \sigma \text{ through } G_n^{ALG} = (V_n^{ALG}, E_n^{ALG})\})$ does exist and is equal to one. On the other hand, it can be shown that the same limit is equal to zero for any sampling-based algorithm (including probabilistically complete ones) if the problem instance is not robustly feasible, unless the samples are drawn from a singular distribution adapted to the problem.[2]

Note that Definition 5.1, in its current form, only applies to instances of Problem 3.1. To define probabilistic completeness for Problem 3.3, we first extend the defi-

[1]Note that, when $\mathcal{X}_{\text{free}}$ is an open set, any feasible path is robustly feasible. Hence, in that case, any feasible problem instance is also robustly feasible. Yet, the notion of robust feasibility is important for our purposes for the following reasons. However, when defining asymptotic optimality, we will require $\mathcal{X}_{\text{free}}$ to be closed in order to ensure the existence optimal solutions. Moreover, the definition of robustly feasible paths will allow us to easily define robustly optimal paths, which will in turn be used to define asymptotic optimality.

[2]More precisely, in this case, we define a *sampling-based algorithm* as one that has access to the problem instance, $(\mathcal{X}_{\text{free}}, x_{\text{init}}, \mathcal{X}_{\text{obs}})$, only through the primitive procedures provided in Section 4.1. (All relevant algorithms presented in this thesis satisfy this property.) Then, it can be shown that, for any sampling-based algorithm ALG, the probability that there exists a feasible path in σ through $G_n^{ALG} = (V_n^{ALG}, E_n^{ALG})$ is equal to zero for all $n \in \mathbb{N}$, in any problem instance that is not robustly feasible, even when ALG is probabilistically complete.

nition of strong δ-clearance as follows. Consider an instance of the feasible path problem with deterministic μ-calculus specifications, denoted by $(x_{\text{init}}, R_1, R_2, \ldots, R_k, \phi_{\text{spec}})$, where x_{init} is the initial configuration, $R_1, R_2, \ldots, R_1 \subset \mathcal{X}$ are open sets, and ϕ_{spec} is a deterministic μ-calculus formula. Let δ be a positive real number. Define the *δ-exterior* of R_i as the set of all configurations that are at most δ apart from R_i, *i.e.*, $R_i^{\delta} := \{x \in \mathcal{X} : \mathcal{B}_\delta(x) \cap R_i \neq \emptyset\}$. The δ-interior of R_i is defined in the usual way, and denoted simply by $R_i^{-\delta}$. A path $\sigma : [0,1] \to \mathcal{X}$ that is feasible for Problem 3.3 is said to be *robustly feasible*, if there exists some $\delta > 0$ such that it is feasible, and moreover its trace is invariant, for all problem instances $(x_{\text{init}}, R_1^{\delta_1'}, R_2^{\delta_2'}, \ldots, R_k^{\delta_k'}, \phi_{\text{spec}})$, where $\delta_1', \delta_2', \ldots, \delta_k' \in [-\delta, \delta]$. A problem instance is said to be robustly feasible, if it admits a robustly feasible path. Then, probabilistic completeness, as defined in Definition 5.1, extends naturally to the instances of Problem 3.3.

Analysis of existing algorithms: It is known that the sPRM and the RRT algorithms are probabilistically complete. Moreover, for these algorithms, the probability of failure to return a feasible solution, when a robustly feasible one exists, converges to zero exponentially fast with increasing number of samples:

Theorem 5.2 (Probabilistic Completeness of sPRM (Kavraki et al., 1998)) *Let $(\mathcal{X}_{\text{free}}, x_{\text{init}}, \mathcal{X}_{\text{goal}})$ be a robustly feasible path planning problem instance. There exists constants $a > 0$ and $n_0 \in \mathbb{N}$, both independent of n, such that, for all $n > n_0$,*

$$\mathbb{P}\big(\{\textit{There exists a feasible path } \sigma \textit{ through } G_n^{\text{sPRM}} = (V_n^{\text{sPRM}}, E_n^{\text{sPRM}})\}\big) > 1 - e^{-an}.$$

Theorem 5.3 (Probabilistic Completeness of RRT (LaValle and Kuffner, 2001)) *Let $(\mathcal{X}_{\text{free}}, x_{\text{init}}, \mathcal{X}_{\text{goal}})$ be a robustly feasible path planning problem instance. There exists constants $a > 0$ and $n_0 \in \mathbb{N}$, both independent of n, such that, for all $n > n_0$,*

$$\mathbb{P}\big(\{\textit{There exists a feasible path } \sigma \textit{ through } G_n^{\text{RRT}} = (V_n^{\text{RRT}}, E_n^{\text{RRT}})\}\big) > 1 - e^{-an}.$$

Probabilistic completeness results do not necessarily extend to the heuristics used in practical implementations of the (s)PRM algorithm. Consider, for example, the k-nearest PRM algorithm with $k = 1$. That is, each vertex is connected to its nearest neighbor and the resulting undirected graph is returned as the output. This PRM variant will be called the 1-nearest PRM, and it will be indicated by the label 1-PRM. The RRT algorithm can be thought of as the incremental version of the 1-nearest PRM algorithm: the RRT algorithm also connects each sample to its nearest neighbor, but forces connectivity of the graph by an incremental construction.

The following theorem shows that the 1-nearest PRM algorithm is not probabilistically complete, although the RRT is (see Theorem 5.3). Furthermore, the probability that the 1-nearest PRM algorithm *fails* to return a feasible path converges to one as the number of samples approaches infinity.

Theorem 5.4 (Incompleteness of 1-nearest PRM) *The k-nearest PRM algorithm is not probabilistically complete for $k = 1$. Furthermore,*

$$\lim_{n\to\infty} \mathbb{P}\big(\{\textit{There exists a feasible path } \sigma \textit{ through } G_n^{\text{1-PRM}} = (V_n^{\text{1-PRM}}, E_n^{\text{1-PRM}})\}\big) = 0.$$

The proof of this theorem is provided in Section 5.2.1.

Similarly, the variable-radius PRM algorithm lacks probabilistic completeness if the radius is shrunk too fast.

Theorem 5.5 (Incompleteness of variable-radius PRM for $r_n \leq \gamma n^{-1/d}$) *There exists a constant γ, independent of n, such that the variable radius PRM algorithm with connection radius $\gamma n^{-1/d}$ is not probabilistically complete. Furthermore, for any such variable radius PRM algorithm ALG,*

$$\lim_{n\to\infty} \mathbb{P}\big(\{\textit{There exists a feasible path } \sigma \textit{ through } G_n^{ALG} = (V_n^{ALG}, E_n^{ALG})\}\big) = 0.$$

The proof of this theorem is given in Section 5.2.2. The constant γ in Theorem 5.5 is the percolation threshold for random r-disc graphs, which is closely related to the continuum percolation threshold (see Theorem 2.3).

Analysis of proposed algorithms: The algorithms PRM*, RRG, and RRT* are all probabilistically complete, for the feasible path planning problem (Problem 3.1). In fact, for the RRG and the RRT*, it is easy to prove a stronger result.

Theorem 5.6 (Probabilistic Completeness of PRM*) *The PRM* algorithm is probabilistically complete.*

Theorem 5.7 (Probabilistic Completeness of RRG) *The RRG algorithm is probabilistically complete. Furthermore, for any robustly feasible problem instance $(\mathcal{X}_{\text{free}}, x_{\text{init}}, \mathcal{X}_{\text{obs}})$, there exists constants $a > 0$ and $n_0 \in \mathbb{N}$, both independent of n, such that, for all $n > n_0$,*

$$\mathbb{P}\big(\{\textit{There exists a feasible path } \sigma \textit{ through } G_n^{\text{RRG}} = (V_n^{\text{RRG}}, E_n^{\text{RRG}})\}\big) > 1 - e^{-an}.$$

Theorem 5.8 (Probabilistic Completeness of RRT*) *The RRT* algorithm is probabilistically complete. Furthermore, for any robustly feasible problem instance $(\mathcal{X}_{\text{free}}, x_{\text{init}}, \mathcal{X}_{\text{obs}})$, there exists constants $a > 0$ and $n_0 \in \mathbb{N}$, both independent of n, such that, for all $n > n_0$,*

$$\mathbb{P}\big(\{\textit{There exists a feasible path } \sigma \textit{ through } G_n^{\text{RRT}^*} = (V_n^{\text{RRT}^*}, E_n^{\text{RRT}^*})\}\big) > 1 - e^{-an}.$$

The proofs of Theorems 5.6-5.8 are given in Section 5.2.3.

The μ-RRT and μ-RRT* algorithms are probabilistically complete, for the feasible path planning problem with deterministic μ-calculus specifications (Problem 3.3).

Theorem 5.9 (Probabilistic Completeness of μ-RRT and μ-RRT*) *The μ-RRT and the μ-RRT* algorithms are probabilistically complete.*

The proof of Theorem-5.9 is provided in Section 5.2.4.

5.1.2 Asymptotic Optimality

In this section, we analyze the algorithms that were introduced in Chapter 4 in terms of their ability to return paths that converge to optimal solutions. We first introduce a precise definition of asymptotic optimality. Then, we show that two of the existing algorithms, namely the RRT and the k-nearest PRM, for any fixed k, lack this property. Finally, we show that the PRM*, RRG, and the RRT* algorithms as well as their k-nearest variants are asymptotically optimal. We also argue the asymptotic optimality of the μ-RRT* algorithm for the optimal path problem with deterministic μ-calculus specifications.

A rigorous definition of asymptotic optimality: Recall from the previous section that a path is robustly feasible, for a given path planning problem instance, if it is feasible and it has strong δ-clearance. We use a similar notion, called weak clearance and introduced below, together with a (semi-)continuity property for the cost functional, to define robustly optimal paths.

Let σ_1, σ_2 be two collision-free paths with the same end points, *i.e.*, $\sigma_1(0) = \sigma_2(0)$ and $\sigma_1(1) = \sigma_1(1)$. Then, σ_1 is said to be *homotopic* to σ_2, if there exists a continuous function $\xi : [0,1] \to \Sigma_{\text{free}}$ such that $\xi(0) = \sigma_1$, $\xi(1) = \sigma_2$, and $\xi(\tau)$ is a collision-free path for all $\tau \in (0,1)$. In this case, the function ξ is called the *homotopy*. Intuitively, any path that is homotopic to σ can be continuously transformed to σ through collision-free paths (see Munkres, 2000).

A collision-free path $\sigma : [0,1] \to \mathcal{X}_{\text{free}}$ is said to have *weak clearance*, if there exists a path σ' that has strong δ-clearance, for some $\delta > 0$, and there exists a homotopy ξ, with $\xi(0) = \sigma'$ and $\xi(1) = \sigma$, such that for all $\tau \in [0,1)$ there exists some $\delta_\tau > 0$ such that the path $\xi(\tau)$ has strong δ_τ-clearance. In other words, a path has weak clearance if it is homotopic to a path through paths that have strong δ-clearance. An important consequence of weak clearance is the following: A path σ with weak clearance may not have strong δ-clearance for any $\delta > 0$; however, there exists a sequence of paths, all of which have strong δ-clearance, and this sequence of paths converges to σ. We make this claim precise later in our analysis (see Lemma 5.47).

The weak clearance property is illustrated, for a path planning problem in Figure 5-2 in a simple example. A more complex example is provided in Figure 5-3. A path that violates the weak clearance property is shown in Figure 5-4.

Note that the notion of weak clearance extends directly to paths that solve path planning problems with complex task specifications (Problems 3.3 and 3.4), since the notion of strong δ-clearance also extends to such paths (see the discussion after the definition of strong δ-clearance given in the previous section). Moreover, the notion of homotopy classes can also be defined in problems with complex task specifications. Strictly speaking, two feasible paths are *homotopic* if they can be continuously deformed to one another through feasible paths. Here, we require a stronger notion, called *trace homotopy*, whereby two paths are declared trace homotopic if they can be continuously deformed to one another through paths with the same trace.

To make precise the notion of limit of a sequence paths, we introduce the set of all paths as a normed space. Recall that Σ is the set of all paths, and $\text{TV}(\cdot)$ is the total

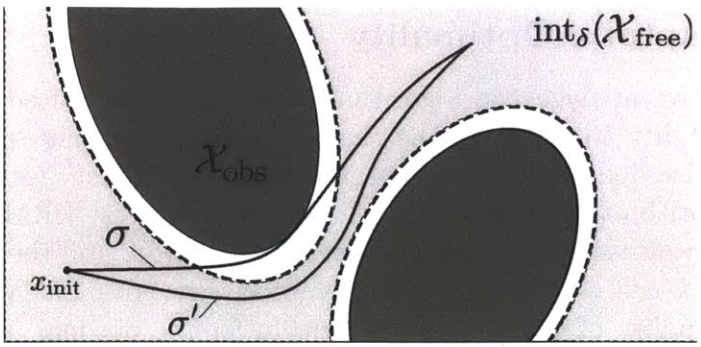


Figure 5-2: An illustration of a path σ that has weak clearance. The path σ', with strong δ-clearance, that is in the same homotopy class with σ is also shown in the figure. Note that σ does not have strong δ-clearance.

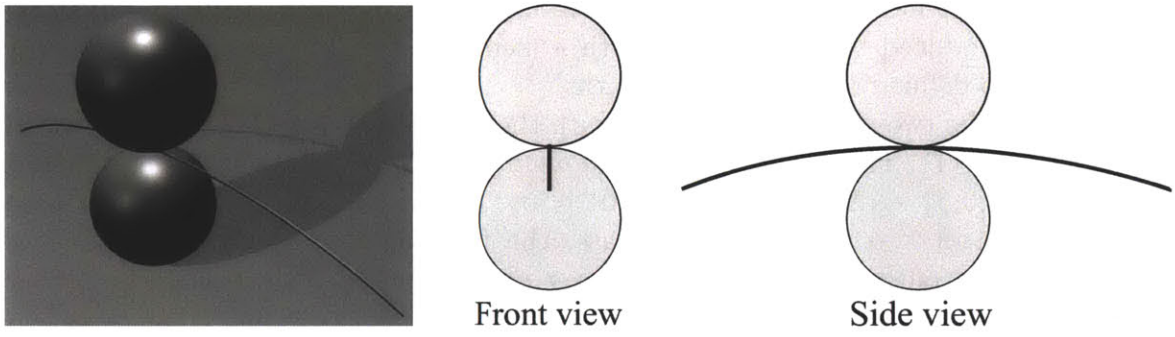


Figure 5-3: An illustration of a path that has weak clearance. The path passes through a point where two obstacles come in contact. Clearly, the path does not have strong δ-clearance for any $\delta > 0$.

variation, *i.e.*, the length, of a path (see Section 3.1). Given two paths $\sigma_1, \sigma_2 \in \Sigma$, which, recall, are continuous functions with (bounded variation) of the form $\sigma_1, \sigma_2 : [0,1] \to \mathbb{R}^d$, the addition operation is defined as $(\sigma_1 + \sigma_2)(\tau) = \sigma_1(\tau) + \sigma_2(\tau)$ for all $\tau \in [0,1]$. Given a path $\sigma \in \Sigma$ and a scalar $\alpha \in \mathbb{R}$, the multiplication by a scalar operation is defined as $(\alpha\,\sigma)(\tau) = \alpha\sigma(\tau)$ for all $\tau \in [0,1]$. Clearly, the set Σ of paths is closed under addition and scalar multiplication. In fact, with these operators, the function space Σ is a vector space.

On the vector space Σ, define the norm $\|\sigma\|_{\text{BV}} := \int_0^1 |\sigma(\tau)|\, d\tau + \text{TV}(\sigma)$, where $|\cdot|$ denotes the usual Euclidean norm in $\mathbb{R}^d$. This norm induces the following distance function:

$$\text{dist}_{BV}(\sigma_1, \sigma_2) := \|\sigma_1 - \sigma_2\|_{\text{BV}} = \int_0^1 |(\sigma_1 - \sigma_2)(\tau)|\, d\tau + \text{TV}(\sigma_1 - \sigma_2).$$

A sequence of paths $\{\sigma_n\}_{n\in\mathbb{N}}$ of paths is said to converge to a path $\bar{\sigma}$, denoted simply by $\lim_{n\to\infty} \sigma_n = \bar{\sigma}$, if $\lim_{n\to\infty} \|\sigma_n - \bar{\sigma}\|_{\text{BV}} = 0$.

A path is said to be *optimal*, if it solves the optimal path planning problem. An optimal path $\sigma^* \in \Sigma_{\text{free}}$ is said to be *robustly optimal*, if it has weak clearance and,

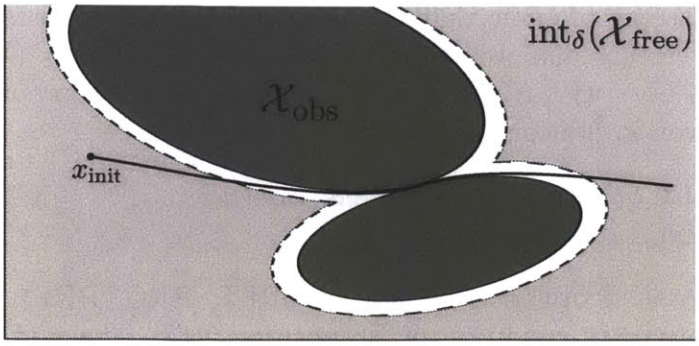


Figure 5-4: An illustration of a path σ that does not have weak clearance. In particular, for any positive value of δ, there is no path in the δ-interior of the free space that is homotopic to σ.

for any sequence $\{\sigma_n\}_{n\in\mathbb{N}}$ of collision-free paths with $\sigma_n \in \Sigma_{\text{free}}$ for all $n \in \mathbb{N}$ and $\lim_{n\to\infty} \sigma_n = \bar{\sigma}$, we have $\lim_{n\to\infty} c(\sigma_n) = c(\sigma^*)$. Clearly, a path planning problem that admits a robustly optimal solution is robustly feasible.

Let ALG be a sampling-based planning algorithm. Let c^* denote the cost of an optimal solution. Denote by Y_n^{ALG} the cost of the minimum-cost feasible path through the graph $G_n^{ALG} = (V_n^{ALG}, E_n^{ALG})$, which ALG returns when it is run with n samples.

Definition 5.10 (Asymptotic Optimality) *An algorithm ALG is said to be asymptotically optimal, if, for any optimal path planning problem instance that admits a robustly optimal solution with finite cost c^*,*

$$\mathbb{P}\big(\{\limsup_{n\to\infty} Y_n^{ALG} = c^*\}\big) = 1.$$

Since $Y_n^{ALG} \geq c^*$ for all $n \in \mathbb{N}$, asymptotic optimality implies that $\lim_{n\to\infty} Y_n^{ALG}$ exists and is equal to c^*, almost surely. Clearly, probabilistic completeness is necessary for asymptotic optimality.

Preliminary results: Before stating our main results, we note a number of interesting properties of sampling-based algorithms regarding asymptotic optimality.

Firstly, an application of Kolmogorov's zero-one law leads to the following fact. Under certain assumptions, the event that a sampling-based algorithm converges to an optimal solution has probability either zero or one. That is, a given sampling-based planning algorithm either converges to an optimal solution in almost all runs or the convergence fails to occur in almost all runs, for a fixed problem instance. See the following proposition, the proof of which can be found in Section 5.3.1.

Proposition 5.11 *Suppose the only randomness in the sampling-based algorithm is introduced by the* `Sample` *procedure. Conditioning on the event that ALG returns a solution eventually,* i.e., $\limsup_{n\to\infty} Y_n^{ALG} < \infty$, *the probability that* $\limsup_{n\to\infty} Y_n^{ALG} = c^*$ *is either zero or one.*

Secondly, it is easy to show that the limit $\lim_{n\to\infty} Y_n^{ALG}$ exists, if the algorithm ALG satisfies a certain monotonicity condition. A graph $G = (V, E)$ is contained in another graph $G' = (V', E')$ if $V \subseteq V'$ and $E \subseteq E'$. The proof for the following proposition is given in Section 5.3.1.

Proposition 5.12 *Suppose, for all sample sequences, $G_n^{ALG} \subseteq G_{n+1}^{ALG}$ for all $n \in \mathbb{N}$. Then, the limit $\lim_{n\to\infty} Y_n^{ALG}$ exists surely.*

The incremental algorithms, namely the RRT, RRG, RRT*, and μ-RRT*, all satisfy the monotonicity condition of Proposition 5.12. The said limit does exists, almost surely, for the other algorithms, in particular the PRM*. However, a rigorous proof requires a more involved coupling argument.

Finally, we make the following assumption throughout this dissertation, to rule out trivial cases where the optimal path may be found with only finitely many samples. Let Σ^* denote the set of all optimal paths, and let $\mathcal{X}_{\text{opt}}$ denote the set of all configurations an optimal path passes through, *i.e.*,

$$\mathcal{X}_{\text{opt}} := \{x \in \mathcal{X}_{\text{free}} : \text{ there exists } \sigma^* \in \Sigma^* \text{ and } \tau \in [0,1] \text{ such that } x = \sigma^*(\tau)\}.$$

Recall that $\mu(\cdot)$ is the usual Lebesgue measure in the d-dimensional Euclidean space.

Assumption 5.13 (Zero-measure optimal paths) *The set of all points traversed by optimal paths has Lebesgue measure zero,* i.e., $\mu(\mathcal{X}_{\text{opt}}) = 0$.

Arguably, this assumption holds in most practical cases. Note that this assumption does not imply that there is a single optimal path. In fact, there are problem instances with uncountably many optimal paths, for which Assumption 5.13 holds.[3] Assumption 5.13 implies that, roughly speaking, no sampling-based planning algorithm can find a solution to the optimal path planning problem in a finite number of iterations, almost surely.[4] The proof of the following theorem is given in Section 5.3.1.

Proposition 5.14 *Suppose Assumption 5.13 holds. Then, the probability that ALG returns a graph containing an optimal at a finite iteration $n \in \mathbb{N}$ is zero,* i.e.,

$$\mathbb{P}\Big(\bigcup\nolimits_{n\in\mathbb{N}} \{Y_n^{ALG} = c^*\}\Big) = 0.$$

Throughout, we assume that Assumption 5.13 holds, which implies that Proposition 5.14 holds also.[5]

[3]Consider, for example, a motion planning problem in the three-dimensional configuration space where a (Euclidean-)ball-shaped obstacle is placed such that its center lies on the straight path that connects the initial and the goal configurations.

[4]When making this claim, we assume that a sampling-based algorithm has access to the configuration space and the problem instance only through the primitive procedures described in Section 4.1.

[5]Note that, although Assumption 5.13 is sufficient for Proposition 5.14 to hold, it is not a necessary condition. There are weaker conditions under which Proposition 5.14 holds. We do not discuss such conditions here, in order to keep the technical argument simple.

Analysis of existing algorithms: Next, we state our results regarding the asymptotic optimality or the lack thereof for the existing algorithms.

First, consider the PRM algorithm and its variants. The PRM algorithm, in its original form, is not asymptotically optimal.

Theorem 5.15 (Non-optimality of the PRM) *The PRM algorithm is not asymptotically optimal.*

The proof of this theorem is provided in Section 5.3.2. The lack of asymptotic optimality in PRM is due to its incremental construction, coupled with the constraint eliminating edges that make unnecessary connections within a connected component. Such a constraint is not present in the batch construction of the sPRM algorithm, which is indeed asymptotically optimal (at the expense of computational complexity; see Section 5.4). The proof of the following theorem is given in Section 5.3.3

Theorem 5.16 (Asymptotic Optimality of sPRM) *The sPRM algorithm is asymptotically optimal.*

On the other hand, as in the case of probabilistic completeness, the heuristics that are often used in the practical implementations of (s)PRM are not necessarily asymptotically optimal.

Theorem 5.17 (Non-optimality of the k-nearest sPRM) *The k-nearest PRM algorithm is not asymptotically optimal, for any fixed $k \in \mathbb{N}$.*

The proof of this theorem is given in Section 5.3.4.

A large class of variable-radius sPRMs, where the connection radius is shrunk at a rate that is faster than in the PRM*, also lack the asymptotic optimality property.

Theorem 5.18 (Non-optimality of variable-radius sPRM with $r(n) \leq \gamma n^{-1/d}$) *The variable-radius sPRM with connection radius $r(n) \leq \gamma\, n^{-1/d}$ is not asymptotically optimal, for all fixed $\gamma > 0$.*

The proof of this theorem is given in Section 5.3.5.

Like widely-used variants of the PRM algorithm, the RRT algorithm is not asymptotically optimal either. The proof of the following theorem is given in Section 5.3.6.

Theorem 5.19 (Non-optimality of the RRT) *The RRT algorithm is not asymptotically optimal.*

A straightforward application of Propositions 5.11 and 5.12 shows that, for those algorithms that are not asymptotically optimal, the probability that they converge to an optimal solution is, in fact, zero. In other words, all these algorithms converge to a non-optimal solution, almost surely, *i.e.*, $\mathbb{P}(\{\limsup_{n\to\infty} Y_n^{ALG} > c^*\}) = 1$, where ALG is one of the algorithms mentioned in Theorems 5.15, 5.17, 5.18, and 5.19.

Analysis of proposed algorithms: Next, we discuss the asymptotic optimality of the algorithms PRM*, RRG, and RRT* as well as their k-nearest variants, in the optimal path planning problem given in Problem 3.2.

The proofs of the following theorems are provided in Sections 5.3.7-5.3.11.

Theorem 5.20 (Asymptotic Optimality of PRM*) *The PRM* algorithm is asymptotically optimal.*

Theorem 5.21 (Asymptotic Optimality of k-nearest PRM*) *The k-nearest PRM* algorithm is asymptotically optimal.*

Theorem 5.22 (Asymptotic Optimality of RRG) *The RRG algorithm is asymptotically optimal.*

Theorem 5.23 (Asymptotic Optimality of k-nearest RRG) *The k-nearest RRG algorithm is asymptotically optimal.*

Theorem 5.24 (Asymptotic Optimality of RRT*) *The RRT* algorithm is asymptotically optimal.*

Theorem 5.25 (Asymptotic Optimality of k-nearest RRT*) *The k-nearest RRT* algorithm is asymptotically optimal.*

The μ-RRT* algorithm and its k-nearest variant are asymptotically optimal for the optimal path planning problem with deterministic μ-calculus specifications (Problem 3.4). The proofs of the following theorems are given in Section 5.3.13.

Theorem 5.26 (Asymptotic Optimality of μ-RRT*) *The μ-RRT* algorithm is asymptotically optimal.*

Theorem 5.27 (Asymptotic Optimality of k-nearest μ-RRT*) *The k-nearest μ-RRT* algorithm is asymptotically optimal.*

5.1.3 Computational Complexity

In this section, we compare the algorithms provided in Chapter 4 in terms of computational complexity. First, we analyze each algorithm in terms of number of calls to the `CollisionFree` procedure. Subsequently, we comment on the computational complexity of the `Nearest` and `Near` procedures. These results lead to a thorough analysis of the time complexity of the algorithms in terms of simple computational operations, such as additions, multiplications, and comparisons.

Throughout this section, we use the following notation for stating asymptotic computational complexity. Let W_n^{ALG} be a function of the graph $G_n^{ALG} = (V_n^{ALG}, E_n^{ALG})$ returned by algorithm ALG when ALG is run with n samples. Clearly, W_n^{ALG} is a random variable that depends on the problem instance. Let $f : \mathbb{N} \to \mathbb{N}$ be a non-decreasing function with $\lim_{n\to\infty} f(n) = \infty$. Following the usual asymptotic

computational complexity notation, the random variable W_n^{ALG} is said to belong to $\Omega(f(n))$, denoted by $W_n^{ALG} \in \Omega(f(n))$, if there exists a problem instance for which $\liminf_{n\to\infty} \mathbb{E}[W_n^{ALG}]/f(n) > 0$. Similarly, W_n^{ALG} is said to belong to $O(f(n))$, denoted by $W_n^{ALG} \in O(f(n))$, if $\limsup_{n\to\infty} \mathbb{E}[W_n^{ALG}]/f(n) < \infty$ for all problem instances. In other words, on one hand, in the case of $W_n^{ALG} \in \Omega(f(n))$, there exists at least one problem instance in which the expected value of the random variable W_n^{ALG} grows at least as fast as $f(n)$; on the other hand, in the case of $W_n^{ALG} \in O(f(n))$, the random variable W_n^{ALG} grows slower than $f(n)$ in all problem instances. We will use these two notations to describe lower and upper bounds for the asymptotic computational complexity of the sampling-based algorithms that this thesis is concerned with.

Number of calls to the `CollisionFree` procedure: Let M_n^{ALG} denote the total number of calls to the `CollisionFree` procedure by algorithm ALG, when ALG is run with n samples.

First, note the following lower bounds for the PRM and the sPRM algorithms. The proofs of Lemmas 5.28 and 5.29 are provided in Sections 5.4.1 and 5.4.2, respectively.

Lemma 5.28 (Collision-checking Complexity of PRM) *For the PRM algorithm,* $M_n^{\text{PRM}} \in \Omega(n^2)$.

Lemma 5.29 (Collision-checking Complexity of sPRM) *For the sPRM algorithm,* $M_n^{\text{sPRM}} \in \Omega(n^2)$.

Clearly, for the k-nearest PRM and the RRT algorithms, $M_n^{k\text{PRM}} = k\,n$ and $M_n^{\text{RRT}} = n$, for all $n \in \mathbb{N}$ in all problem instances. Compare these results with those of Lemmas 5.28 and 5.29. The proposed algorithms have time complexity that is much closer to the more efficient algorithms, RRT and the k-nearest PRM.

Lemma 5.30 (Collision-checking Complexity of PRM*, RRG, and RRT*) *For the PRM*, RRG, and RRT* algorithms,* $M_n^{\text{PRM}^*}, M_n^{\text{RRG}}, M_n^{\text{RRT}^*} \in O(n \log(n))$.

The proof of this lemma is provided in Section 5.4.3.

Finally, for the k-nearest variants of the proposed algorithms, trivially, $M_n^{k\text{PRM}^*} = M_n^{k\text{RRG}} = M_n^{k\text{RRT}^*} = \bar{k} \log(n)$ for some constant $\bar{k}$ that is independent of n.

Computational Complexity of the `CollisionFree` procedure: Now, let us characterize the amount of computational effort required to execute the `CollisionFree` procedure itself. Clearly, the number of simple computational operations required to do so is independent of the number of samples n. However, in general, this number depends on the characteristics of the environment.

The problem of collision checking a path against obstacles is a widely studied problem in the context computational geometry (see Lin and Manocha, 2012, for a survey). There are several practical algorithms that efficiently implement the collision checking procedure. In particular, using data structures based on spatial trees, the algorithm given by Six and Wood (1982) runs in time $O(\log^d(m))$, where d is the dimensionality of the configuration space and m is the number of obstacles (see also Edelsbrunner and Maurer, 1981; Hopcroft et al., 1983).

Computational Complexity of the `Nearest` procedure: The nearest neighbor search problem is also widely studied in the literature, since it has a number of applications in, *e.g.*, computer graphics, database systems, image processing, pattern recognition *etc.* (Samet, 1990a). The computational complexity of the nearest procedure clearly depends on the number of vertices, thus the number of samples n.

A naive algorithm that examines every vertex runs in $O(n)$ time and requires $O(1)$ space. However, in many online real-time applications such as robotics, it is highly desirable to reduce computation time of each iteration within sublinear bounds, *e.g.*, $O(\log n)$ time. Such algorithms may be especially valuable in anytime algorithms that intend to improve the solution as the number of iterations increases.

Fortunately, there exist very efficient algorithms for computing an "approximate" nearest neighbor, if not an exact one. In the sequel, a vertex $v \in V \subset \mathcal{X}$ is said to be an ϵ-approximate nearest neighbor of a configuration x, if $|v - x| \leq (1 + \epsilon)|u - x|$ for all $u \in V$. An approximate nearest neighbor can be computed, for instance, by using balanced-box decomposition (BDD) trees, which achieves $O(c_{d,\epsilon} \log n)$ query time using $O(d\, n)$ space, where $c_{d,\epsilon} \leq d\lceil 1+6\, d/\epsilon \rceil^d$ (Arya et al., 1998). This algorithm is computationally optimal in fixed dimensions, since it closely matches a lower bound for algorithms that use a tree structure stored in roughly linear space (see Arya et al., 1998). These results suggest that the `Nearest` procedure requires at least $O(\log n)$ time answer nearest neighbor queries or an approximation thereof. Finally, note that approximate nearest neighbor computation in the context of PRMs and RRTs was discussed very recently (Yershova and LaValle, 2007; Plaku and Kavraki, 2008).

Computational Complexity of the `Near` procedure: Problems similar to that solved by the `Near` procedure are also widely studied in the literature, often under the name of "range search problems" (Samet, 1990b). Using k-d trees, in fixed dimensions and in the worst case, computing the exact set of vertices that reside in a ball of radius r_n centered at some query point takes $O(n^{1-1/d}+m)$, where m is the number of vertices returned by the search (Lee and Wong, 1977). See also (Chanzy et al., 2000) for an analysis of the average case.

Similar to the nearest-neighbor search, computing approximate solutions to the range search problem is computationally easier. A range search algorithm is said to be ϵ-approximate if it returns all vertices that reside in the ball of size r_n and no vertices outside a ball of radius $(1+\epsilon)r_n$, but may or may not return the vertices that lie outside the former ball and inside the latter box.

Computing ϵ-approximate near neighbors using BBD-trees requires $O(2^d \log n + d(3\sqrt{d}/\epsilon)^d)$ time when using $O(d\, n)$ space, in the worst case (Arya and Mount, 2000). Thus, in fixed dimensions, the complexity of this algorithm is $O(\log n + (1 + \epsilon)^{d-1})$, which is known to be computationally optimal, closely matching a lower bound (Arya and Mount, 2000). More recently, algorithms that can provide trade-offs between time and space were also proposed (see Arya et al., 2005).

Note that the `Near` procedure can be implemented as an approximate range search while the maintaining asymptotic optimality guarantee and without changing the number of calls to the `CollisionFree` procedure up to a constant factor, in all of

the proposed algorithms. Hence, the `Near` procedure can be implemented to run in $O(\log n)$ expected time and linear space, in fixed dimensions.

Computational Complexity of the Incremental Model Checking Algorithm: The μ-RRT and the μ-RRT* algorithms, proposed in Section 4.3.5, employ the incremental model checking algorithm proposed in Section 4.3.4. In this section, we characterize the computational complexity of this model checking algorithm in terms of the size of the Kripke structure that it generates.

$\mathcal{K} = (\mathcal{S}, s_{\text{init}}, \mathcal{R}, \mathcal{L})$ is the Kripke structure generated using the `AddState` and the `AddTransition` procedures. Let $N_{\mathcal{K}}^{\texttt{AS}}$ and $N_{\mathcal{K}}^{\texttt{AD}}$ denote the total number of simple computational operations executed by the `AddState` and the `AddTransition` procedures, respectively, in the process of generating this Kripke structure. During this process, in general, the said procedures are executed multiple times; $N_{\mathcal{K}}^{\texttt{AS}}$ and $N_{\mathcal{K}}^{\texttt{AD}}$ denote the aggregate number of simple operations performed in all these executions. Recall that $\text{card}\,(\cdot)$ is the cardinality operator. Let $|\phi|$ denote the size of the μ-calculus formula ϕ defined as the total number of appearances of all operators, atomic propositions, and variables in ϕ. Let $\text{SF}_\nu(\phi)$ denote the set of all subformulas ψ of ϕ that are of the form $\psi = \nu x.\psi'$ for some $x \in \texttt{Var}$ and $\psi' \in L_1$.

Lemma 5.31 (Time Complexity of Incremental Model Checking) *Suppose the incremental model checking procedure is executed (using the interfacing procedures* `AddState` *and* `AddTransition`*), with the deterministic μ-calculus formula ϕ as input, to generate a Kripke structure $\mathcal{K} = (\mathcal{S}, s_{\text{init}}, \mathcal{R}, \mathcal{L})$. Then, the number of simple operations executed by the* `AddState` *and the* `AddTransition` *procedures satisfy $N_{\mathcal{K}}^{\texttt{AS}} \in O(\text{card}\,(\mathcal{S}))$ and $N_{\mathcal{K}}^{\texttt{AD}} \in O\big(\log(\text{card}\,(\mathcal{S})\ \text{card}\,(\text{SF}_\nu(\phi)))\ \text{card}\,(\mathcal{R})\ |\phi|\big)$.*

The proof of this lemma is given in Section 5.4.4. The overall time complexity of the incremental model checking algorithm satisfies $N_{\mathcal{K}}^{\texttt{AS}} + N_{\mathcal{K}}^{\texttt{AD}} \in O(\log(\text{card}\,(\mathcal{S}))\ \text{card}\,(\mathcal{R})\ |\phi|)$, since $\text{card}\,(\mathcal{S}) \leq \text{card}\,(\mathcal{R})$ by construction.

Time Complexity: Let N_n^{ALG} denote the time complexity of algorithm ALG when it is run with n samples. The time complexity is measured in terms of the number of simple computational operations, such as additions, multiplications, and comparisons, executed by the algorithm. The following theorems are deduced easily from the results provided later in this section. Nevertheless, their proofs are provided in Section 5.4.5.

Among the existing algorithms, the PRM and the sPRM algorithms have the following complexity.

Theorem 5.32 (Time Complexity of PRM and sPRM) *The PRM and the sPRM algorithms satisfy $N_n^{\text{PRM}}, N_n^{\text{sPRM}} \in \Omega(n^2)$.*

The RRT and the k-nearest PRM are substantially more efficient in this measure.

Theorem 5.33 (Time Complexity of k-nearest PRM and RRT) *The k-nearest PRM and the RRT algorithms satisfy $N_n^{k\text{PRM}}, N_n^{\text{RRT}} \in \Omega(n \log n)$.*

Finally, the proposed algorithms match the computational complexity of the most efficient existing algorithms.

