

Incremental Sampling-based Algorithms for Optimal Motion Planning

Sertac Karaman

Emilio Frazzoli

Abstract—During the last decade, incremental sampling-based motion planning algorithms, such as the Rapidly-exploring Random Trees (RRTs) have been shown to work well in practice and to possess theoretical guarantees such as probabilistic completeness. However, no theoretical bounds on the quality of the solution obtained by these algorithms have been established so far. The first contribution of this paper is a negative result: it is proven that, under mild technical conditions, the cost of the best path in the RRT converges almost surely to a non-optimal value. Second, a new algorithm is considered, called the Rapidly-exploring Random Graph (RRG), and it is shown that the cost of the best path in the RRG converges to the optimum almost surely. Third, a tree version of RRG is introduced, called the RRT* algorithm, which preserves the asymptotic optimality of RRG while maintaining a tree structure like RRT. The analysis of the new algorithms hinges on novel connections between sampling-based motion planning algorithms and the theory of random geometric graphs. In terms of computational complexity, it is shown that the number of simple operations required by both the RRG and RRT* algorithms is asymptotically within a constant factor of that required by RRT.

I. INTRODUCTION

The robotic 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. 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 has several applications in other disciplines such as verification, computational biology, and computer animation [1]–[6].

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 to 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. An algorithm that solves this problem is said to be *complete* if it returns a solution when one exists and returns failure otherwise.

Early algorithmic approaches to the motion planning problem mainly focused on developing complete planners for, e.g., polygonal robots moving among polygonal obstacles [7], and later for more general cases using algebraic techniques [8]. These algorithms, however, suffered severely from computational complexity, which prohibited their practical implementations; for instance, the algorithm in [8] had doubly

The authors are with the Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA.

Manuscript submitted to International Journal of Robotics Research.

exponential time complexity. Regarding the computational complexity of the problem, a remarkable result was proven as early as 1979: Reif showed that a most basic version of the motion planning problem, called the piano movers problem, was PSPACE-hard [9], which strongly suggested that complete planners would be unsuitable for practical applications.

Practical planners came around with the development of cell decomposition methods [10], [11], potential fields [12], and roadmap methods [13]. These approaches relaxed the completeness requirement to, for instance, *resolution completeness*, which guarantees completeness only when the resolution parameter of the algorithm is set fine enough. These planners demonstrated remarkable performance in accomplishing various tasks in complex environments within reasonable time bounds [14]. 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 [1], [15].

Arguably, some of the most influential recent advances in robotic motion planning can be found in sampling-based algorithms [16]–[18], which attracted a large amount of attention over the last decade, including very recent work (see, e.g., [19]–[25]). In summary, sampling-based algorithms connect a set of points that are chosen randomly from the obstacle-free space in order to build a roadmap, which is used to form a strong hypothesis of the connectivity of the environment, in particular the connectivity of the initial state and the set of goal states. Informally speaking, sampling-based methods provide large amounts of computational savings by avoiding explicit construction of obstacles in the state space, as opposed to most complete motion planning algorithms. Even though these algorithms do not achieve completeness, they provide *probabilistic completeness* guarantees in the sense that the probability that the planner fails to return a solution, if one exists, decays to zero as the number of samples approaches infinity [26]. Moreover, the rate of decay of the probability of failure is exponential, under the assumption that the environment has good “visibility” properties [26]. 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 [27].

A. Sampling-Based Algorithms

In practical applications, the class of sampling-based motion planning algorithms, including, for instance, Probabilistic Roadmaps (PRMs) [17], [28], and Rapidly-exploring Random

Trees (RRTs) [18], [29], is one of the most widely-used approaches to the motion planning problem [30], [31]. Even though the idea of connecting samples from the state space is essential in both approaches, these two algorithms mainly differ in the way that they construct the roadmap.

The PRM algorithm and its variants are multiple-query methods that first construct a graph (the roadmap), which represents a large portion of the collision-free trajectories, and then answer queries by computing a shortest path that connects the initial state with a final state through the graph. Shortly after their introduction in the literature, PRMs have been reported to extend to high-dimensional state spaces [17], since they avoid explicit construction of obstacles in the state space of the robot. Subsequently, it was shown that the PRM algorithm is probabilistically complete and the probability of failure decays to zero exponentially [28]. During the last decade, the PRM algorithm has been a focus of robotics research. In particular, several improvements were suggested by many authors and the reasons to why it performs well in several practical cases were better understood (see, e.g., [27], [32], [33] 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 have emerged as an online counterparts to the PRMs (see, e.g., [29], [34]). The incremental nature of these algorithms allows termination as soon as a solution is found. Furthermore, slight changes in the environment do not require planning from scratch, since most of the trajectories in the tree are still feasible (this is a property shared with most other roadmap-based methods) or can be improved to produce feasible trajectories quickly. These properties provide significant computational savings making online real-time implementations possible.

A widely-studied incremental algorithm is the RRT. Closely after its introduction in the literature, several theoretical guarantees provided by the RRT algorithm were shown, including probabilistic completeness [29] and exponential rate of decay of the probability of failure [35]. Subsequently, the RRT algorithm has been improved and found many applications in the robotics domain and elsewhere (see for instance [2]–[4], [35]–[37]). In particular, RRTs have been shown to work effectively for systems with differential constraints, nonlinear dynamics, and non-holonomic constraints [18], [35] as well as purely discrete or hybrid systems [36]. Moreover, the RRT algorithm was demonstrated in major robotics events on various experimental robotic platforms [38]–[42].

An interesting extension of the RRT that is worth mentioning at this point is the Rapidly-exploring Random Graph (RRG) algorithm [43]. The RRG algorithm was proposed as an incremental sampling-based method in order to extend RRTs to generate feasible trajectories for satisfying specifications other than the usual “avoid obstacles and reach the goal”. In [43], the authors have shown that the RRG algorithm

can handle any specification given in the form of μ -calculus, which includes the widely-used specification language Linear Temporal Logic (LTL) as a strict subclass. The RRG algorithm incrementally builds a graph, instead of a tree, since several specifications in LTL (e.g., ω -regular properties [44]) require cyclic trajectories, which are not included in trees.

B. Optimal Motion Planning

Generally, the standard robotic motion planning problem does not take the “quality” of the solution into account. That is, the focus of the problem is to find a trajectory that starts from the initial state and reaches one of the goal states while staying inside the obstacle-free portion of the state space. Let us note that, as mentioned earlier, even this form of the problem is computationally challenging [9]. Consequently, there has been a significant amount of research effort devoted to constructing feasible solutions in a computationally effective manner. Hence, most of the existing algorithms pay almost no attention to the other properties of solution, e.g., its length, or its cost. As a result, the analysis of the quality of the solution returned by the algorithm, or algorithms tailored to provably improve the quality of the solution if kept running, have received relatively little attention, even though their importance is mentioned in early seminal papers [18]. In fact, in many field implementations of such algorithms (see, e.g., [39]), it is often the case that a feasible path is found quickly, and additional computation time is devoted to improving the solution with heuristics until the solution is executed.

Yet, the importance of the quality of the solution returned by the planners have been noticed, in particular, from the point of view of incremental sampling-based motion planning. In [45], Urmson and Simmons have proposed heuristics to bias the tree growth in the 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. In [46], Ferguson and Stentz have 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 optimum solution. (Criteria for restarting multiple RRT runs, in a different context, were also proposed in [47].)

A different approach that also offers optimality guarantees is based on graph search algorithms (such as A*) applied over a grid that is generated offline. Recently, these algorithms received a large amount of research attention. In particular, they were extended to run in an anytime fashion [48], [49], deal with dynamic environments [49], [50], and handle systems with differential constraints [51]. They have also been successfully demonstrated on various robotic platforms [51], [52]. However, optimality guarantees of these algorithms are only ensured 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.

C. Statement of Contributions

To the best of our knowledge, this paper provides the first thorough analysis of optimality properties of incremental sampling-based motion planning algorithms. In particular, we show that the probability that the RRT converges to an optimum solution, as the number of samples approaches infinity, is zero under some reasonable technical assumptions. In fact, we give a formal proof of the fact that the RRT algorithm almost always converges to a suboptimal solution. Second, we show that the probability of the same event for the RRG algorithm is one. That is, the RRG algorithm is asymptotically optimal in the sense that it converges to an optimal solution almost surely as the number of samples approaches infinity. Third, we propose a novel variant of the RRG algorithm, called RRT*, which inherits the asymptotic optimality of the RRG algorithm while maintaining a tree structure. To do so, the RRT* algorithm essentially “rewires” the tree structure as it discovers new lower-cost paths reaching the nodes that are already in the tree. Finally, we show that the asymptotic computational complexity of the RRG and RRT* algorithms is essentially the same as that of RRTs. We also provide experimental evidence supporting our theoretical computational complexity results. We discuss the relation between the proposed algorithms and PRM-based methods, and propose a PRM-like algorithm that improves the efficiency of PRM while maintaining the same feasibility and optimality properties.

To our knowledge, the algorithms considered in this paper are the first computationally efficient incremental sampling-based motion planning algorithms with asymptotic optimality guarantees. Indeed, our results imply that these algorithms are optimal also from an asymptotic computational complexity point of view, since they closely match lower bounds for computing nearest neighbors. The key insight is that connections between vertices in the graph should be sought within balls whose radius vanishes with a certain rate as the size of the graph increases, and is based on new connections between motion planning and the theory of random geometric graphs [53], [54].

In this paper, we consider the problem of navigating through a connected bounded subset of a d -dimensional Euclidean space. As in the early seminal papers on incremental sampling-based motion planning algorithms such as [29], we assume no differential constraints, but our methods can be easily extended to planning in configuration spaces and applied to several practical problems of interest. We defer the extension to systems with differential constraints to another paper.

D. Paper Organization

This paper is organized as follows. In Section II, lays the ground in terms of notation and problem formulation. Section III is devoted to the introduction of the RRT and RRG algorithms. In Section IV, these algorithms are analyzed in terms of probabilistic completeness, asymptotic optimality, and computational complexity. The RRT* algorithm is presented in Section V, where it is shown that RRT* inherits the theoretical guarantees of the RRG algorithm. Experimental results are presented and discussed in Section VI. Concluding

remarks are given in Section VII. In order not to disrupt the flow of the presentation, proofs of important results are presented in the Appendix.