Theorem 5.34 (Time Complexity of PRM*, RRG, and RRT*) *The time complexity of PRM*, RRG, and RRT* algorithms and their k-nearest variants satisfy* $N_n^{\mathrm{PRM}} \in O(n \log n)$, *where* $ALG \in \{\mathrm{PRM}^*, k\mathrm{PRM}^*, \mathrm{RRG}, k\mathrm{RRG}, \mathrm{RRT}^*, k\mathrm{RRT}^*\}$.

Theorem 5.35 (Time Complexity of μ-RRT and μ-RRT*) *The time complexity of the μ-RRT and the μ-RRT* algorithms satisfy* $N_n^{\mu\mathrm{RRG}}, N_n^{\mu\mathrm{RRG}^*} \in O(n \log^2 n)$.

The proofs of Theorems 5.32, 5.33, and 5.34 can be found in Section 5.4.5.

Note that these results measure only the computational effort devoted to constructing the graph $G_n^{ALG} = (V_n^{ALG}, E_n^{ALG})$ that is returned by the algorithm. They do not reflect the query time, *i.e.*, time devoted to extracting a feasible path or the path with minimum cost from the graph. In some cases, the query process is simple. For example, in the RRT and the RRT* algorithms the query process consists of finding a vertex in the goal region and tracing it back to the root. In algorithms like the PRM, PRM*, and the RRG, a standard shortest path algorithm (that runs on finite graphs) can be used for the query phase. Given a graph $G = (V, E)$ with a cost function $c : E \to \mathbb{R}_{>0}$ that assigns each edge with a cost, the path the smallest accumulated cost from an initial vertex to all other vertices can be found in $O(\mathrm{card}\,(V) \log(\mathrm{card}\,(V)) + \mathrm{card}\,(E))$ time, where $\mathrm{card}\,(\cdot)$ denotes the cardinality of a set (Schrijver, 2003). Using this result and the intermediate results established in Sections 5.4.1-5.4.3, the time complexity of the query phase for all the algorithms analyzed in this thesis is exactly the same as their time complexity for computing the output graph, which was reported in Theorems 5.32-5.34.

5.2 Proofs on Probabilistic Completeness

In this section, the proofs of our results on the probabilistic completeness or the lack there off of the existing and proposed algorithms are presented.

5.2.1 Incompleteness of 1-nearest PRM

In this section, we provide a proof of Theorem 5.4, establishing the lack of probabilistic completeness in 1-nearest PRM. The proof of this theorem requires two intermediate results, which we provide below.

For simplicity of the presentation, consider the case when $\mathcal{X}_{\mathrm{free}} = \mathcal{X}$. Let $G_n^{\text{1-PRM}} = (V_n^{\text{1-PRM}}, E_n^{\text{1-PRM}})$ denote the graph returned by the 1-nearest sPRM algorithm, when the algorithm is run with n samples. The Euclidean length of an edge $e = (v, v') \in E_n^{\text{1-PRM}}$ is defined as $|v - v'|$, where $|\cdot|$ is the usual Euclidean norm in the d-dimensional Euclidean space. Let L_n denote the total Euclidean length of all the edges present in $G_n^{\text{1-PRM}}$. Recall that ζ_d denotes the volume of the unit ball in the d-dimensional Euclidean space. Let $\zeta_d{}'$ denote the total volume of the union of two unit balls whose centers are a unit distance apart.

Lemma 5.36 (Total length of the 1-nearest neighbor graph (Wade, 2007)) *For all $d \geq 2$, $L_n/n^{1-1/d}$ converges to a constant in mean square,* i.e.,

$$\lim_{n\to\infty} \mathbb{E}\left[\left(\frac{L_n}{n^{1-1/d}} - \left(1+\frac{1}{d}\right)\left(\frac{1}{\zeta_d} - \frac{\zeta_d}{2\,(\zeta_d{}')^{1+1/d}}\right)\right)^2\right] = 0.$$

Proof This lemma is a direct consequence of Theorem 3 in (Wade, 2007). ■

Let N_n denote the number of connected components of G_n^{1PRM}.

Lemma 5.37 (Number of connected components of the 1-nearest neighbor graph) *For all $d \geq 2$, N_n/n converges to a constant in mean square,* i.e.,

$$\lim_{n\to\infty} \mathbb{E}\left[\left(\frac{N_n}{n} - \frac{\zeta_d}{2\,\zeta_d{}'}\right)^2\right] = 0.$$

Proof A reciprocal pair is a pair of vertices each of which is the other one's nearest neighbor. In a graph formed by connecting each vertex to its nearest neighbor, any connected component includes exactly one reciprocal pair whenever the number of vertices is greater than 2 (see, *e.g.* Eppstein et al., 1997). The number of reciprocal pairs in such a graph was shown to converge to $\zeta_d/(2\zeta_d{}')$ in mean square in (Henze, 1987); See also Remark 2 in (Wade, 2007). ■

Proof of Theorem 5.4 Let $\widetilde{L}_n$ denote the average length of a connected component in G_n^{1PRM}, *i.e.*, $\widetilde{L}_n := L_n/N_n$. Let L'_n denote the length of the connected component that includes x_{init}. Since the samples are drawn independently and uniformly, the random variables $\widetilde{L}_n$ and L'_n have the same distribution (although they are clearly dependent). Let γ_L denote the constant that $L_n/n^{1-1/d}$ converges to (see Lemma 5.36). Similarly, let γ_N denote the constant that N_n/n converges to (see Lemma 5.37).

Recall that convergence in mean square implies convergence in probability and hence convergence in distribution (Grimmett and Stirzaker, 2001). Since both $L_n/n^{1-1/d}$ and N_n/n converge in mean square to constants and $\mathbb{P}(\{N_n = 0\}) = 0$ for all $n \in \mathbb{N}$, by Slutsky's theorem (Resnick, 1999), $n^{1/d}\,\widetilde{L}_n = \frac{L_n/n^{1-1/d}}{N_n/n}$ converges to $\gamma := \gamma_L/\gamma_N$ in distribution. In this case, it also converges in probability, since γ is a constant (Grimmett and Stirzaker, 2001). Then, $n^{1/d}\,L'_n$ also converges to γ in probability, since $\widetilde{L}_n$ and L'_n are identically distributed for all $n \in \mathbb{N}$. Thus, L_n converges to 0 in probability, *i.e.*, $\lim_{n\to\infty} \mathbb{P}\left(\{L'_n > \epsilon\}\right) = 0$, for all $\epsilon > 0$.

Let $\epsilon > 0$ be such that $\epsilon < \inf_{x\in\mathcal{X}_{\mathrm{goal}}} |x - x_{\mathrm{init}}|$. Let A_n denote the event that the graph returned by the 1-nearest sPRM algorithm contains a feasible path, *i.e.*, one that starts from x_{init} and reaches the goal region while avoiding obstalces. Clearly, the event $\{L_n > \epsilon\}$ occurs whenever A_n does, *i.e.*, $A_n \subseteq \{L_n > \epsilon\}$. Then, $\mathbb{P}(A_n) \leq \mathbb{P}(\{L_n > \epsilon\})$. Taking the limit superior of both sides

$$\liminf_{n\to\infty} \mathbb{P}(A_n) \;\leq\; \limsup_{n\to\infty} \mathbb{P}(A_n) \;\leq\; \limsup_{n\to\infty} \mathbb{P}(\{L_n > \epsilon\}) \;=\; 0.$$

In other words, the limit $\lim_{n\to\infty} \mathbb{P}(A_n)$ exists and is equal zero. ■

5.2.2 Incompleteness of a Class of Variable-radius PRMs

In this section, we provide a proof of Theorem 5.5, establishing the lack of probabilistic completeness in a large class of variable radius PRM algorithms. The proof of this theorem requires an intermediate result from the random geometric graph theory. First, we recall this result below.

Recall that λ_c is the critical density, or continuum percolation threshold (see Section 2.2). Given a Borel set $\Gamma \subseteq \mathbb{R}^d$, let $G_\Gamma^{\text{disc}}(n, r)$ denote the random r-disc graph formed with vertices independent and uniformly sampled from Γ and edges connecting two vertices, v and v', whenever $|v - v'| < r_n$, where $|\cdot|$ is the usual Euclidean norm in $\mathbb{R}^d$.

Lemma 5.38 ((Penrose, 2003)) *Let $\lambda \in (0, \lambda_c)$ and $\Gamma \subset \mathbb{R}^d$ be a Borel set. Consider a sequence $\{r_n\}_{n\in\mathbb{N}}$ that satisfies $n\, r_n^d \leq \lambda$, $\forall n \in \mathbb{N}$. Let $N_{\max}(G_\Gamma^{\text{disc}}(n, r_n))$ denote the size of the largest component in $G_\Gamma^{\text{disc}}(n, r_n)$. Then, there exist constants $a, b > 0$ and $m_0 \in \mathbb{N}$ such that for all $m \geq m_0$,*

$$\mathbb{P}\left(\left\{N_{\max}(G_\Gamma^{\text{disc}}(n, r_n)) \geq m\right\}\right) \leq n\left(e^{-a\,m} + e^{-b\,n}\right).$$

Proof of Theorem 5.5 Let $\epsilon > 0$ such that $\epsilon < \inf_{x\in X_{\text{goal}}} |x - x_{\text{init}}|$ and that the $2\,\epsilon$-ball centered at x_{init} lies entirely within the obstacle-free space. Let $G_n^{\text{PRM}} = (V_n^{\text{PRM}}, E_n^{\text{PRM}})$ denote the graph returned by this variable-radius sPRM, when the algorithm is run with n samples. Let $G_n = (V_n, E_n)$ denote the the restriction of G_n^{PRM} to the $2\,\epsilon$-ball centered at x_{init}, *i.e.*, $V_n = V_n^{\text{PRM}} \cap \mathcal{B}_{x_{\text{init}}, 2\,\epsilon}$, $E_n = (V_n \times V_n) \cap E_n^{\text{PRM}}$.

Clearly, G_n is equivalent to the random r-disc graph on $\Gamma = \mathcal{B}_{x_{\text{init}}, 2\,\epsilon}$. Let $N_{\max}(G_n)$ denote the number of vertices in the largest connected component of G_n. By Lemma 5.38, there exists constants $a, b > 0$ and $m_0 \in \mathbb{N}$ such that

$$\mathbb{P}(\{N_{\max}(G_n) \geq m\}) \leq n\left(e^{-a\,m} + e^{-b\,n}\right),$$

for all $m \geq m_0$. Then, for all $m = (\epsilon/2)\, n^{1/d} > m_0$,

$$\mathbb{P}\left(\left\{N_{\max}(G_n) \geq \frac{\epsilon}{2}\, n^{1/d}\right\}\right) \leq \frac{\epsilon}{2}\, n^{1/d}\left(e^{-a\,(\epsilon/2)\, n^{1/d}} + e^{-b\,n}\right).$$

Let L_n denote the total length of all the edges in the connected component that includes x_{init}. Since $r_n = \lambda^{1/d} n^{-1/d}$,

$$\mathbb{P}\left(\left\{L_n \geq \frac{\epsilon}{2}\right\}\right) \leq \frac{\epsilon}{2}\, n^{1/d}\left(e^{-a\,(\epsilon/2)\, n^{1/d}} + e^{-b\,n}\right).$$

Since the right hand side is summable, by the Borel-Cantelli lemma the event $\{L_n \geq \epsilon/2\}$ occurs infinitely often with probability zero, *i.e.*, $\mathbb{P}(\limsup_{n\to\infty}\{L_n \geq \epsilon/2\}) = 0$.

Given a graph $G = (V, E)$ define the diameter of this graph as the distance between the farthest pair of vertices in V, *i.e.*, $\max_{v,v'\in V} |v - v'|$. Let D_n denote the diameter of the largest component in G_n. Clearly, $D_n \leq L_n$ holds surely. Thus, $\mathbb{P}\left(\limsup_{n\to\infty}\{D_n \geq \epsilon/2\}\right) = 0$.

Let $I \in \mathbb{N}$ be the smallest number that satisfies $r_I \leq \epsilon/2$. Notice that the edges connected to the vertices $V_n^{\mathrm{PRM}} \cap \mathcal{B}_{x_{\mathrm{init}},\epsilon}$ coincide with those connected to $V_n \cap \mathcal{B}_{x_{\mathrm{init}},\epsilon}$, for all $n \geq I$. Let R_n denote distance of the farthest vertex $v \in V_n^{\mathrm{PRM}}$ to x_{init} in the component that contains x_{init} in G_n^{PRM}. Notice also that $R_n \geq \epsilon$ only if $D_n \geq \epsilon/2$, for all $n \geq I$. That is, for all $n \geq I$, $\{R_n \geq \epsilon\} \subseteq \{D_n \geq \epsilon/2\}$, which implies $\mathbb{P}\left(\limsup_{n\to\infty} \{R_n \geq \epsilon\}\right) = 0$.

Let A_n denote the event that the graph returned by this variable radius sPRM algorithm includes a path that reaches the goal region. Clearly, $\{R_n \geq \epsilon\}$ holds, whenever A_n holds. Hence, $\mathbb{P}(A_n) \leq \mathbb{P}(\{R_n \geq \epsilon\})$. Taking the limit superior,

$$\begin{aligned} \liminf_{n\to\infty} \mathbb{P}(A_n) \;&\leq\; \limsup_{n\to\infty} \mathbb{P}(A_n) \\ &\leq\; \limsup_{n\to\infty} \mathbb{P}\left(\{R_n \geq \epsilon\}\right) \;\leq\; \mathbb{P}\Big(\limsup_{n\to\infty} \{R_n \geq \epsilon\}\Big) = 0, \end{aligned}$$

where the last inequality follows from Fatou's lemma (see, *e.g.*, Resnick, 1999). Hence, we establish that $\lim_{n\to\infty} \mathbb{P}(A_n)$ exists and is equal to zero. ■

5.2.3 Completeness of PRM*, RRG, and RRT*

In this section, we prove the probabilistic completeness of the PRM*, RRG, and RRT* algorithms, for solving Problem 3.1.

First, we prove Theorem 5.6, establishing the probabilistic completeness of PRM*.

Proof of Theorem 5.6 The result is a direct consequence of the asymptotic optimality of the PRM* algorithm (see Theorem 5.20). ■

Recall that Theorems 5.7 and 5.8 establish the probabilistic completeness as well as the exponential decay of the probability of failure to find a feasible solution when one exists. The proofs of these two theorems are almost identical. We present these proofs together below.

Proofs of Theorems 5.7 and 5.8 Let Ω denote the sample space that describes the randomness in the `Sample` procedure. Each element of Ω can be considered an infinite sequence of configurations. Given such $\omega \in \Omega$, let $G_n^{ALG}(\omega) = (V_n^{ALG}(\omega), E_n^{ALG}(\omega))$ denote the graph returned by an algorithm ALG when it is run with the sample sequence encoded by ω.

Notice that, by construction, the set of vertices in the graph constructed by the RRG and the RRT* is exactly the same that in the graph constructed by the RRT, only the edge set is different. That is, $V_n^{\mathrm{RRG}}(\omega) = V_n^{\mathrm{RRT}^*}(\omega) = V_n^{\mathrm{RRT}}(\omega)$ for all sample sequences $\omega \in \Omega$ and all $n \in \mathbb{N}$. Noting that, in all these three graphs, every vertex is connected to the initial configuration x_{init}, we establish that the RRG and the RRT* algorithms contain a path that solves the feasible path planning problem whenever does the RRT. Then, the result follows directly from Theorem 5.3. ■

5.2.4 Completeness of μ-RRT and μ-RRT*

In this section, we prove Theorem 5.9, which establishes the probabilistic completeness of the μ-RRT and the μ-RRT* algorithms.

The probabilistic completeness of μ-RRT* follows trivially from that of the μ-RRT algorithm, since, for any sample sequence, they generate the same arena; the only difference is that the μ-RRT* does slightly more work to keep track of the costs of the incurred solutions. Thus, in what follows we concentrate on the μ-RRT* algorithm.

First, we establish the completeness of the incremental model-checking algorithm introduced in Section 4.3.4, under certain conditions.

Lemma 5.39 *Suppose that the* `AcceptVertex`(A, v) *procedure (see Line 9 of Algorithm 9) returns* `True` *for any input. Then, the incremental model checking algorithm is complete,* i.e., *the algorithm finds a winning strategy for Player 1 whenever Player 1 has a winning strategy on the arena of the L_1 game of $\mathcal{K}$ and ϕ.*

Before proving Lemma 5.39, note the following intermediate result, which states that those vertices deleted from the arena in Lines 2-8 of Algorithm 9 are exactly those starting from which Player 1 loses the game or Player 1 can not get the token to starting from the initial vertex (s_{init}, ϕ).

Lemma 5.40 *Let $A = (\mathcal{V}_1, \mathcal{V}_2, \mathcal{E})$ denote the arena generated by the incremental model-checking algorithm. Let $A' = (\mathcal{V}_1', \mathcal{V}_2', \mathcal{E}')$ denote the arena corresponding to the L_1 game of $\mathcal{K}$ and ϕ. Recall that $\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2$ and $\mathcal{V}' = \mathcal{V}_1' \cup \mathcal{V}_2'$. Then, for any $v \in \mathcal{V}_1 \setminus \mathcal{V}_2$, one of the following holds:* (i) *starting from v Player 1 loses any play or* (ii) *Player 1 can not take the token to v starting from the initial vertex (s_{init}, ϕ).*

Proof We make two key observations. Firstly, in the L_1 game of $\mathcal{K}$ and ϕ, there is only one way for Player 2 to win the game: the token gets to a vertex (s, ϕ), where either *(i)* ψ is of the form $\psi = p$ and $p \notin \mathcal{L}(s)$, or *(ii)* $\psi = \neg p$ and $p \in \mathcal{L}(s)$. Secondly, the only class of vertices where Player 2 has more than one action to play consists of exactly those (s, ψ), where $\psi = \psi' \wedge p$ or $\psi = \psi' \wedge \neg p$, for some L_1 formula ψ' and atomic proposition p. Notice that whenever Algorithm 9 starts deleting vertices from the arena it maintains, it starts exactly with one of those vertices that satisfy the the conditions given above. All vertices that reach a vertex that satisfies these conditions are also deleted for convenience. ■

Now, we are ready to prove Lemma 5.39

Proof of 5.39 Firstly, by Lemma 5.40, Player 1 has a winning strategy in A if and only if Player 1 has a winning strategy in the L_1 game of $\mathcal{K}$ and ϕ, where A is the arena generated by the algorithm. Moreover, further investigating Algorithm 10, the incremental model-checking algorithm keeps track of any winning strategy in which the token ends in a literal vertex as well as winning strategies that repeatedly passes the token through a ν-vertex with maximal color. ■

Finally, we prove the main result of this section.

Proof of Theorem 5.9 As noted earlier, we only show the probabilistic completeness of the μ-RRT algorithm, and the probabilistic completeness of the μ-RRT* algorithm follows from that of μ-RRT.

By Lemma 5.39, the incremental model-checking algorithm is complete, when `AcceptVertex` procedure accepts all instances. However, we have designed this procedure so that it accepts a vertex with probability $1/k$ upon the kth time it is called.[6] This implies that the procedure returns `True` infinitely often, almost surely, as a consequence of the Borel-Cantelli Lemma (see Grimmett and Stirzaker, 2001).

Let $v = (s, \psi)$ be some ν-vertex for which `AcceptVertex`(A, v) returns `False`. Since the problem is robustly optimal, however, a configuration s' will be sampled such that s' is close to s and that $\mathcal{L}(s) = \mathcal{L}(s')$ will be sampled and the vertex (s', ψ) will be accepted by the `AcceptVertex` procedure eventually, with probability one. ■

5.3 Proofs on Asymptotic Optimality

In this section, we provide the proofs of our results on asymptotic optimality. Some of the proofs follow a very similar outline and use the same techniques. In such cases, we will sketch some of the intermediate results to repetitions. In particular, the proofs of Theorems 5.25, 5.26, and 5.27 will only be sketched, since they are merely a combination of the previous results.

5.3.1 Preliminary Results on Asymptotic Optimality

In this section, we provide the proofs for Propositions 5.11, 5.12, and 5.14.

Proof of Proposition 5.11 First, let us recall Kolmogorov's zero-one law. Given a sequence $\{Y_n\}_{n\in\mathbb{N}}$ of random variables, let $\mathcal{F}'_m$ denote the σ-field generated by the sequence $\{Y_n\}_{n=m}^{\infty}$ of random variables. The tail σ-field $\mathcal{T}$ is defined as $\mathcal{T} = \bigcap_{n\in\mathbb{N}} \mathcal{F}'_n$. An event A is said to be a *tail event* if $A \in \mathcal{T}$. An important result in probability theory is that any tail event occurs with probability either zero or one. This theorem is often called the Kolmogorov's zero-one law (Resnick, 1999).

Before proving the proposition, note that conditioning on the event $\{\limsup_{n\to\infty} Y_n^{ALG} < \infty\}$ ensures that Y_n^{ALG} is finite, thus a random variable, for all large n. Consider the sequence $\{Y_n^{ALG}\}_{n\in\mathbb{N}}$ of random variables. Let $\mathcal{F}'_m$ denote the σ-fields generated by $\{Y_n^{ALG}\}_{n=m}^{\infty}$. Then,

$$\begin{aligned}\left\{\limsup_{n\to\infty} Y_n^{ALG} = c^*\right\} \\ = \left\{\limsup_{n\to\infty, n\geq m} Y_n^{ALG} = c^*\right\} \in \mathcal{F}'_m \text{ for all } n \in \mathbb{N}.\end{aligned}$$

Hence, $\left\{Y_n^{ALG} = c^*\right\} \in \bigcap_{n\in\mathbb{N}} \mathcal{F}'_m$ is a tail event. The result stated in the proposition follows from the Kolmogorov's zero-one law. ■

[6] Our choice is due to computational complexity reasons; See Theorem 5.31.

Proof of Proposition 5.12 Let Ω denote the sample space. Its elements, which can be thought of infinite sequences of samples, are denoted by ω. Let $G_n^{ALG}(\omega) = \left(V_n^{ALG}(\omega), E_n^{ALG}(\omega)\right)$ denote the graph returned by the algorithm ALG when it is run with the first n samples in the sample sequence encoded by $\omega \in \Omega$. Let $Y_n^{ALG}(\omega)$ denote the cost of the minimum-cost feasible path through $G_n^{ALG}(\omega)$.

The monotonicity of the graphs, namely the condition that $G_n^{ALG}(\omega) \subseteq G_{n+1}^{ALG}(\omega)$ holds for all $n \in \mathbb{N}$ and all $\omega \in \Omega$, implies that $Y_{n+1}^{ALG}(\omega) \leq Y_n^{ALG}(\omega)$ for all $n \in \mathbb{N}$ and all $\omega \in \Omega$. Since $Y_n^{ALG}(\omega) \geq c^*$, where c^* is the cost of an optimal path that solves the problem, we establish that $\lim_{n\to\infty} Y_n^{ALG}$ exists. ■

Proof of Proposition 5.14 Let B_n denote the event that ALG constructs a graph containing a path with cost exactly equal to c^* when it is run with n samples, *i.e.*, $B_n = \{Y_n^{ALG} = c^*\}$. Let B denote the event that ALG returns a graph containing a path that costs exactly c^* for some finite n. Then, B can be written as $B = \cup_{n\in\mathbb{N}} B_n$. Since $B_n \subseteq B_{n+1}$, we have, by the monotonocity of measures, $\lim_{n\to\infty} \mathbb{P}(B_n) = \mathbb{P}(B)$. By Assumption 5.13 and the definition of the sampling procedure, $\mathbb{P}(B_n) = 0$ for all $n \in \mathbb{N}$, since the probability that the set $\bigcup_{i=1}^{n} \{\texttt{SampleFree}(i)\}$ of points contains a point from a zero-measure set is zero. Hence, $\mathbb{P}(B) = 0$. ■

5.3.2 Non-optimality of PRM

In this section we provide a proof of Theorem 5.15, which establishes that the PRM algorithm is not asymptotically optimal.

Proof of Theorem 5.15 Recall that an algorithm lacks asymptotic optimality if there is at least one problem instance that admits a robustly optimal solution, such that the algorithm fails to converge to an optimal when input this problem instance. To prove this result we provide a particular problem instance, in which the PRM algorithm performs exactly like a particular kind of RRT algorithm. Then, the result we prove in Section 5.3.6, that the RRT algorithm fails to converge to optimal solutions in all problem instances, implies the claim of this theorem.

Consider a convex, obstacle-free environment, *e.g.*, $\mathcal{X}_{\text{free}} = \mathcal{X}$. Choose the connection radius for PRM and the steering parameter of the RRT such that $r, \eta > \sqrt{d}$, so that r, η is larger than the maximum distance between any two points from $\mathcal{X}_{\text{free}}$.

At each iteration, exactly one vertex and one edge is added to the graph, since (i) all connection attempts using the local planner (*e.g.*, straight line connections as considered in this document) are collision-free, and (ii) at the end of each iteration, the graph is connected (*i.e.*, it contains only one connected component).

For $x_{\text{init}} = \texttt{SampleFree}(0)$ and for any $\mathcal{X}_{\text{goal}}$, then $Y_n^{\text{PRM}} = Y_n^{\text{RRT}}$ for all $n \in \mathbb{N}$, surely. In particular, since both PRM and RRT satisfy the monotonicity condition in Proposition 5.12, Theorem 5.19 implies

$$\begin{aligned}\mathbb{P}\left(\{\limsup_{n\to\infty} Y_n^{\text{PRM}} = c^*\}\right) &= \mathbb{P}\left(\{\lim_{n\to\infty} Y_n^{\text{PRM}} = c^*\}\right) \\ &= \mathbb{P}\left(\{\lim_{n\to\infty} Y_n^{\text{RRT}} = c^*\}\right) = 0.\end{aligned}$$

Hence, the result follows. ■

5.3.3 Asymptotic Optimality of sPRM

The asymptotic optimality of the sPRM algorithm follows easily from that of the PRM* algorithm.

Proof of Theorem 5.16 Notice that the the graph returned by the sPRM algorithm contains that returned by the PRM* algorithm. More precisely, by construction, $V_n^{\text{sPRM}}(\omega) = V_n^{\text{PRM}^*}(\omega)$ and $E_n^{\text{sPRM}}(\omega) \supseteq E_n^{\text{PRM}^*}(\omega)$, for all $n \in \mathbb{N}$ and all sample sequences $\omega \in \Omega$. Thus, any path through the graph $G_n^{ALG}(\omega)$ is also a path through $G_n^{ALG}(\omega)$, for all $\omega \in \Omega$. Then, the result follows from the asymptotic optimality of the PRM* algorithm (see Theorem 5.20). ■

5.3.4 Non-optimality of k-nearest PRM

In this section, we prove Theorem 5.17, establishing that the k-nearest PRM algorithm is not asymptotically optimal for any fixed k. For simplicity, the theorem will be proven under the assumption that the underlying point process is Poisson. This assumption provides us with certain spatial independence properties, which significantly simplify the proof. More precisely, it is assumed that the algorithm is run with Poisson(n) samples, where Poisson(n) is a Poisson random variable with mean n. That is, the number of samples that the algorithm is run with is precisely the realization of a Poisson random variable with mean n. Hence, the expected number of samples is equal to n; moreover, for large n, large deviations from n are unlikely, since the Poisson distribution has exponentially-decaying tails (see, *e.g.*, Grimmett and Stirzaker, 2001, for a more precise statement).

With a slight abuse of notation, the cost of the minimum-cost feasible path through the graph returned by the k-nearest PRM algorithm, when the algorithm is run with Poisson(n) number of samples, is also denoted by $Y_n^{k\text{PRM}}$. It is shown that $\mathbb{P}(\{\limsup_{n\to\infty} Y_n^{k\text{PRM}} = c^*\}) = 0$.

Proof of Theorem 5.17 Let σ^* denote an optimal path and s^* denote its length, *i.e.*, $s^* = TV(\sigma^*)$. For each n, consider a tiling of σ^* with disjoint open hypercubes, each with edge length $2\,n^{-1/d}$, such that the center of each cube is a point on σ^*. See Figure 5-5. Let M_n denote the maximum number of tiles that can be generated in this manner and note $M_n \geq \frac{s^*}{2}\, n^{1/d}$. Partition each tile into several open cubes as follows: place an inner cube with edge length $n^{-1/d}$ at the center of the tile and place several outer cubes each with edge length $\frac{1}{2}\, n^{-1/d}$ around the cube at the center as shown in Figure 5-5. Let F_d denote the number of outer cubes. The volumes of the inner cube and each of the outer cubes are n^{-1} and $2^{-d}\, n^{-1}$, respectively.

For $n \in \mathbb{N}$ and $m \in \{1, 2, \ldots, M_n\}$, consider the tile m when the algorithm is run with Poisson(n) samples. Let $I_{n,m}$ denote the indicator random variable for the event that the center cube of this tile contains no samples, whereas every outer cube contains at least $k+1$ samples, in tile m.

The probability that the inner cube contains no samples is $e^{-1/\mu(\mathcal{X}_{\text{free}})}$. The probability that an outer cube contains at least $k+1$ samples is $1 - \mathbb{P}\big(\{\text{Poisson}(2^{-d}/\mu(\mathcal{X}_{\text{free}})) \geq k+1\}\big) = 1 - \mathbb{P}(\{\text{Poisson}(2^{-d}/\mu(\mathcal{X}_{\text{free}})) \leq k\}) = 1 - \frac{\Gamma(k+1, 2^{-d}/\mu(\mathcal{X}_{\text{free}}))}{k!}$, where $\Gamma(\cdot,\cdot)$

is the incomplete gamma function (Abramowitz and Stegun, 1964). Then, noting that the cubes in a given tile are disjoint and using the independence property of the Poisson process (see Lemma 2.8),

$$\mathbb{E}\left[I_{n,m}\right] \;=\; e^{-1/\mu(\mathcal{X}_{\text{free}})}\left(1 - \frac{\Gamma(k+1, 2^{-d}/\mu(\mathcal{X}_{\text{free}}))}{k!}\right)^{F_d} \;>\; 0,$$

which is a constant independent of n; denote this constant by α.

Let $G_n = (V_n, E_n)$ denote the graph returned by the k-nearest PRM algorithm by the end of Poisson(n) iterations. Observe that if $I_{n,m} = 1$, then there is no edge of G_n crossing the cube of side length $\frac{1}{2}\, n^{-1/d}$ that is centered at the center of the inner cube in tile m (shown as the white cube in Figure 5-6). To prove this claim, note the following two facts. First, no point that is outside of the cubes can have an edge that crosses the inner cube. Second, no point in one of the outer cubes has a edge that has length greater than $\frac{\sqrt{d}}{2}\, i^{-1/d}$. Thus, no edge can cross the white cube illustrated in Figure 5-6.

Let σ_n denote the path in G_n that is closest to σ^* in terms of the bounded variation norm. Let $U_n := \|\sigma_n - \sigma^*\|_{\text{BV}}$, where $\|\cdot\|_{\text{BV}}$ denotes the bounded variation norm. Notice that $U_n \geq \frac{1}{2}\, n^{-1/d} \sum_{m=1}^{M_n} I_{n,m} = \frac{1}{2}\, n^{-1/d}\, M_n\, I_{n,1} = \frac{s^*}{4} I_{n,1}$. Then,

$$\mathbb{E}\left[\limsup_{n\to\infty} U_n\right] \;\geq\; \limsup_{n\to\infty} \mathbb{E}\left[U_n\right] \;\geq\; \limsup_{n\to\infty} \frac{s^*}{4}\, \mathbb{E}\left[I_{n,m}\right] \;\geq\; \frac{\alpha\, s^*}{4} \;>\; 0,$$

where the first inequality follows from Fatou's lemma (Resnick, 1999). This implies $\mathbb{P}(\{\limsup_{n\to\infty} U_n > 0\}) > 0$. Since $U_i > 0$ implies $Y_n > c^*$ surely,

$$\mathbb{P}\left(\left\{\limsup_{n\to\infty} Y_n^{k\text{PRM}} > c^*\right\}\right) \geq \mathbb{P}\left(\{\limsup_{n\to\infty} U_n > 0\}\right) > 0.$$

Hence, we establish that $\mathbb{P}\left(\left\{\limsup_{n\to\infty} Y_n^{k\text{PRM}} = c^*\right\}\right) < 1$. In fact, by Proposition 5.11, $\mathbb{P}\left(\left\{\limsup_{n\to\infty} Y_n^{k\text{PRM}} = c^*\right\}\right) = 0$. ■

5.3.5 Non-optimality of a Class of Variable-radius PRMs

In this section, we provide a proof of Theorem 5.18, which shows that the lack of asymptotic optimality in any variable radius PRM with connection radius $r(n) \leq \gamma n^{-1/d}$ for some $\gamma > 0$, where d is the dimensionality of the configuration space.

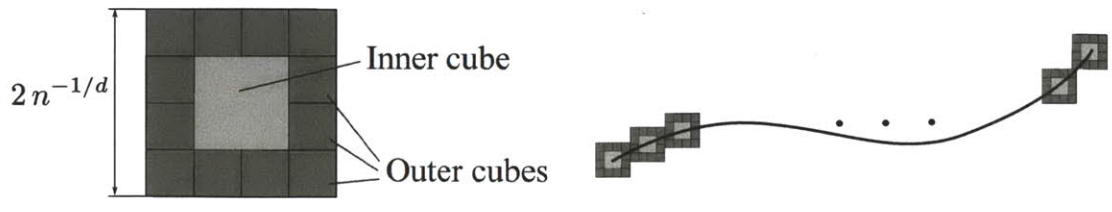


Figure 5-5: An illustration of the tiles covering an optimal path. A single tile is shown in the left; a tiling of the optimal path σ^* is shown on the right.

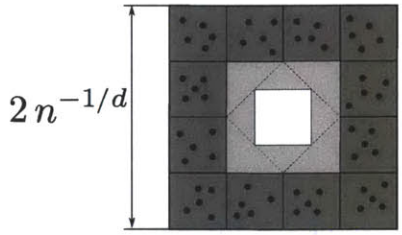


Figure 5-6: The event that the inner cube contains no points and each outer cube contains at least k points of the point process is illustrated. The cube of side length $\frac{1}{2}\,n^{-1/d}$ is shown in white.

Proof of Theorem 5.18 Let σ^* denote a path that is a robust solution to the optimality problem. Let n denote the number of samples that the algorithm is run with. For all n, construct a set $B_n = \{B_{n,1}, B_{n,2}, \ldots, B_{n,M_n}\}$ of openly disjoint balls as follows. Each ball in B_n has radius $r_n = \gamma\, n^{-1/d}$, and lies entirely inside $\mathcal{X}_{\text{free}}$. Furthermore, the balls in B_n "tile" σ^* such that the center of each ball lies on σ^* (see Figure 5-7). Let M_n denote the maximum number of balls, $\bar{s}$ denote the length of the portion of σ^* that lies within the δ-interior of $\mathcal{X}_{\text{free}}$, and $n_0 \in \mathbb{N}$ denote the number for which $r_n \leq \delta$ for all $n \geq n_0$.

Then, for all $n \geq n_0$,

$$M_n \geq \frac{\bar{s}}{2\,\gamma\,\left(\frac{1}{n}\right)^{1/d}} = \frac{\bar{s}}{2\,\gamma}\,n^{1/d}.$$

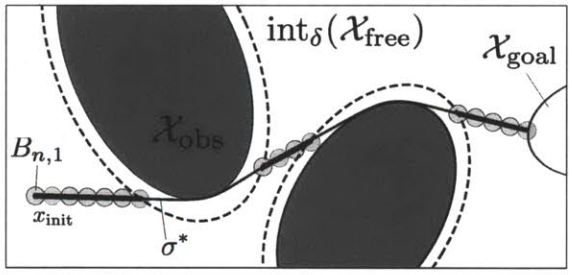


Figure 5-7: An illustration of the covering of the optimal path, σ^*, with openly disjoint balls. The balls cover only a portion of σ^* that lies within the δ-interior of $\mathcal{X}_{\text{free}}$.

Indicate the graph returned by this sPRM algorithm as $G_n = (V_n, E_n)$. Denote the event that the ball $B_{n,m}$ contains no vertex in V_n by $A_{n,m}$. Denote the indicator random variable for the event $A_{n,m}$ by $I_{n,m}$, *i.e.*, $I_{n,m} = 1$ when $A_{n,m}$ holds and $I_{n,m} = 0$ otherwise. Then, for all $n \geq n_0$,

$$\mathbb{E}[I_{n,m}] = \mathbb{P}(A_{n,m}) = \left(1 - \frac{\mu(B_{n,m})}{\mu(\mathcal{X}_{\text{free}})}\right)^n = \left(1 - \frac{\zeta_d\,\gamma^d}{\mu(\mathcal{X}_{\text{free}})}\,\frac{1}{n}\right)^n$$

Let N_n be the random variable that denotes the total number of balls in B_n that

contain no vertex in V_n, *i.e.*, $N_n = \sum_{m=1}^{M_n} I_{n,m}$. Then, for all $n \geq n_0$,

$$\mathbb{E}[N_n] \;=\; \mathbb{E}\left[\sum\nolimits_{m=1}^{M_n} I_{n,m}\right] \;=\; \sum_{m=1}^{M_n} \mathbb{E}[I_{n,m}] \;=\; M_n\, \mathbb{E}[I_{n,1}] \;\geq\; \frac{\bar{s}}{2\,\gamma}\, n^{1/d} \left(1 - \frac{\zeta_d\, \gamma^d}{\mu(\mathcal{X}_{\text{free}})} \frac{1}{n}\right)^n.$$

Consider a ball $B_{n,m}$ that contains no vertices of this sPRM algorithm. Then, no edges of the graph returned by this algorithm cross the ball of radius $\frac{\sqrt{3}}{2} r_n$ centered at the center of $B_{n,m}$. See Figure 5-8.