II. PRELIMINARY MATERIAL

A. Notation

Let \mathbb{N} denote the set of non-negative integers and \mathbb{R} denote the set of reals. Let $\mathbb{R}_{>0}$ and $\mathbb{R}_{\geq 0}$ denote the sets of positive and non-negative reals, respectively. The set $\mathbb{N}_{>0}$ is defined similarly. 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. More generally, if $a, b \in \mathbb{R}^d$, then $[a, b]$ is used to denote the line segment that connects a with b , i.e., $[a, b] := \{\tau a + (1 - \tau)b \mid \tau \in \mathbb{R}, 0 \leq \tau \leq 1\}$. The Euclidean norm is denoted by $\|\cdot\|$. Given a set $X \subset \mathbb{R}^d$, the closure of X is denoted by $\text{cl}(X)$. The closed ball of radius $r > 0$ centered at $x \in \mathbb{R}^d$ is defined as $\mathcal{B}_{x,r} := \{y \in \mathbb{R}^d \mid \|y - x\| \leq r\}$; $\mathcal{B}_{x,r}$ is also called the r -ball centered at x . Given a set $X \subseteq \mathbb{R}^d$, the Lebesgue measure of X , i.e., its volume, is denoted by $\mu(X)$. The volume of the unit ball in \mathbb{R}^d is denoted by ζ_d .

Given a set $X \subset \mathbb{R}^d$, and a scalar $s \geq 0$, a path in X is a continuous function $\sigma : [0, s] \rightarrow X$, where s is the length of the path defined in the usual way as $\sup_{\{n \in \mathbb{N}, 0 = \tau_0 < \tau_1 < \dots < \tau_n = s\}} \sum_{i=1}^n \|\sigma(\tau_i) - \sigma(\tau_{i-1})\|$. Given two paths in X , $\sigma_1 : [0, s_1] \rightarrow X$, and $\sigma_2 : [0, s_2] \rightarrow X$, with $\sigma_1(s_1) = \sigma_2(0)$, their concatenation is denoted by $\sigma_1|\sigma_2$. More precisely, $\sigma = \sigma_1|\sigma_2 : [0, s_1 + s_2] \rightarrow X$ is defined as $\sigma(s) = \sigma_1(s)$ for all $s \in [0, s_1]$, and $\sigma(s) = \sigma_2(s - s_1)$ for all $s \in [s_1, s_1 + s_2]$. More generally, a concatenation of multiple paths $\sigma_1|\sigma_2|\dots|\sigma_n$ is defined as $(\dots((\sigma_1|\sigma_2)|\sigma_3)|\dots|\sigma_n)$. The set of all paths in X with nonzero length is denoted by Σ_X . The straight continuous path between two points $x_1, x_2 \in \mathbb{R}^d$ is denoted by $\text{Line}(x_1, x_2)$.

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 $\{A_i\}_{i \in \mathbb{N}}$, the event $\cap_{j=1}^{\infty} \cup_{i=j}^{\infty} A_i$ is denoted by $\limsup_{i \rightarrow \infty} A_i$ (also called the event that A_i occurs infinitely often). A (real) random variable is a measurable function that maps Ω into \mathbb{R} . An extended (real) random variable can also take the values $\pm\infty$. A sequence of random variables $\{\mathcal{Y}_i\}_{i \in \mathbb{N}}$ is said to converge surely to a random variable \mathcal{Y} if $\lim_{i \rightarrow \infty} \mathcal{Y}_i(\omega) = \mathcal{Y}(\omega)$ for all $\omega \in \Omega$; the sequence is said to converge almost surely if $\mathbb{P}(\{\lim_{i \rightarrow \infty} \mathcal{Y}_i = \mathcal{Y}\}) = 1$. Finally, if $\varphi(\omega)$ is a property that is either true or false for a given $\omega \in \Omega$, the event that denotes the set of all samples ω for which $\varphi(\omega)$ holds, i.e., $\{\omega \in \Omega \mid \varphi(\omega) \text{ holds}\}$, is written as $\{\varphi\}$, e.g., $\{\omega \in \Omega \mid \lim_{i \rightarrow \infty} \mathcal{Y}_i(\omega) = \mathcal{Y}(\omega)\}$ is simply written as $\{\lim_{i \rightarrow \infty} \mathcal{Y}_i = \mathcal{Y}\}$.

Let $f(n)$ and $g(n)$ be two functions with domain and range \mathbb{N} . 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 X be a subset of \mathbb{R}^d . A (directed) graph $G = (V, E)$ on X is composed of a vertex set V and an edge set E , such that V is a finite subset of X , and E is a subset of $V \times V$. The set of all directed graphs on X is denoted by \mathcal{G}_X . A directed path on G is a sequence (v_1, v_2, \dots, 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 $\{u \in V \mid (u, v) \in E\}$ and $\{u \in V \mid (v, u) \in E\}$ 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. Vertices of a tree are often also called nodes.

B. Problem Formulation

In this section, two variants of the path planning problem are presented. First, the feasibility problem in path planning is formalized, then the optimality problem is introduced.

Let X be a bounded connected open subset of \mathbb{R}^d , where $d \in \mathbb{N}$, $d \geq 2$. Let X_{obs} and X_{goal} , called the *obstacle region* and the *goal region*, respectively, be open subsets of X . Let us denote the *obstacle-free space*, i.e., $X \setminus X_{\text{obs}}$, as X_{free} . Let the *initial state*, x_{init} , be an element of X_{free} . In the sequel, a path in X_{free} is said to be a *collision-free path*. A collision-free path that starts at x_{init} and ends in the goal region is said to be a *feasible path*, i.e., a collision-free path $\sigma : [0, s] \rightarrow X_{\text{free}}$ is feasible if and only if $\sigma(0) = x_{\text{init}}$ and $\sigma(s) \in X_{\text{goal}}$.

The *feasibility problem* of path planning is to find a feasible path, if one exists, and report failure otherwise. The problem can be formalized as follows.

Problem 1 (Feasible planning) *Given a bounded connected open set $X \subset \mathbb{R}^d$, an obstacle space $X_{\text{obs}} \subset X$, an initial state $x_{\text{init}} \in X_{\text{free}}$, and a goal region $X_{\text{goal}} \subset X_{\text{free}}$, find a path $\sigma : [0, s] \rightarrow X_{\text{free}}$ such that $\sigma(0) = x_{\text{init}}$ and $\sigma(s) \in X_{\text{goal}}$, if one exists. If no such path exists, then report failure.*

Let $c : \Sigma_{X_{\text{free}}} \rightarrow \mathbb{R}_{>0}$ be a function, called the *cost function*, which assigns a non-negative cost to all nontrivial collision-free paths. The *optimality problem* of path planning asks for finding a feasible path with minimal cost, formalized as follows.

Problem 2 (Optimal planning) *Given a bounded connected open set X , an obstacle space X_{obs} , an initial state x_{init} , and a goal region X_{goal} , find a path $\sigma^* : [0, s] \rightarrow \text{cl}(X_{\text{free}})$ such that (i) $\sigma^*(0) = x_{\text{init}}$ and $\sigma^*(s) \in X_{\text{goal}}$, and (ii) $c(\sigma^*) = \min_{\sigma \in \Sigma_{\text{cl}(X_{\text{free}})}} c(\sigma)$. If no such path exists, then report failure.*

III. ALGORITHMS

In this section, two incremental sampling-based motion planning algorithms, namely the RRT and the RRG algorithms, are introduced. Before formalizing the algorithms, let us note the primitive procedures that they rely on.

Sampling: The function $\text{Sample} : \mathbb{N} \rightarrow X_{\text{free}}$ returns independent identically distributed (i.i.d.) samples from X_{free} .

Steering: Given two points $x, y \in X$, the function $\text{Steer} : (x, y) \mapsto z$ returns a point $z \in \mathbb{R}^d$ such that z is “closer” to y than x is. Throughout the paper, the point z returned by the function Steer will be such that z minimizes $\|z - y\|$ while at the same time maintaining $\|z - x\| \leq \eta$, for a prespecified $\eta > 0$,¹ i.e.,

$$\text{Steer}(x, y) = \operatorname{argmin}_{z \in \mathbb{R}^d, \|z - x\| \leq \eta} \|z - y\|.$$

Nearest Neighbor: Given a graph $G = (V, E) \in \mathcal{G}_{X_{\text{free}}}$ and a point $x \in X_{\text{free}}$, the function $\text{Nearest} : (G, x) \mapsto v$ returns a vertex $v \in V$ that is “closest” to x in terms of a given distance function. In this paper, we will use Euclidean distance (see, e.g., [18] for alternative choices), i.e.,

$$\text{Nearest}(G = (V, E), x) = \operatorname{argmin}_{v \in V} \|x - v\|.$$

Near Vertices: Given a graph $G = (V, E) \in \mathcal{G}_{X_{\text{free}}}$, a point $x \in X_{\text{free}}$, and a number $n \in \mathbb{N}$, the function $\text{Near} : (G, x, n) \mapsto V'$ returns a set V' of vertices such that $V' \subseteq V$. The Near procedure can be thought of as a generalization of the nearest neighbor procedure in the sense that the former returns a collection of vertices that are close to x , whereas the latter returns only one such vertex that is the closest. Just like the Nearest procedure, there are many ways to define the Near procedure, each of which leads to different algorithmic properties. For technical reasons to become clear later, we define $\text{Near}(G, x, n)$ to be the set of all vertices within the closed ball of radius r_n centered at x , where

$$r_n = \min \left\{ \left(\frac{\gamma}{\zeta_d} \frac{\log n}{n} \right)^{1/d}, \eta \right\},$$

and γ is a constant. Hence, the volume of this ball is $\min\{\gamma \frac{\log n}{n}, \zeta_d \eta^d\}$.

Collision Test: Given two points $x, x' \in X_{\text{free}}$, the Boolean function $\text{ObstacleFree}(x, x')$ returns True iff the line segment between x and x' lies in X_{free} , i.e., $[x, x'] \subset X_{\text{free}}$.

Both the RRT and the RRG algorithms are similar to most other incremental sampling-based planning algorithms (see Algorithm 1). Initially, the algorithms start with the graph that includes the initial state as its single vertex and no edges; then, they incrementally grow a graph on X_{free} by sampling a state x_{rand} from X_{free} at random and extending the graph towards x_{rand} . In the sequel, every such step of sampling followed by extensions (Lines 2-5 of Algorithm 1) is called a single *iteration* of the incremental sampling-based algorithm.

Hence, the body of both algorithms, given in Algorithm 1, is the same. However, RRGs and RRTs differ in the choice of the vertices to be extended. The *Extend* procedures for the RRT and the RRG algorithms are provided in Algorithms 2 and 3, respectively. Informally speaking, the RRT algorithm extends the nearest vertex towards the sample. The RRG algorithm first extends the nearest vertex, and if such extension is successful, it also extends all vertices returned by the Near procedure. In both cases, all extensions resulting in collision-free trajectories

¹This steering procedure is used widely in the robotics literature, since its introduction in [29]. Our results also extend to the Rapidly-exploring Random Dense Trees (see, e.g., [30]), which are slightly modified versions of the RRTs that do not require tuning any prespecified parameters such as η in this case.

are added to the graph as edges, and their terminal points as new vertices.

Algorithm 1: Body of RRT and RRG Algorithms

```

1  $V \leftarrow \{x_{\text{init}}\}; E \leftarrow \emptyset; i \leftarrow 0;$ 
2 while  $i < N$  do
3    $G \leftarrow (V, E);$ 
4    $x_{\text{rand}} \leftarrow \text{Sample}(i); i \leftarrow i + 1;$ 
5    $(V, E) \leftarrow \text{Extend}(G, x_{\text{rand}});$ 

```

Algorithm 2: Extend_{RRT}

```

1  $V' \leftarrow V; E' \leftarrow E;$ 
2  $x_{\text{nearest}} \leftarrow \text{Nearest}(G, x);$ 
3  $x_{\text{new}} \leftarrow \text{Steer}(x_{\text{nearest}}, x);$ 
4 if ObstacleFree( $x_{\text{nearest}}, x_{\text{new}}$ ) then
5    $V' \leftarrow V' \cup \{x_{\text{new}}\};$ 
6    $E' \leftarrow E' \cup \{(x_{\text{nearest}}, x_{\text{new}})\};$ 
7 return  $G' = (V', E')$ 

```

Algorithm 3: Extend_{RRG}

```

1  $V' \leftarrow V; E' \leftarrow E;$ 
2  $x_{\text{nearest}} \leftarrow \text{Nearest}(G, x);$ 
3  $x_{\text{new}} \leftarrow \text{Steer}(x_{\text{nearest}}, x);$ 
4 if ObstacleFree( $x_{\text{nearest}}, x_{\text{new}}$ ) then
5    $V' \leftarrow V' \cup \{x_{\text{new}}\};$ 
6    $E' \leftarrow E' \cup \{(x_{\text{nearest}}, x_{\text{new}}), (x_{\text{new}}, x_{\text{nearest}})\};$ 
7    $X_{\text{near}} \leftarrow \text{Near}(G, x_{\text{new}}, |V|);$ 
8   for all  $x_{\text{near}} \in X_{\text{near}}$  do
9     if ObstacleFree( $x_{\text{new}}, x_{\text{near}}$ ) then
10       $E' \leftarrow E' \cup \{(x_{\text{near}}, x_{\text{new}}), (x_{\text{new}}, x_{\text{near}})\};$ 
11 return  $G' = (V', E')$ 

```

Notice that the graph G maintained by the RRT algorithm is a tree, whereas that maintained by the RRG can, in general, include vertices with more than one incoming neighbor.

IV. ANALYSIS

A. Convergence to a Feasible Solution

In this section, the feasibility problem is considered. It is proven that the RRG algorithm inherits the probabilistic completeness as well as the exponential decay of the probability of failure (as the number of samples increase) from the RRT. These results imply that the RRT and RRG algorithms have the same performance in producing a solution to the feasibility problem as the number of samples increase. Before formalizing these claims, let us introduce some notation and preliminaries.

Note that the only randomness in Algorithms 1-3 may arise from the sampling procedure denoted as `Sample`.² To model

²We will not address the case in which the sampling procedure is deterministic, but refer the reader to [55], which contains an in-depth discussion of the relative merits of randomness and determinism in sampling-based motion planning algorithms.

this randomness, we define the sample space Ω to be the infinite cartesian product X_{free}^∞ . Intuitively, a single element ω of Ω denotes a sequence $\{\text{Sample}(i)\}_{i \in \mathbb{N}}$ of infinitely many samples drawn from X_{free} , and Ω denotes the set of all such sequences.

Sets of vertices and edges of the graphs maintained by the RRT and the RRG algorithms, then, can be defined as functions from the sample space Ω to appropriate sets. More precisely, let $\{\mathcal{V}_i^{\text{RRT}}\}_{i \in \mathbb{N}}$ and $\{\mathcal{V}_i^{\text{RRG}}\}_{i \in \mathbb{N}}$, sequences of functions defined from Ω into finite subsets of X_{free} , be the sets of vertices in the RRT and the RRG, respectively, at the end of iteration i . By convention, we define $\mathcal{V}_0^{\text{RRT}} = \mathcal{V}_0^{\text{RRG}} = \{x_{\text{init}}\}$. Similarly, let $\mathcal{E}_i^{\text{RRT}}$ and $\mathcal{E}_i^{\text{RRG}}$, defined for all $i \in \mathbb{N}$, denote the set of edges in the RRT and the RRG, respectively, at the end of iteration i . Clearly, $\mathcal{E}_0^{\text{RRT}} = \mathcal{E}_0^{\text{RRG}} = \emptyset$.

An important lemma used for proving the equivalency between the RRT and the RRG algorithms is the following.

Lemma 3 For all $i \in \mathbb{N}$ and all $\omega \in \Omega$, $\mathcal{V}_i^{\text{RRT}}(\omega) = \mathcal{V}_i^{\text{RRG}}(\omega)$ and $\mathcal{E}_i^{\text{RRT}}(\omega) \subseteq \mathcal{E}_i^{\text{RRG}}(\omega)$.

Lemma 3 implies that the paths discovered by the RRT algorithm by the end of iteration i is, essentially, a subset of those discovered by the RRG by the end of the same iteration.

An algorithm addressing Problem 1 is said to be *probabilistically complete* if it finds a feasible path with probability approaching one as the number of iterations approaches infinity. Note that there exists a collision-free path starting from x_{init} to any vertex in the tree maintained by the RRT, since the RRT maintains a connected graph on X_{free} that necessarily includes x_{init} . Using this fact, the probabilistic completeness property of the RRT is stated alternatively as follows.

Theorem 4 (see [18]) If there exists a feasible solution to Problem 1, then $\lim_{i \rightarrow \infty} \mathbb{P}(\{\mathcal{V}_i^{\text{RRT}} \cap X_{\text{goal}} \neq \emptyset\}) = 1$.

Moreover, under certain assumptions on the environment, the probability that the RRT fails to find a feasible path, even though one exists, decays to zero exponentially (see, e.g., [18], [34]). We state these assumptions here and then state the theorem in our notation.

An *attraction sequence* [18] is defined as a finite sequence $\mathcal{A} = \{A_1, A_2, \dots, A_k\}$ of sets as follows: (i) $A_0 = \{x_{\text{init}}\}$, and (ii) for each set A_i , there exists a set B_i , called the *basin* such that for any $x \in A_{i-1}$, $y \in A_i$, and $z \in X \setminus B_i$, there holds $\|x - y\| \leq \|x - z\|^3$. Given an attraction sequence \mathcal{A} of length k , let p_k denote $\min_{i \in \{1, 2, \dots, k\}} \left(\frac{\mu(A_i)}{\mu(X_{\text{free}})} \right)$.

The following theorem states that the probability that the RRT algorithm fails to return a solution, when one exists, decays to zero exponentially fast.

Theorem 5 (see [18]) If there exists an attraction sequence \mathcal{A} of length k , then $\mathbb{P}(\{\mathcal{V}_i^{\text{RRT}} \cap X_{\text{goal}} = \emptyset\}) \leq e^{-\frac{1}{2}(i p_k - 2k)}$.

As noted in [18], the convergence gets faster as the length of the attraction sequence gets smaller and the volume of

³Since we do not consider any differential constraints, some of the other assumptions introduced in [18] are immediately satisfied.

the minimum volume set in the attraction sequence gets larger. Such properties are achieved when the environment does not involve narrow passages and, essentially, has good “visibility” properties. (See, e.g., [27] for dependence of the performance of probabilistic planners such as PRMs on the visibility properties of the environment).

With Lemma 3 and Theorems 4 and 5, the following theorem is immediate.

Theorem 6 *If there exists a feasible solution to Problem 1, then $\lim_{i \rightarrow \infty} \mathbb{P}(\{\mathcal{V}_i^{\text{RRG}} \cap X_{\text{goal}} \neq \emptyset\}) = 1$. Moreover, if an attraction sequence \mathcal{A} of length k exists, then $\mathbb{P}(\{\mathcal{V}_i^{\text{RRG}} \cap X_{\text{goal}} = \emptyset\}) \leq e^{-\frac{1}{2}(i p_k - 2^k)}$.*

B. Asymptotic Optimality

This section is devoted to the investigation of optimality properties of the RRT and the RRG algorithms. First, under some mild technical assumptions, we show that the probability that the RRT converges to an optimal solution is zero. However, the convergence of this random variable is guaranteed, which implies that the RRT converges to a *non-optimum* solution with probability one. On the contrary, we subsequently show that the RRG algorithm converges to an optimum solution almost-surely.

Let $\{\mathcal{Y}_i^{\text{RRT}}\}_{i \in \mathbb{N}}$ be a sequence of extended random variables that denote the cost of a minimum-cost path contained within the tree maintained by the RRT algorithm at the end of iteration i . The extended random variable $\mathcal{Y}_i^{\text{RRG}}$ is defined similarly. Let c^* denote the cost of a minimum-cost path in $\text{cl}(X_{\text{free}})$, i.e., the cost of a path that solves Problem 2.

Let us note that the limits of these two extended random variable sequences as i approaches infinity exist. More formally, notice that $\mathcal{Y}_{i+1}^{\text{RRT}}(\omega) \leq \mathcal{Y}_i^{\text{RRT}}(\omega)$ holds for all $i \in \mathbb{N}$ and all $\omega \in \Omega$. Moreover, we have that $\mathcal{Y}_i^{\text{RRT}}(\omega) \geq c^*$ for all $i \in \mathbb{N}$ and all $\omega \in \Omega$, by optimality of c^* . Hence, $\{\mathcal{Y}_i^{\text{RRT}}\}_{i \in \mathbb{N}}$ is a surely non-increasing sequence of random variables that is surely lower-bounded by c^* . Thus, for all $\omega \in \Omega$, the limit $\lim_{i \rightarrow \infty} \mathcal{Y}_i^{\text{RRT}}(\omega)$ exists. The same argument also holds for the sequence $\{\mathcal{Y}_i^{\text{RRG}}\}_{i \in \mathbb{N}}$.