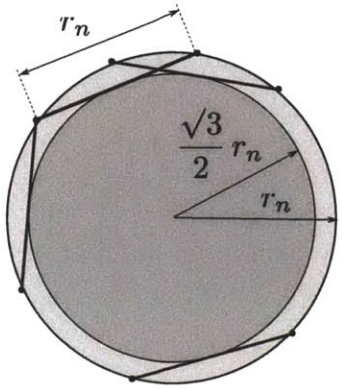


Figure 5-8: If the outer ball does not contain vertices of the PRM graph, then no edge of the graph corresponds to a path crossing the inner ball.

Let P_n denote the (finite) set of all acyclic paths that reach the goal region in the graph returned by this sPRM algorithm when the algorithm is run with n samples. Let U_n denote the total variation of the path that is closest to σ^* among all paths in P_n, *i.e.*, $U_n := \min_{\sigma_n \in P_n} \|\sigma_n - \sigma^*\|_{\text{BV}}$. Then,

$$\mathbb{E}[U_n] \;\geq\; \mathbb{E}\left[\gamma \left(\frac{1}{n}\right)^{1/d} N_n\right] \;\geq\; \frac{\bar{s}}{2} \left(1 - \frac{\zeta_d\, \gamma^d}{\mu(\mathcal{X}_{\text{free}})} \frac{1}{n}\right)^n.$$

Taking the limit superior of both sides, the following inequality can be established:

$$\begin{aligned} \mathbb{E}\left[\limsup_{n\to\infty} U_n\right] \;&\geq\; \limsup_{n\to\infty} \mathbb{E}\left[U_n\right] \\ &\geq\; \limsup_{n\to\infty} \frac{\bar{s}}{2} \left(1 - \frac{\zeta_d\, \gamma^d}{\mu(\mathcal{X}_{\text{free}})} \frac{1}{n}\right)^n \;=\; \frac{\bar{s}}{2}\, e^{-\frac{\zeta_d\, \gamma^d}{\mu(\mathcal{X}_{\text{free}})}} > 0, \end{aligned}$$

where the first inequality follows from Fatou's lemma (Resnick, 1999). Hence, we have $\mathbb{P}(\{\limsup_{n\to\infty} U_n > 0\}) > 0$, which implies that $\mathbb{P}\left(\left\{\limsup_{n\to\infty} Y_n^{ALG} > c^*\right\}\right) > 0$. That is, $\mathbb{P}\left(\left\{\limsup_{n\to\infty} Y_n^{ALG} = c^*\right\}\right) < 1$. In fact, $\mathbb{P}\left(\left\{\limsup_{n\to\infty} Y_n^{ALG} = c^*\right\}\right) = 0$ by the Kolmogorov zero-one law (see Proposition 5.11). ■

5.3.6 Non-optimality of RRT

In this section, we provide a proof of Theorem 5.19, thus establishing that the RRT algorithm is not asymptotically optimal. In fact, our proof shows directly, *i.e.*, without using Kolmogorov's zero-one law (Proposition 5.11), that the RRT algorithm converges to an optimal solution with probability zero. Along the way, we provide a number of results which were previously unknown, including the expected length of the longest path in the RRT.

For simplicity, the theorem will be proven assuming that (i) the environment contains no obstacles, *i.e.*, $\mathcal{X}_{\text{free}} = [0,1]^d$, and (ii) the parameter η of the steering procedure is set large enough, *e.g.*, $\eta \geq \text{diam}\,(\mathcal{X}_{\text{free}}) = \sqrt{d}$. One one hand, considering this case is enough to prove that the RRT algorithm is not asymptotically optimal, as it demonstrates a case for which the RRT algorithm fails to converge to an optimal solution, although the problem instance is clearly robustly optimal. On the other hand, these assumptions are not essential, and the claims extend to the more general case, but the technical details of the proof are considerably more complicated.

The proof can be outlined as follows. Order the vertices in the RRT according to the iteration at which they are added to the tree. The set of vertices that contains the k-th child of the root along with all its descendants in the tree is called the k-th branch of the tree. First, it is shown that a necessary condition for the asymptotic optimality of RRT is that infinitely many branches of the tree contain vertices outside a small ball centered at the initial condition. Then, the RRT algorithm is shown to violate this condition, with probability one.

A necessary condition: First, we provide a necessary condition for the RRT algorithm to be asymptotically optimal.

Lemma 5.41 *Let $0 < R < \inf_{y \in \mathcal{X}_{\text{goal}}} |y - x_{\text{init}}|$, where $|\cdot|$ denotes the usual Euclidean norm. The event $\{\lim_{N\to\infty} Y_n^{\text{RRT}} = c^*\}$ occurs only if the k-th branch of the RRT contains vertices outside the R-ball centered at x_{init} for infinitely many k.*

Proof Let $\{x_1, x_2, \dots\}$ denote the set of children to the root vertex in the order they are added to the tree. Let $\Gamma(x_k)$ denote the optimal cost of a path starting from the root vertex, passing through x_k, and reaching the goal region. By our assumption that the measure of the set of all points that are on the optimal path is zero (see Assumption 27 and Lemma 28), the probability that $\Gamma(x_k) = c^*$ is zero for all $k \in \mathbb{N}$. Hence,

$$\mathbb{P}\Big(\bigcup\nolimits_{k\in\mathbb{N}} \{\Gamma(x_k) = c^*\}\Big) \;\leq\; \sum_{k=1}^{\infty} \mathbb{P}\big(\{\Gamma(x_k) = c^*\}\big) = 0.$$

Let A_k denote the event that at least one vertex in the k-th branch of the tree is outside the ball of radius R centered at x_{init} in some iteration of the RRT algorithm. Consider the case when the event $\{\limsup_{k\to\infty} A_k\}$ does not occur and the events $\{\Gamma(x_k) > c^*\}$ occur for all $k \in \mathbb{N}$. Then, A_k occurs for only finitely many k. Let K denote the largest number such that A_K occurs. Then, the cost of the best path in the tree is at least $\sup\{\Gamma(x_k) \,|\, k \in \{1, 2, \dots, K\}\}$, which is strictly larger than c^*,

since $\{\Gamma(x_k) > c^*\}$ for all finite k. Thus, $\lim_{n\to\infty} Y_n^{\text{RRT}} > c^*$ must hold. That is, we have argued that

$$\Big(\limsup_{k\to\infty} A_k\Big)^c \cap \Big(\bigcap_{k\in\mathbb{N}} \{\Gamma(x_k) > c^*\}\Big) \subseteq \Big\{\lim_{n\to\infty} Y_n^{\text{RRT}} > c^*\Big\}.$$

Taking the complement of both sides and using monotonicity of probability measures,

$$\begin{aligned}
\mathbb{P}\Big(\big\{\lim_{n\to\infty} Y_n^{\text{RRT}} = c^*\big\}\Big) &\leq \mathbb{P}\Big(\big(\limsup_{k\to\infty} A_k\big) \cup \big(\bigcup_{k\in\mathbb{N}} \{\Gamma(x_k) = c^*\}\big)\Big), \\
&\leq \mathbb{P}\Big(\limsup_{k\to\infty} A_k\Big) + \mathbb{P}\Big(\bigcup_{k\in\mathbb{N}} \{\Gamma(x_k) = c^*\}\Big),
\end{aligned}$$

where the last inequality follows from the union bound. The lemma follows from the fact that the last term in the right hand side is equal to zero as shown above. ■

Length of the first path in a branch: The following result provides a useful characterization of the RRT structure.

Lemma 5.42 *Let $U = \{X_1, X_2, \ldots, X_n\}$ be a set of independently sampled and uniformly distributed points in the d-dimensional unit cube, $[0,1]^d$. Let X_{n+1} be a point that is sampled independently from all the other points according to the uniform distribution on $[0,1]^d$. Then, the probability that among all points in U the point X_i is the one that is closest to X_{n+1} is $1/n$, for all $i \in \{1, 2, \ldots, n\}$. Moreover, the expected distance from X_{n+1} to its nearest neighbor in U is $n^{-1/d}$.*

Proof Since the probability distribution is uniform, the probability that X_{n+1} is closest to X_i is the same for all $i \in \{1, 2, \ldots, n\}$, which implies that this probability is equal to $1/n$. The expected distance to the closest point in U is an application of the order statistics of the uniform distribution. ■

An immediate consequence of this result is that each vertex of the RRT has unbounded degree, almost surely, as the number of samples approaches infinity.

One can also define a notion of infinite paths in the RRT, as follows. Let Λ be the set of infinite sequences of natural numbers $\alpha = (\alpha_1, \alpha_2, \ldots)$. For any $i \in \mathbb{N}$, let $\pi_i : \Sigma \to \mathbb{N}^i, (\alpha_1, \alpha_2, \ldots, \alpha_i, \ldots) \mapsto (\alpha_1, \alpha_2, \ldots, \alpha_i)$, be a function returning the prefix of length i of an infinite sequence in Λ. The lexicographic ordering of Λ is such that, given $\alpha, \beta \in \Sigma$, $\alpha \leq \beta$ if and only if there exists $j \in \mathbb{N}$ such that $\alpha_i = \beta_i$ for all $i \in \mathbb{N}$, $i \leq j - 1$, and $\alpha_j \leq \beta_j$. This is a total ordering of Λ, since $\mathbb{N}$ is a totally ordered set. Given $\alpha \in \Lambda$ and $i \in \mathbb{N}$, let $\mathcal{L}_{\pi_i(\alpha)}$ be the sum of the distances from the root vertex x_{init} to its α_1-th child, from this vertex to its α_2-th child, etc., for a total of i terms. Because of Lemma 5.42, this construction is well defined, almost surely, for a sufficiently large number of samples. For any infinite sequence $\alpha \in \Lambda$, let $\mathcal{L}_\alpha = \lim_{i\to+\infty} \mathcal{L}_{\pi_i(\alpha)}$; the limit exists since $\mathcal{L}_{\pi_i(\alpha)}$ is non-decreasing in i.

Consider infinite strings of the form $\mathbf{k} = (k, 1, 1, \ldots)$, $k \in \mathbb{N}$, and introduce the shorthand $\mathcal{L}_{\mathbf{k}} := \mathcal{L}_{(k,1,1,\ldots)}$. The following lemma shows that, for any $k \in \mathbb{N}$, $\mathcal{L}_{\mathbf{k}}$ has finite expectation, which immediately implies that $\mathcal{L}_{\mathbf{k}}$ takes only finite values with

probability one. The lemma also provides a couple of other useful properties of $\mathcal{L}_{\mathbf{k}}$, which will be used later on.

Lemma 5.43 *The expected value* $\mathbb{E}[\mathcal{L}_{\mathbf{k}}]$ *is non-negative and finite, and monotonically non-increasing, in the sense that* $\mathbb{E}[\mathcal{L}_{\mathbf{k+1}}] \leq \mathbb{E}[\mathcal{L}_{\mathbf{k}}]$, *for any* $k \in \mathbb{N}$. *Moreover,* $\lim_{k\to\infty} \mathbb{E}[\mathcal{L}_{\mathbf{k}}] = 0$.

Proof Under the simplifying assumptions that there are no obstacles in the unit cube and η is large enough, the vertex set V_n^{RRT} of the graph maintained by the RRT algorithm is precisely the first n samples and each new sample is connected to its nearest neighbor in V_n^{RRT}.

Define Z_i as a random variable describing the contribution to $\mathcal{L}_{\mathbf{1}}$ realized at iteration i; in other words, Z_i is the distance of the i-th sample to its nearest neighbor among the first $i-1$ samples if the i-th sample is on the path used in computing $\mathcal{L}_{\mathbf{1}}$, and zero otherwise. Then, using Lemma 5.42,

$$\mathbb{E}[\mathcal{L}_{\mathbf{1}}] = \mathbb{E}\left[\sum_{i=1}^{\infty} Z_i\right] = \sum_{i=1}^{\infty} \mathbb{E}[Z_i] = \sum_{i=1}^{\infty} i^{-1/d}\, i^{-1} = \texttt{Zeta}(1+1/d),$$

where the second equality follows from the monotone convergence theorem and `Zeta` is the Riemann zeta function. Since $\texttt{Zeta}(y)$ is finite for any $y > 1$, $\mathbb{E}[\mathcal{L}_{\mathbf{1}}]$ is a finite number for all $d \in \mathbb{N}$.

Let N_k be the iteration at which the first sample contributing to $\mathcal{L}_k$ is generated. Then, an argument similar to the one given above yields

$$\mathbb{E}[\mathcal{L}_{\mathbf{k+1}}] = \sum_{i=N_k+1}^{\infty} i^{-(1+1/d)} = \mathbb{E}[\mathcal{L}_{\mathbf{1}}] - \sum_{i=1}^{N_k} i^{-(1+1/d)}.$$

Then, clearly, $\mathbb{E}[\mathcal{L}_{\mathbf{k+1}}] < \mathbb{E}[\mathcal{L}_{\mathbf{k}}]$ for all $k \in \mathbb{N}$. Moreover, since $N_k \geq k$, it is the case that $\lim_{k\to\infty} \mathbb{E}[\mathcal{L}_{\mathbf{k}}] = 0$. ■

Length of the longest path in a branch: Given $k \in \mathbb{N}$, and the sequence $\mathbf{k} = (k, 1, 1, \ldots)$, the quantity $\sup_{\alpha\geq\mathbf{k}} \mathcal{L}_\alpha$ is an upper bound on the length of any path in the k-th branch of the RRT, or in any of the following branches. The next result bounds the probability that this quantity is very large.

Lemma 5.44 *For any* $\epsilon > 0$,

$$\mathbb{P}\left(\left\{\sup_{\alpha\geq\mathbf{k}} \mathcal{L}_\alpha > \epsilon\right\}\right) \leq \frac{\mathbb{E}[\mathcal{L}_{\mathbf{k}}]}{\epsilon}.$$

First, we state and prove the following intermediate result.

Lemma 5.45 $\mathbb{E}[\mathcal{L}_\alpha] \leq \mathbb{E}[\mathcal{L}_{\mathbf{k}}]$, *for all* $\alpha \geq \mathbf{k}$.

Proof The proof is by induction. Since $\alpha \geq \mathbf{k}$, then $\pi_1(\alpha) \geq k$, and Lemma 5.43 implies that $\mathbb{E}[\mathcal{L}_{(\pi_1(\alpha),1,1,\ldots)}] \leq \mathbb{E}[\mathcal{L}_{\mathbf{k}}]$. Moreover, it is also the case that, for any $i \in \mathbb{N}$ (and some abuse of notation), $\mathbb{E}[\mathcal{L}_{(\pi_{i+1}(\alpha),1,1,\ldots)}] \leq \mathbb{E}[\mathcal{L}_{(\pi_i(\alpha),1,1,\ldots)}]$, by a similar argument considering a tree rooted at the last vertex reached by the finite path $\pi_i(\alpha)$. Since $(\pi_{i+1}(\alpha), 1, 1, \ldots) \geq (\pi_i(\alpha), 1, 1, \ldots) \geq (k, 1, 1, \ldots)$, the result follows. ■

Proof of Lemma 5.44 Define the random variable $\bar{\alpha} := \inf\{\alpha \geq \mathbf{k} \,|\, \mathcal{L}_\alpha > \epsilon\}$, and set $\bar{\alpha} := \mathbf{k}$ if $\mathcal{L}_\alpha \leq \epsilon$ for all $\alpha \geq \mathbf{k}$. Note that $\bar{\alpha} \geq \mathbf{k}$ holds surely. Hence, by Lemma 5.45, $\mathbb{E}[\mathcal{L}_{\bar{\alpha}}] \leq \mathbb{E}[\mathcal{L}_{\mathbf{k}}]$. Let I_ϵ be the indicator random variable for the event $S_\epsilon := \{\sup_{\alpha \geq \mathbf{k}} \mathcal{L}_\alpha > \epsilon\}$. Then,

$$\mathbb{E}[\mathcal{L}_{\mathbf{k}}] \geq \mathbb{E}[\mathcal{L}_{\bar{\alpha}}] = \mathbb{E}[\mathcal{L}_{\bar{\alpha}} I_\epsilon] + \mathbb{E}[\mathcal{L}_{\bar{\alpha}}(1 - I_\epsilon)] \geq \epsilon\, \mathbb{P}(S_\epsilon),$$

where the last inequality follows from the fact that $\mathcal{L}_{\bar{\alpha}}$ is at least ϵ whenever the event S_ϵ occurs. ■

A useful corollary of Lemmas 5.43 and 5.44 is the following.

Corollary 5.46 *For any* $\epsilon > 0$,

$$\lim_{k \to \infty} \mathbb{P}(\{\sup_{\alpha \geq \mathbf{k}} \mathcal{L}_\alpha > \epsilon\}) = 0.$$

Violation of the necessary condition: Finally, the proof of the theorem follows showing that the necessary condition in Lemma 5.41 is violated.

Proof of Theorem 5.19 Recall from Lemma 5.41 that a necessary condition for asymptotic optimality is that the k-th branch of the RRT contains vertices outside the R-ball centered at x_{init} for infinitely many k, where $0 < R < \inf_{y \in \mathcal{X}_{\text{goal}}} |y - x_{\text{init}}|$. Clearly, the latter event can occur only if longest path in the k-th branch of the RRT is longer than R for infinitely many k. That is,

$$\mathbb{P}\left(\left\{\lim_{n \to \infty} Y_n^{\text{RRT}} = c^*\right\}\right) \leq \mathbb{P}\left(\limsup_{k \to \infty} \left\{\sup_{\alpha \geq \mathbf{k}} \mathcal{L}_\alpha > R\right\}\right).$$

The event on the right hand side is monotonic in the sense that $\{\sup_{\alpha > \mathbf{k}+1} \mathcal{L}_\alpha > R\} \supseteq \{\sup_{\alpha \geq \mathbf{k}} \mathcal{L}_\alpha > R\}$ for all $k \in \mathbb{N}$. Hence, $\lim_{k \to \infty}\{\sup_{\alpha \geq \mathbf{k}} \mathcal{L}_\alpha > R\}$ exists. In particular,

$$\begin{aligned}&\mathbb{P}\big(\limsup_{k \to \infty}\{\sup_{\alpha \geq \mathbf{k}} \mathcal{L}_\alpha > R\}\big)\\ &\qquad = \mathbb{P}\big(\lim_{k \to \infty}\{\sup_{\alpha \geq \mathbf{k}} \mathcal{L}_\alpha > R\}\big) = \lim_{k \to \infty} \mathbb{P}\big(\{\sup_{\alpha \geq \mathbf{k}} \mathcal{L}_\alpha > R\}\big),\end{aligned}$$

where the last equality follows from the continuity of probability measures. Since $\lim_{k \to \infty} \mathbb{P}\left(\{\sup_{\alpha \geq \mathbf{k}} \mathcal{L}_\alpha > R\}\right) = 0$ for all $R > 0$ by Corollary 5.46, we conclude that $\mathbb{P}(\{\lim_{n \to \infty} Y_n^{\text{RRT}} = c^*\}) = 0$, which completes the proof. ■

5.3.7 Asymptotic Optimality of PRM*

In this section, we prove Theorem 5.20, thus establishing the asymptotic optimality of the the PRM* algorithm. The proof is long and technical. In what follows, we first outline the proof; subsequently, we provide the details.

Outline of the proof: Let σ^* denote a robust optimal path. By definition, σ^* has weak clearance. First, define a sequence $\{\delta_n\}_{n\in\mathbb{N}}$ such that $\delta_n > 0$ for all $n \in \mathbb{N}$ and δ_n approaches zero as n approaches infinity. Construct a sequence $\{\sigma_n\}_{n\in\mathbb{N}}$ of paths such that σ_n has strong δ_n-clearance for all $n \in \mathbb{N}$ and σ_n converges to σ^* as n approaches infinity.

Second, define a sequence $\{q_n\}_{n\in\mathbb{N}}$. For all $n \in \mathbb{N}$, construct a set $B_n = \{B_{n,1}, B_{n,2}, \ldots, B_{n,M}\}$ of overlapping balls, each with radius q_n, that collectively "cover" the path σ_n. See Figures 5-9 and 5-10. Let $x_m \in B_{n,m}$ and $x_{m+1} \in B_{n,m+1}$ be any two points from two consecutive balls in B_n. Construct B_n such that (i) x_m and x_{m+1} have distance no more than the connection radius $r(n)$ and (ii) the straight path connecting x_m and x_{m+1} lies entirely within the obstacle free space. These requirements can be satisfied by setting δ_n and q_n to certain constant fractions of $r(n)$.

Let A_n denote the event that each ball in B_n contains at least one vertex of the graph returned by the PRM* algorithm, when the algorithm is run with n samples. Third, show that A_n occurs for all large n, with probability one. Clearly, in this case, the PRM* algorithm will connect the vertices in consecutive balls with an edge, and any path formed in this way will be collision-free.

Finally, show that any sequence of paths generated in this way converges to the optimal path σ^*. Using the robustness of σ^*, show that the cost of the best path in the graph returned by the PRM* algorithm converges to $c(\sigma^*)$ almost surely.

Construction of the sequence $\{\sigma_n\}_{n\in\mathbb{N}}$ of paths: The following lemma establishes a connection between the notions of strong and weak clearance.

Lemma 5.47 *Let σ^* be a path be a path that has weak clearance. Let $\{\delta_n\}_{n\in\mathbb{N}}$ be a sequence of real numbers such that $\lim_{n\to\infty}\delta_n = 0$ and $0 \leq \delta_n \leq \delta$ for all $n \in \mathbb{N}$. Then, there exists a sequence $\{\sigma_n\}_{n\in\mathbb{N}}$ of paths such that $\lim_{n\to\infty}\sigma_n = \sigma^*$ and σ_n has strong δ_n-clearance for all $n \in \mathbb{N}$.*

Proof First, define a sequence $\{\mathcal{X}_n\}_{n\in\mathbb{N}}$ of subsets of $\mathcal{X}_{\text{free}}$ such that $\mathcal{X}_n$ is the closure of the δ_n-interior of $\mathcal{X}_{\text{free}}$, *i.e.*,

$$\mathcal{X}_n := \text{cl}(\text{int}_{\delta_n}(\mathcal{X}_{\text{free}}))$$

for all $n \in \mathbb{N}$. Note that, by definition, (i) $\mathcal{X}_n$ are closed subsets of $\mathcal{X}_{\text{free}}$, and (ii) any point $\mathcal{X}_n$ has distance at least δ_n to any point in the obstacle set $\mathcal{X}_{\text{obs}}$.

Then, construct the sequence $\{\sigma_n\}_{n\in\mathbb{N}}$ of paths, where $\sigma_n \in \Sigma_{\mathcal{X}_n}$, as follows. Let $\psi : [0,1] \to \Sigma_{\text{free}}$ denote the homotopy with $\psi(0) = \sigma^*$; the existence of ψ is

guaranteed by weak clearance of σ^*. Define

$$\alpha_n := \max_{\alpha \in [0,1]} \{\alpha : \psi(\alpha) \in \Sigma_{\mathcal{X}_n}\} \quad \text{and} \quad \sigma_n := \psi(\alpha_n).$$

Since $\Sigma_{\mathcal{X}_n}$ is closed, the maximum in the definition of α_n is attained. Moreover, since $\psi(1)$ has strong δ-clearance and $\delta_n \leq \delta$, $\sigma_n \in \Sigma_{\mathcal{X}_n}$, which implies the strong δ_n-clearance of σ_n.

Clearly, $\bigcup_{n\in\mathbb{N}} \mathcal{X}_n = \mathcal{X}_{\text{free}}$, since $\lim_{n\to\infty} \delta_n = 0$. Also, by the weak clearance of σ^*, for any $\alpha \in (0, 1]$, there exists some $\delta_\alpha \in (0, \delta]$ such that $\psi(\alpha)$ has strong δ_α-clearance. Then, $\lim_{n\to\infty} \alpha_n = 0$, which implies $\lim_{n\to\infty} \sigma_n = \sigma^*$. ■

Recall that the connection radius of the PRM* algorithm was defined as

$$r_n = \gamma_{\text{PRM}} \left(\frac{\log n}{n}\right)^{1/d} > 2(1 + 1/d)^{1/d} \left(\frac{\mu(X_{\text{free}})}{\zeta_d}\right)^{1/d} \left(\frac{\log n}{n}\right)^{1/d}$$

(see Algorithm 4 and the definition of the `Near` procedure in Section 4.1). Let θ_1 be a small positive constant; the precise value of θ_1 will be provided shortly in the proof of Lemma 5.49. Define

$$\delta_n := \min\left\{\delta, \frac{1+\theta_1}{2+\theta_1} r_n\right\}, \qquad \text{for all } n \in \mathbb{N}.$$

By definition, $0 \leq \delta_n \leq \delta$ holds. Moreover, $\lim_{n\to\infty} \delta_n = 0$, since $\lim_{n\to\infty} r_n = 0$. Then, by Lemma 5.47, there exists a sequence $\{\sigma_n\}_{n\in\mathbb{N}}$ of paths such that $\lim_{n\to\infty} \sigma_n = \sigma^*$ and σ_n has strong δ_n-clearance for all $n \in \mathbb{N}$.

Construction of the sequence $\{B_n\}_{n\in\mathbb{N}}$ of sets of balls: First, a construction of a finite set of balls that collectively "cover" a path σ_n is provided. The construction is illustrated in Figure 5-9.

Definition 5.48 (Covering balls) *Given a path $\sigma_n : [0, 1] \to \mathcal{X}$, and the real numbers $q_n, l_n \in \mathbb{R}_{>0}$, the set* `CoveringBalls`$(\sigma_n, q_n, l_n)$ *is defined as a set $\{B_{n,1}, B_{n,2}, \ldots, B_{n,M_n}\}$ of M_n balls of radius q_n such that $B_{n,m}$ is centered at $\sigma(\tau_m)$, and*

- *the center of $B_{n,1}$ is $\sigma(0)$,* i.e., *$\tau_1 = 0$,*
- *the centers of two consecutive balls are exactly l_n apart,* i.e., *$\tau_m := \min\{\tau \in [\tau_{m-1}, 1] : |\sigma(\tau) - \sigma(\tau_{m-1})| \geq l_n\}$ for all $m \in \{2, 3, \ldots, M_n\}$,*
- *and $M - 1$ is the largest number of balls that can be generated in this manner while the center of the last ball, B_{n,M_n} is $\sigma(1)$,* i.e., *$\tau_{M_n} = 1$.*

For each $n \in \mathbb{N}$, define

$$q_n := \frac{\delta_n}{1+\theta_1}.$$

Construct the set $B_n = \{B_{n,1}, B_{n,2}, \ldots, B_{n,M_n}\}$ of balls as $B_n :=$ `CoveringBalls`$(\sigma_n, q_n, \theta_1 q_n)$ using Definition 5.48 (see Figure 5-9). By construction, each ball in B_n has

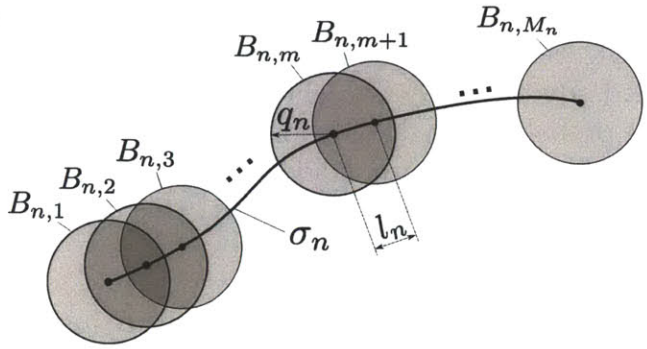


Figure 5-9: An illustration of the `CoveringBalls` construction. A set of balls that collectively cover the trajectory σ_n is shown. All balls have the same radius, q_n. The spacing between the centers of two consecutive balls is l_n.

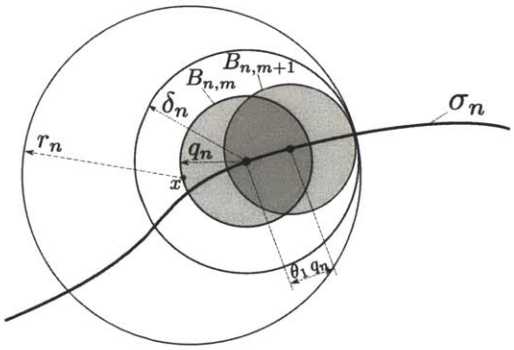


Figure 5-10: An illustration of the covering balls for PRM* algorithm. The δ_n-ball is guaranteed to be inside the obstacle-free space. The connection radius r_n is also shown as the radius of the connection ball centered at a vertex $x \in B_{n,m}$. The vertex x is connected to all other vertices that lie within the connection ball.

radius q_n and the centers of consecutive balls in B_n are $\theta_1 q_n$ apart (see Figure 5-10 for an illustration of covering balls with this set of parameters). The balls in B_n collectively cover the path σ_n.

The probability that each ball in B_n contains at least one vertex: Recall that $G_n^{\text{PRM}^*} = (V_n^{\text{PRM}^*}, E_n^{\text{PRM}^*})$ denotes the graph returned by the PRM* algorithm, when the algorithm is run with n samples. Let $A_{n,m}$ denote the event that the ball $B_{n,m}$ contains at least one vertex of the graph generated by the PRM* algorithm, *i.e.*, $A_{n,m} = \left\{B_{n,m} \cap V_n^{\text{PRM}^*} \neq \emptyset\right\}$. Let A_n denote the event that all balls in B_n contain at least one vertex of the PRM* graph, *i.e.*, $A_n = \bigcap_{m=1}^{M_n} A_{n,m}$.

Lemma 5.49 *If* $\gamma_{\text{PRM}} > 2\,(1 + 1/d)^{1/d} \left(\frac{\mu(X_{\text{free}})}{\zeta_d}\right)^{1/d}$, *then there exists a constant* $\theta_1 > 0$ *such that the event that every ball in* B_n *contains at least one vertex of the PRM* graph occurs for all large enough* n *with probability one,* i.e.,

$$\mathbb{P}\left(\liminf_{n\to\infty} A_n\right) = 1.$$

Proof The proof is based on a Borel-Cantelli argument which can be summarized as follows. Recall that A_n^c denotes the complement of A_n. First, the sum $\sum_{n=1}^{\infty} \mathbb{P}(A_n^c)$ is shown to be bounded. By the Borel-Cantelli lemma (Grimmett and Stirzaker, 2001), this implies that the probability that A_n holds infinitely often as n approaches infinity is zero. Hence, the probability that A_n holds infinitely often is one. In the rest of the proof, an upper bound on $\mathbb{P}(A_n)$ is computed, and this upper bound is shown to be summable.

First, compute a bound on the number of balls in B_n as follows. Let s_n denote the length of σ_n, *i.e.*, $s_n := \mathrm{TV}(\sigma_n)$. Recall that the balls in B_n were constructed such that the centers of two consecutive balls in B_n have distance $\theta_1 \, q_n$. The segment of σ_n that starts at the center of $B_{n,m}$ and ends at the center of $B_{n,m+1}$ has length at least $\theta_1 q_n$, except for the last segment, which has length less than or equal to $\theta_1 q_n$. Let $n_0 \in \mathbb{N}$ be the number such that $\delta_n < \delta$ for all $n \geq n_0$. Then, for all $n \geq n_0$,

$$\begin{aligned} \mathrm{card}\,(B_n) = M_n \;&\leq\; \frac{s_n}{\theta_1 q_n} = \frac{(1+\theta_1) s_n}{\theta_1 \delta_n} = \frac{(2+\theta_1)\, s_n}{\theta_1 \, r_n} \\ &=\; \frac{(2+\theta_1)\, s_n}{\theta_1 \, \gamma_{\mathrm{PRM}}} \left(\frac{n}{\log n} \right)^{1/d}. \end{aligned}$$

Second, compute the volume of a single ball in B_i as follows. Recall that $\mu(\cdot)$ denotes the usual Lebesgue measure, and ζ_d denotes the volume of a unit ball in the d-dimensional Euclidean space. For all $n \geq n_0$,

$$\begin{aligned} \mu(B_{n,m}) = \zeta_d \, q_n^d &= \zeta_d \left(\frac{\delta_n}{1+\theta_1} \right)^d \\ &= \zeta_d \left(\frac{r_n}{2+\theta_1} \right)^d = \zeta_d \left(\frac{\gamma_{\mathrm{PRM}}}{2+\theta_1} \right)^d \frac{\log n}{n} \end{aligned}$$

For all $n \geq I$, the probability that a single ball, say $B_{n,1}$, does not contain a vertex of the graph generated by the PRM* algorithm, when the algorithm is run with n samples, is

$$\begin{aligned} \mathbb{P}\left(A_{n,1}^c\right) &= \left(1 - \frac{\mu(B_{n,1})}{\mu(X_{\mathrm{free}})} \right)^n \\ &= \left(1 - \frac{\zeta_d}{\mu(X_{\mathrm{free}})} \left(\frac{\gamma_{\mathrm{PRM}}}{2+\theta_1} \right)^d \frac{\log n}{n} \right)^n \end{aligned}$$

Using the inequality $(1 - 1/f(n))^r \leq e^{-r/f(n)}$, the right-hand side can be bounded as

$$\mathbb{P}(A_{n,1}) \leq e^{-\frac{\zeta_d}{\mu(X_{\mathrm{free}})} \left(\frac{\gamma_{\mathrm{PRM}}}{2+\theta_1} \right)^d \log n} = n^{-\frac{\zeta_d}{\mu(X_{\mathrm{free}})} \left(\frac{\gamma_{\mathrm{PRM}}}{2+\theta_1} \right)^d}.$$

Hence,

$$\begin{aligned}
\mathbb{P}\left(A_n^c\right) = \mathbb{P}\left(\bigcup_{m=1}^{M_n} A_{n,m}^c\right) &\leq \sum_{m=1}^{m_n} \mathbb{P}\left(A_{n,m}^c\right) = M_n\, \mathbb{P}(A_{n,1}^c) \\
&\leq \frac{(2+\theta_1)s_n}{\theta_1\, \gamma_{\text{PRM}}} \left(\frac{n}{\log n}\right)^{1/d} i^{-\frac{\zeta_d}{\mu(X_{\text{free}})}\left(\frac{\gamma_{\text{PRM}}}{2+\theta_1}\right)^d} \\
&= \frac{(2+\theta_1)s_n}{\theta_1\, \gamma_{\text{PRM}}} \frac{1}{(\log n)^d}\, n^{-\left(\frac{\zeta_d}{\mu(X_{\text{free}})}\left(\frac{\gamma_{\text{PRM}}}{2+\theta_1}\right)^d - \frac{1}{d}\right)}
\end{aligned}$$

where the first inequality follows from the union bound.

Finally, $\sum_{n=1}^{\infty} \mathbb{P}(A_n^c) < \infty$ holds, if $\frac{\zeta_d}{\mu(X_{\text{free}})}\left(\frac{\gamma_{\text{PRM}}}{2+\theta_1}\right)^d - \frac{1}{d} > 1$, which can be satisfied for any $\gamma_{PRM} > 2(1+1/d)^{1/d}\left(\frac{\mu(X_{\text{free}})}{\zeta_d}\right)^{1/d}$ by appropriately choosing θ_1. Then, by the Borel-Cantelli lemma (Grimmett and Stirzaker, 2001), $\mathbb{P}(\limsup_{n\to\infty} A_n^c) = 0$, which implies $\mathbb{P}(\liminf_{n\to\infty} A_n) = 1$. ■

Connecting the vertices in subsequent balls in B_n: Let $Z_n := \{x_1, x_2, \ldots, x_{M_n}\}$ be any set of points such that $x_m \in B_{n,m}$ for each $m \in \{1, 2, \ldots, M_n\}$. The following lemma states that for all $n \in \mathbb{N}$ and all $m \in \{1, 2, \ldots, M_n - 1\}$, the distance between x_m and x_{m+1} is less than the connection radius, r_n, which implies that the PRM* algorithm will attempt to connect the two points x_m and x_{m+1} if they are in the vertex set of the PRM* algorithm.

Lemma 5.50 *If $x_{n,m} \in B_{n,m}$ and $x_{n,m+1} \in B_{n,m+1}$, then $|x_{n,m+1} - x_{n,m}| \leq r_n$, for all $n \in \mathbb{N}$ and all $m \in \{1, 2, \ldots, M_i - 1\}$, where $|\cdot|$ is the usual Euclidean norm.*

Proof Recall that each ball in B_n has radius $q_n = \frac{\delta_n}{(1+\theta_1)}$. Given any two points $x_m \in B_{n,m}$ and $x_{m+1} \in B_{n,m+1}$, all of the following hold: (i) x_m has distance q_n to the center of $B_{n,m}$, (ii) x_{m+1} has distance q_n to the center of $B_{n,m+1}$, and (iii) centers of $B_{n,m}$ and $B_{n,m+1}$ have distance $\theta_1\, q_n$ to each other. Then,

$$|x_{n,m+1} - x_{n,m}| \leq (2+\theta_1)\, q_n = \frac{2+\theta_1}{1+\theta_1}\, \delta_n \leq r_n,$$

where the first inequality is obtained by an application of the triangle inequality and the last inequality follows from the definition of $\delta_n = \min\{\delta, \frac{1+\theta_1}{2+\theta_1}\, r_n\}$. ■

By Lemma 5.50, conclude that the RPM* algorithm will attempt to connect any two vertices in consecutive balls in B_n. The next lemma shows that any such connection attempt will, in fact, be successful. That is, the path connecting $x_{n,m}$ and $x_{n,m+1}$ is collision-free for all $m \in \{1, 2, \ldots, M_n\}$.

Lemma 5.51 *For all $n \in \mathbb{N}$ and all $m \in \{1, 2, \ldots, M_n\}$, if $x_m \in B_{n,m}$ and $x_{m+1} \in B_{n,m+1}$, then the line segment connecting $x_{n,m}$ and $x_{n,m+1}$ lies in the obstacle-free space,* i.e.,

$$\alpha\, x_{n,m} + (1-\alpha)\, x_{n,m+1} \in X_{\text{free}}, \qquad \text{for all } \alpha \in [0,1].$$

Proof Recall that σ_n has strong δ_n-delta clearance and that the radius q_n of each ball in B_n was defined as $q_n = \frac{\delta_n}{1+\theta_1}$, where $\theta_1 > 0$ is a constant. Hence, any point along the trajectory σ_n has distance at least $(1+\theta_1)\, q_n$ to any point in the obstacle set. Let y_m and y_{m+1} denote the centers of the balls $B_{n,m}$ and $B_{n,m+1}$, respectively. Since $y_m = \sigma(\tau_m)$ and $y_{m+1} = \sigma(\tau_{m+1})$ for some τ_m and τ_{m+1}, y_m and y_{m+1} also have distance $(1+\theta_1)q_n$ to any point in the obstacle set.

Clearly, $|x_m - y_m| \leq q_n$. Moreover, the following inequality holds:

$$\begin{aligned} |x_{m+1} - y_m| \;&\leq |(x_m - y_{m+1}) + (y_{m+1} - y_m)| \\ &\leq \; |x_{m+1} - y_{m+1}| + |y_{m+1} - y_m| \;\leq\; q_n + \theta_1\, q_n = (1+\theta_1)\, q_n. \end{aligned}$$

where the second inequality follows from the triangle inequality and the third inequality follows from the construction of balls in B_n.

For any convex combination $x_\alpha := \alpha\, x_m + (1-\alpha)\, x_{m+1}$, where $\alpha \in [0,1]$, the distance between x_α and y_m can be bounded as follows:

$$\begin{aligned} \big|\big(\alpha\, x_m + (1+\alpha)\, x_{m+1}\big) - y_m\big| \;&=\; \big|\alpha\,(x_m - y_m) + (1+\alpha)\,(x_{m+1} - y_m)\big| \\ &=\; \alpha\,|x_m - y_m| + (1+\alpha)\,|x_{m+1} - y_m| \\ &=\; \alpha\, q_n + (1+\alpha)\,(1+q_n) \;\leq\; (1+\theta_1)\, q_n, \end{aligned}$$

where the second equality follows from the linearity of the norm. Hence, any point along the line segment connecting x_m and x_{m+1} has distance at most $(1+\theta_1)\, q_n$ to y_m. Since, y_m has distance at least $(1+\theta_1)q_n$ to any point in the obstacle set, the line segment connecting x_m and x_{m+1} is collision-free. ■

Convergence to the optimal path: Let P_n denote the set of all paths in the graph $G_n^{\text{PRM}^*} = (V_n^{\text{PRM}^*}, E_n^{\text{PRM}^*})$. Let σ'_n be the path that is closest to σ_n in terms of the bounded variation norm among all those paths in P_n, *i.e.*, $\sigma'_n := \min_{\sigma' \in P_n} \|\sigma' - \sigma_n\|_{\text{BV}}$. Note that the sequence $\{\sigma'_n\}_{n\in\mathbb{N}}$ is a random sequence of paths, since the graph $G_n^{\text{PRM}^*}$, hence the set P_n of paths is random. The following lemma states that the bounded variation distance between σ'_n and σ_n approaches to zero, with probability one.

Lemma 5.52 *The random variable* $\|\sigma'_n - \sigma_n\|_{\text{BV}}$ *converges to zero almost surely,* i.e.,

$$\mathbb{P}\left(\{\lim_{n\to\infty} \|\sigma'_n - \sigma_n\|_{\text{BV}} = 0\}\right) = 1.$$

Proof The proof of this lemma is based on a Borel-Cantelli argument. It is shown that $\sum_{n\in\mathbb{N}} \mathbb{P}(\|\sigma'_n - \sigma_n\|_{\text{BV}} > \epsilon)$ is finite for any $\epsilon > 0$, which implies that $\|\sigma'_n - \sigma_n\|_{\text{BV}}$ converges to zero almost surely by the Borel-Cantelli lemma (Grimmett and Stirzaker, 2001). This proof uses a Poissonization argument in one of the intermediate steps. That is, a particular result is shown to hold in the Poisson process described in Lemma 2.8. Subsequently, the result is de-Poissonized, *i.e.*, shown to hold also for the original process.

Fix some $\epsilon > 0$. Let $\alpha, \beta \in (0,1)$ be two constants, both independent of n. Recall that q_n is the radius of each ball in the set B_n of balls covering the path σ_n. Let

$I_{n,m}$ denote the indicator variable for the event that the ball $B_{n,m}$ has no point that is within a distance $\beta\, q_n$ from the center of $B_{n,m}$. For a more precise definition, let $\beta\, B_{n,m}$ denote the ball that is centered at the center of $B_{n,m}$ and has radius $\beta\, r_n$. Then,

$$I_{n,m} := \begin{cases} 1, & \text{if } (\beta\, B_{n,m}) \cap V^{\mathrm{PRM}^*} = \emptyset, \\ 0, & \text{otherwise.} \end{cases}$$

Let K_n denote the number of balls in B_n that do not contain a vertex that is within a $\beta\, q_n$ distance to the center of that particular ball, *i.e.*, $K_n := \sum_{m=1}^{M_n} I_{n,m}$.

Consider the event that $I_{n,m}$ holds for at most an α fraction of the balls in B_n, *i.e.*, $\{K_n \leq \alpha\, M_n\}$. This event is important for the following reason. Recall that the vertices in subsequent balls in B_n are connected by edges in $G_n^{\mathrm{PRM}^*}$ by Lemmas 5.50 and 5.51. If only at most an α fraction of the balls do not have a vertex that is less than a distance of $\beta\, r_n$ from their centers (hence, a $(1-\alpha)$ fraction have at least one vertex within a distance of $\beta\, r_n$ from their centers), *i.e.*, $\{K_n \leq \alpha\, M_n\}$ holds, then the bounded variation difference between σ'_n and σ_n is at most $(\sqrt{2}\,\alpha + \beta(1-\alpha))L \leq \sqrt{2}\,(\alpha+\beta)L$, where L is a finite bound on the length of all paths in $\{\sigma_n\}_{n\in\mathbb{N}}$, *i.e.*, $L := \sup_{n\in\mathbb{N}} \mathrm{TV}(\sigma_n)$. That is,

$$\{K_n \leq \alpha\, M_n\} \subseteq \left\{ \|\sigma'_n - \sigma_n\|_{\mathrm{BV}} \leq \sqrt{2}\,(\alpha+\beta)\, L \right\}$$

Taking the complement of both sides and using the monotonicity of probability measures,

$$\mathbb{P}\left(\left\{ \|\sigma'_n - \sigma_n\|_{\mathrm{BV}} > \sqrt{2}\,(\alpha+\beta)\, L \right\} \right) \leq \mathbb{P}\left(\{K_n \geq \alpha\, M_n\} \right).$$

In the rest of the proof, it is shown that the right hand side of the inequality above is summable for all small $\alpha, \beta > 0$, which implies that $\mathbb{P}\left(\{\|\sigma'_n - \sigma_n\|_{\mathrm{BV}} > \epsilon\}\right)$ is summable for all small $\epsilon > 0$.

For this purpose, the process that provides independent uniform samples from $\mathcal{X}_{\mathrm{free}}$ is approximated by an equivalent Poisson process described in Section 2.2. A more precise definition is given as follows. Let $\{X_1, X_2, \ldots, X_n\}$ denote the binomial point process corresponding to the `SampleFree` procedure. Let $\nu < 1$ be a constant independent of n. Recall that Poisson$(\nu\, n)$ denotes the Poisson random variable with intensity $\nu\, n$ (hence, mean value $\nu\, n$). Then, the process $\mathcal{P}_{\nu\, n} := \{X_1, X_2, \ldots, X_{\mathrm{Poisson}(\nu\, n)}\}$ is a Poisson process restricted to $\mu(\mathcal{X}_{\mathrm{free}})$ with intensity $\nu\, n / \mu(\mathcal{X}_{\mathrm{free}})$ (see Lemma 2.8). Thus, the expected number of points of this Poisson process is $\nu\, n$.

Clearly, the set of points generated by one process is a subset of the those generated by the other. However, since $\nu < 1$, in most trials the Poisson point process $\mathcal{P}_{\nu\, n}$ is a subset of the binomial point process.

Define the random variable $\widetilde{K}_n$ denote the number of balls of that fail to have one sample within a distance $\beta\, r_n$ to their centers, when the underlying point process is $\mathcal{P}_{\nu\, n}$ (instead of the independent uniform samples provided by the `SampleFree` procedure). In other words, $\widetilde{K}_n$ is the random variable that is defined similar to K_n, except that the former is defined with respect to the points of $\mathcal{P}_{\nu\, n}$ whereas the latter is defined with respect to the n samples returned by `SampleFree` procedure.

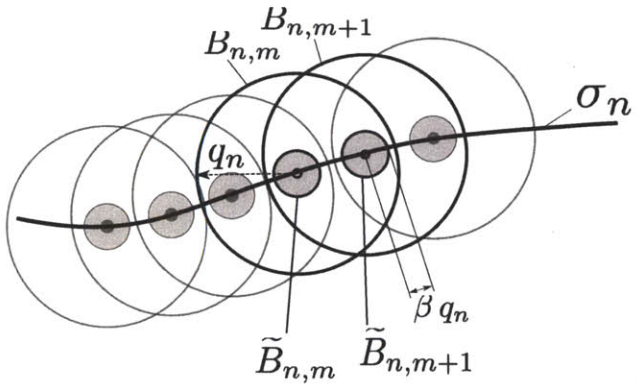


Figure 5-11: The set $\widetilde{B}_{n,m}$ of non-intersection balls is illustrated.

Since $\{\widetilde{K}_n > \alpha\, M_n\}$ is a decreasing event, *i.e.*, the probability that it occurs increases if $\mathcal{P}_{\nu n}$ includes fewer samples, the following bound holds (see, *e.g.*, Penrose, 2003):

$$\mathbb{P}\big(\{K_n \geq \alpha\, M_n\}\big) \leq \mathbb{P}\big(\{\widetilde{K}_n \geq \alpha\, M_n\}\big) + \mathbb{P}(\{\text{Poisson}(\nu\, n) \geq n\}).$$

Since a Poisson random variable has exponentially-decaying tails, the second term on the right hand side can be bounded as

$$\mathbb{P}(\{\text{Poisson}(\nu\, n) \geq n\}) \leq e^{-cn},$$

where $c > 0$ is a constant.

The first term on the right hand side can be computed directly as follows. First, for all small β, the balls of radius $\beta\, r_n$ are all disjoint (see Figure 5-11). Denote this set of balls by $\widetilde{B}_{n,m} = \{\widetilde{B}_{n,1}, \widetilde{B}_{n,2}, \ldots, \widetilde{B}_{n,M_n}\}$. More precisely, $\widetilde{B}_{n,m}$ is the ball of radius $\beta\, q_n$ centered at the center of $B_{n,m}$. Second, observe that the event $\{K_n > \alpha\, M_n\}$ is equivalent to the event that at least an α fraction of all the balls in $\widetilde{B}_n$ include at least one point of the process $\mathcal{P}_{\nu n}$. Since, the point process $\mathcal{P}_{\nu n}$ is Poisson and the balls in $\widetilde{B}_n$ are disjoint for all small enough β, the probability that a single ball in $\widetilde{B}_n$ does not contain a sample is $p_n := \exp(-\zeta_d\, (\beta q_n)^d\, \nu\, n/\mu(\mathcal{X}_{\text{free}})) \leq \exp(-c\, \beta\, \nu\, n)$ for some constant c. Third, by the independence property of the Poisson point process, the number of balls in $\widetilde{B}_n$ that do not include a point of the point process $\mathcal{P}_{\nu n}$ is a binomial random variable with parameters M_n and p_n. Then, for all large n,

$$\mathbb{P}\big(\{\widetilde{K}_n \geq \alpha\, M_n\}\big) \leq \mathbb{P}\left(\{\text{Binomial}(M_n, p_n) \geq \alpha M_n\}\right) \leq \exp(-M_n\, p_n),$$

where recall Binomial(M_n, p_n) is a binomial random variable with parameter M_n, p_n.

Combining the two inequalities above, the following bound is obtained for the original sampling process

$$\mathbb{P}\left(\{K_n \geq \alpha\, M_n\}\right) \leq e^{-c\, n} + e^{-M_n\, p_n}.$$

Summing up both sides,

$$\sum_{n=1}^{\infty} \mathbb{P}\left(\{K_n \geq \alpha\, n\}\right) < \infty.$$

This argument holds for all $\alpha, \beta, \nu > 0$. Hence, for all $\epsilon > 0$,

$$\sum_{n=1}^{\infty} \mathbb{P}\left(\{\|\sigma'_n - \sigma_n\|_{\mathrm{BV}} > \epsilon\}\right) < \infty.$$

Then, by the Borel-Cantelli lemma, $\mathbb{P}\left(\{\lim_{n\to\infty} \|\sigma'_n - \sigma_n\|_{\mathrm{BV}}\}\right) = 0$. ■

Finally, the following lemma states that the cost of the minimum cost path in the graph returned by the PRM* algorithm converges to the optimal cost c^* with probability one. Recall that $Y_n^{\mathrm{PRM}^*}$ denotes the cost of the minimum-cost path in the graph returned by the PRM* algorithm, when the algorithm is run with n samples.

Lemma 5.53 *Under the assumptions of Theorem 5.20, the cost of the minimum-cost path present in the graph returned by the* PRM* *algorithm converges to the optimal cost c^* as the number of samples approaches infinity, with probability one,* i.e.*,*

$$\mathbb{P}\left(\left\{\lim_{n\to\infty} Y_n^{\mathrm{PRM}^*} = c^*\right\}\right) = 1.$$

Proof Recall that σ^* denotes the optimal path, and that $\lim_{n\to\infty} \sigma_n = \sigma^*$ holds surely. By Lemma 5.52, $\lim_{n\to\infty} \|\sigma'_n - \sigma_n\|_{\mathrm{BV}} = 0$ holds with probability one. Thus, by repeated application of the triangle inequality, $\lim_{n\to\infty} \|\sigma'_n - \sigma^*\|_{\mathrm{BV}} = 0$, *i.e.*,

$$\mathbb{P}\left(\left\{\lim_{n\to\infty} \|\sigma'_n - \sigma^*\|_{\mathrm{BV}} = 0\right\}\right) = 1.$$

Then, by the robustness of the optimal path σ^*, it follows that

$$\mathbb{P}\left(\left\{\lim_{n\to\infty} c(\sigma'_n) = c^*\right\}\right) = 1.$$

That is the costs of the paths $\{\sigma'_n\}_{n\in\mathbb{N}}$ converges to the optimal cost almost surely, as the number of samples approaches infinity. ■

5.3.8 Asymptotic Optimality of k-nearest PRM*

In this section, we prove Theorem 5.21, which establishes the asymptotic optimality of the k-nearest PRM* algorithm. The proof follows a similar outline when compared to the proof of Theorem 5.20. In particular, it is again shown that the algorithm is able to connects samples that fall into balls of suitable size. However, in this case, showing that the connections are in fact established requires a substantially different technique. In what follows, detailed arguments are provided whenever the proof differs from the proof of Theorem 5.20. When the argument is only sketched when it is very similar. Below, an outline is given, followed by the detailed argument.

Outline of the proof: i Let σ^* be a robust optimal path with weak clearance. First, define the sequence $\{\sigma_n\}_{n\in\mathbb{N}}$ of paths as in the proof of Theorem 5.20.

Second, define a sequence $\{q_n\}_{n\in\mathbb{N}}$ and tile σ_n with a set $B_n = \{B_{n,1}, B_{n,2}, \ldots, B_{n,M}\}$ of overlapping balls of radius q_n. See Figures 5-9 and 5-12. Let $x_m \in B_{n,m}$ and $x_{m+1} \in B_{n,m+1}$ be any two points from subsequent balls in B_n. Construct B_n such that the straight path connecting x_m and x_{m+1} lies entirely inside the obstacle free space. Also, construct a set B'_n of balls such that (i) $B'_{n,m}$ and $B_{n,m}$ are centered at the same point and (ii) $B_{n,m}$ contains $B_{n,m}$, and $B_{n,m+1}$, for all $m \in \{1, 2, \ldots, M_n - 1\}$.

Let A_n denote the event that each ball in B_n contains at least one vertex, and A'_n denote the event that each ball in B'_n contains at most $k(n)$ vertices of the graph returned by the k-nearest PRM* algorithm. Third, show that A_n and A'_n occur together for all large n, with probability one. Clearly, this implies that the PRM* algorithm will connect vertices in subsequent ball in B_n with an edge, and any path formed by connecting such vertices will be collision-free.

Finally, show that any sequence of paths formed in this way converges to σ^*. Using the robustness of σ^*, show that the best path in the graph returned by the k-nearest PRM* algorithm converges to $c(\sigma^*)$ almost surely.

Construction of the sequence $\{\sigma_n\}_{n\in\mathbb{N}}$ of paths: Let $\theta_1, \theta_2 \in \mathbb{R}_{>0}$ be two constants, the precise values of which will be provided shortly. Define

$$\delta_n := \min\left\{\delta,\ (1+\theta_1)\left(\frac{(1+1/d+\theta_2)\,\mu(X_{\text{free}})}{\zeta_d}\right)^{1/d}\left(\frac{\log n}{n}\right)^{1/d}\right\}.$$

Since $\lim_{n\to\infty}\delta_n = 0$ and $0 \le \delta_n \le \delta$ for all $n \in \mathbb{N}$, by Lemma 5.47, there exists a sequence $\{\sigma_n\}_{n\in\mathbb{N}}$ of paths such that $\lim_{n\to\infty}\sigma_n = \sigma^*$ and σ_n is strongly δ_n-clear for all $n \in \mathbb{N}$.

Construction of the sequence $\{B_n\}_{n\in\mathbb{N}}$ of sets of balls: Define

$$q_n := \frac{\delta_n}{1+\theta_1}.$$

For each $n \in \mathbb{N}$, use Definition 5.48 to construct a set $B_n = \{B_{n,1}, B_{n,2}, \ldots, B_{n,M_n}\}$ of overlapping balls that collectively cover σ_n as $B_n := \texttt{CoveringBalls}(\sigma_n, q_n, \theta_1 q_n)$ (see Figures 5-9 and 5-12 for an illustration).

The probability that each ball in B_n contains at least one vertex: Recall that $G_n^{k\text{PRM}^*} = (V_n^{k\text{PRM}^*}, E_n^{k\text{PRM}^*})$ denotes the graph returned by the k-nearest PRM* algorithm, when the algorithm is run with n samples. Let $A_{n,m}$ denote the event that the ball $B_{n,m}$ contains at least one vertex from $V_n^{k\text{PRM}^*}$, *i.e.*, $A_{n,m} = \left\{B_{n,m} \cap V_n^{k\text{PRM}^*} \neq \emptyset\right\}$. Let A_n denote the event that all balls in $B_{n,m}$ contains at least one vertex of $G_n^{k\text{PRM}^*}$, *i.e.*, $A_n = \bigcap_{m=1}^{M_n} A_{n,m}$.

Recall that A_n^c denotes the complement of the event A_n, $\mu(\cdot)$ denotes the Lebesgue measure, and ζ_d is the volume of the unit ball in the d-dimensional Euclidean space.

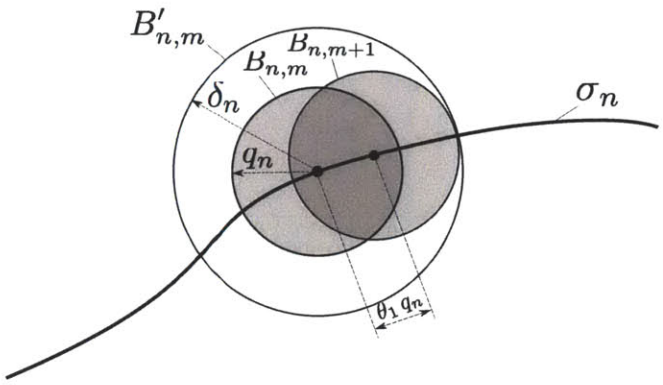


Figure 5-12: An illustration of the covering balls for the k-nearest PRM* algorithm. The δ_n ball is guaranteed to contain the balls $B_{n,m}$ and $B_{n,m+1}$.

Let s_n denote the length of σ_n.

Lemma 5.54 *For all $\theta_1, \theta_2 > 0$,*

$$\mathbb{P}(A_n^c) \leq \frac{s_n}{\theta_1} \left(\frac{\zeta_d}{\theta_1 \, (1 + 1/d + \theta_2) \, \mu(X_{\text{free}})} \right)^{1/d} \frac{1}{(\log n)^{1/d} \; n^{1+\theta_2}}.$$

In particular, $\sum_{n=1}^{\infty} \mathbb{P}(A_n^c) < \infty$ for all $\theta_1, \theta_2 > 0$.

Proof Let $n_0 \in \mathbb{N}$ be a number for which $\delta_n < \delta$ for all $n > n_0$. A bound on the number of balls in B_n can computed as follows. For all $n > n_0$,

$$M_n = \text{card}\,(B_n) \leq \frac{s_n}{\theta_1 \, q_n} = \frac{s_n}{\theta_1} \left(\frac{\zeta_d}{(1 + 1/d + \theta_2)\mu(X_{\text{free}})} \right)^{1/d} \left(\frac{n}{\log n} \right)^{1/d}.$$

The volume of each ball B_n can be computed as

$$\mu(B_{n,k}) = \zeta_d (q_n)^d = (1 + 1/d + \theta_2) \, \mu(X_{\text{free}}) \frac{\log n}{n}.$$

The probability that the ball $B_{n,m}$ does not contain a vertex of the k-nearest PRM* algorithm can be bounded as

$$\mathbb{P}(A_{n,m}^c) = \left(1 - \frac{\mu(B_{n,m})}{\mu(X_{\text{free}})} \right)^n = \left(1 - (1 + 1/d + \theta_2) \frac{\log n}{n} \right)^n \leq n^{-(1+1/d+\theta_2)}.$$

Finally, the probability that at least one of the balls in B_n contains no vertex of

the k-nearest PRM* can be bounded as

$$\begin{aligned}\mathbb{P}(A_n) &= \mathbb{P}\left(\bigcup_{m=1}^{M_n} A_{n,m}\right) \leq \sum_{m=1}^{M_n} \mathbb{P}(A_{n,m}) = M_n\, \mathbb{P}(A_{n,1}) \\ &\leq \frac{s_n}{\theta_1}\left(\frac{\zeta_d}{(1+1/d+\theta_2)\,\mu(X_{\text{free}})}\right)^{1/d}\left(\frac{n}{\log n}\right)^{1/d} n^{-(1+1/d+\theta_2)} \\ &= \frac{s_n}{\theta_1}\left(\frac{\zeta_d}{(1+1/d+\theta_2)\,\mu(X_{\text{free}})}\right)^{1/d}\frac{1}{(\log n)^{1/d}\ n^{1+\theta_2}}.\end{aligned}$$

Clearly, $\sum_{n=1}^{\infty} \mathbb{P}(A_n^c) < \infty$ for all $\theta_1, \theta_2 > 0$. ■

Construction of the sequence $\{B'_n\}_{n\in\mathbb{N}}$ of sets of balls: Construct a set $B'_n = \{B_{n,1}, B_{n,2}, \ldots, B_{n,M_n}\}$ of balls as $B'_n := \texttt{CoveringBalls}(\sigma_n, \delta_n, \theta_1 q_n)$ so that each ball in B'_n has radius δ_n and the spacing between two balls is $\theta_1 q_n$ (see Figure 5-12).

Clearly, the centers of balls in B'_n coincide with the centers of the balls in B_n, *i.e.*, the center of $B'_{n,m}$ is the same as the center of $B_{n,m}$ for all $m \in \{1, 2, \ldots, M_n\}$ and all $n \in \mathbb{N}$. However, the balls in B'_n have a larger radius than those in B_n.

The probability that each ball in B'_n contains at most $k(n)$ vertices: Recall that the k-nearest PRM algorithm connects each vertex in the graph with its $k(n)$ nearest vertices when the algorithm is run with n samples, where $k(n) = k_{\text{PRM}} \log n$. Let A'_n denote the event that all balls in B'_n contain at most $k(n)$ vertices of $G_n^{k\text{PRM}^*}$.

Recall that A'^c_n denotes the complement of the event A_n.

Lemma 5.55 *If $k_{\text{PRM}} > e\,(1+1/d)$, then there exists some $\theta_1, \theta_2 > 0$ such that*

$$\mathbb{P}(A'^c_n) \leq \frac{s_n}{\theta_1}\left(\frac{\zeta_d}{(1+1/d+\theta_2)\mu(X_{\text{free}})}\right)^{1/d}\frac{1}{(\log i)^{1/d}\ n^{-(1+\theta_1)^d(1+1/d+\theta_2)}}.$$

In particular, $\sum_{n=1}^{\infty} \mathbb{P}(A'^c_n) < \infty$ for some $\theta_1, \theta_2 > 0$.

Proof Let $n_0 \in \mathbb{N}$ be a number for which $\delta_n < \delta$ for all $n > n_0$. As shown in the proof of Lemma 5.54, the number of balls in B'_n satisfies

$$M_n = \text{card}\,(B'_n) \leq \frac{s_n}{\theta_1 q_n} = \frac{s_n}{\theta_1}\left(\frac{\zeta_d}{(1+1/d+\theta_2)\mu(X_{\text{free}})}\right)^{1/d}\left(\frac{n}{\log n}\right)^{1/d}.$$

For all $n > n_0$, the volume of $B'_{n,m}$ can be computed as

$$\mu(B'_{n,m}) = \zeta_d\,(\delta_n)^d = (1+\theta_1)^d\,(1+1/d+\theta_2)\,\mu(X_{\text{free}})\,\frac{\log n}{n}.$$

Let $I_{n,m,i}$ denote the indicator random variable of the event that sample i falls

into ball $B'_{n,m}$. The expected value of $I_{n,m,i}$ can be computed as

$$\mathbb{E}[I_{n,m,i}] = \frac{\mu(B'_{n,m})}{\mu(X_{\text{free}})} = (1+\theta_1)^d\,(1+1/d+\theta_2)\,\frac{\log n}{n}.$$

Let $N_{n,m}$ denote the number of vertices that fall inside the ball $B'_{n,m}$, *i.e.*, $N_{n,m} = \sum_{i=1}^{n} I_{n,m,i}$. Then,

$$\mathbb{E}[N_{n,m}] = \sum_{i=1}^{n} \mathbb{E}[I_{n,m,i}] = n\ \mathbb{E}[I_{n,m,1}] = (1+\theta_1)^d(1+1/d+\theta_2)\log n.$$

Since $\{I_{n,m,i}\}_{i=1}^{n}$ are independent identically distributed random variables, large deviations of their sum, $M_{n,m}$, can be bounded by the following Chernoff bound (see, *e.g.*, Dubhashi and Panconesi, 2009):

$$\mathbb{P}\big(\,\{\,N_{n,m} > (1+\epsilon)\,\mathbb{E}[N_{n,m}]\,\}\,\big) \;\leq\; \left(\frac{e^{\epsilon}}{(1+\epsilon)^{(1+\epsilon)}}\right)^{\mathbb{E}[N_{n,m}]},$$

for all $\epsilon > 0$. In particular, for $\epsilon = e-1$,

$$\mathbb{P}\big(\,\{\,N_{n,m} > e\,\mathbb{E}[N_{n,m}]\,\}\,\big) \;\leq\; e^{-\mathbb{E}[N_{n,m}]} \;=\; e^{-(1+\theta_1)^d(1+1/d+\theta_2)\log n} \;=\; n^{-(1+\theta_1)^d(1+1/d+\theta_2)}.$$

Since $k(n) > e\,(1+1/d)\,\log n$, there exists some $\theta_1, \theta_2 > 0$ independent of n such that $e\,\mathbb{E}[N_{n,k}] = e\,(1+\theta_1)\,(1+1/d+\theta_2)\,\log n \leq k(n)$. Then, for the same values of θ_1 and θ_2,

$$\mathbb{P}\big(\,\{\,N_{n,m} > k(n)\,\}\,\big) \;\leq\; \mathbb{P}\big(\,\{\,N_{n,m} > e\,\mathbb{E}[N_{n,m}]\,\}\,\big) \;\leq\; n^{-(1+\theta_1)^d(1+1/d+\theta_2)}.$$

Finally, consider the probability of the event that at least one ball in B_n contains more than $k(n)$ nodes. Using the union bound together with the inequality above

$$\mathbb{P}\left(\bigcup_{m=1}^{M_n}\big\{N_{n,m} > k(n)\big\}\right) \;\leq\; \sum_{m=1}^{M_n+1}\mathbb{P}\big(\big\{N_{n,m} > k(n)\big\}\big) \;=\; M_n\;\mathbb{P}\big(\{N_{n,1} > k(n)\}\big)$$

Hence,

$$\begin{aligned}\mathbb{P}(A_n'^{c}) &= \mathbb{P}\Big(\bigcup_{m=1}^{M_n}\big\{N_{n,m} > k(n)\big\}\Big)\\ &\leq\; \frac{s_n}{\theta_1}\left(\frac{\zeta_d}{(1+1/d+\theta_2)\mu(X_{\text{free}})}\right)^{1/d}\frac{1}{(\log n)^{1/d}\;\; n^{-(1+\theta_1)^d(1+1/d+\theta_2)}}.\end{aligned}$$

Clearly, $\sum_{n=1}^{\infty}\mathbb{P}(A_n'^{c}) < \infty$ for the same values of θ_1 and θ_2. ■

Connecting the vertices in the subsequent balls in B_n: First, note the following lemma.

Lemma 5.56 *If $k_{\text{PRM}} > e\,(1+1/d)^{1/d}$, then there exists $\theta_1, \theta_2 > 0$ such that the event that each ball in B_n contains at least one vertex and each ball in B'_n contains at most $k(n)$ vertices occurs for all large n, with probability one,* i.e.,

$$\mathbb{P}\left(\liminf_{n\to\infty}(A_n \cap A'_n)\right) = 1.$$

Proof Consider the event $A_n^c \cup A_n'^c$, which is the complement of $A_n \cap A'_n$. Using the union bound,

$$\mathbb{P}\left(A_n^c \cup A_n'^c\right) \leq \mathbb{P}(A_n^c) + \mathbb{P}(A_n'^c).$$

Summing both sides,

$$\sum_{n=1}^{\infty} \mathbb{P}(A_n^c \cup A_N'^c) \leq \sum_{n=1}^{\infty} \mathbb{P}(A_n^c) + \sum_{n=1}^{\infty} \mathbb{P}(A_n'^c) < \infty,$$

where the last inequality follows from Lemmas 5.54 and 5.55. Then, by the Borel-Cantelli lemma, $\mathbb{P}\left(\limsup_{n\to\infty}(A_n^c \cup A_n'^c)\right) = \mathbb{P}\left(\limsup_{n\to\infty}(A_n \cap A'_n)^c\right) = 0$, which implies $\mathbb{P}\left(\liminf_{n\to\infty}(A_n \cap A'_n)\right) = 1$. ■

Note that for each $m \in \{1, 2, \ldots, M_n - 1\}$, both $B_{n,m}$ and $B_{n,m+1}$ lies entirely inside the ball $B'_{n,m}$ (see Figure 5-12). Hence, whenever the balls $B_{n,m}$ and $B_{n,m+1}$ contain at least one vertex each, and $B'_{n,m}$ contains at most $k(n)$ vertices, the k-nearest PRM* algorithm attempts to connect all vertices in $B_{n,m}$ and $B_{n,m+1}$ with one another.

The following lemma guarantees that connecting any two points from two consecutive balls in B_n results in a collision-free trajectory. The proof of the lemma is essentially the same as that of Lemma 5.51.

Lemma 5.57 *For all $n \in \mathbb{N}$ and all $m \in \{1, 2, \ldots, M_n\}$, if $x_m \in B_{n,m}$ and $x_{m+1} \in B_{n,m+1}$, then the line segment connecting x_m and x_{m+1} lies in the obstacle-free space,* i.e.,

$$\alpha\, x_m + (1-\alpha)\, x_{m+1} \in X_{\text{free}}, \qquad \text{for all } \alpha \in [0,1].$$

Convergence to the optimal path: The proof of the following lemma is similar to that of Lemma 5.52, and is omitted here.

Let P_n denote the set of all paths in the graph returned by kPRM* algorithm at the end of n iterations. Let σ'_n be the path that is closest to σ_n in terms of the bounded variation norm among all those paths in P_n, *i.e.*, $\sigma'_n := \min_{\sigma' \in P_n} \|\sigma' - \sigma_n\|_{\text{BV}}$.

Lemma 5.58 *The random variable $\|\sigma'_n - \sigma_n\|_{\text{BV}}$ converges to zero almost surely,* i.e.,

$$\mathbb{P}\left(\{\lim_{n\to\infty} \|\sigma'_n - \sigma_n\|_{\text{BV}} = 0\}\right) = 1.$$

A corollary of the lemma above is that $\lim_{n\to\infty} \sigma'_n = \sigma^*$ with probability one. Then, the result follows by the robustness of the optimal solution (see the proof of Lemma 5.53 for details).

5.3.9 Asymptotic Optimality of RRG

This section is devoted to the proof of Theorem 5.22, which establishes the asymptotic optimality of the RRG algorithm.

Outline of the proof: The proof of this theorem is similar to that of Theorem 5.20. The main difference is the definition of C_n that denotes the event that the RRG algorithm has sufficiently explored the obstacle free space. More precisely, C_n is the event that for any point x in the obstacle free space, the graph maintained by the RRG algorithm algorithm includes a vertex that can be connected to x.

Construct the sequence $\{\sigma_n\}_{n\in\mathbb{N}}$ of paths and the sequence $\{B_n\}_{n\in\mathbb{N}}$ of balls as in the proof of Theorem 5.20. Let A_n denote the event that each ball in B_n contains a vertex of the graph maintained by the RRG by the end of iteration n. Compute n by conditioning on the event that C_i holds for all $i \in \{\lfloor \theta_3 \, n \rfloor, \ldots, n\}$, where $0 < \theta_3 < 1$ is a constant. Show that the probability that C_i fails to occur for any such i is small enough to guarantee that A_n occurs for all large n with probability one. Complete the proof as in the proof of Theorem 5.20.

Definitions of $\{\sigma_n\}_{n\in\mathbb{N}}$ and $\{B_n\}_{n\in\mathbb{N}}$: Let $\theta_1 > 0$ be a constant. Define δ_n, σ_n, q_n, and B_n as in the proof of Theorem 5.20.

Probability that each ball in B_n contains at least one vertex: Let $A_{n,m}$ be the event that the ball $B_{n,m}$ contains at least one vertex of the RRG at the end of n iterations. Let A_n be the event that all balls in B_n contain at least one vertex of the RRG at the end of iteration n, *i.e.*, $A_n = \bigcap_{m=1}^{M_n} A_{n,m}$, where M_n is the number of balls in B_n. Recall that γ_{RRG} is the constant used in defining the connection radius of the RRG algorithm (see Algorithm 5).

Lemma 5.59 *If* $\gamma_{\text{RRG}} > 2(1+1/d)^{1/d} \left(\frac{\mu(X_{\text{free}})}{\zeta_d}\right)^{1/d}$ *then there exists* $\theta_1 > 0$ *such that* A_n *occurs for all large* n *with probability one,* i.e.,

$$\mathbb{P}\left(\liminf_{n\to\infty} A_n\right) = 1.$$

The proof of this lemma requires two intermediate results, which are provided next.

Recall that η is the parameter used in the `Steer` procedure (see the definition of `Steer` procedure in Section 4.1). Let C_n denote the event that for any point $x \in X_{\text{free}}$, the graph returned by the RRG algorithm includes a vertex v such that $|x - v| \leq \eta$ and the line segment joining v and x is collision-free. The following lemma establishes an bound on the probability that this event fails to occur at iteration n.

Lemma 5.60 *There exists constants* $a, b \in \mathbb{R}_{>0}$ *such that* $P(C_n^c) \leq a\, e^{-b\, n}$ *for all* $n \in \mathbb{N}$.

Proof Partition X_{free} into finitely many convex sets such that each partition is bounded by a ball a radius η. Such a finite partition exists by the boundedness

of X_{free}. Denote this partition by $X'_1, X'_2, \ldots, X'_M$. Since the probability of failure decays to zero with an exponential rate, for any $m \in \{1, 2, \ldots, M\}$, the probability that p_m fails to contain a vertex of the RRG decays to zero with an exponential rate, *i.e.*,

$$\mathbb{P}\left(\{\nexists x \in V_n^{\text{RRG}} \cap X'_m\}\right) \leq a_m \, e^{-b_m \, n}$$

The probability that at least one partition fails to contain one vertex of the RRG also decays to zero with an exponential rate. That is, there exists $a, b \in \mathbb{R}_{>0}$ such that

$$\mathbb{P}\left(\bigcup_{m=1}^{M} \{\nexists x \in V_n^{\text{RRG}} \cap X'_m\}\right) \leq \sum_{m=1}^{M} \mathbb{P}\left(\{\nexists x \in V_n^{\text{RRG}} \cap X'_m\}\right) \leq \sum_{m=1}^{M} a_m \, e^{-b_m n} \leq a \, e^{-b \, n},$$

where the first inequality follows from the union bound.

■

Let $0 < \theta_3 < 1$ be a constant independent of n. Consider the event that C_i occurs for all i that is greater than $\theta_3 \, n$, *i.e.*, $\bigcap_{i=\lfloor \theta_3 \, n \rfloor}^{n} C_i$. The following lemma analyzes the probability of the event that $\bigcap_{i=\lfloor \theta_3 \, n \rfloor}^{n} C_i$ fails to occur.