1) *Almost Sure Suboptimality of the RRT:* Let us note the following assumptions, which will be required to show the almost-sure sub-optimality of the RRT. Let Σ^* denote the set of all optimal paths, i.e., the set of all paths that solve Problem 2, and X_{opt} denote the set of states that an optimal path in Σ^* passes through, i.e.,

$$X_{\text{opt}} = \cup_{\sigma^* \in \Sigma^*} \cup_{\tau \in [0, s^*]} \{\sigma^*(\tau)\}$$

Assumption 7 (Zero-measure Optimal Paths) *The set of all points in the state-space that an optimal trajectory passes through has measure zero, i.e., $\mu(X_{\text{opt}}) = 0$.*

Assumption 8 (Sampling Procedure) *The sampling procedure is such that the samples $\{\text{Sample}(i)\}_{i \in \mathbb{N}}$ are drawn from an absolutely continuous distribution with a continuous density function $f(x)$ bounded away from zero on X_{free} .*

Assumption 9 (Monotonicity of the Cost Function) *For all $\sigma_1, \sigma_2 \in \Sigma_{X_{\text{free}}}$, the cost function c satisfies the following: $c(\sigma_1) \leq c(\sigma_1 | \sigma_2)$.*

Assumptions 7 rules out trivial cases, in which the RRT algorithm can sample exactly an optimal path with non-zero probability. Note that this assumption is placed on the problem instance rather than the algorithm. Most cost functions and problem instances of interest satisfy this assumption, including, e.g., the Euclidean length of the path. Let us note that this assumption does not imply that there is a single optimal path; indeed, there are problem instances with uncountably many optimal paths, for which Assumption 7 holds. Assumption 8 also ensures that the sampling procedure can not be tuned to construct the optimal path exactly. Finally, Assumption 9 merely states that extending a path to produce a longer path can not decrease its cost.

Recall that d denotes the dimensionality of the state space. The negative result of this section is formalized as follows.

Theorem 10 *Let Assumptions 7, 8, and 9 hold. Then, the probability that the cost of the minimum-cost path in the RRT converges to the optimal cost is zero, i.e.,*

$$\mathbb{P}\left(\left\{\lim_{i \rightarrow \infty} \mathcal{Y}_i^{\text{RRT}} = c^*\right\}\right) = 0,$$

whenever $d \geq 2$.

As noted before, the limit $\lim_{i \rightarrow \infty} \mathcal{Y}_i^{\text{RRT}}(\omega)$ exists and is a random variable. However, Theorem 10 directly implies that this limit is strictly greater than c^* with probability one, i.e., $\mathbb{P}(\{\lim_{i \rightarrow \infty} \mathcal{Y}_i^{\text{RRT}} > c^*\}) = 1$. In other words, we establish, as a corollary, that the RRT algorithm converges to a suboptimal solution with probability one.

Remark 11 (Effectiveness of Multiple RRTs) *Since the cost of the best path returned by the RRT algorithm converges to a random variable, say $\mathcal{Y}_{\infty}^{\text{RRT}}$, Theorem 10 provides new insight explaining the effectiveness of approaches as in [46]. In fact, running multiple instances of the RRT algorithm amounts to drawing multiple samples of $\mathcal{Y}_{\infty}^{\text{RRT}}$.*

2) *Almost Sure Optimality of the RRG:* Let us note the following set of assumptions, which will be required to show the asymptotic optimality of the RRG.

Assumption 12 (Additivity of the Cost Function) *For all $\sigma_1, \sigma_2 \in \Sigma_{X_{\text{free}}}$, the cost function c satisfies the following: $c(\sigma_1 | \sigma_2) = c(\sigma_1) + c(\sigma_2)$.*

Assumption 13 (Continuity of the Cost Function) *The cost function c is Lipschitz continuous in the following sense: there exists some constant κ such that for any two paths $\sigma_1 : [0, s_1] \rightarrow X_{\text{free}}$ and $\sigma_2 : [0, s_2] \rightarrow X_{\text{free}}$,*

$$|c(\sigma_1) - c(\sigma_2)| \leq \kappa \sup_{\tau \in [0, 1]} \|\sigma_1(\tau s_1) - \sigma_2(\tau s_2)\|.$$

Assumption 14 (Obstacle Spacing) *There exists a constant $\delta \in \mathbb{R}_+$ such that for any point $x \in X_{\text{free}}$, there exists $x' \in$*

X_{free} , such that (i) the δ -ball centered at x' lies inside X_{free} , i.e., $\mathcal{B}_{x', \delta} \subset X_{\text{free}}$, and (ii) x lies inside the δ -ball centered at x' , i.e., $x \in \mathcal{B}_{x', \delta}$.

Assumption 13 ensures that two paths that are very close to each other have similar costs. Let us note that several cost functions of practical interest satisfy Assumptions 12 and 13. An example that is widely used, e.g., in optimal control, is the line integral of a continuous function $g : X_{\text{free}} \rightarrow \mathbb{R}_{>0}$ over trajectories of bounded length, i.e., $c(\sigma) = \int_0^s g(\sigma(\tau))d\tau$. Assumption 14 is a rather technical assumption, which ensures existence of some free space around the optimal trajectories to allow convergence. We also assume for simplicity that the sampling is uniform, although our results can be easily extended to more general sampling procedures.

Recall that d is the dimensionality of the state-space X , and γ is the constant defined in the `Near` procedure. The positive result that states the asymptotic optimality of the RRG algorithm can be formalized as follows.

Theorem 15 *Let Assumptions 12, 13, and 14 hold, and assume that Problem 1 admits a feasible solution. Then, the cost of the minimum-cost path in the RRG converges to c^* almost surely, i.e.,*

$$\mathbb{P}\left(\left\{\lim_{i \rightarrow \infty} \mathcal{Y}_i^{\text{RRG}} = c^*\right\}\right) = 1,$$

wherever $d \geq 2$ and $\gamma > \gamma_L := 2^d(1 + 1/d)\mu(X_{\text{free}})$.

C. Computational Complexity

The objective of this section is to compare the computational complexity of RRTs and RRGs. It is shown that these algorithms share essentially the same asymptotic computational complexity in terms of the number of calls to *simple operations* such as comparisons, additions, and multiplications. Simple operations should not be confused with *primitive procedures* outlined in Section III, which are higher level functions that may have different asymptotic computational complexities in terms of the number of simple operations that they perform. Note that the objective of this section is not the evaluation of the computational complexity in absolute terms, but to compare the relative complexity of the two algorithms, hence the input of interest is the number of iterations, as opposed to parameters describing the problem instance.

Let us first investigate the computational complexity of the RRT and the RRG algorithms in terms of the number of calls to the primitive procedures. Notice that, in every iteration, the number of calls to `Sample`, `Steer`, and `Nearest` procedures are the same in both algorithms. However, number of calls to `Near` and `ObstacleFree` procedures differ: the former is never called by the RRT and is called at most once by the RRG, whereas the latter is called exactly once by the RRT and at least once by the RRG. Informally speaking, we first show that the expected number of calls to the `ObstacleFree` procedure by the RRG algorithm is $O(\log n)$, where n is the number of vertices in the graph.

Let $\mathcal{O}_i^{\text{RRG}}$ be a random variable that denotes the number of calls to the `ObstacleFree` procedure by the RRG algorithm in iteration i . Notice that, as an immediate corollary of Lemma 3,

the number of vertices in the RRT and RRG algorithms is the same at any given iteration. Let \mathcal{N}_i be the number of vertices in these algorithms at the end of iteration i . The following theorem establishes that the expected number of calls to the `ObstacleFree` procedure in iteration i by the RRG algorithm scales logarithmically with the number of vertices in the graph as i approaches infinity.

Lemma 16 *In the limit as i approaches infinity, the random variable $\mathcal{O}_i^{\text{RRG}} / \log(\mathcal{N}_i)$ is no more than a constant in expectation, i.e.,*

$$\limsup_{i \rightarrow \infty} \mathbb{E}\left[\frac{\mathcal{O}_i^{\text{RRG}}}{\log(\mathcal{N}_i)}\right] \leq \phi,$$

where $\phi \in \mathbb{R}_{>0}$ is a constant that depends only on the problem instance.

Hence, informally speaking, we have established that the RRG algorithm has an overhead of order $\log n$ calls to the `ObstacleFree` procedure and one call to the `Near` procedure, where n is the number of vertices in the RRG; otherwise, both algorithms scale the same in terms of number of calls to the primitive procedures.

However, some primitive procedures clearly take more computation time to execute than others. For a more fair comparison, we also evaluate the time required to execute each primitive procedure in terms of the number of simple operations (also called *steps* in this paper) that they perform. This analysis shows that the expected number of simple operations performed by the RRG is asymptotically within a constant factor of that performed by the RRT, which establishes that the RRT and the RRG algorithms have the same asymptotic computational complexity in terms of number of calls to the simple operations.

First, notice that `Sample`, `Steer`, and `ObstacleFree` procedures can be performed in a constant number of steps, i.e., independent of the number of vertices in the graph. (Although all these procedures clearly depend on parameters such as the dimensionality of the state-space and the number of obstacles, these are fixed for each problem instance.)

Second, let us consider the computational complexity of the `Nearest` procedure. Indeed, the nearest neighbor search problem has been widely studied in the literature, since it has many applications in, e.g., computer graphics, database systems, image processing, data mining, pattern recognition, etc. [56], [57]. Let us assume that the distance computation between two points can be done in $O(d)$ time (note that for the purposes of comparing distances, the Euclidean distance can be evaluated without computing any roots). Clearly, a brute-force algorithm that examines every vertex runs in $O(dn)$ time and requires $O(1)$ space. However, in many online real-time applications such as robotics, it is highly desirable to reduce the computation time of each iteration under sublinear bounds, especially for anytime algorithms, which provide better solutions as the number of iterations increase.

Fortunately, it is possible to design nearest neighbor computation algorithms that run in sublinear time. Generally, such efficient computations of nearest neighbors are based on constructing a tree structure online so as to reduce the

query time [57]. Indeed, using k -d trees, nearest neighbor search can be performed within sublinear time bounds, in the worst case [57]. In fact, let us note that the k -d tree algorithm was extended to arbitrary topological manifolds and used for nearest neighbor computation in RRTs [58].