Lemma 5.61 *For any* $\theta_3 \in (0, 1)$,

$$\sum_{n=1}^{\infty} \mathbb{P}\left(\left(\bigcap_{i=\lfloor \theta_3 n \rfloor}^{n} C_i\right)^c\right) \; < \; \infty.$$

Proof The following inequalities hold:

$$\begin{aligned}
\sum_{n=1}^{\infty} \mathbb{P}\left(\left(\bigcap_{i=\lfloor \theta_3 \, n \rfloor}^{n} C_i\right)^c\right) \; &= \; \sum_{n=1}^{\infty} \mathbb{P}\left(\bigcup_{i=\lfloor \theta_3 \, n \rfloor}^{n} C_i^c\right) \\
&\leq \; \sum_{n=1}^{\infty} \sum_{i=\lfloor \theta_3 \, i \rfloor}^{n} \mathbb{P}(C_i^c) \; \leq \; \sum_{n=1}^{\infty} \sum_{i=\lfloor \theta_3 \, n \rfloor}^{n} a \, e^{-b \, i},
\end{aligned}$$

where the last inequality follows from Lemma 5.60. The right-hand side is finite for all $a, b > 0$. ■

Proof of Lemma 5.59 It is shown that $\sum_{n=1}^{\infty} \mathbb{P}\left(A_n^c\right) < \infty$, which, by the Borel-Cantelli Lemma (see Grimmett and Stirzaker, 2001), implies that A_n^c occurs infinitely often with probability zero, *i.e.*, $\mathbb{P}(\limsup_{n \to \infty} A_n^c) = 0$, which in turn implies $\mathbb{P}(\liminf_{n \to \infty} A_n) = 1$.

Let $n_0 \in \mathbb{N}$ be a number for which $\delta_n < \delta$ for all $n > n_0$. First, for all $n > n_0$, the number of balls in B_n can be bounded by (see the proof of Lemma 5.49 for details)

$$M_n = \text{card}\left(B_n\right) \leq \frac{(2 + \theta_1) \, s_n}{\theta_1 \, \gamma_{\text{RRG}}} \left(\frac{n}{\log n}\right)^{1/d}.$$

Second, for all $n > n_0$, the volume of each ball in B_n can be calculated as (see the proof of Lemma 5.49)

$$\mu(B_{n,m}) \;=\; \zeta_d \left(\frac{\gamma_{\text{PRM}}}{2+\theta_1}\right)^d \frac{\log n}{n},$$

where ζ_d is the volume of the unit ball in the d-dimensional Euclidean space.

Third, conditioning on the event $\bigcap_{i=\lfloor \theta_3 n \rfloor}^{n} C_i$, each new sample will be added to the graph maintained by the RRG algorithm as a new vertex between iterations $i = \lfloor \theta_3 n \rfloor$ and $i = n$. Thus,

$$\begin{aligned}
\mathbb{P}\left(A_{n,m}^c \,\middle|\, \bigcap_{i=\lfloor \theta_3 n \rfloor}^{n} C_i\right) &\leq \left(1 - \frac{\mu(B_{n,m})}{\mu(X_{\text{free}})}\right)^{n-\lfloor \theta_3 n \rfloor} \leq \left(1 - \frac{\mu(B_{n,m})}{\mu(X_{\text{free}})}\right)^{(1-\theta_3)\, n} \\
&\leq \left(1 - \frac{\zeta_d}{\mu(X_{\text{free}})} \left(\frac{\gamma_{\text{RRG}}}{2+\theta_1}\right)^d \frac{\log n}{n}\right)^{(1-\theta_3)\, n} \\
&\leq e^{-\frac{(1-\theta_3)\,\zeta_d}{\mu(X_{\text{free}})}\left(\frac{\gamma_{\text{RRG}}}{2+\theta_1}\right)^d \log n} \leq n^{-\frac{(1-\theta_3)\,\zeta_d}{\mu(X_{\text{free}})}\left(\frac{\gamma_{\text{RRG}}}{2+\theta_1}\right)^d},
\end{aligned}$$

where the fourth inequality follows from $(1 - 1/f(n))^{g(n)} \leq e^{g(n)/f(n)}$.

Fourth,

$$\begin{aligned}
\mathbb{P}\left(A_n^c \,\middle|\, \bigcap_{i=\lfloor \theta_3 n \rfloor}^{n} C_i\right) &\leq \mathbb{P}\left(\bigcup_{m=1}^{M_n} A_{n,m}^c \,\middle|\, \bigcap_{i=\lfloor \theta_3 n \rfloor}^{n} C_i\right) \\
&\leq \sum_{m=1}^{M_n} \mathbb{P}\left(A_{n,m}^c \,\middle|\, \bigcap_{i=\lfloor \theta_3 n \rfloor}^{n} C_i\right) \\
&= M_n\; \mathbb{P}\left(A_{n,1}^c \,\middle|\, \bigcap_{i=\lfloor \theta_3 n \rfloor}^{n} C_i\right) \\
&\leq \frac{(2+\theta_1)\, s_n}{\theta_1\, \gamma_{\text{RRG}}} \left(\frac{n}{\log n}\right)^{1/d} n^{-\frac{(1-\theta_3)\,\zeta_d}{\mu(X_{\text{free}})}\left(\frac{\gamma_{\text{RRG}}}{2+\theta_1}\right)^d}.
\end{aligned}$$

Hence,

$$\mathbb{P}\left(A_n^c \,\middle|\, \bigcap_{i=\lfloor \theta_3 n \rfloor}^{n} C_i\right) \;<\; \infty,$$

whenever $\frac{(1-\theta_3)\,\zeta_d}{\mu(X_{\text{free}})}\left(\frac{\gamma_{\text{RRG}}}{2+\theta_1}\right)^d - 1/d > 1$, *i.e.*, $\gamma_{\text{RRG}} > (2+\theta_1)(1+1/d)^{1/d}\left(\frac{\mu(X_{\text{free}})}{(1-\theta_3)\,\zeta_d}\right)^{1/d}$, which is satisfied by appropriately choosing the constants θ_1 and θ_3, since $\gamma_{\text{RRG}} > 2\,(1+1/d)^{1/d}\left(\frac{\mu(X_{\text{free}})}{\zeta_d}\right)^{1/d}$.

Finally,

$$\begin{aligned}
\mathbb{P}\left(A_n^c \mid \bigcap_{i=\lfloor \theta_3 n \rfloor}^{n} C_i\right) &= \frac{\mathbb{P}\left(A_n^c \cap \left(\cap_{i=\lfloor \theta_3 n \rfloor}^{n} C_i\right)\right)}{\mathbb{P}\left(\bigcap_{i=\lfloor \theta_3 n \rfloor}^{n} C_i\right)} \\
&\geq \mathbb{P}(A_n^c \cap (\cap_{i=\lfloor \theta_3 n \rfloor}^{n} C_i)) \\
&= 1 - \mathbb{P}(A_n \cup (\cap_{i=\lfloor \theta_3 n \rfloor}^{n} C_i)^c) \\
&\geq 1 - \mathbb{P}(A_n) - \mathbb{P}\big((\cap_{i=\lfloor \theta_3 n \rfloor}^{n} C_i)^c\big) \\
&= \mathbb{P}(A_n^c) - \mathbb{P}\big((\cap_{i=\lfloor \theta_3 n \rfloor}^{n} C_i)^c\big).
\end{aligned}$$

Taking the infinite sum of both sides yields

$$\sum_{n=1}^{\infty} \mathbb{P}(A_n^c) \leq \sum_{n=1}^{\infty} \mathbb{P}\left(A_n^c \mid \bigcap_{i=\lfloor \theta_3 n \rfloor}^{n} C_n\right) + \sum_{n=1}^{\infty} \mathbb{P}\left(\left(\bigcap_{i=\lfloor \theta_3 n \rfloor}^{n} C_n\right)\right).$$

The first term on the right hand side is shown to be finite above. The second term is finite by Lemma 5.61. Hence, $\sum_{n=1}^{\infty} \mathbb{P}(A_n) < \infty$. Then, by the Borel Cantelli lemma, A_n^c occurs infinitely often with probability zero, which implies that its complement A_n occurs for all large n, with probability one. ■

Convergence to the optimal path: The proof of the following lemma is similar to that of Lemma 5.52, and is omitted here.

Let P_n denote the set of all paths in the graph returned by RRG algorithm at the end of n iterations. Let σ'_n be the path that is closest to σ_n in terms of the bounded variation norm among all those paths in P_n, *i.e.*, $\sigma'_n := \min_{\sigma' \in P_n} \|\sigma' - \sigma_n\|_{\mathrm{BV}}$.

Lemma 5.62 *The random variable* $\|\sigma'_n - \sigma_n\|_{\mathrm{BV}}$ *converges to zero almost surely,* i.e.,

$$\mathbb{P}\left(\{\lim_{n\to\infty} \|\sigma'_n - \sigma_n\|_{\mathrm{BV}} = 0\}\right) = 1.$$

A corollary of the lemma above is that $\lim_{n\to\infty} \sigma'_n = \sigma^*$ with probability one. Then, the result follows by the robustness of the optimal solution (see the proof of Lemma 5.53 for details).

5.3.10 Asymptotic Optimality of k-nearest RRG

This section is devoted to the proof of Theorem 5.23, which states that the RRG algorithm is asymptotically optimal.

Outline of the proof: The proof of this theorem is a combination of that of Theorem 5.21 and 5.22.

Define the sequences $\{\sigma_n\}_{n\in\mathbb{N}}$, $\{B_n\}_{n\in\mathbb{N}}$, and $\{B'_n\}_{n\in\mathbb{N}}$ as in the proof of Theorem 5.21. Define the event C_n as in the proof of Theorem 5.22. Let A_n denote the event that each ball in B_n contains at least one vertex, and A'_n denote the event that

each ball in B'_n contains at most $k(n)$ vertices of the graph maintained by the RRG algorithm, by the end of iteration n. Compute A_n and A'_n by conditioning on the event that C_i holds for all $i = \theta_3 n$ to n. Show that this is enough to guarantee that A_n and A'_n hold together for all large n, with probability one.

Definitions of $\{\sigma_n\}_{n\in\mathbb{N}}$, $\{B_n\}_{n\in\mathbb{N}}$, and $\{B'_n\}_{n\in\mathbb{N}}$: Let $\theta_1, \theta_2 > 0$ be two constants. Define δ_n, σ_n, q_n, B_n, and B'_n as in the proof of Theorem 5.21.

The probability that each ball in B_n contains at least one vertex: Let $A_{n,m}$ denote the event that the ball $B_{n,m}$ contains at least one vertex of the graph maintained by the k-nearest RRG algorithm by the end of iteration n. Let A_n denote the event that all balls in $B_{n,m}$ contain at least one vertex of the same graph, *i.e.*, $A_n = \bigcup_{m=1}^{M_n} A_{n,m}$. Let s_n denote the length of σ_n, *i.e.*, $TV(\sigma_n)$. Recall η is the parameter in the `Steer` procedure. Let C_n denote the event that for any point $x \in X_{\text{free}}$, the k-nearest RRG algorithm includes a vertex v such that $|x - v| \leq \eta$.

Lemma 5.63 *For any $\theta_1, \theta_2 > 0$ and any $\theta_3 \in (0,1)$,*

$$\mathbb{P}\Big(A_n^c \,\Big|\, \bigcap_{i=\lfloor\theta_3 n\rfloor}^{n} C_i\Big) \leq \frac{s_n}{\theta_1}\left(\frac{\zeta_d}{\theta_1\,(1+1/d+\theta_2)\,\mu(X_{\text{free}})}\right)^{1/d} \frac{1}{(\log n)^{1/d}\; n^{(1-\theta_3)(1+1/d+\theta_2)-1/d}}.$$

In particular, $\sum_{n=1}^{\infty} \mathbb{P}(A_n^c \mid \bigcap_{i=\lfloor\theta_3 n\rfloor}^{n} C_i) < \infty$ for any $\theta_1, \theta_2 > 0$ and some $\theta_3 \in (0,1)$.

Proof Let $n_0 \in \mathbb{N}$ be a number for which $\delta_n < \delta$ for all $n > n_0$. Then, for all $n > n_0$,

$$M_n = \text{card}\,(B_n) \leq \frac{s_n}{\theta_1\, q_n} = \frac{s_n}{\theta_1}\left(\frac{\zeta_d}{(1+1/d+\theta_2)\mu(X_{\text{free}})}\right)^{1/d}\left(\frac{n}{\log n}\right)^{1/d}.$$

The volume of each ball B_n can be computed as

$$\mu(B_{n,k}) = \zeta_d (q_n)^d = (1+1/d+\theta_2)\,\mu(X_{\text{free}})\frac{\log n}{n}.$$

Given $\bigcap_{i=\lceil\theta_3 n\rceil}^{n} C_i$, the probability that the ball $B_{n,m}$ does not contain a vertex of the k-nearest PRM* algorithm can be bounded as

$$\begin{aligned}\mathbb{P}\Big(A_{n,m}^c \,\Big|\, \bigcap_{i=\lceil\theta_3 n\rceil}^{n} C_i\Big) &= \left(1 - \frac{\mu(B_{n,m})}{\mu(X_{\text{free}})}\right)^{(1-\theta_3)\,n} \\ &= \left(1 - (1+1/d+\theta_2)\frac{\log n}{n}\right)^{(1-\theta_3)\,n} \leq n^{-(1-\theta_3)\,(1+1/d+\theta_2)}.\end{aligned}$$

Finally, the probability that at least one of the balls in B_n contains no vertex of

the k-nearest PRM* can be bounded as

$$\begin{aligned}\mathbb{P}(A_n) &= \mathbb{P}\left(\bigcup_{m=1}^{M_n} A_{n,m}\right) \leq \sum_{m=1}^{M_n} \mathbb{P}(A_{n,m}) = M_n\, \mathbb{P}(A_{n,1}) \\ &\leq \frac{s_n}{\theta_1}\left(\frac{\zeta_d}{(1+1/d+\theta_2)\,\mu(X_{\text{free}})}\right)^{1/d} \frac{1}{(\log n)^{1/d}\ n^{(1-\theta_3)\,(1+1/d+\theta_2)-1/d}}.\end{aligned}$$

Clearly, for all $\theta_1, \theta_2 > 0$, there exists some $\theta_3 \in (0,1)$ such that $\sum_{n=1}^{\infty} \mathbb{P}(A_n^c) < \infty$. ■

The probability that each ball in B'_n contains at most $k(n)$ vertices: Let A'_n denote the event that all balls in B'_n contain at most $k(n)$ vertices of the graph maintained by the RRG algorithm, by end of iteration n.

Lemma 5.64 *If $k_{\text{PRM}} > e\,(1+1/d)$, then there exists $\theta_1, \theta_2, \theta_3 > 0$ such that*

$$\begin{aligned}&\mathbb{P}\left(A_n'^c \,\middle|\, \bigcap_{i=\lfloor \theta_3\, n\rfloor}^{n} C_i\right) \\ &\qquad \leq \frac{s_n}{\theta_1}\left(\frac{\zeta_d}{(1+1/d+\theta_2)\mu(X_{\text{free}})}\right)^{1/d} \frac{1}{(\log n)^{1/d}\ n^{-(1-\theta_3)(1+\theta_1)^d(1+1/d+\theta_2)}}.\end{aligned}$$

In particular, $\sum_{n=1}^{\infty} \mathbb{P}(A_n^c \mid \bigcap_{i=\lfloor \theta_3\, n\rfloor}^{n} C_i) < \infty$ for some $\theta_1, \theta_2 > 0$ and some $\theta_3 > 0$.

Proof Let $n_0 \in \mathbb{N}$ be a number for which $\lambda_n < \delta$ for all $n > n_0$. Then, the number of balls in B'_n and the volume of each ball can be computed as

$$M_n = \text{card}\,(B'_n) \leq \frac{s_n}{\theta_1 q_n} = \frac{s_n}{\theta_1}\left(\frac{\zeta_d}{(1+1/d+\theta_2)\mu(X_{\text{free}})}\right)^{1/d}\left(\frac{n}{\log n}\right)^{1/d}.$$

$$\mu(B'_{n,m}) = \zeta_d\,(\lambda_n)^d = (1+\theta_1)^d\,(1+1/d+\theta_2)\,\mu(X_{\text{free}})\,\frac{\log n}{n}.$$

Let $I_{n,m,i}$ denote the indicator random variable of the event that sample i falls into ball $B'_{n,m}$. The expected value of $I_{n,m,i}$ can be computed as

$$\mathbb{E}[I_{n,m,i}] = \frac{\mu(B'_{n,m})}{\mu(X_{\text{free}})} = (1+\theta_1)^d\,(1+1/d+\theta_2)\,\frac{\log n}{n}.$$

Let $N_{n,m}$ denote the number of vertices that fall inside the ball $B'_{n,m}$ between iterations $\lfloor \theta_3\, n\rfloor$ and n, *i.e.*, $N_{n,m} = \sum_{i=\lfloor \theta_3\, n\rfloor}^{n} I_{n,m,i}$. Then,

$$\mathbb{E}[N_{n,m}] = \sum_{i=\lfloor \theta_3\, n\rfloor}^{n} \mathbb{E}[I_{n,m,i}] = (1-\theta_3)\,n\ \mathbb{E}[I_{n,m,1}] = (1-\theta_3)\,(1+\theta_1)^d\,(1+1/d+\theta_2)\log n.$$

Since $\{I_{n,m,i}\}_{i=1}^{n}$ are independent identically distributed random variables, large deviations of their sum, $M_{n,m}$, can be bounded by the following Chernoff bound (Dubhashi

and Panconesi, 2009):

$$\mathbb{P}\big(\{\, N_{n,m} > (1+\epsilon)\,\mathbb{E}[N_{n,m}]\,\}\big) \;\le\; \left(\frac{e^{\epsilon}}{(1+\epsilon)^{(1+\epsilon)}}\right)^{\mathbb{E}[N_{n,m}]},$$

for all $\epsilon > 0$. In particular, for $\epsilon = e - 1$,

$$\mathbb{P}\big(\{\, N_{n,m} > e\,\mathbb{E}[N_{n,m}]\,\}\big) \;\le\; e^{-\mathbb{E}[N_{n,m}]} \;=\; n^{-(1-\theta_3)\,(1+\theta_1)^d\,(1+1/d+\theta_2)}.$$

Since $k(n) > e\,(1 + 1/d)\,\log n$, there exists some $\theta_1, \theta_2 > 0$ and $\theta_3 \in (0,1)$, independent of n, such that $e\,\mathbb{E}[N_{n,k}] = e\,(1-\theta_3)\,(1+\theta_1)\,(1+1/d+\theta_2)\,\log n \le k(n)$. Then, for the same values of θ_1 and θ_2,

$$\mathbb{P}\big(\{\, N_{n,m} > k(n)\,\}\big) \;\le\; \mathbb{P}\big(\{\, N_{n,m} > e\,\mathbb{E}[N_{n,m}]\,\}\big) \;\le\; n^{-(1-\theta_3)\,(1+\theta_1)^d\,(1+1/d+\theta_2)}.$$

Finally, consider the probability of the event that at least one ball in B_n contains more than $k(n)$ nodes. Using the union bound together with the inequality above

$$\mathbb{P}\left(\bigcup\nolimits_{m=1}^{M_n} \{N_{n,m} > k(n)\}\right) \;\le\; \sum_{m=1}^{M_n+1} \mathbb{P}\big(\{N_{n,m} > k(n)\}\big) \;=\; M_n\;\mathbb{P}\big(\{N_{n,1} > k(n)\}\big)$$

Hence,

$$\begin{aligned}\mathbb{P}\Big(A_n'^{c} \,\Big|\, \bigcap\nolimits_{i=\lfloor\theta_3\, n\rfloor}^{n} C_i\Big) &= \mathbb{P}\left(\bigcup\nolimits_{m=1}^{M_n} \{N_{n,m} > k(n)\}\right)\\ &\le \frac{s_n}{\theta_1}\left(\frac{\zeta_d}{(1+1/d+\theta_2)\mu(X_{\text{free}})}\right)^{1/d} \frac{1}{(\log n)^{1/d}\;\; n^{-(1-\theta_3)\,(1+\theta_1)^d(1+1/d+\theta_2)}}.\end{aligned}$$

Clearly, $\sum_{n=1}^{\infty} \mathbb{P}\big(A_n'^{c} \,\big|\, \cap_{i=\lfloor\theta_3\, n\rfloor}^{n} C_i\big) < \infty$ for the same values of θ_1, θ_2, and θ_3. . ■

Connecting the vertices in subsequent balls in B_n:

Lemma 5.65 *If $k_{\text{PRM}} > e\,(1 + 1/d)^{1/d}$, then there exists $\theta_1, \theta_2 > 0$ such that the event that each ball in B_n contains at least one vertex and each ball in B_n' contains at most $k(n)$ vertices occurs for all large n, with probability one,* i.e.,

$$\mathbb{P}\left(\liminf_{n\to\infty}(A_n \cap A_n')\right) = 1.$$

First note the following lemma.

Lemma 5.66 *For any $\theta_3 \in (0,1)$,*

$$\sum_{n=1}^{\infty} \mathbb{P}\left(\left(\bigcap\nolimits_{i=\lfloor\theta_3 n\rfloor}^{n} C_n\right)^{c}\right) \;<\; \infty.$$

Proof Since the RRG algorithm and the k-nearest RRG algorithm have the same

vertex sets, *i.e.*, $V_n^{\text{RRG}} = V_n^{k\text{RRG}}$ for all $n \in \mathbb{N}$, the lemma follows from Lemma 5.61. ■

Proof of Lemma 5.65 Note that

$$\begin{aligned}\mathbb{P}\left((A_n^c \cup A_n'^c) \mid \bigcap_{i=\lfloor \theta_3 n \rfloor}^{n} C_i\right) &= \frac{\mathbb{P}\left(A_n^c \cap \left(\cap_{i=\lfloor \theta_3 n \rfloor}^{n} C_i\right)\right)}{\mathbb{P}\left(\bigcap_{i=\lfloor \theta_3 n \rfloor}^{n} C_i\right)} \\ &\geq \mathbb{P}\left((A_n^c \cup A_n'^c) \cap \left(\cap_{i=\lfloor \theta_3 n \rfloor}^{n} C_i\right)\right) \\ &\geq \mathbb{P}(A_n^c \cup A_n'^c) - \mathbb{P}\big((\cap_{i=\lfloor \theta_3 n \rfloor}^{n} C_i)^c\big),\end{aligned}$$

where the last inequality follows from the union bound. Rearranging and using the union bound,

$$\mathbb{P}(A_n^c \cup A_n'^c) \leq \mathbb{P}\big(A_n^c \mid \cap_{i=\lfloor \theta_3 n \rfloor}^{n} C_i\big) + \mathbb{P}\big(A_n^c \mid \cap_{i=\lfloor \theta_3 n \rfloor}^{n} C_i\big) + \mathbb{P}\big(\big(\cap_{i=\lfloor \theta_3 n \rfloor}^{n} C_i\big)^c\big).$$

Summing both sides,

$$\begin{aligned}&\sum_{n=1}^{\infty} \mathbb{P}(A_n^c \cup A_n'^c) \\ &\leq \sum_{n=1}^{\infty} \mathbb{P}\left(A_n^c \mid \bigcap_{i=\lfloor \theta_3 n \rfloor}^{n} C_i\right) + \mathbb{P}\left(A_n'^c \mid \bigcap_{i=\lfloor \theta_3 n \rfloor}^{n} C_i\right) + \sum_{n=1}^{\infty} \mathbb{P}\left(\left(\bigcap_{i=\lfloor \theta_3 n \rfloor}^{n} C_i\right)^c\right),\end{aligned}$$

where the right hand side is finite by Lemmas 5.63, 5.64, and 5.66, by picking θ_3 close to one. Hence, $\sum_{n=1}^{\infty} \mathbb{P}(A_n^c \cup A_n'^c) < \infty$. Then, by the Borel-Cantelli lemma, $\mathbb{P}(\limsup_{n\to\infty}(A_n^c \cup A_n'^c)) = 0$, or equivalently $\mathbb{P}(\liminf_{n\to\infty}(A_n \cap A_n')) = 1$. ■

Convergence to the optimal path: The proof of the following two lemmas are essentially the same as that of Lemma 5.52, and is omitted here. Let P_n denote the set of all paths in the graph returned by kRRG algorithm at the end of n iterations. Let σ_n' be the path that is closest to σ_n in terms of the bounded variation norm among all those paths in P_n, *i.e.*, $\sigma_n' := \min_{\sigma' \in P_n} \|\sigma' - \sigma_n\|_{\text{BV}}$.

Lemma 5.67 *The random variable* $\|\sigma_n' - \sigma_n\|_{\text{BV}}$ *converges to zero almost surely,* i.e.,

$$\mathbb{P}\left(\{\lim_{n\to\infty} \|\sigma_n' - \sigma_n\|_{\text{BV}} = 0\}\right) = 1.$$

A corollary of the lemma above is that $\lim_{n\to\infty} \sigma_n' = \sigma^*$ with probability one. Then, the result follows by the robustness of the optimal solution (see the proof of Lemma 5.53 for details).

5.3.11 Asymptotic Optimality of RRT*

This section is devoted to the proof of Theorem 5.24, which establishes the optimality of the RRT* algorithm. For simplicity, we assume that the steering parameter η is large enough, *i.e.*, $\eta \geq \text{diam}(\mathcal{X})$, although the results hold for any $\eta > 0$.

Marked point process: Consider the following marked point process. Let $\{X_1, X_2, \ldots, X_n\}$ be a independent uniformly distributed points drawn from X_{free} and let $\{Y_1, Y_2, \ldots, Y_n\}$ be independent uniform random variables with support $[0, 1]$. Each point X_i is associated with a mark Y_i that describes the order of X_i in the process. More precisely, a point X_i is assumed to be drawn after another point $X_{i'}$ if $Y_{i'} < Y_i$. We will also assume that the point process includes the point x_{init} with mark $Y = 0$.

Consider the graph formed by adding an edge $(X_{i'}, X_i)$, whenever (i) $Y_{i'} < Y_i$ and (ii) $\|X_i - X_{i'}\| \leq r_n$ both hold. Notice that, formed in this way, G_n includes no directed cycles. Denote this graph by $G_n = (V_n, E_n)$. Also, consider a subgraph G'_n of G_n formed as follows. Let $c(X_i)$ denote the cost of best path starting from x_{init} and reaching X_i. In G'_n, each vertex X_i has a single parent X_i with the smallest cost $c(X_i)$. Since the graph is built incrementally, the cost of the best path reaching X_i will be the same as the one reaching $X_{i'}$ in both G_n and G'_n. Clearly, G'_n is equivalent to the graph returned by the RRT* algorithm at the end of n iterations, if the steering parameter η is large enough.

Let Y_n and the Y'_n denote the costs of the best paths starting from x_{init} and reaching the goal region in G_n and G'_n, respectively. Then, $\limsup_{n\to\infty} Y_n = \limsup_{n\to\infty} Y'_n$ surely. In the rest of the proof, it is shown that $\mathbb{P}(\{\limsup_{n\to\infty} Y_n\}) = 1$, which implies that $\mathbb{P}(\{\limsup_{n\to\infty} Y'_n\}) = 1$, which in turn implies the result.

Definitions of $\{\sigma_n\}_{n\in\mathbb{N}}$ and $\{B_n\}_{n\in\mathbb{N}}$: Let σ^* denote an optimal path. Define

$$\delta_n := \min\{\delta, 4\, r_n\},$$

where r_n is the connection radius of the RRT* algorithm. Let $\{\sigma_n\}_{n\in\mathbb{N}}$ be the sequence paths, the existence of which is guaranteed by Lemma 5.47.

For each $n \in \mathbb{N}$, construct a sequence $\{B_n\}_{n\in\mathbb{N}}$ of balls that cover σ_n as $B_n = \{B_{n,1}, B_{n,2}, \ldots, B_{n,M_n}\} := \texttt{CoveringBalls}(\sigma_n, r_n, 2\, r_n)$ (see Definition 5.48), where r_n is the connection radius of the RRT* algorithm, *i.e.*, $r_n = \gamma_{\text{RRT}^*} \left(\frac{\log n}{n}\right)^{1/d}$. Clearly, the balls in B_n are openly disjoint, since the spacing between any two consecutive balls is $2\, r_n$.

Connecting the vertices in subsequent balls in B_n: For all $m \in \{1, 2, \ldots, M_n\}$, let $A_{n,m}$ denote the event that there exists two vertices $X_i, X_{i'} \in V_n^{\text{RRT}^*}$ such that $X_i \in B_{n,m}$, $X_{i'} \in B_{n,m+1}$ and $Y_{i'} \leq Y_i$, where Y_i and $Y_{i'}$ are the marks associated with points X_i and $X_{i'}$, respectively. Notice that, in this case, X_i and $X_{i'}$ will be connected with an edge in G_n. Let A_n denote the event that $A_{n,m}$ holds for all $m \in \{1, 2, \ldots, M\}$, *i.e.*, $A_n = \bigcap_{m=1}^{M} A_{n,m}$.

Lemma 5.68 *If $\gamma_{\text{RRT}^*} > 4 \left(\frac{\mu(\mathcal{X}_{\text{free}})}{\zeta_d}\right)^{1/d}$, then A_n occurs for all large n, with probability one,* i.e.,

$$\mathbb{P}\left(\liminf_{n\to\infty} A_n\right) = 1.$$

Proof The proof of this result is based on a Poissonization argument. Let Poisson(λ) be a Poisson random variable with parameter $\lambda = \theta\, n$, where $\theta \in (0,1)$ is a constant independent of n. Consider the point process that consists of exactly Poisson($\theta\, n$) points, *i.e.*, $\{X_1, X_2, \ldots, X_{\text{Poisson}(\theta\, n)}\}$. This point process is a Poisson point process with intensity $\theta\, n\, /\mu(X_{\text{free}})$ by Lemma 2.8.

Let $\tilde{A}_{n,m}$ denote the event that there exists two vertices X_i and $X_{i'}$ in the vertex set of the RRT* algorithm such that X_i and $X_{i'}$ are connected with an edge in $\tilde{G}_n$, where $\tilde{G}_n$ is the graph returned by the RRT* when the algorithm is run for Poisson($\theta\, n$) many iterations, *i.e.*, Poisson($\theta\, n$) samples are drawn from $\mathcal{X}_{\text{free}}$.

Clearly, $\mathbb{P}(A^c_{n,m}) = \mathbb{P}(\tilde{A}^c_{n,m} \,|\, \{\text{Poisson}(\theta\, n) = n\})$. Moreover,

$$\mathbb{P}(A^c_{n,m}) \;\leq\; \mathbb{P}(\tilde{A}^c_{n,m}) + \mathbb{P}(\{\text{Poisson}(\theta\, n) > n\}).$$

since $\mathbb{P}(A^c_{n,m})$ is non-increasing with n (see, *e.g.*, Penrose, 2003). Since $\theta < 1$, $\mathbb{P}(\{\text{Poisson}(\theta\, n) > n\}) \;\leq\; e^{-a\, n}$, where $a > 0$ is a constant independent of n.

To compute $\mathbb{P}(\tilde{A}^c_{n,m})$, a number of definitions are provided. Let $N_{n,m}$ denote the number of vertices that lie in the interior of $B_{n,m}$. Clearly, $\mathbb{E}[N_{n,m}] = \frac{\zeta_d\, \gamma^d_{\text{RRT}^*}}{\mu(X_{\text{free}})}\, \log n$, for all $m \in \{1, 2, \ldots, M_n\}$. For notational simplicity, define $\alpha := \frac{\zeta_d\, \gamma^d_{\text{RRT}^*}}{\mu(X_{\text{free}})}$. Let $\epsilon \in (0,1)$ be a constant independent of n. Define the event

$$C_{n,m,\epsilon} \;:=\; \{N_{n,m} \geq (1-\epsilon)\, \mathbb{E}[N_{n,m}]\} = \{N_{n,m} \geq (1-\epsilon)\, \alpha\, \log n\}$$

Since $N_{n,m,\epsilon}$ is binomially distributed, its large deviations from its mean can be bounded as follows (Penrose, 2003),

$$\mathbb{P}\left(C^c_{n,m,\epsilon}\right) = \mathbb{P}(\{N_{n,m,\epsilon} \leq (1-\epsilon)\, \mathbb{E}[N_{n,m}]\}) \leq e^{-\alpha\, H(\epsilon)\, \log n} = n^{-\alpha H(\epsilon)},$$

where $H(\epsilon) = \epsilon + (1-\epsilon)\log(1-\epsilon)$. Notice that $H(\epsilon)$ is a continuous function of ϵ with $H(0) = 0$ and $H(1) = 1$. Hence, $H(\epsilon)$ can be made arbitrary close to one by taking ϵ close to one.

Then,

$$\begin{aligned}
\mathbb{P}(\tilde{A}^c_{n,m}) \;&=\; \mathbb{P}(\tilde{A}^c_{n,m} \,|\, C_{n,m,\epsilon} \cap C_{n,m+1,\epsilon})\, \mathbb{P}(C_{n,m,\epsilon} \cap C_{n,m+1,\epsilon}) \\
&\quad + \mathbb{P}(\tilde{A}^c_{n,m} \,|\, (C_{n,m,\epsilon} \cap C_{n,m+1,\epsilon})^c)\, \mathbb{P}((C_{n,m,\epsilon} \cap C_{n,m+1,\epsilon})^c) \\
&\leq\; \mathbb{P}(\tilde{A}^c_{n,m} \,|\, C_{n,m,\epsilon} \cap C_{n,m+1,\epsilon})\, \mathbb{P}(C_{n,m,\epsilon} \cap C_{n,m+1,\epsilon}) + \mathbb{P}(C^c_{n,m,\epsilon}) + \mathbb{P}(C^c_{n,m+1,\epsilon}),
\end{aligned}$$

where the last inequality follows from the union bound.

First, using the spatial independence of the underlying point process,

$$\mathbb{P}\left(C_{n,m,\epsilon} \cap C_{n,m+1,\epsilon}\right) = \mathbb{P}\left(C_{n,m,\epsilon}\right)\, \mathbb{P}\left(C_{n,m+1,\epsilon}\right) \leq n^{-2\,\alpha\, H(\epsilon)}.$$

Second, observe that $\mathbb{P}(A^c_{n,m} \,|\, N_{n,m} = k, N_{n,m+1} = k')$ is a non-increasing function of both k and k', since the probability of the event $\tilde{A}_{n,m}$ can not increase with the

increasing number of points in both balls, $B_{n,m}$ and $B_{n,m+1}$. Then,

$$\begin{aligned}
&\mathbb{P}(\tilde{A}^c_{n,m} \,|\, C_{n,m,\epsilon} \cap C_{n,m+1,\epsilon}) \\
&\quad = \mathbb{P}(\tilde{A}^c_{n,m} \,|\, \{N_{n,m} \geq (1-\epsilon)\,\alpha\,\log N_{n,m}, N_{n,m+1} \geq (1-\epsilon)\,\alpha\,\log N_{n,m+1}\}) \\
&\quad \leq \mathbb{P}(\tilde{A}^c_{n,m} \,|\, \{N_{n,m} = (1-\epsilon)\,\alpha\,\log N_{n,m}, N_{n,m+1} = (1-\epsilon)\,\alpha\,\log N_{n,m+1}\}).
\end{aligned}$$

The term on the right hand side is one minus the probability that the maximum of $\alpha \log n$ number of uniform samples drawn from $[0,1]$ is smaller than the minimum of $\alpha \log n$ number of samples again drawn from $[0,1]$, where all the samples are drawn independently. This probability can be calculated as follows. From the order statistics of uniform distribution, the minimum of $\alpha \log n$ points sampled independently and uniformly from $[0,1]$ has the following probability distribution function:

$$f_{\min}(x) = \frac{(1-x)^{\alpha \log n - 1}}{\texttt{Beta}(1, \alpha\,\log(n))},$$

where $\texttt{Beta}(\cdot,\cdot)$ is the Beta function (also called the Euler integral) (Abramowitz and Stegun, 1964). The maximum of the same number of independent uniformly distributed random variables with support $[0,1]$ has the following cumulative distribution function:

$$F_{\max}(x) = x^{\alpha \log n}$$

Then,

$$\begin{aligned}
\mathbb{P}(\tilde{A}^c_{n,m} \,|\, C_{n,m,\epsilon} \cap C_{n,m+1,\epsilon}) &\leq \int_0^1 F_{\max}(x)\, f_{\min}(x)\, dx \\
&= \frac{\texttt{Gamma}((1-\epsilon)\,\alpha\,\log n)\,\texttt{Gamma}((1-\epsilon)\,\epsilon\,\log n)}{2\,\texttt{Gamma}(2(1-\epsilon)\,\alpha\,\log(n))} \\
&\leq \frac{((1-\epsilon)\,\alpha\,\log n)!\,((1-\epsilon)\,\alpha\,\log n)!}{2\,(2\,(1-\epsilon)\,\alpha\,\log n)!} \\
&= \frac{((1-\epsilon)\,\alpha\,\log n)!}{2(2(1-\epsilon)\,\alpha\,\log n)(2(1-\epsilon)\,\alpha\,\log n - 1)\cdots 1} \\
&\leq \frac{1}{2^{(1-\epsilon)\,\alpha\,\log n}} = n^{-\log(2)\,(1-\epsilon)\,\alpha},
\end{aligned}$$

where $\texttt{Gamma}(\cdot)$ is the gamma function (see Abramowitz and Stegun, 1964).

Then,

$$\mathbb{P}(\tilde{A}^c_{n,m}) \leq n^{-\alpha\left(2\,H(\epsilon)+\log(2)\,(1-\epsilon)\right)} + 2\,n^{-\alpha\,H(\epsilon)}.$$

Since $2\,H(\epsilon) + \log(2)\,(1-\epsilon)$ and $H(\epsilon)$ are both continuous and increasing in the interval $(0.5, 1)$, the former is equal to $2 - \log(4) > 0.5$ and the latter is equal to 1 as ϵ approaches one from below, there exists some $\bar{\epsilon} \in (0.5, 1)$ such that both

$2\,H(\bar{\epsilon}) + \log(2)\,(1-\bar{\epsilon}) > 0.5$ and $H(\bar{\epsilon}) > 0.5$. Thus,

$$\mathbb{P}(\tilde{A}^c_{n,m}) \leq n^{-\alpha/2} + 2\,n^{-\alpha/2} = 3\,n^{-\alpha/2}.$$

Hence,

$$\begin{aligned}\mathbb{P}(A^c_{n,m}) &\leq \mathbb{P}(\tilde{A}^c_{n,m}) + \mathbb{P}(\text{Poisson}(\theta\,n) > n) \\ &\leq 3\,n^{-\alpha/2} + e^{-a\,n}\end{aligned}$$

Recall that A_n denotes the event that $A_{n,m}$ holds for all $m \in \{1, 2, \ldots, M_n\}$. Then,

$$\mathbb{P}(A^c_n) = \mathbb{P}\left(\left(\bigcap\nolimits_{m=1}^{M_n} A_{n,m}\right)^c\right) = \mathbb{P}\left(\bigcup\nolimits_{m=1}^{M_n} A^c_{n,m}\right) \leq \sum_{m=1}^{M_n} \mathbb{P}\left(A^c_{n,m}\right) = M_n\,\mathbb{P}(A^c_{n,1}),$$

where the last inequality follows from the union bound. The number of balls in B_n can be bounded as

$$|B_n| = M_n \leq \beta\left(\frac{n}{\log n}\right)^{1/d},$$

where β is a constant. Combining this with the inequality above,

$$\mathbb{P}(A^c_n) \leq \beta\left(\frac{n}{\log n}\right)^{1/d}\left(3\,n^{-\alpha/2} + e^{-a\,n}\right),$$

which is summable for $\alpha > 2\,(1 + 1/d)$. Thus, by the Borel-Cantelli lemma, the probability that A^c_n occurs infinitely often is zero, *i.e.*, $\mathbb{P}(\limsup_{n\to\infty} A^c_n) = 0$, which implies that A_n occurs for all large n with probability one, *i.e.*, $\mathbb{P}(\liminf_{n\to\infty} A_n) = 1$. ■

Convergence to the optimal path: The proof of the following lemma is similar to that of Lemma 5.52, and is omitted here.

Let P_n denote the set of all paths in the graph returned by RRT* algorithm at the end of n iterations. Let σ'_n be the path that is closest to σ_n in terms of the bounded variation norm among all those paths in P_n, *i.e.*, $\sigma'_n := \min_{\sigma' \in P_n} \|\sigma' - \sigma_n\|$.

Lemma 5.69 *The random variable* $\|\sigma'_n - \sigma_n\|_{\text{BV}}$ *converges to zero almost surely,* i.e.,

$$\mathbb{P}\left(\{\lim_{n\to\infty} \|\sigma'_n - \sigma_n\|_{\text{BV}} = 0\}\right) = 1.$$

A corollary of the lemma above is that $\lim_{n\to\infty} \sigma'_n = \sigma^*$ with probability one. Then, the result follows by the robustness of the optimal solution (see the proof of Lemma 5.53 for details).

5.3.12 Asymptotic Optimality of k-nearest RRT*

In this section the proof of Theorem 5.25 is provided. The proof is almost identical to that of Theorem 5.24 given in detail in Section 5.3.11, with minor exceptions. Instead

of repeating the whole argument, here we only sketch the proof, providing detailed arguments in places with substantial differences. Again, for simplicity, we assume that the steering parameter η is large enough, *i.e.*, $\eta \geq \text{diam}(\mathcal{X})$.

As in the previous section, the incremental nature of sampling is represented by a marked point process, where each sample is assigned a mark that imposes a strict ordering of the samples. That is, $\{X_1, X_2, \ldots, X_n\}$ are assumed to be independent uniformly distributed samples drawn from $\mathcal{X}_{\text{free}}$, and $\{Y_1, Y_2, \ldots, Y_n\}$ are marks associated with the samples. Rather than their indices, the ordering of the samples is determined by their marks, *i.e.*, we consider X_i to be sampled before X_j if $Y_i < Y_j$. Consider the graph build by connecting each X_i to the k_i-nearest vertices among $\{X_j : j < i\}$, which is a subgraph of the k-nearest RRT* graph. We show that this subgraph contains a sequence of paths that converge to an optimal path.

We define a sequence $\{\sigma_n\}_{n\in\mathbb{N}}$ of paths and a sequence $\{B_n\}_{n\in\mathbb{N}}$ of sets of covering balls as in Section 5.3.11. The argument in the same section shows that each such ball contains a sample, almost surely. However, a different argument must be carried out to show that samples that fall into consecutive balls are connected by an edge. This can be done, as in the proof of Theorem 5.23 given in Section 5.3.10). More precisely, we first define a larger sequence $\{B'_n\}_{n\in\mathbb{N}}$ sets of balls, such that $B'_{n,m}$ contains $B'_{n,m}$ for all m and all n. Again, following the argument in Section 5.3.10, it can be shown that the outer balls in B_n contain no more than k samples (in fact, at any point up until iteration n), hence samples in subsequent balls are connected to each other. Following the argument in Lemma 5.68, it can be shown in each inner ball in B_n contains at least one vertex sampled in the correct order. Hence, an edge is generated between two the vertices in two subsequent balls in B_n, for all n.

Finally, to prove almost-sure convergence towards an optimal path, Lemma 5.52 is applied, as we have done in Section 5.3.11.

5.3.13 Asymptotic Optimality of μ-RRT*

In this section, we prove the asymptotic optimality of the μ-RRT* algorithm and its k-nearest variant. The proof is almost identical to the proof of Theorem 5.24. However, a number of technical difficulties arise. We outline these difficulties below, and explain how they can be overcome. In order not to repeat the arguments provided earlier, we refrain from providing a full proof.

Firstly, we shall show that, whenever the problem instance at hand admits a robustly optimal solution (infinite horizon path), say (σ_p^*, σ_s^*), there exists a sequence $\{(\sigma_{p,n}, \sigma_{s,n})\}_{n\in\mathbb{N}}$ of paths, all with the same trace as (σ_p^*, σ_s^*), and converges to (σ_p^*, σ_s^*). We have presented a similar result for the optimal path planning problem in Lemma 5.47, which must be extended to the optimal path planning with deterministic μ-calculus specifications. The extension is trivial, except that we need to ensure convergence towards two paths, σ_p^* and σ_s^*, at the same time, instead of only one.

Secondly, we shall define a suitable sequence $\{B_n\}_{n\in\mathbb{N}}$ of balls that cover the path $(\sigma_{p,n}, \sigma_{s,n})$, in a such a way that, whenever each ball in B_n contains a sample and samples in subsequent balls are connected, we obtain a path, say $(\sigma'_{p,n}, \sigma'_{s,n})$, that has the same trace with $(\sigma_{p,n}, \sigma_{s,n})$, thus with (σ_p^*, σ_s^*). In the context of optimal path

planning problem, *i.e.*, for the simple task specification of "reaching the goal region while avoiding the obstacles," it is trivial to show that this property holds when the balls are naively placed to cover σ_n. However, in the case of arbitrary deterministic μ-calculus specifications, the balls must be carefully placed. Such a placement follows that we have considered in the proof of Theorem 5.24, except at the boundaries of R_i the balls must be placed so that *(i)* in any given ball, all configurations satisfy the same atomic propositions, and *(ii)* all points in adjacent balls can be connected by an edge in the μ-RRT* algorithm.

Aside from these technical difficulties, the proof follows that of Theorem 5.24.

5.4 Proofs on Computational Complexity

5.4.1 Collision-checking Complexity of PRM

In this section we provide a proof for Lemma 5.28 by providing a problem instance that admits robustly optimal solution; However, the PRM algorithm calls the `CollisionFree` procedure at least order n^2 times in this particular problem instance, where n is the number of samples that the algorithm is run with.

Proof of Theorem 5.28 Consider a problem instance $(\mathcal{X}_{\text{free}}, x_{\text{init}}, \mathcal{X}_{\text{goal}})$, where $\mathcal{X}_{\text{free}}$ is composed of two openly-disjoint sets $\mathcal{X}_1$ and $\mathcal{X}_2$ (see Figure 5-13). The set $\mathcal{X}_2$ is a hyperrectangle-shaped set such that one of its sides has length $r/2$, where r is the connection radius of the PRM algorithm.

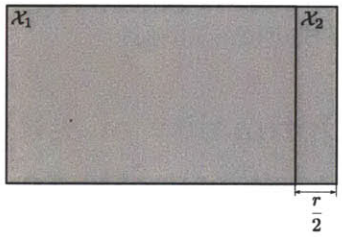


Figure 5-13: An illustration of $\mathcal{X}_{\text{free}} = \mathcal{X}_1 \cup \mathcal{X}_2$.

Any r-ball centered at a point in $\mathcal{X}_2$ will certainly contain some part of $\mathcal{X}_2$ with measure bounded away from zero. Define $\bar{\mu}$ as the volume of the smallest region in $\mathcal{X}_2$ that can be intersected by an r-ball centered at $\mathcal{X}_2$, *i.e.*, $\bar{\mu} := \inf_{x \in \mathcal{X}_2} \mu(\mathcal{B}_{x,r} \cap \mathcal{X}_1)$. Clearly, $\bar{\mu} > 0$.

Thus, for any sample X_n that falls into $\mathcal{X}_2$, the PRM algorithm will attempt to connect X_n to a certain number of vertices that lies in a subset $\mathcal{X}_1'$ of $\mathcal{X}_1$ such that $\mu(\mathcal{X}_1') \geq \bar{\mu}$. The expected number of vertices in $\mathcal{X}_1'$ is at least $\bar{\mu}\, n$. Moreover, none of these vertices can be in the same connected component with X_n. Thus, the algorithm attempts at least $\bar{\mu}\, n$ connections at each iterations, which implies that $\mathbb{E}[M_n^{\text{PRM}}]/n^2 > \bar{\mu}$. The result is obtained by taking the limit inferior of both sides. ■

5.4.2 Collision-checking Complexity of sPRM

In this section, we prove a stronger result than claimed in Lemma 5.29. We show that for *all* problem instances $P = (\mathcal{X}_{\text{free}}, x_{\text{init}}, \mathcal{X}_{\text{goal}})$, $\liminf_{n\to\infty} \mathbb{E}[M_n^{\text{sPRM}}/n] > 0$.

Proof of Lemma 5.29 Clearly, M_n, the number of calls to the `CollisionFree` procedure in iteration n, is equal to the number of nodes inside the ball of radius r centered at the last sample point X_n.

The number of calls to the `CollisionFree` procedure for each sample is precisely the number of (undirected) edges connected to that sample. Thus, the number of edges in the graph returned by the sPRM algorithm is precisely equal to twice the number of calls to the `CollisionFree` procedure (see Lines 5 and 6 in Algorithm 2).

Recall that $G_n^{\text{sPRM}} = (V_n^{\text{sPRM}}, E_n^{\text{sPRM}})$ denotes the graph returned by the sPRM algorithm when it is run with n samples. In the reminder of the proof, we find a lower bound for $\mathbb{E}[\text{card}\left(E_n^{\text{sPRM}}\right)]$.

Recall that we denote the connection radius of Algorithm 2 by r. Let $\bar{\mu}$ denote the volume of the smallest region that can be formed by intersecting $\mathcal{X}_{\text{free}}$ with an r-ball centered at a point inside $\mathcal{X}_{\text{free}}$, *i.e.*, $\bar{\mu} := \inf_{x\in\mathcal{X}_{\text{free}}} \mu(\mathcal{B}_{x,r} \cap \mathcal{X}_{\text{free}})$. Since $\mathcal{X}_{\text{free}}$ is the closure of an open set, $\bar{\mu} > 0$.

In sPRM, the edges are created by connecting each sample to other samples within a ball of radius r. The volume of the $\mathcal{X}_{\text{free}}$ that lies inside this ball is at least $\bar{\mu}$. Then, the number of edges connected to each sample is lower bounded by the value of a binomial random variable with parameters $\bar{\mu}/\mu(\mathcal{X}_{\text{free}})$ and n, since the underlying point process is binomial. Since this is true for each sample, $\mathbb{E}[\text{card}\left(E_n^{\text{sPRM}}\right)] \geq \frac{\bar{\mu}}{\mu(\mathcal{X}_{\text{free}})} n^2$. Then, $\mathbb{E}[M_n/n] \geq \frac{\bar{\mu}}{2\,\mathcal{X}_{\text{free}}} n^2$ for all large $n \in \mathbb{N}$. Taking the limit inferior of both sides gives the result. ■

5.4.3 Collision-checking Complexity of PRM*, RRG, and RRT*

In this section, we provide a proof of Lemma 5.30, which establishes that the number of calls to the `CollisionFree` procedure in all the proposed algorithms is at most $O(n \log n)$, where n is the number of samples that the algorithm is run with.

Proof of Lemma 5.30 First, consider PRM*. Recall that r_n denotes the connection radius of the PRM* algorithm. Recall that the r_n interior of $\mathcal{X}_{\text{free}}$, denoted by $\text{int}_{r_n}(\mathcal{X}_{\text{free}})$, is defined as the set of all points x, for which the r_n-ball centered at x lies entirely inside $\mathcal{X}_{\text{free}}$. Let A denote the event that the sample X_n drawn at the last iteration falls into the r_n interior of $\mathcal{X}_{\text{free}}$. Then,

$$\mathbb{E}\left[M_n^{\text{PRM}^*}\right] = \mathbb{E}\left[M_n^{\text{PRM}^*} \mid A\right] \mathbb{P}(A) + \mathbb{E}\left[M_n^{\text{PRM}^*} \mid A^c\right] \mathbb{P}(A^c).$$

Let $n_0 \in \mathbb{N}$ be the smallest number such that $\mu(\text{int}_{r_n}(\mathcal{X}_{\text{free}})) > 0$. Clearly, such n_0 exists, since $\lim_{n\to\infty} r_n = 0$ and $\mathcal{X}_{\text{free}}$ has non-empty interior. Recall that ζ_d is the volume of the unit ball in the d-dimensional Euclidean space and that the connection

radius of the PRM* algorithm is $r_n = \gamma_{\text{PRM}}(\log n/n)^{1/d}$. Then, for all $n \geq n_0$

$$\mathbb{E}\big[M_n^{\text{PRM}^*} \mid A\big] = \frac{\zeta_d \, \gamma_{\text{PRM}}}{\mu(\text{int}_{r_n}(\mathcal{X}_{\text{free}}))} \, n \log n.$$

On the other hand, given that $X_n \notin \text{int}_{r_n}(\mathcal{X}_{\text{free}})$, the r_n-ball centered at X_n intersects a fragment of $\mathcal{X}_{\text{free}}$ that has volume less than the volume of an r_n-ball in the d-dimensional Euclidean space. Then, for all $n > n_0$, $\mathbb{E}\big[M_n^{\text{PRM}^*} \mid A^c\big] \leq \mathbb{E}\big[M_n^{\text{PRM}^*} \mid A\big]$.

Hence, for all $n \geq n_0$,

$$\frac{\mathbb{E}\left[M_n^{\text{PRM}^*}\right]}{n \log n} \leq \frac{\zeta_d \, \gamma_{\text{PRM}}}{\mu(\text{int}_{r_n}(\mathcal{X}_{\text{free}}))} \leq \frac{\zeta_d \, \gamma_{\text{PRM}}}{\mu(\text{int}_{r_{n_0}}(\mathcal{X}_{\text{free}}))}.$$

Next, consider the RRG. Recall that η is the parameter provided in the `Steer` procedure (see Section 4.1). Let D denote the diameter of the set $\mathcal{X}_{\text{free}}$, *i.e.*, $D := \sup_{x,x' \in \mathcal{X}_{\text{free}}} |x - x'|$, where $|\cdot|$ denotes the usual Euclidean norm. Clearly, whenever $\eta \geq D$, $V^{\text{PRM}^*} = V^{\text{RRG}} = V^{\text{RRT}^*}$ surely, and the claim holds.

To prove the claim when $\eta < D$, let C_n denote the event that for any point $x \in \mathcal{X}_{\text{free}}$ the RRG algorithm has a vertex $x' \in V_n^{\text{RRG}}$ such that $|x - x'| \leq \eta$. As shown in the proof of Theorem 5.22 (see Lemma 5.60), there exists $a, b > 0$ such that $\mathbb{P}(C_n^c) \leq a\, e^{-b\, n}$. Then,

$$\mathbb{E}\left[M_n^{\text{RRG}}\right] = \mathbb{E}\left[M_n^{\text{RRG}} \,\middle|\, C_n\right] \mathbb{P}(C_n) + \mathbb{E}\left[M_n^{\text{RRG}} \,\middle|\, C_n^c\right] \mathbb{P}(C_n^c),$$

Clearly, $\mathbb{E}\left[M_n^{\text{RRG}} \,\middle|\, C_n^c\right] \leq n^2$, since it is bounded by the maximum number of edges. Hence, the second term of the sum on the right hand side converges to zero as n approaches infinity. On the other hand, given that C_n holds, the new vertex that will be added to the graph at iteration n, if such a vertex is added at all, will be the same as the last sample, X_n. To complete the argument, given any set of n points placed inside $\mu(X_{\text{free}})$, let N_n denote the number of points that are inside a ball of radius r_n that is centered at a point X_n sampled uniformly at random from $\mu(X_{\text{free}})$. The expected number of points inside this ball is no more than $\frac{\mu(\zeta_d \, r_n^d)}{\mu(X_{\text{free}})} \log n$. Hence, $\mathbb{E}[M_n^{\text{RRG}} \mid C_n] < \frac{\zeta_d \, \gamma_{\text{PRM}})}{\mu(X_{\text{free}})} n \log n$, which implies the existence of a constant $\phi_1 \in \mathbb{R}_{\geq 0}$ such that $\limsup_{n \to \infty} \mathbb{E}[M_n^{\text{RRG}}]/(n \log n) \leq \phi_1$.

Since $M_n^{\text{RRT}^*} = M_n^{\text{RRG}}$ holds surely, $\limsup_{n \to \infty} \mathbb{E}[M_n^{\text{RRG}}]/(n \log n) \leq \phi_1$ also. ■

5.4.4 Time Complexity of Incremental Model Checking

In this section, we prove Lemma 5.31, which characterizes the computational complexity of the incremental model checking algorithm proposed in Section 4.3.4.

Before providing the proof, we establish a number of intermediate results.

Lemma 5.70 *Let ϕ be a deterministic μ-calculus formula. Then, the number of subformulas of ϕ is at most order $|\phi|$,* i.e., $\text{card}\left(\{\psi \in L_1 : \psi \leq \phi\}\right) \in O(|\phi|)$.

Proof Consider the parse tree of ϕ. Each subformula ψ corresponds to a sub-tree, rooted at the final operator in ψ. The number of such sub-trees in the parse tree of ϕ is at most order the total number of appearances of operators, atomic propositions, and variables in ϕ. Thus, the result follows. ■

Lemma 5.71 *The size of the game arena* $A = (\mathcal{V}_1, \mathcal{V}_2, \mathcal{E})$ *of the* L_1 *game of* $\mathcal{K} = (\mathcal{S}, s_{\text{init}}, \mathcal{R}, \mathcal{L})$ *satisfies* $\text{card}(\mathcal{V}_1 \cup \mathcal{V}_2) \in O(\text{card}(\mathcal{S})\ |\phi|)$ *and* $\text{card}(\mathcal{E}) \in O(\text{card}(\mathcal{R})\ |\phi|)$.

Proof Let $\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2$. Let $\text{SF}(\phi)$ denote the set of subformulas of ϕ. By definition, $\mathcal{V} \subseteq \mathcal{S} \times \text{SF}(\phi)$. Hence, by Lemma 5.70, we have $\text{card}(\mathcal{V}_1 \cup \mathcal{V}_2) \in O(\text{card}(\mathcal{S})\ |\phi|)$.

We have that $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V} = (\mathcal{S} \times \text{SF}(\phi)) \times (\mathcal{S} \times \text{SF}(\phi))$. However, we have $(v, v') \in \mathcal{E}$, where $v = (s, \psi)$ and $v' = (s', \psi')$, only if $(s, s') \in \mathcal{R}$. Thus, $\text{card}(\mathcal{E}) \in O(\text{card}(\mathcal{R})\ |\phi|)$. ■

Lemma 5.72 *Suppose the incremental model checking algorithm is run, with input* ϕ *as the formula, to generate the Kripke structure* $\mathcal{K} = (\mathcal{S}, s_{\text{init}}, \mathcal{R}, \mathcal{L})$. *Then, the number of* ν*-vertices stored in the global data structure* `Nv` *(see the paragraph on global data structures in Section 4.3.4) satisfies* $\text{card}(\texttt{Nv}) \in O(\log(\text{card}(\mathcal{S})))$.

Proof Consider the incremental process generating $\mathcal{K}$. Initially, $\texttt{Nv} = \emptyset$. The only place that the set `Nv` is altered is in Lines 9-10 in Algorithm 9. Notice that these lines are executed at most once for any vertex $(s, \psi) \in \mathcal{V}_1 \cup \mathcal{V}_2$ of the arena $A = (\mathcal{V}_1, \mathcal{V}_2, \mathcal{E})$ of the L_1 game of $\mathcal{K}$ and ϕ. By Lemma 5.71, $\text{card}(\mathcal{V}_1 \cup \mathcal{V}_2) \in O(\text{card}(\mathcal{S}))$. However, not every vertex in $\mathcal{V}_1 \cup \mathcal{V}_2$ is added to `Nv`. Only those that are ν-vertices and those that pass the test of `AcceptVertex`$(A, (s, \psi))$ procedure are inserted. Recall that `AcceptVertex` is a randomized procedure that returns `True` with probability $1/k$, independent for each call, where k is the number of times that the procedure is called. Then, the expected number of times that it returns `True` is the Harmonic number, which is upper bounded by $\log(k)$. Since $k \leq \text{card}(\mathcal{S})\ \text{card}(\text{SF}_\nu(\phi))$, the result follows. ■

Proof of Lemma 5.31 The `AddState` procedure (see Algorithm 7) only executes a single line and returns. Hence, $N_{\mathcal{K}}^{\texttt{AS}} \in O(\text{card}(\mathcal{S}))$.

The argument for characterizing the complexity of $N_{\mathcal{K}}^{\texttt{AD}}$ is more involved. Each time it is called, the procedure runs the `UpdateArena` and `UpdateGlobalDataStructures` a number of times. Let us analyze the computational complexity induced by these procedures separately.

Firstly, the `UpdateArena` procedure updates the arena $A = (\mathcal{V}_1, \mathcal{V}_2, \mathcal{E})$ and the global data structures `Nv` and `Lv`. Note that, any line executed by this algorithm is executed at most a constant number of times for each edge in $\mathcal{E}$, which satisfies $\mathcal{E} \in O(\text{card}(\mathcal{R})\ |\phi|)$ by by Lemma 5.71. Thus, in the process of building the Kripke structure $\mathcal{K}$, the number of total simple computational operations performed by the `UpdateArena` procedure is $O(\text{card}(\mathcal{R})\ |\phi|)$.

Secondly, the `UpdateGlobalDataStructures` is also run for each edge at most a constant number of times, thus $O(\text{card}(\mathcal{R})\ |\phi|)$ times. However, the number of times

its for loop is run depends on the size of $\texttt{RV}(s, \psi)$, which is at most the number of ν-vertices in `Nv`, which in turn is bounded by $O(\log(\text{card}\,(\mathcal{S})))$ by Lemma 5.72. Hence, the number of total simple computational operations performed by the procedure `UpdateGlobalDataStructures` is $O(\log(\text{card}\,(\mathcal{S})\,\text{card}\,(\text{SF}_\nu(\phi)))\,\text{card}\,(\mathcal{R})\;|\phi|)$. ■

5.4.5 Time Complexity of the Algorithms

In the section, the proofs of Theorems 5.32, 5.33, and 5.34 are provided. Collectively, these theorems characterize the computational complexity of all algorithms introduced in Chapter 4, in terms of the number of simple computational operations performed when the algorithm is run with n samples. In this context, a simple computational operation is one that takes a constant number of clock ticks (independent of n) on the central processing unit, such as additions, multiplications, or comparisons.

The proofs follow easily from the lemmas and the discussions given in Section 5.4. Nevertheless, they are provided below for completeness.

Proof of Theorem 5.32 By Lemmas 5.28 and 5.29, both the PRM and the sPRM algorithms spend at least order of n^2 expected time in executing the `CollisionFree` procedure. Thus, the result follows. ■

Proof of Theorem 5.33 The result follows directly from the fact that both algorithms take at least order $n \log n$ time to execute nearest neighbor queries. ■

Proof of Theorem 5.34 Since we would like to provide an upper bound on computational complexity, we have to take into account all the steps executed by the algorithm. By Lemma 5.30, the expected number of calls to `CollisionFree` procedure is at most $n \log n$. Since execution time of the `CollisionFree` procedure is a constant that is independent of n, the expected number of simple operations executed for collision checking is at most order $n \log n$.

The `Nearest` and `Near` procedures are called only a constant number of times for each sample. Thus, they are called at most n times, and the procedure takes at most $\log n$ expected time to execute in each such call. Thus, the expected number of simple operations executed for `Nearest` and `Near` procedures is at most order $n \log n$.

It is easy to see that any other intermediate operation requires no more than order $n \log n$ expected number of simple operations to be executed.

This argument establishes that the asymptotic computational complexity of the PRM*, RRG, and the RRT* algorithms is $O(n \log n)$. The μ-RRT and the μ-RRT* algorithms, on the other hand, also call the `AddState` and `AddTransition` procedures of the incremental model checking and synthesis algorithm for the deterministic μ-calculus. The asymptotic computational complexity added by these procedures depends on the `AcceptVertex` procedure. Since we accept the kth vertex with probability $1/k$, an expected number of $O(\log n)$ vertices are accepted. This renders the total complexity of the μ-RRT and the μ-RRT* algorithms becomes $O(n \log^2 n)$. ■

Chapter 6

Conclusions and Remarks

6.1 Summary

Sampling-based motion planning have received increasing attention during the last decade. A number of algorithmic approaches were developed and rigorously analyzed. In particular, some of the leading paradigms, such as the Probabilistic RoadMaps (PRMs) and the Rapidly-exploring Random Trees (RRTs), have been demonstrated on several robotic platforms, and found applications in diverse fields well outside the robotics domain, ranging from computational geometry to drug discovery. However, a large portion of this intense research effort has been limited to the classical feasible path planning problem, which asks for a path that starts from an initial configuration and reaches a goal configuration while avoiding collision with obstacles.

In this dissertation, we have presented a new class of algorithms that extend the application domain of sampling-based algorithms to two novel directions: optimal path planning and path planning with complex task specifications. Moreover, we have analyzed some of the widely-used existing sampling-based algorithms in terms of probabilistic completeness, computational complexity, and asymptotic optimality.

The optimal path planning problem asks for a path that solves the feasible path planning problem while minimizing a cost function. Algorithms tailored to solve the optimal path planning problem can provide high quality solutions, which may be invaluable in a number of applications in robotics and beyond. In this text, we have first shown that naive extensions of the existing algorithms fail to properly address the optimal path planning problem. More precisely, we have shown that the existing algorithms either lack asymptotic optimality, i.e., almost-sure convergence to optimal solutions, or they lack computational efficiency: on one hand, neither the RRT nor the k-nearest PRM (for any fixed k) is asymptotically optimal; on the other hand, the simple PRM algorithm, where the connections are sought within fixed radius balls, is not computationally as efficient as the RRT or other PRM variants. Subsequently, we have proposed two novel algorithms, called PRM* and RRT*, that guarantee asymptotic optimality, without sacrificing computational efficiency. In fact, we have shown that the proposed algorithms have the same asymptotic computational complexity when compared to the most efficient existing algorithms, such as the RRT.

The feasible path planning problem asks for a path that satisfies a simple specification: "reach the goal region while avoiding collision with obstacles." We have considered an extension of this problem with complex task specifications given in the form of deterministic μ-calculus. Example specifications include reachability, safety, ordering, liveness, as well as a logical and temporal combination of there off. To address this problem, we have proposed an incremental sampling-based algorithm that is provably correct and probabilistically complete. That is, roughly speaking, the proposed algorithm generates a correct-by-design path that satisfies the given specification, when such a path exists, with probability approaching one as the number of samples increases to infinity. This algorithm is composed of two key ingredients, which may be of independent interest. First is a novel incremental sampling-based algorithm, called the RRG, that generates a representative set of paths in the form of a graph, with guaranteed convergence to feasible paths. Second is a novel incremental local model-checking algorithm for the deterministic μ-calculus. Moreover, with the help of these tools and the ideas behind the RRT* algorithm we have constructed a sampling-based algorithm that also guarantees asymptotic optimality.

6.2 Current and Future Work

During the past couple of years, PRM*, RRT*, and their variants have spurred a growing interest. Recent research has combined the RRT*-like connection strategy, *i.e.*, connection to $O(\log n)$ neighbors, with other, often novel, algorithmic components to address a variety of problems. Just to name a few, an efficient sampling-based algorithm that guarantees convergence to a constant-factor-optimal solution has been proposed by Marble and Bekris (2012). A class of planning problems that involve sensor noise have been addressed by Bry and Roy (2011). RRT* variants that can trade off rapid exploration and fast convergence towards optimality were proposed by Altorevitz et al. (2011). Massively parallel versions of the RRT* algorithm was proposed by Bialkowski et al. (2011). The RRT* algorithm was extended to handle a class of planning problem on manifolds by Jaillet and Porta (2012) and to deal with a certain class of differential constraints by Webb and van den Berg (2012).

In the this section, we review some of our recent work in sampling-based algorithms that are inspired by those presented in this dissertation. First, we discuss the asymptotically-optimal sampling-based planning under differential constraints. Second, we review some of our recent work on extending the domain of sampling-based algorithms to control and estimation problems, including differential games, stochastic optimal control, and optimal filtering. In both cases, we point a number of open problems and directions for future work.

6.2.1 Sampling-based Planning on sub-Riemannian Manifolds

So far, we have only considered path planning problems, where we ignored the constraints implied by the dynamics governing the robot. The introduction of dynamics, often represented by a set of differential constraints, changes the nature of a motion planning problem significantly. In fact, in most cases, the algorithms proposed in this dissertation can not be directly applied to problems involving differential constraints.

Although important advances were made in addressing optimal motion planning problems with (non-trivial) differential constraints, many important problems remain open. In this section, we discuss two important variations of the optimal path planning problem. The first variation, discussed in Section 6.2.1, relaxes the implicit requirement that the free space has non-empty interior. The second variant, presented in Section 6.2.1, introduces differential constraints into the problem definition. In both cases, we outline the necessary modifications for the proposed algorithms proposed to address the said extensions, and point out some of the open problems.

Path Planning on Riemannian Manifolds

In this section, we generalize the path planning problem. In the generalized version, the configuration space is a Riemannian manifold rather than the unit cube. First, we introduce some fundamental concepts from Riemannian geometry. Subsequently, we define the problem of path planning on Riemannian manifolds. Finally, we discuss a number of modifications to the proposed algorithms in order to address this problem. This section mainly serves as a precursor to the next section, where we extend the planning problems considered in Chapter 3 with differential constraints.

Riemannian Geometry: Below we introduce some fundamental concepts in Riemannian geometry. Our presentation is brief; the interested reader is referred to the standard texts, such as (Boothby, 1975; do Carmo, 1992), for a detailed exposition. Our notation is fairly standard and closely follows that of Boothby (1975).

Let $n, m \in \mathbb{N}$. Let $V \subseteq \mathbb{R}^n$ and $W \subseteq \mathbb{R}^m$ be two open sets. A function $f : V \to W$ is said to be *smooth* if all its partial derivatives exist and are continuous. The function f is said to be a *diffeomorphism* if f is a bijection and both f and its inverse f^{-1} are smooth functions, in which case V and W are said to be *diffeomorphic*.

A set $M \subseteq \mathbb{R}^n$ is said to be a *smooth d-dimensional manifold*, if, for all $p \in M$, there exists an open set $V \subset \mathbb{R}^n$ such that $V \cap M$ is diffeomorphic to an open set $V' \subset \mathbb{R}^d$.[1] An *interval*, often denoted by I, is a convex subset of $\mathbb{R}$. A *smooth curve* on M is a smooth function $\gamma : I \to M$ for some interval I that includes the origin.

Given a point $p \in M$, a vector $v \in \mathbb{R}^n$ is said to be a *tangent vector* of M at p, if there exists a smooth curve $\gamma : \mathbb{R} \to M$ such that $\gamma(0) = p$ and $\dot{\gamma}(0) = v$. The *tangent space* of M at p is defined as $T_pM := \{\dot{\gamma}(0) \,|\, \gamma \text{ is a smooth curve on } M \text{ and } \gamma(0) = p\}$.

A smooth function $Y : M \to \mathbb{R}^n$ is called a *smooth vector field* on M, if $Y(p) \in T_pM$ for all $p \in M$. The set of all smooth vector fields on M is denoted by $VF(M)$.

A *Riemannian metric* on M is a family of positive-definite inner products, denoted by $g_p := T_pM \times T_pM \to \mathbb{R}$, parametrized by $p \in M$, such that, for any two smooth vector fields $Y, Y' \in VF(M)$, the function $p \mapsto g_p(Y(p), Y'(p))$ is smooth as a function from M to $\mathbb{R}$. A smooth manifold M endowed with a Riemannian metric g is called a *Riemannian manifold*. The pair (M, g) is called a *Riemannian geometry*.

[1] In general, a smooth manifold is a Hausdorff space with a countable basis such that any point on the manifold has a neighborhood that is diffeomorphic to an open subset of the Euclidean space (see Munkres, 2000). For all practical purposes, however, in this text we are only interested in manifolds that can be embedded into an Euclidean space.

Let $|\cdot|$ denote the norm induced by the Riemannian metric g, *i.e.*, $|v| = \sqrt{g_p(v,v)}$ for all $v \in T_pM$ and all $p \in M$. Then, the length of a curve $\gamma : I \to M$ is defined as

$$L(\gamma) := \int_I |\dot{\gamma}(t)|\, dt.$$

The *Riemannian distance* between $p, p' \in M$ is defined as

$$d(p,p') := \inf \big\{ L(\gamma) : \gamma \text{ is a smooth curve with } \gamma(0) = p \text{ and } \gamma(1) = p' \big\}.$$

Define the *Riemannian ball* of radius ϵ as

$$B(p,\epsilon) := \big\{ p \in M : d(p,p') \leq \epsilon \big\}.$$

Path Planning Problems on Riemannian Manifolds: In what follows, we extend the path planning problems defined in Chapter 3 with a configuration space that is represented by a Riemannian manifold. Let $M \subset \mathbb{R}^n$ be a d-dimensional Riemannian manifold. Let $\mathcal{X}_{\text{obs}}, \mathcal{X}_{\text{goal}} \subset M$ be the goal set and the obstacle set, respectively. Suppose that both sets are open relative to M. Define the free space as $\mathcal{X}_{\text{free}} := M \setminus \mathcal{X}$. Let $x_{\text{init}} \in M$ denote the initial configuration. Then, the feasible path planning problem on a Riemannian manifold M is to find a finite-length Riemannian curve that *(i)* starts from the initial configuration, *(ii)* reaches a final configuration, and *(iii)* stays entirely within the free space.

Other path planning problems extend similarly. The new class of problems have two major differences when compared to those we presented in Chapter 3. Firstly, the configuration space is a d-dimensional Riemannian manifold, instead of $(0,1)^d$. Second, we seek a finite-length Riemannian curve on the configuration space manifold, rather than a path, *i.e.*, a continuous function with bounded variation.

Example Clearly, $\mathbb{R}^n$ is an n-dimensional manifold since it is diffeomorphic to itself. One of the most basic examples of a non-trivial manifold is the unit circle defined as $S^1 := \{(x_1, x_2) \in \mathbb{R}^n \,|\, \sqrt{x_1^2 + x_2^2} = 1\} \subset \mathbb{R}^2$. The configuration space of an n-link planar manipulator can be represented by its n joint angles, hence the manifold formed by the Cartesian product of n unit circles, *i.e.*, $M = S^1 \times S^1 \times \cdots \times S^1$. A more compact representation is $M' = [0, 2\pi]^n$. Note that, although M and M' are embedded in $\mathbb{R}^{2n}$ and $\mathbb{R}^n$, respectively, both manifolds are of the same dimension.

Examples that are more complex include closed kinematic chains. Consider, for example, the n-link closed planar chain with both of the end effectors pinned down to the origin of the Euclidean plane. In that case, the manifold M that represents the configuration space is a subset of $S^1 \times S^1 \times \cdots \times S^1$ such that any point $p \in M$ respects the constraint that the end effector is at the origin. That is, $M := \{(\alpha_1, \alpha_2, \ldots, \alpha_n) \in [0, 2\pi]^n : \sum_{i=1}^n \sin(\alpha_i) = 0 \text{ and } \sum_{i=1}^n \cos(\alpha_i) = 0\}$. Clearly, the manifold M is a measure-zero subset of $\mathbb{R}^n$, the Euclidean space with smallest dimensionality that M can be embedded into. The interested reader is referred to (Trinkle and Milgram, 2002; Milgram and Trinkle, 2004; Cortes and Simeon, 2005) for similar examples. ■

Extensions of Algorithms: The novel asymptotically-optimal algorithms presented in Chapter 4 can not be used directly solve path planning problems on arbitrary Riemannian manifolds. Below, we discuss the necessary modifications that enable them to address the new class of problems. The modifications occur only in the primitive procedures presented in Section 4.1. We redefine the primitive procedures `Sample`, `SampleFree`, `Nearest`, `Near`, and `Steer` defined in Section 4.1 as follows.

The `Sample` and the `SampleFree` procedures draw independent identically distributed samples from M and $M \setminus \mathcal{X}_{\text{obs}}$, respectively. Given a configuration $x \in M$ and a finite set $V \subset M$ of configurations, the `Nearest` procedure returns the configuration that is closes to x among those in V, with respect to the Riemannian distance function, *i.e.*, $\texttt{Nearest}(V, x) := \arg\min_{v \in V} d(x, v)$, where $d(\cdot, \cdot)$ denotes the Riemannian distance function. Hence, the only difference is that the distances are evaluated in terms of the Riemannian distance function, rather than the Euclidean distance. All the remaining procedures, namely `Near` and `Steer`, are modified in the same manner.