Worst case logarithmic time, however, is hard to achieve in many cases of practical interest. Note that when $d = 1$, a binary tree [59] will allow answering nearest neighbor queries in $O(\log n)$ time using $O(n)$ space, in the worst case. When $d = 2$, similar logarithmic time bounds can also be achieved by, for instance, computing a Voronoi diagram [60]. However, the time complexity of computing Voronoi diagrams is known to be $O(n^{\lceil d/2 \rceil})$ when $d > 2$. Several algorithms that achieve a worst-case query time that is logarithmic in the number of vertices and polynomial in the number of dimensions were designed (see, e.g., [61]). However, these algorithms use space that is exponential in d , which may be impractical. Unfortunately, no exact nearest neighbor algorithm that does not scale exponentially with the number of dimensions and runs queries in logarithmic time using roughly linear space is known [62].

Fortunately, computing an “approximate” nearest neighbor instead of an exact one is computationally easier. In the sequel, a vertex y is said to be an ε -approximate nearest neighbor of a point x if $\|y - x\| \leq (1 + \varepsilon) \|z - x\|$, where z is the true nearest neighbor of x . An approximate nearest neighbor can be computed using balanced-box decomposition (BBD) trees, which achieves $O(c_{d,\varepsilon} \log n)$ query time using $O(dn)$ space [62], where $c_{d,\varepsilon} \leq d^{\lceil 1 + 6d/\varepsilon \rceil^d}$. 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 [62]. Let us note that using approximate nearest neighbor computation in the context of both PRMs and RRTs was also discussed very recently [63].

Assuming that the `Nearest` procedure computes an approximate nearest neighbor using the algorithm given in [62], in fixed dimensions the `NearestNeighbor` algorithm has to run in $\Omega(\log n)$ time as formalized in the following lemma. Let $\mathcal{M}_i^{\text{RRT}}$ be the random variable that denotes the number of steps taken by the RRT algorithm in iteration i .

Lemma 17 *Assuming that `Nearest` is implemented using the algorithm given in [62], which is optimal in fixed dimensions, the number of steps executed by the RRT algorithm at each iteration is at least order $\log(\mathcal{N}_i)$ in expectation in the limit, i.e., there exists a constant $\phi_{\text{RRT}} \in \mathbb{R}_{>0}$ such that*

$$\liminf_{i \rightarrow \infty} \mathbb{E} \left[\frac{\mathcal{M}_i^{\text{RRT}}}{\log(\mathcal{N}_i)} \right] \geq \phi_{\text{RRT}}.$$

Likewise, problems similar to that solved by the `Near` procedure are also widely-studied in the literature, generally under the name of *range search problems*, as they have many applications in, for instance, computer graphics and spatial database systems [57]. In the worst case and in fixed dimensions, computing the exact set of vertices that reside in a ball of radius r_n centered at a query point x takes $O(n^{1-1/d} + m)$ time using k -d trees [64], where m is the number of vertices returned by the search. In [65], a thorough

analysis of the average case performance is also provided under certain conditions.

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 $(1 + \varepsilon) r_n$, but may or may not return the vertices that lie outside the former ball and inside the latter ball. Computing ε -approximate solutions using BBD-trees requires $O(2^d \log n + d^2 (3\sqrt{d}/\varepsilon)^{d-1})$ time when using $O(dn)$ space, in the worst case [66]. Thus, in fixed dimensions, the complexity of this algorithm is $O(\log n + (1/\varepsilon)^{d-1})$, which is known to be optimal, closely matching a lower bound [66]. More recently, algorithms that can provide trade-offs between time and space were also proposed [67].

Note that the `Near` procedure can be implemented as an approximate range search while maintaining the asymptotic optimality guarantee. Notice that the expected number of vertices returned by the `Near` procedure also does not change, except by a constant factor. Hence, the `Near` procedure can be implemented to run in order $\log n$ expected time in the limit and linear space in fixed dimensions.

Let $\mathcal{M}_i^{\text{RRG}}$ denote the number of steps performed by the RRG algorithm in iteration i . Then, together with Lemma 16, the discussion above implies the following lemma.

Lemma 18 *The number of steps executed by the RRG algorithm at each iteration is at most order $\log(\mathcal{N}_i)$ in expectation in the limit, i.e., there exists a constant $\phi_{\text{RRG}} \in \mathbb{R}_{>0}$ such that*

$$\limsup_{i \rightarrow \infty} \mathbb{E} \left[\frac{\mathcal{M}_i^{\text{RRG}}}{\log(\mathcal{N}_i)} \right] \leq \phi_{\text{RRG}}.$$

Finally, by Lemmas 17 and 18, we conclude that the RRT and the RRG algorithms have the same asymptotic computational complexity as stated in the following theorem.

Theorem 19 *Under the assumptions of Lemmas 17 and 18, there exists a constant $\phi \in \mathbb{R}_{>0}$ such that*

$$\limsup_{i \rightarrow \infty} \mathbb{E} \left[\frac{\mathcal{M}_i^{\text{RRG}}}{\mathcal{M}_i^{\text{RRT}}} \right] \leq \phi.$$

In section VI, we also provide some experimental evidence supporting Theorem 19.

D. On the efficient construction of Probabilistic RoadMaps

At this point, let us note that our results also imply an efficient PRM planning algorithm. The PRM algorithm samples a set of n vertices from the state-space, and checks connectivity of each pair of samples via the `Steer` and `ObstacleFree` procedures to build a graph (called the roadmap) as follows. Any two vertices x_1 and x_2 are connected by an edge, if the path `Steer`(x_1, x_2) that connects these two samples lies within the obstacle-free space⁴. It is known that this algorithm

⁴In practice, some variations of the PRM algorithm attempt to connect every vertex to the vertices that lie within a fixed distance. However, this variation does not change the theoretical properties such as probabilistic completeness or the asymptotic computational complexity.

is probabilistically complete and, under certain assumptions, the probability of failure, if a solution exists, decays to zero exponentially. However, notice that, from an asymptotic computational complexity point of view, creating this graph requires $O(n^2)$ calls to the `Steer` and `ObstacleFree` procedures. On the contrary, our results imply that in a modified version of this algorithm, if each vertex is attempted connection to vertices within a ball of volume $\gamma_L \frac{\log n}{n}$, then, theoretically speaking, probabilistic completeness can be achieved, while incurring $O(n \log n)$ expected computational cost. Pursuing this direction is outside the scope of this work, and is left to future work.

V. A TREE VERSION OF THE RRG ALGORITHM

Maintaining a tree structure rather than a graph may be advantageous in some applications, due to, for instance, relatively easy extensions to motion planning problems with differential constraints, or to cope with modeling errors. The RRG algorithm can also be slightly modified to maintain a tree structure, while preserving the asymptotic optimality properties as well the computational efficiency. In this section a tree version of the RRG algorithm, called RRT*, is introduced and analyzed.

A. The RRT* Algorithm

Given two points $x, x' \in X_{\text{free}}$, recall that $\text{Line}(x, x') : [0, s] \rightarrow X_{\text{free}}$ denotes the path defined by $\sigma(\tau) = \tau x + (s - \tau)x'$ for all $\tau \in [0, s]$, where $s = \|x' - x\|$. Given a tree $G = (V, E)$ and a vertex $v \in V$, let `Parent` be a function that maps v to the unique vertex $v' \in V$ such that $(v', v) \in E$.

The RRT* algorithm differs from the RRT and the RRG algorithms only in the way that it handles the `Extend` procedure. The body of the RRT* algorithm is presented in Algorithm 1 and the `Extend` procedure for the RRT* is given in Algorithm 4. In the description of the RRT* algorithm, the cost of the unique path from x_{init} to a vertex $v \in V$ is denoted by `Cost`(v). Initially, `Cost`(x_{init}) is set to zero.

Similarly to the RRT and RRG, the RRT* algorithm first extends the nearest neighbor towards the sample (Lines 2-3). However, it connects the new vertex, x_{new} , to the vertex that incurs the minimum accumulated cost up until x_{new} and lies within the set X_{near} of vertices returned by the `Near` procedure (Lines 6-13). RRT* also extends the new vertex to the vertices in X_{near} in order to “rewire” the vertices that can be accessed through x_{new} with smaller cost (Lines 14-17).

B. Convergence to a Feasible Solution

For all $i \in \mathbb{N}$, let $\mathcal{V}_i^{\text{RRT}^*}$ and $\mathcal{E}_i^{\text{RRT}^*}$ denote the set of vertices and the set of edges of the graph maintained by the RRT* algorithm, at the end of iteration i . Let $\mathcal{V}_0^{\text{RRT}^*}(\omega) = \{x_{\text{init}}\}$ and $\mathcal{E}_0^{\text{RRT}^*}(\omega) = \emptyset$ for all $\omega \in \Omega$.

The following lemma is the equivalent of Lemma 3.

Lemma 20 *For all $i \in \mathbb{N}$ and all $\omega \in \Omega$, $\mathcal{V}_i^{\text{RRT}^*}(\omega) = \mathcal{V}_i^{\text{RRG}}(\omega)$, and $\mathcal{E}_i^{\text{RRT}^*}(\omega) \subseteq \mathcal{E}_i^{\text{RRG}}(\omega)$.*

Algorithm 4: Extend_{RRT*}

```

1  $V' \leftarrow V; E' \leftarrow E;$ 
2  $x_{\text{nearest}} \leftarrow \text{Nearest}(G, x);$ 
3  $x_{\text{new}} \leftarrow \text{Steer}(x_{\text{nearest}}, x);$ 
4 if ObstacleFree( $x_{\text{nearest}}, x_{\text{new}}$ ) then
5    $V' \leftarrow V' \cup \{x_{\text{new}}\};$ 
6    $x_{\min} \leftarrow x_{\text{nearest}};$ 
7    $X_{\text{near}} \leftarrow \text{Near}(G, x_{\text{new}}, |V|);$ 
8   for all  $x_{\text{near}} \in X_{\text{near}}$  do
9     if ObstacleFree( $x_{\text{near}}, x_{\text{new}}$ ) then
10       $c' \leftarrow \text{Cost}(x_{\text{near}}) + c(\text{Line}(x_{\text{near}}, x_{\text{new}}));$ 
11      if  $c' < \text{Cost}(x_{\text{new}})$  then
12         $x_{\min} \leftarrow x_{\text{near}};$ 
13    $E' \leftarrow E' \cup \{(x_{\min}, x_{\text{new}})\};$ 
14   for all  $x_{\text{near}} \in X_{\text{near}} \setminus \{x_{\min}\}$  do
15     if ObstacleFree( $x_{\text{new}}, x_{\text{near}}$ ) and
16        $\text{Cost}(x_{\text{near}}) >$ 
17        $\text{Cost}(x_{\text{new}}) + c(\text{Line}(x_{\text{new}}, x_{\text{near}}))$  then
18        $x_{\text{parent}} \leftarrow \text{Parent}(x_{\text{near}});$ 
19        $E' \leftarrow E' \setminus \{(x_{\text{parent}}, x_{\text{near}})\};$ 
20        $E' \leftarrow E' \cup \{(x_{\text{new}}, x_{\text{near}})\};$ 
21 return  $G' = (V', E')$ 

```

From Lemma 20 and Theorem 6, the following theorem, which asserts the probabilistic completeness and the exponential decay of failure probability of the RRT* algorithm, is immediate.