Finally, the `CollisionFree` procedure is implemented such that, instead of the straight path, it checks whether the minimum-length Riemannian curve connecting the two given configurations lies entirely in the free space.

Given these modifications for the primitive procedures, it can be shown that the proposed algorithms achieve asymptotic optimality, while preserving computational efficiency, for the optimal path planning problem on a Riemannian manifold.

Arguably, however, sampling from arbitrary Riemannian manifolds and computing Riemannian distances, in a computationally efficient manner, is not a trivial task. In fact, the importance of the problem of computing nearest neighbors with respect to the Riemannian distance has not gone unnoticed: Yershova and LaValle (2007) have proposed an algorithm based on kd-trees. Technical difficulties also arise when computing minimum-length curves on a Riemannian manifold. Yet, it is relatively easier to compute feasible curves on the manifolds studied in (Yershova and LaValle, 2007). Similarly, the task of drawing independent samples, for example, according to the uniform distribution, is an easy task in many examples that arise in robotics, *e.g.*, in all examples discussed in (Yershova and LaValle, 2007), although technical challenges may arise in a number of other examples.

This discussion may be considered an outline of a program to design asymptotically-optimal and computationally-efficient sampling-based algorithms for path planning problems on Riemannian manifolds. As we have outlined above, many challenges remain. It is worth noting at this point that the modifications we propose are by no means necessary, but they are sufficient. In fact, one may be able to relax a number of these requirements, and still guarantee both asymptotic optimality and computational efficiency. For instance, a suitable approximation of the Riemannian distance may suffice, rather than exact computations.

Let us note that, very recently, Jaillet and Porta (2012) discussed the extension of the RRT* algorithm for the optimal path planning problem on a large class of Riemannian manifolds. Their work considers a certain class of manifolds represented by a set of inequalities. The proposed algorithm, called the AtlasRRT*, constructs a family of local parametrizations, also called an atlas, currently with the RRT* tree.

Trajectory Planning and Sub-Riemannian Geometry

In this section, we introduce a problem that extends the classical path planning problem with differential constraints. Next, we introduce a number of fundamental concepts in sub-Riemannian geometry. Subsequently, we point out an important connection between the trajectory planning problem and a certain motion planning problem on sub-Riemannian manifolds. The latter problem leads us to a number of modifications to the proposed algorithms necessary to maintain asymptotic optimality and computational efficiency. Finally, we review some of the open problems.

Trajectory Planning Problems: Consider the following dynamical system:

$$\dot{x}(t) = f(x(t), u(t)), \qquad x(0) = x_0 \tag{6.1}$$

where $x(t) \in X \subseteq \mathbb{R}^n$, $u(t) \in U \subseteq \mathbb{R}^m$, $f : X \times U \to \mathbb{R}^n$ is Lipschitz continuous in both of its arguments. The dynamical system is said to be *smooth* if $f(\cdot, \cdot)$ is a smooth function, *i.e.*, all partial derivatives of f exists and are continuous. A trajectory $x : [0, T] \to X$ is said to be *dynamically feasible*, if there exists a Lebesgue-measurable input signal $u : [0, T] \to U$ such that u and x satisfy Equation (6.1).

The state $x_0 \in X$ (see Equation (6.1)) is called the *initial state*. Let $X_{\text{obs}} \subset X$ and $X_{\text{goal}} \subset X$, called the *obstacle set* and the *goal set*, be open relative to X. Let $\texttt{Cost} : \Sigma \to \mathbb{R}_{\geq 0}$ be a *cost function* that assigns each non-trivial trajectory a non-zero cost. Then, the *trajectory planning problem* is to find a dynamically-feasible trajectory $x : [0, T] \to X$ that (i) starts from the initial state, *i.e.*, $x(0) = x_0$, (ii) avoids collision with obstacles, *i.e.*, $x(t) \notin X_{\text{obs}}$ for all $t \in [0, T]$, and (iii) reaches the goal region, *i.e.*, $x(T) \in X_{\text{goal}}$. The *optimal trajectory planning problem* is to find a trajectory $x^* : [0, T] \to X$ that solves the trajectory planning problem while minimizing the cost function c. The feasible and optimal trajectory problems extend the feasible and optimal path planning problems, given in Problems 3.1 and 3.2, with dynamics. Similarly, trajectory planning problems with complex task specifications can also be defined as extensions of Problems 3.3 and 3.4.

We use the term *motion planning* collectively for trajectory planning and path planning problems. In the literature, trajectory planning problems are often called planning problems with differential constraints (*e.g.*, in LaValle, 2006); in this case, the differential constraint is the constraint imposed by Equation (6.1).

Note that trajectory planning is a generalization of path planning. With the (trivial) differential constraint $\dot{x} = u$, where $x \in X$, $u \in U$, and $X = U = [0, 1]^d$, we recover the path planning problems described in Chapter 3. In fact, a larger class of trajectory planning problems are equivalent planning problems on Riemannian manifolds, in which case the algorithms we have proposed in Chapter 4 can be applied with minimal changes (see the discussion in Section 6.2.1).

A more interesting class of trajectory planning problems are those that can not be directly reduced to a planning problem on a Riemannian manifold. These are exactly those problems in which the degree of non-holonomy of the underlying system, defined in the differential-geometric sense, is greater than unity. Such systems

can be fully characterized using sub-Riemannian geometry. Below, we define the degree of non-holonomy of a system, after introducing a number of preliminaries from sub-Riemannian geometry. Subsequently, we outline a number of modifications to the proposed algorithms in order to address a certain class of trajectory planning problems. Along the way, we provide a number of examples to illustrate some of the important connections between sub-Riemannian geometry and planning under differential constraints. These connections were partially uncovered in seminal publications on non-holonomic planning, such as (Laumond et al., 1998).

Sub-Riemannian Geometry: In Section 6.2.1, we briefly introduced a number of fundamental notions in Riemannian geometry, including the definitions of smooth manifolds, smooth curves, tangent spaces, smooth vector fields, and Riemannian metrics, which lead to the definition of Riemannian geometry. In this section, we build upon those definitions by introducing a number of new concepts, which help us define a sub-Riemannian geometry. Along the way, we point out the links between sub-Riemannian geometry and trajectory planning problems. For a thorough exposition to sub-Riemannian geometry, the reader is referred to the monographs by Strichartz (1986); Bellaiche (1996); Gromov (1996); Montgomery (2002). Our notation closely follows that of Montgomery (2002).

Let $k, l \in \mathbb{N}$. A set $E \subset M \times \mathbb{R}^l$ is said to be a *smooth vector bundle* of rank k over the manifold M, if the set $E_p := \{v \in \mathbb{R}^l \,|\, (p, v) \in E\}$ is a k-dimensional linear subspace of $\mathbb{R}^l$. The set E_p is also called the *fiber* of E over p. In particular, the vector bundle $TM := \{(p, v) \,|\, p \in M, v \in T_pM\}$ is called the *tangent bundle* of M. Note that its fiber, T_pM, is an m-dimensional linear subspace of $\mathbb{R}^n$. A *subbundle* of a vector bundle is a subset that is a vector bundle on its own right.

A *distribution* on a manifold M is a subbundle $\mathcal{H}$ of the tangent bundle TM. A *sub-Riemannian geometry* on a manifold M is a distribution $\mathcal{H}$ on M together with an inner product $g(\cdot, \cdot) : \mathcal{H} \times \mathcal{H} \to \mathbb{R}_{\geq 0}$. The set $\mathcal{H}$ is also called the *horizontal distribution*. The fiber of $\mathcal{H}$ at a point $p \in M$ is denoted by $\mathcal{H}_p$. A curve $\gamma : I \to M$ defined on the manifold M is said to be a *horizontal curve*, if it is tangent to $\mathcal{H}$, *i.e.*, $\dot{\gamma}(t) = \mathcal{H}_{\gamma(t)}$ for all $t \in I$.

The length of a smooth horizontal curve $\gamma : I \to M$ is defined as in the case of Riemannian geometry, *i.e.*, $L(\gamma) := \int_I |\dot{\gamma}(t)|\, dt$. The *sub-Riemannian distance* between two points $p, p' \in M$ is defined as

$$d_s(p, p') := \inf \left\{ L(\gamma) : \gamma \text{ is a smooth horizontal curve with } \gamma(0) = p \text{ and } \gamma(1) = p' \right\}$$

Define the sub-Riemannian ball of radius ϵ as $B_s(p, \epsilon) := \left\{ p \in M : d_s(p, p') \leq \epsilon \right\}$.

Contrast these definitions with the definition of the Riemannian distance and the Riemannian ball given in Section 6.2.1. In the computation of the sub-Riemannian distance, the curve connecting the two points is forced to be horizontal, thus forced to respect the constraints in the tangent space posed by the horizontal distribution.

Example Suppose the configuration space of a robot is represented by a manifold M. Then, the dynamics governing this robot, *i.e.*, constraints on its velocities while moving within its configuration space, can be represented by a distribution.

Consider, for instance, the Reeds-Shepp vehicle (see Reeds and Shepp, 1990). Its configuration space can be represented by the manifold $M = \mathbb{R}^2 \times S^1$ encoding its planar position and its orientation. To succinctly describe its dynamics, define the state variables (x, y, θ), where $(x, y) \in \mathbb{R}^2$ describes the position and $\theta \in [-\pi, \pi]$ describes the orientation of the vehicle. Then, its dynamics satisfies

$$\begin{pmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{\theta} \end{pmatrix} = f\left(\begin{pmatrix} x(t) \\ y(t) \\ \theta(t) \end{pmatrix}, \begin{pmatrix} u_1(t) \\ u_2(t) \end{pmatrix}\right) = \begin{pmatrix} u_1(t)\cos(\theta(t)) \\ u_1(t)\sin(\theta(t)) \\ u_1(t)\,u_2(t) \end{pmatrix},$$

where $|u_1(t)|, |u_2(t)| \leq 1$. The input signals u_1 and u_2 denote the forward velocity and the steering angle, respectively.

The right hand side of the equation above induces a distribution $\mathcal{H}$ on the manifold M. More precisely, $\mathcal{H}_p$ is the family of vector fields $Y(p) = \{f(p, u) : u \in U\}$, for all $p \in M$. At the origin, *i.e.*, at $(x, y, \theta) = (0, 0, 0)$, this distribution is spanned by the vectors $(1, 0, 0)$ and $(0, 0, 1)$. These vectors span a two dimensional plane along the coordinates x and θ. This merely indicates that the car can move in the longitudinal direction and it can change its orientation; but it can not move sideways, as the vector $(0, 1, 0)$ is not included in the span of the distribution at the origin. ■

Two important results in sub-Riemannian geometry are the Chow's theorem and the Ball-Box theorem (Gromov, 1996). In what follows, we state these theorems, after introducing a number of new definitions. Along the way, we define the degree of non-holonomy for a horizontal distribution.

Let Y, Z be two vector fields defined on a smooth manifold M. The *Lie bracket* of Y, Z is a vector field defined as

$$[Y, Z](p) := \dot{Z}(p)Y(p) - \dot{Y}(p)Z(p),$$

for all $p \in M$, where $\dot{Y}(p) : T_pM \to \mathbb{R}^n$ denotes the derivative of the Y at point $p \in M$. More formally, the derivative of Y at point $p \in M$ in the direction $v \in T_pM$, denoted by $\dot{Y}(p)\,v$, is defined as follows. Let $\gamma : \mathbb{R} \to M$ be a smooth curve such that $\gamma(0) = p$ and $\dot{\gamma}(0) = v$, and define $\dot{Y}(p)\,v = \frac{d}{dt}\big|_{t=0} Y(\gamma(t)) = \lim_{s\to 0} \frac{Y(\gamma(s)) - Y(p)}{s}$. It can be shown that this limit does exist; moreover, the limit is independent of the particular curve chosen to evaluate it.

For a Riemannian geometry, let $\mathrm{Lie}(\mathcal{F})$ denote the set of all vector fields formed by $\mathcal{F}$ along with all iterated Lie brackets of vector fields from $\mathcal{F}$, *i.e.*, $\mathrm{Lie}(\mathcal{F}) := \mathcal{F} \cup \{[Y_1, [Y_2, \ldots [Y_{n-1}, Y_n] \ldots] \,|\, Y_i \in \mathcal{F}, n \in \mathbb{N}\}$. It can be shown that $\mathrm{Lie}(\mathcal{F})$, also called the *Lie hull* of $\mathcal{F}$, is a Lie algebra (see, *e.g.*, Montgomery, 2002).

The definition of a Lie hull extends naturally to sub-Riemannian geometry. Given a distribution $\mathcal{H}$ on M, recall that $\mathcal{H}_p$ denotes its fiber at the point $p \in M$. Define $\mathcal{H}^1 := \{Y \in VF(M) \,|\, Y(p) \in \mathcal{H}_p\}$. For all $k \in \mathbb{N}$ and $k > 1$, define $\mathcal{H}^{k+1} := \mathcal{H}^k \cup [\mathcal{H}^1, \mathcal{H}^k]$, where $[\mathcal{H}^1, \mathcal{H}^k] = \mathrm{Span}\{[Y, Z] \,|\, Y \in \mathcal{H}^1, Z \in \mathcal{H}^k\}$. Define the fiber of $\mathcal{H}^k$ at point $p \in M$ as $\mathcal{H}^k_p := \{Y(p) \,|\, Y \in \mathcal{H}^k\}$. Then, the Lie hull of $\mathcal{H}$ is simply $\mathrm{Lie}(\mathcal{H}) := \bigcup_{k\geq 1} \mathcal{H}^k$. Its fiber over $p \in M$ is $\mathrm{Lie}(\mathcal{H})_p := \{Y(p) \,|\, Y \in \mathrm{Lie}(\mathcal{H})\}$. The

distribution $\mathcal{H}$ is said to be *bracket generating*, if the Lie hull of $\mathcal{H}$ spans the whole tangent space of M, *i.e.*, $\text{Lie}(\mathcal{H})_p = T_pM$ for all $p \in M$.

A central result in sub-Riemannian geometry is stated in the following theorem.

Theorem 6.1 (Chow's Theorem (Chow, 1970; Montgomery, 2002)) *Let $\mathcal{H}$ be a bracket generating distribution on a connected manifold M. Then, any two points of M can be joined with a horizontal path.*

The condition that the distribution is bracket generating, is called Chow's condition in sub-Riemannian geometry (Montgomery, 2002), the linear algebra rank condition in control theory (Bullo and Lewis, 2004; Jurdevic, 1997; Isidori, 1994), and Hörmander's condition in the context of partial differential equations (Hormander, 1967).

Example We continue the previous example. Given any configuration (x, y, θ) of the Reeds-Shepp car, the vector fields $Y_1 = (\cos\theta, \sin\theta, 0)$ and $Y_2 = (0, 0, 1)$ span the distribution induced by its dynamics. The Lie bracket of these two vector fields can be calculated as $[Y_1, Y_2] = (-\sin\theta, \cos\theta, 0)$. Notice that the vector fields $\{Y_1, Y_2, [Y_1, Y_2]\}$ span the whole tangent space, satisfying Chow's condition. Then, Chow's theorem merely tells us that, given two configurations of the Reeds-Shepp car, there exists a dynamically-feasible trajectory, *i.e.*, a horizontal curve, that joins the two. In fact, even though the car can not move sideways directly, it can travel to a position that lies to its side by moving forward and backward while steering suitably. ■

Let $\mathcal{H}$ be a bracket generating distribution. Then, for any $p \in \mathcal{H}$, there exists an integer $r(p)$ such that $\mathcal{H}_p \subset \mathcal{H}_p^2 \subset \cdots \subset \mathcal{H}_p^{r(p)} = T_pM$. The smallest such integer is also called the *degree of nonholonomy* of the distribution $\mathcal{H}$ at point p (Bellaiche et al., 1998; Montgomery, 2002). When the degree of non-holonomy is equal to one, $\mathcal{H}_p$ already spans T_pM; thus, no effective constraint is imposed on the tangent space. In other words, using a linear combination of the vector fields in $\mathcal{H}$, we can recover the whole tangent space. However, when $\mathcal{H}_p \neq T_pM$, the distribution $\mathcal{H}$ imposes a non-trivial constraint in the tangent space. As a consequence, no horizontal curve can move in the direction of $T_pM \setminus \mathcal{H}_p$ directly. Yet, it is possible to move in the directions spanned by $\mathcal{H}_p^2 \setminus \mathcal{H}_p$, by switching between two vector fields $Y, Z \in \mathcal{H}$ for which $[Y, Z] \in \mathcal{H}^2 \setminus \mathcal{H}$. The movement in the directions spanned by $\mathcal{H}_p^2 \setminus \mathcal{H}_p$, however, occurs more "slowly" than the movement in the directions spanned by $\mathcal{H}_p$. How slowly this movement occurs is described by the ball-box theorem. Below, we provide a number of preliminary definitions and state the ball-box theorem.

Let $n_k(p)$ denote the dimensionality of the space spanned by $\mathcal{H}_p^k$. The list $(n_1(p), n_2(p), \ldots, n_{r(p)}(p))$ of integers is called the *growth vector* of $\mathcal{H}$ at p. When $\mathcal{H}$ is bracket generating, the dimensionality of $\mathcal{H}_p^{r(p)}$ equals to that of T_pM, *i.e.*, $n_{r(p)} = n$. The point p is said to be a *regular point*, if there exists an open neighborhood of M around p such that the growth vector is constant; otherwise, p is said to be a *singular point*.

Given a vector field Y, let $\exp tY$ denote its flow. That is, $(\exp tY) : M \to M$ is such that $(\exp tY)(p)$ is equal to $y(t)$ where $y : [0, t] \to M$ is a solution to the following differential equation: $\frac{d}{dt}y(\tau) = Y(y(\tau))$ and $y(0) = p$. In other words,

$(\exp tY)(p)$ denotes the point that a system evolving on the manifold M reaches under the influence of the vector field Y for t time units starting from the point p.

Suppose that p is a regular point, and denote the growth vector at this point simply by $(n_1, n_2, \ldots, n_r)$, where $n_r = n$. Define the vector fields $Y_1, Y_2, \ldots, Y_n$ as follows: the set $\{Y_1, Y_2, \ldots, Y_{n_1}\}$ of vector fields spans $\mathcal{H}$; the set $\{Y_1, Y_2, \ldots, Y_{n_2}\}$ spans $\mathcal{H}^2$; and so on. Define the joint flow of all these vector fields as $\Phi(t_1, t_2, \ldots, t_n; p) := \big((\exp t_1 Y_1) \circ (\exp t_2 Y_2) \circ \cdots \circ (\exp t_n Y_n)\big)(p)$, where $\circ$ denotes the functional composition operator. Hence, $\Phi(t_1, t_2, \ldots, t_n; p)$ denotes the point reached, starting from p, under the influence of vector field Y_n for t_n time units, then Y_{n-1} for t_{n-1} time units, and so on. Since p is a regular point, such a map can be defined in some neighborhood of p. For notational convenience, define the weights $w_i := k$ if $Y_i(p) \in \mathcal{H}_p^k$ and $Y_i(p) \notin \mathcal{H}_p^{k+1}$ for all $i \in \{1, 2, \ldots, n\}$, and define $w := (w_1, w_2, \ldots, w_n)$. Finally, define the *w-weighted box* of size $\epsilon > 0$ centered at $p \in M$ as

$$\mathrm{Box}^w(p, \epsilon) := \{\Phi(t_1, t_2, \ldots, t_n; p) \,|\, |t_k| \leq \epsilon^{w_k}\}.$$

Recall that the sub-Riemannian ball of radius ϵ is defined as

$$B_s(p, \epsilon) := \big\{p \in M : d_s(p, p') \leq \epsilon\big\}.$$

The following theorem, attributed to Mitchell, Gershkovich, and Nagel-Stein-Wainger by Gromov (1996), is an important result in sub-Riemannian geometry.

Theorem 6.2 (Ball-Box Theorem (see Montgomery (2002))) *Suppose that $\mathcal{H}$ satisfies Chow's condition. Then, there exists positive constants ϵ_0, c, C with $c < C$ such that for all $\epsilon < \epsilon_0$ and for all $p \in M$,*

$$\mathrm{Box}^w(p, c\epsilon) \subset B_s(p, \epsilon) \subset \mathrm{Box}^w(p, C\epsilon).$$

Informally speaking, the ball-box theorem estimates the size of a small sub-Riemannian ball up to a constant factor. It states that the set of states that can be reached by a horizontal path starting from p contains a weighted box of radius $c\,\epsilon$ and is contained in a box of radius $C\,\epsilon$. The orientation of these boxes at a particular point is determined by the vector fields $\{Y_1, Y_2, \ldots, Y_n\}$ evaluated at that point. Note that Chow's theorem can be deduced from the ball-box theorem (Montgomery, 2002).

Example Denote by $\mathcal{H}$ the distribution induced by the dynamics of the Reeds-Shepp car. From the previous example, $\mathcal{H} \subset \mathcal{H}^2 = T_pM$, and the vector fields $Y_1 = (\cos\theta, \sin\theta, 0)$, $Y_2 = (0, 0, 1)$, $Y_3 = (-\sin\theta, \cos\theta, 0)$ are such that $\mathrm{Span}\{Y_1, Y_2\} = \mathcal{H}$ while $\mathrm{Span}\{Y_1, Y_2, Y_3\} = \mathcal{H}^2$. Since the dimensionality of the space spanned by $\mathcal{H}$ and $\mathcal{H}^2$ are equal to two and three, respectively, the growth vector is $(2, 3)$ for all points on the manifold describing the configuration space of the Reeds-Shepp car. The weight vector, on the other hand, can be computed as $w = (1, 2, 1)$.

Suppose that the car is at the origin, *i.e.*, $(x, y, \theta) = \mathbf{0} := (0, 0, 0)$. Then, we calculate $Y_1(\mathbf{0}) = (1, 0, 0)$, $Y_2(\mathbf{0}) = (0, 0, 1)$, and $Y_3(\mathbf{0}) = (0, 1, 0)$. In this case, the ball-box theorem implies that the set of states reachable from $\mathbf{0}$ within t time units

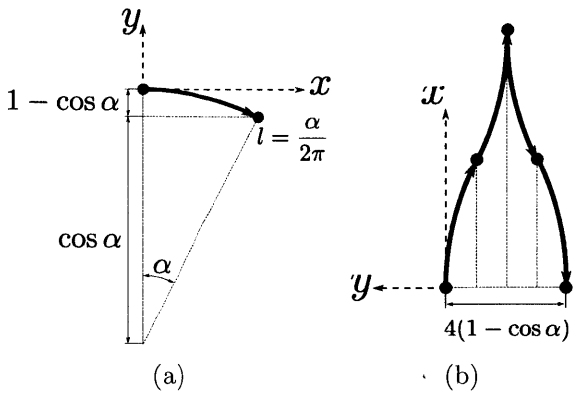


Figure 6-1: Figure (a) shows the resulting trajectory of the vehicle when it moves forward while turning right. Figure (b) shows the resulting trajectory after four maneuvers. The x- and y-axes are also shown in the figures. Note that Figure (b) is rotated 90 degrees when compared to Figure (a).

contains a box of size $[-ct, ct] \times [-ct^2, ct^2] \times [-ct, ct]$ for all small enough $t > 0$, where $c > 0$ is a constant. In other words, the car can move a distance of ct in the longitudinal direction within t time units, and it can turn with a constant speed, both of which are apparent from the equations describing its dynamics. However, the ball-box theorem also states that the car can move a distance of ct^2 sideways by combining its maneuvers, which is not immediately obvious.

Let us explicitly provide a sequence of maneuvers that translates the car sideways for at least a distance of ct^2 within time t. Consider the following four maneuvers applied sequentially in this order: (i) forward and right, *i.e.*, $u_1(t) = u_2(t) = -1$, (ii) forward and left, *i.e.*, $u_1(t) = 1, u_2(t) = 1$, (iii) backward and right, *i.e.*, $u_1(t) = -1, u_2(t) = -1$, and (iv) backward and left, *i.e.*, $u_1(t) = -1, u(t) = 1$, with each set of controls applied for $T/4$ time units. As seen in Figure 6-1, with each move the car moves a distance of $1 = \cos(\alpha) = 1 - \cos((\pi/2)\,t)$ along the y-axis. Using the Taylor expansion of the cosine function, $\cos\alpha = 1 - \frac{\alpha^2}{2!} + \frac{\alpha^4}{4!} - \cdots$, we conclude that total distance traveled by the car along the y-axis satisfies $4\,(1 - \cos((\pi/2)\,t)) = ct^2 + o(t^2)$, where $o(t^2)$ denotes terms that are smaller than t^2 asymptotically and c is a constant. Hence, after the aforementioned four moves, the car ends up roughly at the point $(0, ct^2, 0)$, confirming the result of the ball-box theorem.

Note that the ball-box theorem also predicts that no move executed within t time units can move a distance more than Ct^2, for all small enough t. ■

Links between sub-Riemannian Geometry, Control Theory, and Motion Planning: We define the motion planning problem on sub-Riemannian manifolds as follows. Let $M \subset \mathbb{R}^n$ be a d-dimensional Riemannian manifold. Let $\mathcal{X}_{\text{obs}}, \mathcal{X}_{\text{goal}} \subset M$ denote the obstacle set and the goal set, respectively, both open relative to M, and define the free space as $\mathcal{X}_{\text{free}} := M \setminus \mathcal{X}_{\text{obs}}$. Let x_{init} denote the initial state. Let $\mathcal{H}$ be a distribution defined on M. Then, the motion planning problem on a sub-Riemannian

manifold is to find a finite-length sub-Riemannian curve γ on $(M, \mathcal{H})$, such that γ *(i)* starts from the initial state, *(ii)* reaches a final state in $\mathcal{X}_{\text{goal}}$, and *(iii)* stays entirely within the free space. Clearly, by construction, any sub-Riemannian curve respects the distribution $\mathcal{H}$, *i.e.*, tangent vectors along the curve are in $\mathcal{H}$.

The trajectory planning problem and the motion planning problem on a sub-Riemannian manifold are tightly linked. The state space X in the former is represented by the manifold M in the latter. The dynamics is represented by the differential constraint $\dot{x} = f(x, u)$, where $u \in U$, in the former, and by the distribution $\mathcal{H}$ in the latter. In fact, the $\mathcal{H}$ is merely the set $\{f(p, u) : u \in U\}$ for all $p \in M$.

Extensions of Algorithms: A set of modifications is necessary for the proposed algorithms to address trajectory planning problems, or equivalently planning problems on sub-Riemannian manifolds. As in previous section, the modifications only occur in the primitive procedures introduced in Section 4.1.

We modify the primitive procedures as follows. The `Sample` procedure returns independent and identically distributed samples from M. Thus, the `Sample` procedure is implemented as in Section 6.2.1.

The primitive procedures that run proximity queries, namely `Nearest` and `Near` procedures, are modified such that the distance metric is the sub-Riemannian metric. That is, $\texttt{Nearest}(V, x) := \arg\min_{v \in V} d_s(x, v)$, where $d_s(\cdot, \cdot)$ is the sub-Riemannian distance function, and the `Near` procedure is defined similarly. Note that precise computation of the sub-Riemannian distance is by no means trivial. However, the ball-box theorem provides us with an approximation in terms of Euclidean distances:

$$\texttt{Near}(V, x) := V \bigcap \text{Box}^w \left(x, \gamma \left(\frac{\log \text{card}\,(V)}{\text{card}\,(V)} \right)^{1/D} \right),$$

where w is the associated weight vector as defined above, $\text{Box}^w(x, r)$ is the weighted box of size r, with weight vector w, centered at the point x, and $D := \sum_{i=1}^{n} w_i$. Incidentally, the integer D coincides with the Hausdorff dimension of the distribution $\mathcal{H}$ (see Montgomery, 2002). It can be shown, using the ball-box theorem, that this implementation of the `Near` procedure, in fact, leads to asymptotic optimality. Moreover, the expected number of vertices contained in this box is $O(\log \text{card}\,(V))$, which in turn implies that, for example, the number of times `CollisionFree` procedure is called is no more than $O(n \log n)$, when the algorithm is run with n samples.

Finally, the `CollisionFree` procedure is implemented such that it checks for collision the minimum-length horizontal curve connecting the two given configurations. The minimum length horizontal curve, *i.e.*, the optimal trajectory between two states, can be computed exactly for a class of dynamical systems, such as a double integrator, Dubins' car (Dubins, 1957), or the Reeds-Shepp car (Reeds and Shepp, 1990). However, computing even a feasible trajectory between two states amounts to solving a boundary value problem, which may be analytically and computationally challenging. Designing algorithms that can get around solving boundary value problems, for example, using suitable approximations instead, remains largely an open problem.

6.2.2 Anytime Sampling-based Algorithms for Control Problems

One of the defining properties of the RRT* algorithm is its *anytime* flavor: the algorithm finds a feasible solution quickly, *e.g.*, almost as fast as the RRT, and improves this solution towards an optimal one when it is provided with more computation time. In other words, the RRT* algorithm constructs an approximation to the solution of the optimal path planning problem in an anytime manner. The approximation, initially, is crude; However, when more computation time is allowed, the approximation is refined towards an exact solution to the problem (of optimal motion planning).

Arguably, anytime algorithms may be especially valuable in robotics applications, where the computation time that can be devoted to planning is rarely known *a priori*. In fact, in many robotics applications, the end of the planning phase is triggered by an uncontrolled event. Once the computation time for planning is up, the robot is forced to implement a feasible plan to make progress towards the goal. Furthermore, in some cases, the planning is an online process, where the robot can improve its plan even during execution time, *i.e.*, as parts of the plan is being executed, other parts of it are improved on the fly (see, *e.g.*, Karaman et al., 2011). Arguably, anytime sampling-based algorithms, like the RRT*, may be invaluable planning tools for such applications of robotics and similar problems arising in many other domains.

Anytime sampling-based algorithms like the RRT* have inspired a class of novel algorithms tailored to solve, in an anytime manner, fundamental problems that frequently arise in optimal control and estimation. In this section, we survey recent progress in extending the application domain of sampling-based algorithms to some of these problems. Along the way, we point out a number of open problems.

Differential Games

Broadly speaking, a (non-cooperative) differential game involves multiple players, each of which seeks to maximize her own payoff, subject to differential constraints. Initial applications of differential games have mainly been in the military domain (Isaacs, 1965). However, it was quickly realized that this class of problems have vast civilian applications, such as air traffic control (see, *e.g.*, Tomlin et al., 1998).

In the rest of this section, we first extend the motion planning problem with adversarial agents, where the robot tries to avoid capture by the adversarial agents in a complex environment populated with obstacles. Subsequently, we outline a sampling-based algorithm that solves this problem with probabilistic guarantees on soundness and completeness. This problem and the algorithm were presented in (Karaman and Frazzoli, 2010b). We close this section with a brief overview of the open problems.

A Class of Pursuit-Evasion Games: Below, we describe a two-player zero-sum differential game in which one of the players, called the evader, tries to escape in minimum time to a goal set, while the second player, called the pursuer, tries to capture the evader, before doing so. More formally, consider a time-invariant dynamical system described by the differential equation $\dot{x}(t) = f(x(t), u_{\mathrm{e}}(t), u_{\mathrm{p}}(t))$, where

$x : t \mapsto x(t) \in X \subset \mathbb{R}^d$ is the state trajectory, $u_\mathrm{e} : t \mapsto u_\mathrm{e}(t) \in U_\mathrm{e} \subset \mathbb{R}^{m_\mathrm{e}}$ is the evader's control input, $u_\mathrm{p} : t \mapsto u_\mathrm{p}(t) \in U_\mathrm{p} \subset \mathbb{R}^{m_\mathrm{p}}$ is the pursuer's control input. The sets X, U_e, and U_p are assumed to be compact, the control signals u_e and u_p are bounded and measurable, and $f(z, w_\mathrm{e}, w_\mathrm{p})$ is locally Lipschitz in z and piecewise continuous in w_e and w_p. Let X_obs, X_goal, and X_capt denote the obstacle set, goal set, and capture set, which we assume to be open.

Given an initial condition $x(0) \in X \setminus X_\mathrm{obs}$ and the control inputs of the evader and the pursuer, a unique state trajectory can be computed. The final time of the game is given by $T = \inf\{t \in \mathbb{R}_{\geq 0} : x(t) \in \mathrm{cl}\,(X_\mathrm{goal} \cup X_\mathrm{capt})\}$, *i.e.*, the time at which either the evader escapes to the escape set or the pursuer captures the evader. Since this is a zero-sum game, only one objective function will be considered, which is defined as follows: $L(u_\mathrm{e}, u_\mathrm{p}) = T$, if $x(T) \in \mathrm{cl}(X_\mathrm{goal})$; and $L(u_\mathrm{e}, u_\mathrm{p}) = +\infty$, otherwise. The evader tries to minimize this objective function by escaping to the goal region in minimum time, while the pursuer tries to maximize it by capturing the evader before she reaches the goal.

Let $\mathrm{BR} : U_\mathrm{e} \to U_\mathrm{p}$ denote the transformation that maps each evader trajectory to the best response of the pursuer, *i.e.*, $\mathrm{BR}(u_\mathrm{e}) := \arg\max_{u_\mathrm{p}} L(u_\mathrm{e}, u_\mathrm{p})$. In the game described above, the evader picks her strategy so that $L^* = L(u_\mathrm{e}^*, \mathrm{BR}(u_\mathrm{e}^*)) \leq L(u_\mathrm{e}, e_\mathrm{p})$ for all u_e and all u_p. Let $u_\mathrm{p}^* := \mathrm{BR}(u_\mathrm{e}^*)$. Then, the pair $(u_\mathrm{e}^*, u_\mathrm{p}^*)$ is called the (open-loop) *Stackelberg equilibrium* of this differential game (Başar and Olsder, 1982).

As common in pursuit-evasion games, the dynamics of the agents in the problem considered in this section further possesses a separable structure, in the following sense. It is assumed that the state can be partitioned as $x = (x_\mathrm{e}, x_\mathrm{p}) \in X_\mathrm{e} \times X_\mathrm{p} = X$, the obstacle set can be similarly partitioned as $X_\mathrm{obs} = (X_\mathrm{obs,e} \times X_\mathrm{p}) \cup (X_\mathrm{e} \times X_\mathrm{obs,p})$, where $X_\mathrm{obs,e} \subset X_\mathrm{e}$ and $X_\mathrm{obs,p} \subset X_\mathrm{p}$, the goal set is such that $X_\mathrm{goal} = (X_{e,\mathrm{goal}} \times X_\mathrm{p}) \setminus X_\mathrm{capt}$, where $X_{e,\mathrm{goal}} \subset X_\mathrm{e}$, and the dynamics are decoupled as follows:

$$\frac{d}{dt}x(t) = \frac{d}{dt}\begin{bmatrix} x_\mathrm{e}(t) \\ x_\mathrm{p}(t) \end{bmatrix} = f(x(t), u(t)) = \begin{bmatrix} f_\mathrm{e}\,(x_\mathrm{e}(t), u_\mathrm{e}(t)) \\ f_\mathrm{p}\,(x_\mathrm{p}(t), u_\mathrm{p}(t)) \end{bmatrix}, \quad \text{for all } t \in \mathbb{R}_{\geq 0},$$

It is also assumed that the initial condition is an equilibrium state for the pursuer, *i.e.*, there exists $u_\mathrm{p}' \in U_\mathrm{p}$ such that $f_\mathrm{p}(x_{\mathrm{init,p}}, u_\mathrm{p}') = 0$.

The algorithmic question in this problem is to compute T, the final time of the game, and compute u_e^*, the Stackelberg strategy for the evader, whenever T is finite.

The pursuit-evasion game described above generalizes the (minimum-time) optimal motion planning problem with an adversarial agent, namely the pursuer. In addition to reaching the goal region (in minimum time) while avoiding collision with obstacles, the evader must also avoid capture by the pursuer. Arguably, the problem we describe above is the simplest extension of optimal motion planning with adversarial agents. We discuss a number of other extensions later in this section.

Sampling-based Algorithm: In (Karaman and Frazzoli, 2010b), we propose a sampling-based algorithm that solves the pursuit-evasion game described above, with probabilistic guarantees. The algorithm computes the set of all states that the evader

can reach without being captured by the pursuer, in an anytime manner. For this purpose, two instances of the RRT* algorithm is employed, one rooted at the initial configuration of the evader and another rooted at that of the pursuer. The two RRT* instances are run independently with the following exception. The vertices of the tree that belongs to the evader's RRT* are deleted, when it is ensured that the evader can be captured via a path in the tree that belongs to the pursuer's RRT*. If the evader's RRT* tree is rich enough to include a trajectory that reaches the goal region while avoiding collision with obstacles, the algorithm returns this trajectory. The details of the algorithm can be found in (Karaman and Frazzoli, 2010b).

The output of the algorithm is a trajectory for the evader, which converges to a Stackelberg strategy (as defined above) in some suitable sense. The nature of this convergence can be made precise through a number of formal properties that the algorithm is equipped with. Below, we describe these properties, after introducing some preliminary definitions.

An algorithm is said to be *sound* for the pursuit-evasion problem, if the trajectory it returns, whenever it returns one, is feasible, *i.e.*, the trajectory *(i)* starts from evader's initial state, *(ii)* reaches the goal region, *(iii)* avoids collision with obstacles, and *(iv)* avoids capture by the pursuer. The trajectory returned by the algorithm described above, when the algorithm returns one, reaches the goal region and avoids collision with obstacles, by construction, since it is generated by an RRT* rooted However, the returned trajectory avoids capture by the pursuer, if the pursuer's RRT* is dense enough, which occurs, roughly speaking, with probability converging to one as the number of samples approaches infinity. Hence, the algorithm we describe above is *probabilistically sound*, *i.e.*, given that the algorithm returns a trajectory infinitely often, $\liminf_{n\to\infty} \mathbb{P}(\{\sigma_n \text{ is feasible}\}) = 1$, where σ_n is the trajectory returned by the algorithm by the end of iteration n. In comparison, the solution returned by all of the sampling-based algorithms we presented in Chapter 4 are sound, *i.e.*, the solution returned by the algorithm, if it returns one, is feasible.

The algorithm we present in this section is *probabilistically complete*, like many other sampling-based algorithms, *i.e.*, $\liminf_{n\to\infty} \mathbb{P}(\{\text{the algorithm returns a feasible solution in iteration } n\}) = 1$, whenever there exists a feasible solution to the problem.

Finally, the algorithm is *asymptotically optimal*, in the sense that the trajectory returned by the algorithm converges to a minimum-time solution, almost surely.

Extensions and Open Problems: The pursuit-evasion problem introduced above is a relatively simple differential game. In fact, arguably, it is the simplest extension of the optimal path planning problem with adversarial agents. Many open problems remain on the path to designing general-purpose anytime sampling-based algorithms for differential games.

In particular, the case when the evader's and the pursuer's dynamics are coupled may captures a class of planning under uncertainty problems, where the actions of the pursuer represents the disturbance. Then, the objective is to plan a trajectory that is feasible no matter what disturbance is observed during execution.

Another direction with a vast potential for impact is to relax the relax the re-

quirements on the information structure. In the problem discussed in this section, the evader does not observe at all the actions of the pursuer, as the game progresses. In some practical cases, the pursuer may, in fact, not be visible to the evader during execution, although the initial state of the pursuer is known. However, in many other cases, the evader may be able to observe the state of the pursuer in execution time. In that case, the solution provided by the algorithm presented in this section is sound. However, it is conservative. Thus, the algorithm is not complete for this problem. Designing anytime sampling-based algorithms that return feedback control strategies that converge to, for example, Nash equilibrium solutions under feedback information structure (see Başar and Olsder, 1982), remains an open problem.

Stochastic Optimal Control

In the optimal trajectory problem introduced in Section 6.2.1, we assumed that the dynamics is deterministic. In this section, we extend this problem with noisy dynamics. In what follows, we define the continuous-time and continuous-space stochastic optimal control problem. Subsequently, we describe the key ideas behind an anytime sampling-based algorithm tailored to solve this problem. We end this section with a list of open problems in this direction. The material we present in this section is largely taken from (Huynh et al., 2012), which the interested reader is referred to.

Problem Description: Let $n, m, k \in \mathbb{N}$, and let $\mathbb{R}^{n\times k}$ denote the set of all n by k real matrices. We denote the k-dimensional Brownian motion by $\{w(t) : t \geq 0\}$. Let $S \subset \mathbb{R}^n$ and $U \subset \mathbb{R}^m$ be compact sets with non-empty interior. Consider the stochastic dynamical system

$$dx(t) = f(x(t), u(t))dt + F(x(t), u(t))dw(t),$$

where $f : S \times U \to \mathbb{R}^n$ and $F : S \times U \to \mathbb{R}^{n\times k}$. A *policy* is a function of the form $\mu : S \to U$. Under policy μ, the state of the system is a stochastic process, denoted by $\{x_\mu(t) : t \geq 0\}$, that a solution to the following integral equation:

$$x_\mu(t) = x(0) + \int_0^t f(x_\mu(\tau), \mu(x_\mu(\tau)))\, d\tau + \int_0^t F(x_\mu(\tau), \mu(x_\mu(\tau)))\, dw(\tau)$$

for all $t \leq T_\mu$, where T_μ is the *first exit time* of the system defined as $T_\mu := \inf\{t : x_\mu(t) \notin \text{int}(S)\}$. Define the *cost rate function* and the *terminal cost function* as $\phi : S \times U \to \mathbb{R}$ and $\psi : S \to \mathbb{R}$, respectively. Then, the *cos-to-go function* under policy μ starting from state z is defined as

$$J_\mu(z) = \mathbb{E}\left[\int_0^{T_\mu} \phi(x_\mu(t), \mu(x_\mu(t)))dt + \psi(x_\mu(t)) \,\middle|\, x_0 = z\right].$$

The *optimal cost-to-go function* is defined as $J^*(z) = \inf_{\mu\in\Pi} J_\mu(z)$, where Π denotes the set of all (admissible) policies. The policy μ^*, for which $J_{\mu^*}(z) = J^*(z)$ for all $z \in X$, is called the *optimal policy.*

The algorithmic question in stochastic optimal control is to compute the optimal cost-to-go function and the optimal policy, for a given problem instance described by the tuple (S, U, f, F, g, h). The stochastic optimal control problem as described above extends the optimal motion planning problem with stochastic dynamics. By suitably choosing the terminal cost function $\psi(\cdot)$, the optimal policy can be encouraged to stay away from the obstacle set and eventually reach the goal set. That is, $\psi(x) = \upsilon$ for all $x \in \partial\mathcal{X}_{\text{obs}} \cup \partial[0,1]^d$ and $\psi(x) = \epsilon$ for all $x \in \partial\mathcal{X}_{\text{goal}}$, where $\mathcal{X}_{\text{obs}}$ and $\mathcal{X}_{\text{goal}}$ are the obstacle set and the goal set, $[0,1]^d$ is the configuration space, and the ∂ operator denotes the boundary of a set. The particular values of ϵ and υ can be tuned in order to discourage the controller from taking risk, by steering away from the obstacle boundary, thus decreasing the chances of a collision; or vice verse.

Sampling-based Algorithms: A sampling-based algorithm that solves the problem presented above, with probabilistic guarantees, is presented in (Huynh et al., 2012). The key idea behind this algorithm is to generate a sequence of discrete-time finite-state Markov decision processes, the trajectories of which converges in distribution to the trajectories of the original stochastic dynamical system. To compute the optimal value function, the algorithm also employs an incremental value iteration algorithm, based on the asynchronous parallel value iteration algorithm presented by Bertsekas and Tsitsiklis (1997). The resulting sampling-based algorithm guarantees convergence in some suitable sense to the optimal value function, almost surely, as the number of samples approaches infinity. Moreover, the algorithm has attractive computational properties that allow real-time implementations in online settings. The reader is referred to (Huynh et al., 2012) for details.

Extensions and Open Problems: The stochastic optimal control problem discussed in this section is fairly general, aside from the assumption that the state is fully observed. Yet, a number of practical generalizations have not yet been addressed.

First, only Brownian noise model is used in the formulation above. In this model, the perturbation due to noise is a topologically continuous trajectory, almost surely. Using sampling-based algorithms to address continuous-time continuous-space problems involving other noise models remains open.

Second, we have considered a relatively simple objective function: the expected value of the line integral of a cost rate function with respect to the trajectory and a terminal cost function. Algorithms that minimize, for example, the variance of the same cost function may be valuable in applications, where high risk is prohibited.

Optimal State Estimation

Throughout the thesis, we focused on problems that assumed perfect observation of the state of the robot as well as that of its surroundings. In many applications, however, only noisy observations of the state are available, in which case estimating the current state of the system becomes an important problem (Thrun et al., 2005).

In this section, we introduce a continuous-time continuous-space state estimation problem, for which an anytime sampling-based algorithm was recently proposed. This

material is presented in detail in (Chaudhari et al., 2012), which the interested reader is referred to. We also point out a number of open problems on the path to truly anytime sampling-based algorithms for planning problems under uncertainty, in particular an extension of the stochastic optimal control problem with noisy observations.

Problem Description: Let $\{w(t) : t \geq 0\}$ be a k-dimensional standard Brownian motion, and let $\{v(t) : t \geq 0\}$ be an l-dimensional standard Brownian motion independent of $w(t)$. Consider the following autonomous stochastic dynamical system:

$$\begin{aligned} dx(t) &= f(x(t))\,dt + F(x(t))\,dw(t), \\ dy(t) &= g(x(t))\,dt + G(x(t))\,dv(t), \end{aligned}$$

where $x(t) \in X \subset \mathbb{R}^n$ denotes the state and $y(t) \in Y \subset \mathbb{R}^q$ denotes the observation, at time t, and $f(\cdot), F(\cdot), g(\cdot), G(\cdot)$ are Lipschitz-continuous functions with suitable domain and range.

A widely studied state estimation problems is filtering, where one seeks to estimate the state $\{x(t) : t \geq 0\}$ from observations. More precisely, given a set $\{y(s) : 0 \leq s \leq t\}$ of observations, the filtering problem is to find a square-integrable and measurable estimate $\{\hat{x}(t) : 0 \leq s \leq t\}$ such that $\mathbb{E}\big[|\hat{x}(t) - x(t)|\big]$ is minimized. It is well known that this estimate satisfies $\hat{x}(t) = \mathbb{E}\big[x(t) \,|\, \{y(s) : 0 \leq s \leq t\}\big]$.

Sampling-based Algorithms: Recently, Chaudhari et al. (2012) proposed a sampling-based algorithm to address the filtering problem described above. The key idea behind the algorithm is to construct a discrete approximation of the dynamics, incrementally, in conjunction with an incremental algorithm that computes the optimal estimate on this approximation. The approximation, in this case, is a discrete Hidden Markov Model (HMM) generated via sampling in such a way that the trajectories of the HMM converges, in distribution, to the trajectories of the original continuous-time stochastic dynamical system, as the number of samples approaches infinity. Some other estimation problems, such as the maximum a posteriori (MAP) estimation, are also addressed using sampling-based algorithms in (Chaudhari et al., 2012).

Extensions and Open Problems: In many applications of robotics, state estimators are used in conjunction with a controller, which often aims to optimize a cost function similar to the one presented in Section 6.2.2, subject to stochastic dynamics and noisy observations (see, *e.g.*, Thrun et al., 2005). The design of such controllers has received a growing attention; in particular, a number of algorithms have been proposed to address the discrete-time, finite-state versions of the problem (see, *e.g.*, Pineau et al., 2003; Smith and Simmons, 2005; Kurniawati et al., 2008). However, much progress is needed to address fairly general continuous-time continuous-space formulations of the problem using anytime sampling-based algorithms.

Bibliography

M. Abramowitz and I. A. Stegun. *Handbook of Mathematical Functions.* With Formulas, Graphs, and Mathematical Tables. Courier Dover Publications, 1964.

R Altorevitz, S. Patil, and A Derbakova. Rapidly-Exploring Roadmaps:Weighing Exploration vs. Refinement in Optimal Motion Planning. In *IEEE International Conference on Robotics and Automation*, 2011.

R. Alur, T. A. Henzinger, G. Lafferriere, and G. J. Pappas. Discrete Abstractions of Hybrid Systems. *Proceedings of the IEEE*, 88(7):971–984, 2000.

S. Arya and D. M. Mount. Approximate range searching. *Computational Geometry*, 17:135–152, 2000.

S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu. An Optimal Algorithm for Approximate Nearest Neighbor Searching in Fixed Dimensions. *Journal of the ACM*, 45(6):891–923, 1998.

S. Arya, T. Malamatos, and D. M. Mount. Space-time Tradeoffs for Approximate Spherical Range Counting. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 535–544, 2005.

P. Balister, B. Bollobás, A. Sarkar, and M. Walters. Connectivity of random k-nearest-neighbour graphs. *Advances in Applied ...*, January 2005.

P. Balister, B. Bollobás, and A. Sarkar. Percolation, connectivity, coverage and colouring of random geometric graphs. *Handbook of Large-Scale Random ...*, January 2008.

J. Barraquand and J. Latombe. Robot Motion Planning: Distributed Representation Approach. *International Journal of Robotics Research*, 10(6):628–649, 1991.

J. Barraquand, L. E. Kavraki, J. Latombe, T. Li, R. Motwani, and P. Raghavan. A Random Sampling Scheme for Path Planning. *International Journal of Robotics Research*, 16(6):759–774, 1997.

T. Başar and G. J. Olsder. Dynamic Noncooperative Game Theory, January 1982.

A. Bellaiche. The Tangent Space in Sub-Riemannian Geometry. In A. Bellaiche and J Risler, editors, *Sub-Reimannian Geometry*, pages 1–78. Birkhauser, 1996.

A. Bellaiche, F. Jean, and J. J. Risler. Geometry of Nonholonomic Systems. In J. P. Laumond, editor, *Robot Motion Planning and Control*, pages 55–91. Springer, 1998.

C. Belta, A. Bicchi, M. Egerstedt, E. Frazzoli, E. Klavins, and G. J. Pappas. Symbolic Planning and Control of Robot Motion. *IEEE Robotics and Automation Magazine*, 14(1):61–70, March 2007.

D. Berenson, J. Kuffner, and H. Choset. An Optimization Approach to Planning for Mobile Manipulation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2008.

D. Berenson, T. Simeon, and S. S. Srinivasa. Addressing Cost-Space Chasms in Manipulation Planning. In *IEEE International Conference on Robotics and Automation*, 2011.

D. P. Bertsekas and J. N. Tsitsiklis. Parallel and distributed computation, 1997.

A. Bhatia and E. Frazzoli. Incremental Search Methods for Reachability Analysis of Continuous and Hybrid Systems. In *Hybrid Systems: Computation and Control*, pages 451–471. Springer, February 2004.

A. Bhatia, L. E. Kavraki, and M. Y. Vardi. Sampling-based Motion Planning with Temporal Goals. In *IEEE Conference on Robotics and Automation (ICRA)*, 2010.

J. Bialkowski, S. Karaman, and E. Frazzoli. Massively Parallelizing the RRT and the RRT*. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011.

B. Bollobás and O. Riordan. *Percolation*. Cambridge Univ Pr, 2006.

W. M. Boothby. An Introduction to Differentiable Manifolds and Riemannian Geometry, 1975.

M. S. Branicky, S. M. LaValle, K. Olson, and L. Yang. Quasi-Randomized Path Planning. In *IEEE Conference on Robotics and Automation*, 2001.

M. S. Branicky, M. M. Curtiss, J. A. Levine, and S. B. Morgan. RRTs for Nonlinear, Discrete, and Hybrid Planning and Control. In *IEEE Conference on Decision and Control*, pages 657–663, 2003.

M. S. Branicky, M. M. Curtiss, J. Levine, and S. Morgan. Sampling-based planning, control and verification of hybrid systems. *IEE Proceedings - Control Theory and Applications*, 153(5):575, 2006.

R. A. Brooks and T. Lozano-Perez. A Subdivision Algorithm in Configuration Space for Findpath with Rotation. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 799–806, 1983.

J. Bruce and M. Veloso. Real-Time Randomized Path Planning. In *RoboCup 2002: Robot Soccer World Cup VI (Lecture Notes in Computer Science vol. 2752)*, pages 1–6. Springer, 2003.

A. Bry and N. Roy. Rapidly-exploring Random Belief Trees for Motion Planning Under Uncertainty. In *IEEE Conference on Robotics and Automation*, 2011.

F. Bullo and A. D. Lewis. *Geometric Control of Mechanical Systems.* Springer, 2004.

J. Canny. *The Complexity of Robot Motion Planning.* The MIT Press, 1988.

P. Chanzy, L. Devroye, and C. Zamora-Cura. Analysis of range search for random k-d trees. *Acta Informatica*, 37(4-5):355–383, December 2000.

P. Chaudhari, S. Karaman, and E. Frazzoli. Sampling-based Algorithms for Filtering using Markov Chain Approximations. In *IEEE Conference on Decision and Control (submitted)*, 2012.

B. Chazelle, T. Ottmann, E. Soisalon-Soininen, and D. Wood. The Complexity and Decidability of Separation. In *11th Colloquium on Automata, Languages and Programming*, pages 119–127, Antwerp, Belgium, 1984. Springer.

H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun. *Principles of Robot Motion.* Theory, Algorithms, and Implementation. The MIT Press, 2005.

W. L. Chow. Uber Systeme von Linearen Partiellen Differentialglaichungen erster Ordnung. *Math. Ann.*, 117:98–105, 1970.

E. M. Clarke, O. Grumberg, and D. Peled. *Model checking.* The MIT Press, 1999.

J. Cortes and T. Simeon. Sampling-Based Motion Planning under Kinematic Loop-Closure Constraints. In *Algorithmic Foundations of Robotics VI (Springer Tracts in Advanced Robotics, Vol. 17/2005)*, pages 75–90. Springer, 2005.

J. Cortes, L. Jaillet, and T. Simeon. Molecular Disassembly with RRT-Like Algorithms. In *IEEE Conference on Robotics and Automation*, pages 3301–3306, 2007.

M. P. do Carmo. *Riemannian Geometry.* Birkhauser, 1992.

D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel. Path Planning for Autonomous Driving in Unknown Environments. In *Experimental Robotics*, pages 55–64. Springer, 2009.

D. Dor and U. Zwick. SOKOBAN and other motion planning problems. *Computational Geometry*, 13(4):215–228, 1999.

D. P. Dubhashi and A. Panconesi. Concentration of Measure for the Analysis of Randomized Algorithms, 2009.

L. E. Dubins. On Curves of Minimal Length with a Constraint on Average Curvature, and with Prescribed Initial and Terminal Positions and Tangents. *American Journal of Mathematics*, 79(3):497–516, 1957.

H. Edelsbrunner and H. A. Maurer. On the Intersection of Orthogonal Objects. *Information Processing Letters*, 13(4,5):177–181, 1981.

E. A. Emerson and E. M. Clarke. Using Branching Time Temporal Logic to Synthesize Synchronization Skeletons. *Science of Computer Programming*, 2(3):241–266, 1982.

E. A. Emerson, C. S. Jutla, and A. P. Sistla. On model checking for the mu-calculus and its fragments. *Theoretical Computer Science*, 258(1-2):491–522, 2001.

D. Eppstein, M. S. Paterson, and F. F. Yao. On nearest-neighbor graphs. *Discrete and Computational Geometry*, 17:263–282, January 1997.

G. E. Fainekos, A. Girard, H. Kress-Gazit, and G. J. Pappas. Temporal logic motion planning for dynamic robots. *Automatica*, 45(2):343–352, 2009.

D. Ferguson and A. Stentz. Anytime RRTs. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006.

P. W. Finn and L. E. Kavraki. Computational Approaches to Drug Design. *Algorithmica*, 25(2-3):347–371, 1999.

G. W. Flake and E. B. Baum. Rush Hour is PSPACE-complete, or "Why you should generously tip parking lot attendants". *Theoretical Computer Science*, 270(1-2): 895–911, 2002.

E. Frazzoli, M. A. Dahleh, and E. Feron. Real-Time Motion Planning for Agile Autonomous Vehicles. *Journal of Guidance, Control, and Dynamics*, 25(1):116–129, 2002.

S. S. Ge and Y. J. Cui. Dynamic Motion Planning for Mobile Robots Using Potential Field Method. *Autonomous Robots*, 13(3):207–222, 2002.

E. N. Gilbert. Random Plane Networks. *Journal of the Society of Industrial and Applied Mathematics*, 9(4):533–543, 1961.

A. Girard and G. J. Pappas. Approximation Metrics for Discrete and Continuous Systems. *IEEE Transactions on Automatic Control*, 52(5):782–798, 2007.

A. Girard and G. J. Pappas. Approximate Bisimulation: A Bridge Between Computer Science and Control Theory. *European Journal of Control*, (5-6):568–578, 2011.

E. Gradel, W. Thomas, and T. Wilke. *Automata, Logics, and Infinite Games.* Springer, 2002.

G. Grimmett and D. Stirzaker. *Probability and random processes.* Oxford University Press, USA, 2001.

M. Gromov. Carnot-Caratheodory spaces seen from within. In *Sub-Riemannian Geometry*, pages 79–323. Birkhauser, 1996.

R. A. Hearn. *Games, Puzzles, and Computation.* PhD thesis, Massachusetts Institute of Technology, 2006.

N. Henze. On the fraction of random points with specified nearest-neighbour interrelations and degree of attraction. *Advances in applied probability*, 19:873–895, January 1987.

T. A. Henzinger, R. Majumdar, and J. Raskin. A Classification of Symbolic Transition Systems. *ACM Transactions on Computational Logic*, 6(1):1–32, 2005.

G. J. Holzmann. The Model Checker SPIN. *IEEE Transactions on Software Engineering*, 23(5):279–295, June 1997.

J. E. Hopcroft, D. Joseph, and S. Whitesides. On the movement of robot arms in 2-dimensional bounded regions. In *Symposium on Foundations of Computer Science*, pages 280–289, 1982.

J. E. Hopcroft, J. T. Schwartz, and M. Sharir. Efficient Detection of Intersections among Spheres. *International Journal of Robotics Research*, 2:77–80, 1983.

J. E. Hopcroft, D. Joseph, and S. Whitesides. Movement problems for 2-dimensional linkages. *SIAM Journal of Computing*, 13(3):610–629, 1984a.

J. E. Hopcroft, J. T. Schwartz, and M. Sharir. On the complexity of motion planning for multiple independent objects: PSPACE-hardness of the "warehouseman's problem". *International Journal of Robotics Research*, 3(4):76–88, 1984b.

L. Hormander. Hypoelliptic second order differential equations. *Acta Mathematica*, 119(1):147–171, 1967.

D. Hsu, J. Latombe, and R. Motwani. Path planning in expansive configuration spaces. *Proceedings of the 1997 IEEE International Conference on Robotics and Automation*, 3:2719–2726, 1997a.

D. Hsu, J. Latombe, and R. Motwani. Path Planning in Expansive Configuration Spaces. In *IEEE Conference on Robotics and Automation*, pages 2719–2726, 1997b.

D. Hsu, R. Kindel, J. Latombe, and S. Rock. Randomized Kinodynamic Motion Planning with Moving Obstacles. *The International Journal of Robotics Research*, 21(3):233–255, 2002.

D. Hsu, J. P. Latombe, and H. Kurniawati. On the probabilistic foundations of probabilistic roadmap planning. *The International Journal of Robotics Research*, 25(7):627–643, January 2006.

V. Huynh, S. Karaman, and E. Frazzoli. Incremental Sampling-based Algorithms for Stochastic Optimal Control. In *IEEE Conference on Robotics and Automation*, 2012.

Y. K. Hwang and N. Ahuja. Potential Field Approach to Path Planning. *IEEE Transactions on Robotics and Automation*, 8(1):23–32, 1992.

R. Isaacs. *Differential Games.* A Mathematical Theory With Applications to Warfare and Pursuit, Control and Optimization. John Wiley and Sons, Inc., 1965.

A. Isidori. *Nonlinear Control Systems.* Springer, 3rd edition, 1994.

L. Jaillet and J. M. Porta. Asymptotically-optimal Path Planning on Manifolds. In *Robotics: Science and Systems*, 2012.

L. Jaillet, J. Cortes, and T. Simeon. Sampling-Based Path Planning on Configuration-Space Costmaps. *IEEE Transactions on Robotics*, 26(4):635–646, 2010.

J. Jeon, S. Karaman, and E. Frazzoli. Anytime Computation of Time-Optimal Off-Road Vehicle Maneuvers using the RRT*. In *IEEE Conference on Decision and Control*, 2011.

V. Jurdevic. *Geometric Control Theory.* Geometric Control Theory, 1997.

S. Karaman and E. Frazzoli. Optimal Kinodynamic Motion Planning using Incremental Sampling-based Methods. In *IEEE Conference on Decision and Control (CDC)*, Atlanta, GA, December 2010a.

S. Karaman and E. Frazzoli. Incremental Sampling-based Algorithms for a class of Pursuit-Evasion Games. In *Workshop on Algorithmic Foundations of Robotics (WAFR)*, Singapore, December 2010b.

S. Karaman and E. Frazzoli. Linear temporal logic vehicle routing with applications to multi-UAV mission planning. *Journal of Robust and Nonlinear Control*, 21(12): 1372–1395, 2011a.

S. Karaman and E. Frazzoli. Sampling-based Algorithms for Optimal Motion Planning. *International Journal of Robotics Research*, 30(7):846–894, 2011b.

S. Karaman and E. Frazzoli. Sampling-based Algorithms for Optimal Motion Planing with Deterministic mu-calculus Specifications. In *American Control Conference*, 2012.

S. Karaman, M. Walter, A. Perez, E. Frazzoli, and S. Teller. Anytime Motion Planning using the RRT*. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

L. Kavraki, P. Svestka, J. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, January 1996.

L. Kavraki, M. Kolountzakis, and J. Latombe. Analysis of probabilistic roadmaps for path planning. *IEEE Transactions on Robotics and Automation*, 14(1):166–171, 1998.

L. E. Kavraki and J. Latombe. Randomized Preprocessing of Configuration Space for Fast Path Planning. In *IEEE Conference on Robotics and Automation*, pages 2138–2139, 1994.

O. Khatib. Real-time Obstacle Avoidance for Manipulators and Mobile Robots. *International Journal of Robotics Research*, 5(1):90–98, 1986.

M. Kloetzer and C. Belta. A Fully Automated Framework for Control of Linear Systems from Temporal Logic Specifications. *IEEE Transactions on Automatic Control*, 53(1):287–297, 2008.

Y. Koren and J. Borenstein. Potential field methods and their inherent limitations for mobile robot navigation. In *IEEE International Conference on Robotics and Automation*, pages 1398–1404, 1991.

E. Koyuncu, N. K. Ure, and G. Inalhan. Integration of Path/Maneuver Planning in Complex Environments for Agile Maneuvering UCAVs. *Journal of Intelligent and Robotic Systems*, 57(1-4):143–170, 2009.

H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas. Where's Waldo? Sensor-Based Temporal Logic Motion Planning. In *IEEE International Conference on Robotics and Automation*, 2007.

H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas. Temporal-Logic-Based Reactive Mission and Motion Planning. *IEEE Transactions on Robotics*, 25(6):1370–1381, 2009.

J. J. Kuffner, S. Kagami, K. Nishiwaki, M. Inaba, and H. Inoue. Dynamically-Stable Motion Planning for Humanoid Robots. *Autonomous Robots*, 15:105–118, 2002.

H. Kurniawati, D. Hsu, and W S Lee. SARSOP: Efficient Point-Based POMDP Planning by Approximating Optimally Reachable Belief Spaces. In *Proceedings of Robotics: Science and Systems*, 2008.

Y. Kuwata, J. Teo, G. Fiore, S. Karaman, E. Frazzoli, and J. P. How. Real-Time Motion Planning with Applications to Autonomous Urban Driving. *IEEE Transactions on Control Systems*, 17(5):1105–1118, 2009.

A. Ladd and L. Kavraki. Measure theoretic analysis of probabilistic path planning. *Robotics and Automation*, January 2004.

J. Latombe. Motion Planning: A Journey of Molecules, Digital Actors, and Other Artifacts. *International Journal of Robotics Research*, 18(11):1119–1128, 1999.

J. P. Laumond, S. Sekhavat, and F. Lamiraux. Guidelines in Nonholonomic Motion Planning for Mobile Robots. In *Robot Motion Planning and Control*, pages 171–253. Springer, 1998.

S. M. LaValle. *Planning algorithms.* Cambridge Univ Pr, May 2006.

S. M. LaValle and J. Kuffner. Randomized kinodynamic planning. *The International Journal of Robotics Research*, 20(5):378–400, 2001.

D. T. Lee and C. K. Wong. Worst-Case Analysis for Region and Partial Region Searches in Multidimensional Binary Search Trees and Balanced Quad Trees. *Acta Informatica*, 9(1):23–29, 1977.

M. Likhachev and D. Ferguson. Planning Long Dynamically Feasible Maneuvers for Autonomous Vehicles. *The International Journal of Robotics Research*, 28(8): 933–945, July 2009.

M. Likhachev, D. Ferguson, G. Gordon, A. Stentz, and S. Thrun. Anytime search in dynamic graphs. *Artificial Intelligence*, 172(14):1613–1643, 2008.

M. Lin and D. Manocha. Collision and Proximity Queries. In *Handbook of Discrete and Computational Geometry*, pages 1–21. Chapman and Hall/CRC, July 2012.

S. R. Lindemann and S. M. LaValle. Current Issues in Sampling-Based Motion Planning. In *Springer Tracts in Advanced Robotics*, pages 36–54. Springer, 2005.

Y. Liu and N. I. Badler. Real-time Reach Planning for Animated Characters Using Hardware Acceleration. pages 1–8, June 2003.

T. Lozano-Perez. Spatial Planning: A Configuration Space Approach. *IEEE Transactions on Computers*, 32(2):108–120, 1983.

T. Lozano-Perez and M. Wesley. An Algorithm for Planning Collision-Free Paths Among Polyhedral Obstacles. *Communications of the ACM*, 22(10):560–570, 1979.

Z. Manna and P. Wolper. Synthesis of Communicating Processes from Temporal Logic Specifications. *ACM Transactions on Programming Languages and Systems*, 6(1):68–93, 1984.

J D Marble and K. E. Bekris. Towards Small Asymptotically Near-Optimal Roadmaps. In *IEEE International Conference on Robotics and Automation*, 2012.

R. Meester and R. Roy. *Continuum percolation.* Cambridge Univ Pr, 1996.

R J Milgram and J C Trinkle. The geometry of configuration spaces for closed chains in two and three dimensions. *Homology, Homotopy, and Applications*, 6(1):237–267, 2004.

R. Montgomery. *A Tour of Subriemannian Geometries, Their Geodesics, and Applications.* American Mathematical Society, 2002.

J. R. Munkres. *Topology.* Prentice Hall, 2nd edition, 2000.

H. Niederreiter. *Random Number Generation and Quasi-Monte Carlo Methods.* Society for Industrial Mathematics, 1992.

G. J. Pappas. Bisimilar linear systems. *Automatica*, 39(12):2035–2047, 2003.

D. Peled, P. Pelliccione, and P. Spletini. Model Checking. In *Wiley Encyclopedia of Computer Science and Engineering*, pages 1904–1920. Wiley, 2009.

M. Penrose. *Random geometric graphs.* Oxford University Press, USA, 2003.

A. Perez, S. Karaman, M. Walter, A. Shkolnik, E. Frazzoli, and S. Teller. Asymptotically-optimal Path Planning for Manipulation using Incremental Sampling-based Algorithms. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011.

J Pineau, G. Gordon, and S. Thrun. Point-based value iteration: An anytime algorithm for POMDPs. In *International Joint Conference on Artificial Intelligence*, 2003.

E. Plaku and L. E. Kavraki. Quantitative Analysis of Nearest-Neighbors Search in High-Dimensional Sampling-based Motion Planning. In *Algorithmic Foundations of Robotics VIII (Springer Tracts in Advanced Robotics 47/2008)*, pages 3–18. Springer, 2008.

E. Plaku, K. E. Bekris, B. Y. Chen, A. M. Ladd, and L. E. Kavraki. Sampling-Based Roadmap of Trees for Parallel Motion Planning. *IEEE Transactions on Robotics*, 21(4):587–608, 2005.

A. Pnueli. The Temporal Logic of Programs. In *18th Annual Symposium on the Foundations of Computer Science*, pages 46–57, 1977.

S. Prentice and N. Roy. The Belief Roadmap: Efficient Planning in Belief Space by Factoring the Covariance. *The International Journal of Robotics Research*, 28 (11-12):1448–1465, 2009.

J. Quintanilla and S. Torquato. Efficient measurement of the percolation threshold for fully penetrable discs. *Journal of Physics A: Mathematical and General*, 33: 399–407, 2000.

J. A. Reeds and L. A. Shepp. Optimal paths for a car that goes both forwards and backwards. *Pacific Journal of Mathematics*, 145(2):367–393, 1990.

J. Reif. Complexity of the mover's problem and generalizations. In *Proceedings of the 20th Symposium on the Foundations of Computer Science*, pages 421–427, 1979.

J. Reif. Complexity of the Generalized Mover's Problem. In J Schwartz, J Hopcroft, and M. Sharir, editors, *Planning, Geometry, and Complexity of Robot Motion*, pages 267–281. Ablex Publishing Corp., 1987.

J. Reif and J Storer. Shortest Paths in Euclidean Space with Polyhedral Obstacles. Technical Report CS-85-121, 1985.

S. I. Resnick. *A Probability Path.* Birkhäuser, 1999.

E. Rimon and D. E. Koditschek. Exact robot navigation using artificial potential functions . *IEEE Transactions on Robotics and Automation*, 8(5):501–518, 1992.

M. Sahimi. *Applications of percolation theory.* CRC, 1994.

H. Samet. *Applications of spatial data structures.* Computer graphics, Image Processing, and GIS. Addison Wesley Publishing Company, 1990a.

H. Samet. *The design and analysis of spatial data structures.* Addison Wesley Publishing Company, 1990b.

K. Schneider. *Verification of Reactive Systems.* Formal Methods and Algorithms. Springer-Verlag New York Incorporated, 2004.

A. Schrijver. *Combinatorial optimization.* Springer Verlag, 2003.

J. T. Schwartz and M. Sharir. On the "Piano Movers" Problem. II. General Techniques for Computing Topological Properties of Real Algebraic Manifolds. *Advances in Applied Mathematics*, 4:298–351, 1983a.

J. T. Schwartz and M. Sharir. On the "Piano Movers" Problem I. The Case of a Two-Dimensional Rigid Polygonal Body Moving Amidst Polygonal Obstacles. *Communications on Pure and Applied Mathematics*, 36(3):345–398, 1983b.

M. Sharir. Algorithmic Motion Planning. In *Handbook of discrete and computaional geometry*, pages 733–754. CDC Press, 1997.

M. Sipser. *Introduction To The Theory Of Computation.* Thomson, 2nd edition, 2006.

H. W. Six and D. Wood. Counting and Reporting Intersections of d-Ranges. *IEEE Transactions on Computers*, 31(3):181–187, 1982.

S. L. Smith, J. Tumova, C. Belta, and D. Rus. Optimal path planning under temporal logic constraints. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3288–3293, 2010.

T. Smith and R. Simmons. Point-Based POMDP Algorithms: Improved Analysis and Implementation. In *Proceedings of Uncertainty in Artificial Intelligence*, 2005.

O. Sokolsky and S. A. Smolka. Incremental Model Checking in the Modal mu-calculus. In *Computer Aided Verification (Lecture Notes in Computer Science, Vol. 818/1994)*, pages 351–363. Springer, 1994.

A. Stentz. The Focussed D* Algorithm for Real-Time Replanning. In *International Joint Conference on Artificial Intelligence*, 1995.

M. Stilman, J. Schamburek, J. Kuffner, and T. Asfour. Manipulation Planning Among Movable Obstacles. In *IEEE International Conference on Robotics and Automation*, pages 3327–3332, January 2007.

D. Stoyan, W. S. Kendall, and J. Mecke. *Stochastic Geometry and Its Applications.* Wiley, 2nd edition, 1995.

R. S. Strichartz. Sub-Riemannian Geometry. *Journal of Differential Geometry*, 24: 221–263, 1986.

P. Tabuada and G. J. Pappas. Model Checking LTL over Controllable Linear Systems Is Decidable. In *Hybrid Systems: Computation and Control*, pages 498–513, 2003.

P. Tabuada and G. J. Pappas. Linear Time Logic Control of Discrete-Time Linear Systems. *IEEE Transactions on Automatic Control*, 51(12):1862–1877, 2006.

R. Tedrake, I. R. Manchester, M. Tobenkin, and J. W. Roberts. LQR-trees: Feedback Motion Planning via Sums-of-Squares Verification. *The International Journal of Robotics Research*, 29(8):1038–1052, 2010.

S. Teller, A. Correa, R. Davis, L. Fletcher, E. Frazzoli, J. Glass, J. P. How, J. Jeon, S. Karaman, B. Luders, N. Roy, T. Sainath, and M. R. Walter. A Voice-Commandable Robotic Forklift Working Alongside Humans in Minimally-Prepared Outdoor Environments. In *IEEE International Conference on Robotics and Automation*, pages 526–533, Anchorage, AK, May 2010.

W. Thomas. *Automata on infinite objects.* Handbook of theoretical computer science (vol. B), 1991.

S. Thrun, W. Burgard, and D. Fox. *Probabilistic robotics.* The MIT Press, 2005.

C. Tomlin, G. J. Pappas, and S. S. Sastry. Conflict Resolution for Air Traffic Management:A Study in Multiagent Hybrid Systems. *IEEE Transactions on Automatic Control*, 43(4):509–521, 1998.

J. C. Trinkle and R. J. Milgram. Complete Path Planning for Closed Kinematic Chains with Spherical Joints. *The International Journal of Robotics Research*, 21 (9):773–789, 2002.

S. M. Udupa. Collision detection and avoidance in computer controlled manipulators. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, pages 737–748, 1977.

C. Urmson and R. Simmons. Approaches for heuristically biasing RRT growth. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2003.

A. Wade. Explicit laws of large numbers for random nearest-neighbour-type graphs. *Advances in applied probability*, January 2007.

A. R. Wade. Asymptotic theory for the multidimensional random on-line nearest-neighbour graph. *Stochastic Processes and their Applications*, January 2009.

D. J. Webb and J. van den Berg. Kinodynamic RRT*: Optimal Motion Planning for Systems with Linear Differential Constraints. *arXiv*, pages 1–8, 2012.

T. Wongpiromsarn, U. Topcu, and R. M. Murray. Receding Horizon Temporal Logic Planning for Dynamical Systems. In *IEEE Conference on Decision and Control*, October 2009.

F. Xue and P. Kumar. The number of neighbors needed for connectivity of wireless networks. *Wireless Networks*, January 2004.

A. Yershova and S. M. LaValle. Improving Motion Planning Algorithms by Efficient Nearest-Neighbor Searching. *IEEE Transactions on Robotics*, 23(1):151–157, 2007.

A. Yershova, L. Jaillet, T. Simeon, and S. M. LaValle. Dynamic-Domain RRTs: Efficient Exploration by Controlling the Sampling Domain. In *IEEE International Conference on Robotics and Automation*, pages 3856–3861, 2005.

M. Zucker, J. J. Kuffner, and M. S. Branicky. Multipartite RRTs for Rapid Replanning in Dynamic Environments. In *IEEE Conference on Robotics and Automation*, pages 1603–1609, 2007.