Theorem 21 *If there exists a feasible solution to Problem 1, then $\lim_{i \rightarrow \infty} \mathbb{P}(\{\mathcal{V}_i^{\text{RRT}^*} \cap X_{\text{goal}} \neq \emptyset\}) = 1$. Moreover, if an attraction sequence \mathcal{A} of length k exists, then $\mathbb{P}(\{\mathcal{V}_i^{\text{RRT}^*} \cap X_{\text{goal}} = \emptyset\}) \leq e^{-\frac{1}{2}(i p_k - 2^k)}$.*

C. Asymptotic Optimality

Let $\mathcal{Y}_i^{\text{RRT}^*}$ be a random variable that denotes the cost of a minimum cost path in the tree maintained by the RRT* algorithm, at the end of iteration i . The following theorem ensures the asymptotic optimality of the RRT* algorithm.

Theorem 22 *Let Assumptions 12, 13, and 14 hold. Then, the cost of the minimum cost path in the RRT* converges to c^* almost surely, i.e., $\mathbb{P}(\{\lim_{i \rightarrow \infty} \mathcal{Y}_i^{\text{RRT}^*} = c^*\}) = 1$.*

D. Computational Complexity

Let $\mathcal{M}_i^{\text{RRT}^*}$ be the number of steps performed by the RRT* algorithm in iteration i . The following theorem follows from Lemma 20 and Theorem 19.

Theorem 23 *Under the assumptions of Theorem 19, there exists a constant $\phi \in \mathbb{R}_{>0}$ such that*

$$\limsup_{i \rightarrow \infty} \mathbb{E} \left[\frac{\mathcal{M}_i^{\text{RRT}^*}}{\mathcal{M}_i^{\text{RRT}}} \right] \leq \phi.$$

VI. SIMULATIONS

This section is devoted to an experimental study of the algorithms. Three different problem instances are considered and the RRT and RRT* algorithms are compared with respect to their running time and cost of the solution achieved. Both algorithms were implemented in C and run on a computer with 2.66 GHz processor and 4GB RAM running the Linux operating system.

We consider three problem instances. In the first two, the cost function is the Euclidean path length. The first scenario includes no obstacles. Both algorithms are run in a square environment. The trees maintained by the algorithms are shown in Figure 1 at several stages. The figure illustrates that, in this case, the RRT algorithm does not improve the feasible solution to converge to 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 2, which shows that in the limit the RRT algorithm has cost very close to a $\sqrt{2}$ factor the optimal solution (see [68] for a similar result in a deterministic setting), 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 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 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 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 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 6. Notice that the tree either avoids the high cost region or crosses it quickly, and vice-versa for the low-cost region. (Incidentally, this behavior corresponds to the well known Snell-Descartes law for refraction of light, see [69] for a path-planning application.)

To compare the running time, both algorithms were run alongside in an environment with no obstacles up until one million iterations. Figure 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 complexity analysis of Section IV-C, this ratio converges to a constant value. The same figure is produced for the second scenario and provided in Figure 8.

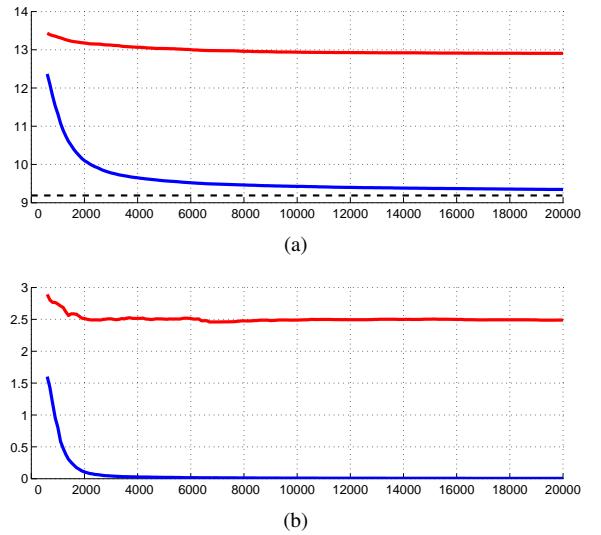


Fig. 2. The cost of the best paths in the RRT (shown in red) and the RRT* (shown in blue) plotted against iterations averaged over 500 trials in (a). The optimal cost is shown in black. The variance of the trials is shown in (b).

VII. CONCLUSION

This paper presented the results of a thorough analysis of the RRT and RRG algorithms for optimal motion planning. It is shown that, as the number of samples increases, the RRT algorithm converges to a sub-optimal solution almost surely. On the other hand, it is proven that the RRG algorithm has the asymptotic optimality property, i.e., almost sure convergence to an optimum solution, which the RRT algorithm lacked. The paper also proposed a novel algorithm called the RRT*, which inherits the asymptotic optimality property of the RRG, while maintaining a tree structure rather than a graph. The RRG and the RRT* were shown to have no significant overhead when compared to the RRT algorithm in terms of asymptotic computational complexity. Experimental evidence that demonstrate the effectiveness of the algorithms proposed and support the theoretical claims were also provided.

The results reported in this paper can be extended in a number of directions, and applied to other sampling-based algorithms other than RRT. First of all, the proposed approach, building on the theory of random graphs to adjust the length

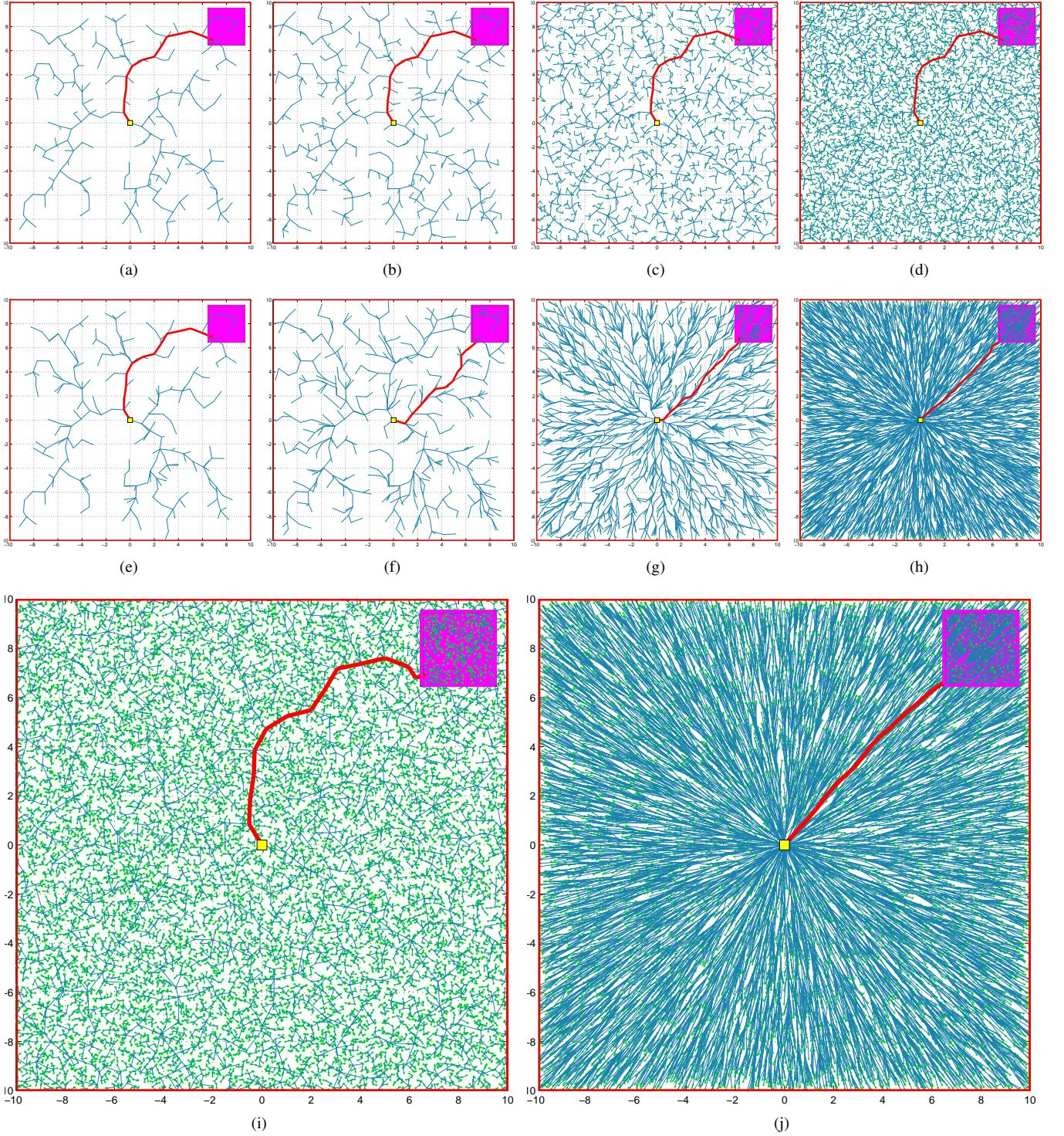


Fig. 1. A Comparison of the RRT* and RRT algorithms on a simulation example with no obstacles. Both algorithms were run with the same sample sequence. Consequently, in this case, the vertices of the trees at a given iteration number are the same for both of the algorithms; only the edges differ. The edges formed by the RRT* algorithm are shown in (a)-(d) and (j), whereas those formed by the RRT algorithm are shown in (e)-(h) and (i). 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.

of new connections can enhance the computational efficiency of PRM-based algorithms. Second, the algorithms and the analysis should be modified to address motion planning problems in the presence of differential constraints, also known

as kino-dynamic planning problems. A third direction is the optimal planning problem in the presence of temporal/logic constraints on the trajectories, e.g., expressed using formal specification languages such as Linear Temporal Logic, or

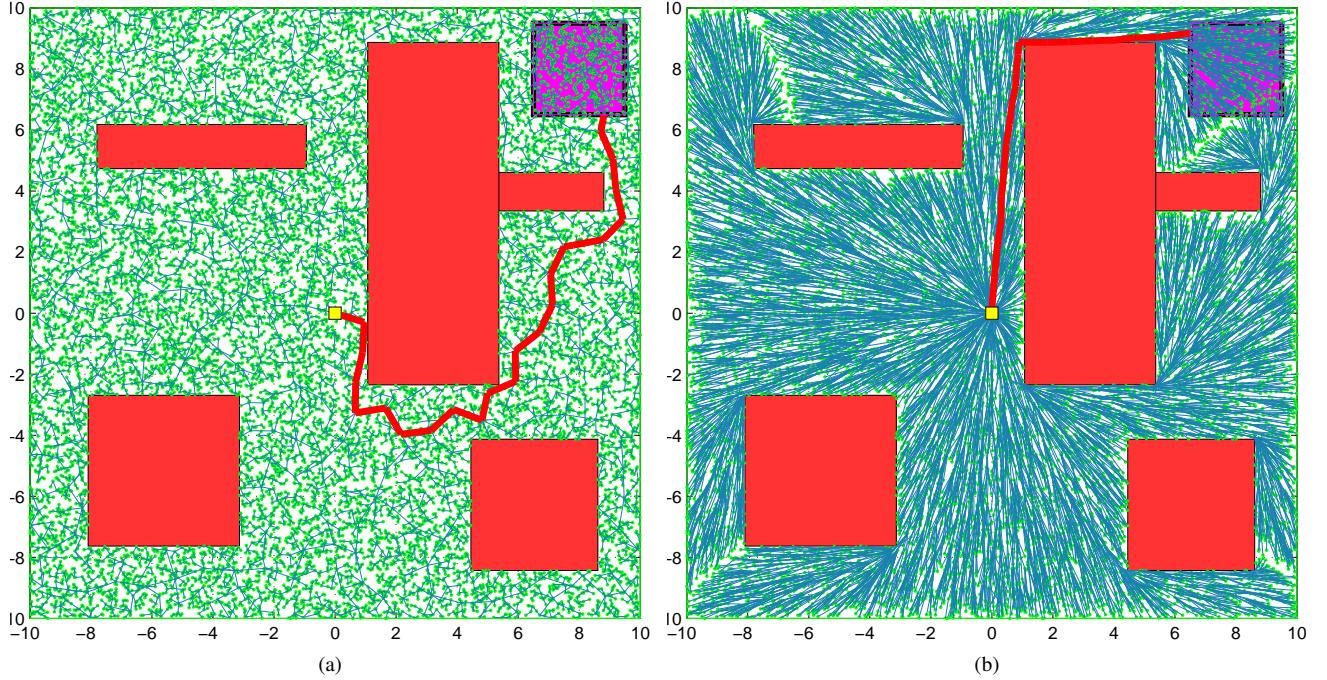


Fig. 3. A Comparison of the RRT (shown in (a)) and RRT* (shown in (b)) algorithms on a simulation example with obstacles. Both algorithms were run with the same sample sequence for 20,000 samples. The cost of best path in the RRT and the RRG were 21.02 and 14.51, respectively.

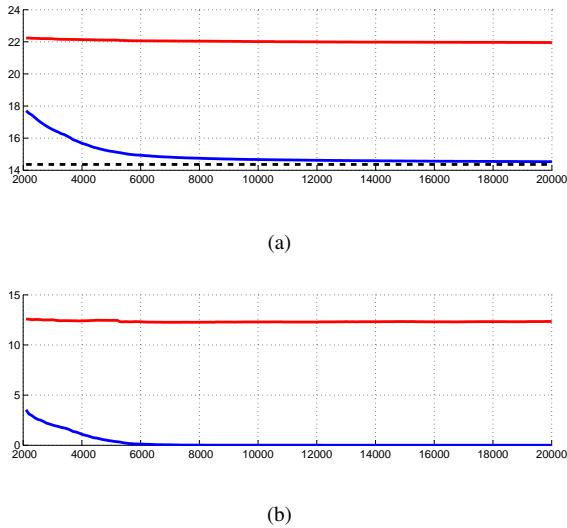


Fig. 5. An environment cluttered with obstacles is considered. The cost of the best paths in the RRT (shown in red) and the RRT* (shown in blue) plotted against iterations averaged over 500 trials in (a). The optimal cost is shown in black. The variance of the trials is shown in (b).

the μ -calculus. Such constraints correspond to, e.g., rules of the road constraints for autonomous ground vehicles, mission specifications for autonomous robots, and rules of engagement in military applications. Ultimately, incremental sampling-based algorithms with asymptotic optimality properties may provide the basic elements for the on-line solution of differential games, as those arising when planning in the presence of dynamic obstacles.

Finally, it is noted that the proposed algorithms may have applications outside of the robotic motion planning domain. In fact, the class of incremental sampling algorithm described in this paper can be readily extended to deal with problems described by partial differential equations, such as the eikonal equation and the Hamilton-Jacobi-Bellman equation.

ACKNOWLEDGMENTS

The authors are grateful to Professors M.S. Branicky and G.J. Gordon for their insightful comments on a draft version of this paper. This research was supported in part by the Michigan/AFRL Collaborative Center on Control Sciences, AFOSR grant no. FA 8650-07-2-3744. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the supporting organizations.

REFERENCES

- [1] J. Latombe. Motion planning: A journey of robots, molecules, digital actors, and other artifacts. *International Journal of Robotics Research*, 18(11):1119–1128, 1999.
- [2] A. Bhatia and E. Frazzoli. Incremental search methods for reachability analysis of continuous and hybrid systems. In R. Alur and G.J. Pappas, editors, *Hybrid Systems: Computation and Control*, number 2993 in Lecture Notes in Computer Science, pages 142–156. Springer-Verlag, Philadelphia, PA, March 2004.
- [3] M. S. Branicky, M. M. Curtis, J. Levine, and S. Morgan. Sampling-based planning, control, and verification of hybrid systems. *IEEE Proc. Control Theory and Applications*, 153(5):575–590, Sept. 2006.
- [4] J. Cortes, L. Jailet, and T. Simeon. Molecular disassembly with RRT-like algorithms. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2007.
- [5] Y. Liu and N.I. Badler. Real-time reach planning for animated characters using hardware acceleration. In *IEEE International Conference on Computer Animation and Social Characters*, pages 86–93, 2003.

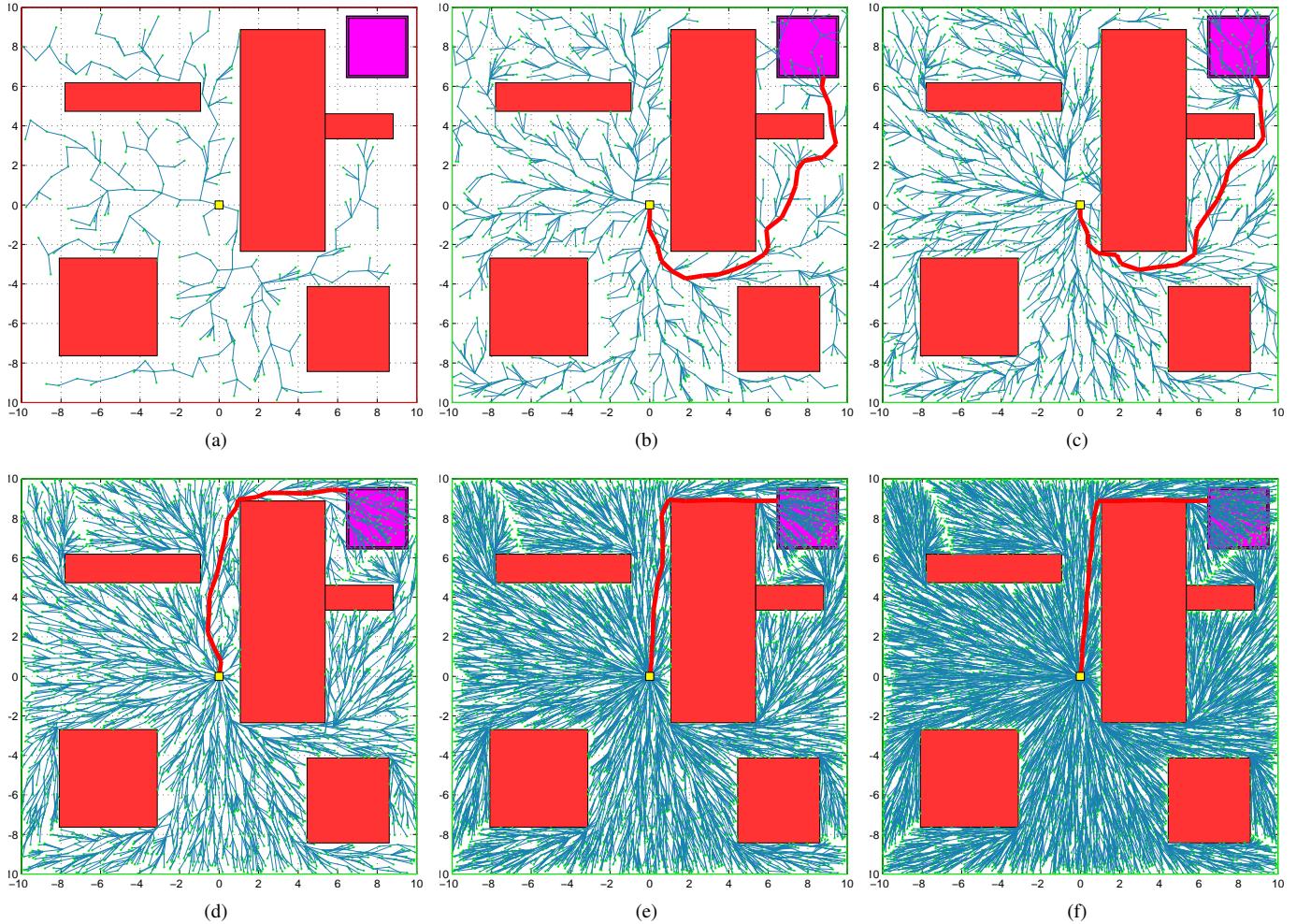


Fig. 4. RRT* algorithm shown after 500 (a), 1,500 (b), 2,500 (c), 5,000 (d), 10,000 (e), 15,000 (f) iterations.

- [6] P.W. Finn and L.E. Kavraki. Computational approaches to drug design. *Algorithmica*, 25:347–371, 1999.
- [7] T. Lozano-Perez and M. A. Wesley. An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM*, 22(10):560–570, 1979.
- [8] 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, 1983.
- [9] J.H. Reif. Complexity of the mover’s problem and generalizations. In *Proceedings of the IEEE Symposium on Foundations of Computer Science*, 1979.
- [10] R. Brooks and T. Lozano-Perez. A subdivision algorithm in configuration space for findpath with rotation. In *International Joint Conference on Artificial Intelligence*, 1983.
- [11] J. Barraquand and J. C. Latombe. Robot motion planning: A distributed representation approach. *International Journal of Robotics Research*, 10(6):628–649, 1993.
- [12] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics Research*, 5(1):90–98, 1986.
- [13] J. Canny. *The Complexity of Robot Motion Planning*. MIT Press, 1988.
- [14] S. S. Ge and Y.J. Cui. Dynamic motion planning for mobile robots using potential field method. *Autonomous Robots*, 13(3):207–222, 2002.
- [15] Y. Koren and J. Borenstein. Potential field methods and their inherent limitations for mobile robot navigation. In *IEEE Conference on Robotics and Automation*, 1991.
- [16] L. Kavraki and J. Latombe. Randomized preprocessing of configuration space for fast path planning. In *IEEE International Conference on Robotics and Automation*, 1994.
- [17] L.E. 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*, 12(4):566–580, 1996.
- [18] S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. *International Journal of Robotics Research*, 20(5):378–400, May 2001.
- [19] S. Prentice and N. Roy. The belief roadmap: Efficient planning in belief space by factoring the covariance. *International Journal of Robotics Research*, 28(11–12):1448–1465, 2009.
- [20] R. Tedrake, I. R. Manchester, M. M. Tobakian, and J. W. Roberts. LQR-trees: Feedback motion planning via sums of squares verification. *International Journal of Robotics Research (to appear)*, 2010.
- [21] B. Luders, S. Karaman, E. Frazzoli, and J. P. How. Bounds on tracking error using closed-loop rapidly-exploring random trees. In *American Control Conference*, 2010.
- [22] D. Berenson, J. Kuffner, and H. Choset. An optimization approach to planning for mobile manipulation. In *IEEE International Conference on Robotics and Automation*, 2008.
- [23] A. Yershova and S. Lavalle. Motion planning in highly constrained spaces. Technical report, University of Illinois at Urbana-Champaign, 2008.
- [24] M. Stilman, J. Schamburek, J. Kuffner, and T. Asfour. Manipulation planning among movable obstacles. In *IEEE International Conference on Robotics and Automation*, 2007.
- [25] E. Koyuncu, N.K. Ure, and G. Inalhan. Integration of path/maneuver planning in complex environments for agile maneuvering UCAsVs. *Journal of Intelligent and Robotic Systems*, 57(1–4):143–170, 2010.
- [26] J. Barraquand, L. Kavraki, J. Latombe, T. Li, R. Motwani, and P. Raghavan. A random sampling scheme for path planning. *International Journal of Robotics Research*, 16:759–774, 1997.
- [27] D. Hsu, J. Latombe, and H. Kurniawati. On the probabilistic foundations

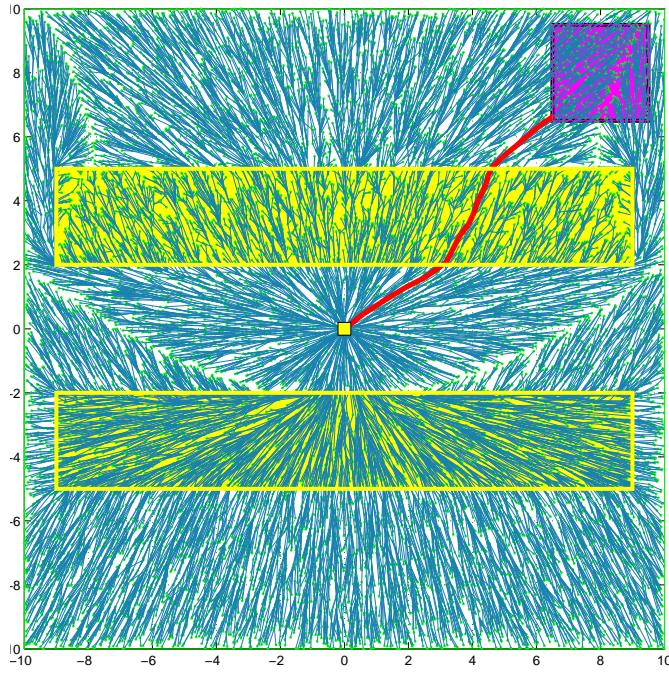


Fig. 6. RRT* algorithm at the end of iteration 20,000 in an environment with no obstacles. The upper yellow region is the high-cost region, whereas the lower yellow region is low-cost.

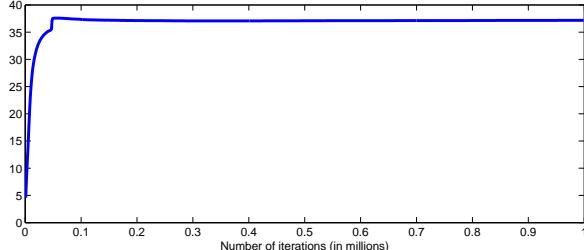


Fig. 7. A comparison of the running time of the RRT* and the RRT algorithms. The ratio of the running time of the RRT* over that of the RRT up until each iteration is plotted versus the number of iterations.

- of probabilistic roadmap planning. *International Journal of Robotics Research*, 25(7), 2006.
- [28] L. E. Kavraki, M. N. Kolountzakis, and J. Latombe. Analysis of probabilistic roadmaps for path planning. *IEEE Transactions on Robotics and Automation*, 14(1):166–171, 1998.
 - [29] J.J. Kuffner and S.M. LaValle. RRT-connect: An efficient approach to single-quert path planning. In *Proceedings of the IEEE International Conference on Robotics and Automation*, 2000.
 - [30] S. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.
 - [31] S.R. Lindemann and S.M. LaValle. Current issues in sampling-based motion planning. In P. Dario and R. Chatila, editors, *Eleventh International Symposium on Robotics Research*, pages 36–54. Springer, 2005.
 - [32] M. S. Branicky, S. M. LaValle, K. Olson, and L. Yang. Quasi-randomized path planning. In *IEEE Conference on Robotics and Automation*, 2001.
 - [33] A. L. Ladd and L. Kavraki. Measure theoretic analysis of probabilistic path planning. *IEEE Transactions on Robotics and Automation*, 20(2):229–242, 2004.
 - [34] D. Hsu, R. Kindel, J. Latombe, and S. Rock. Randomized kinodynamic motion planning with moving obstacles. *International Journal of Robotics Research*, 21(3):233–255, 2002.
 - [35] E. Frazzoli, M. Dahleh, and E. Feron. Real-time motion planning for agile autonomous vehicles. *Journal of Guidance, Control, and*

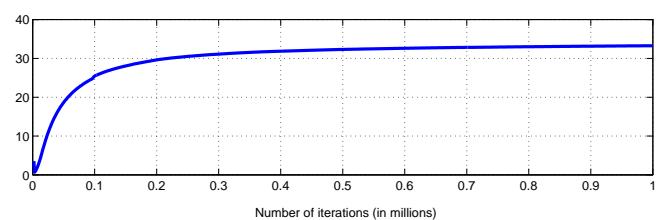


Fig. 8. A comparison of the running time of the RRT* and the RRT algorithms in an environment with obstacles. The ratio of the running time of the RRT* over that of the RRT up until each iteration is plotted versus the number of iterations.

- Dynamics*, 25(1):116–129, 2002.
- [36] M. S. Branicky, M. M. Curtis, J. A. Levine, and S. B. Morgan. RRTs for nonlinear, discrete , and hybrid planning and control. In *IEEE Conference on Decision and Control*, 2003.
 - [37] M. Zucker, J. Kuffner, and M. Branicky. Multiple RRTs for rapid replanning in dynamic environments. In *IEEE Conference on Robotics and Automation*, 2007.
 - [38] J. Bruce and M.M. Veloso. *Real-Time Randomized Path Planning for Robot Navigation*, volume 2752 of *Lecture Notes in Computer Science*, chapter RoboCup 2002: Robot Soccer World Cup VI, pages 288–295. Springer, 2003.
 - [39] 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.
 - [40] S. Teller, M. R. Walter, M. Antone, A. Correa, R. Davis, L. Fletcher, E. Frazzoli, J. Glass, J.P. How, A. S. Huang, J. Jeon, S. Karaman, B. Luders, N. Roy, and T. Sainath. A voice-commandable robotic forklift working alongside humans in minimally-prepared outdoor environments. In *IEEE International Conference on Robotics and Automation*, 2010.
 - [41] A. Shkolnik, M. Levashev, I. R. Manchester, and R. Tedrake. Bounding on rough terrain with the LittleDog robot. Under review.
 - [42] 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.
 - [43] S. Karaman and E. Frazzoli. Sampling-based motion planning with deterministic μ -calculus specifications. In *IEEE Conference on Decision and Control (CDC)*, 2009.
 - [44] E.M. Clarke, O. Grumberg, and D.A. Peled. *Model Checking*. Springer, 1999.
 - [45] C. Urmson and R. Simmons. Approaches for heuristically biasing RRT growth. In *Proceedings of the IEEE/RSJ International Conference on Robotics and Systems (IROS)*, 2003.
 - [46] D. Ferguson and A. Stentz. Anytime RRTs. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2006.
 - [47] N. A. Wedge and M.S. Branicky. On heavy-tailed runtimes and restarts in rapidly-exploring random trees. In *Twenty-third AAAI Conference on Artificial Intelligence*, 2008.
 - [48] M. Likhachev, G. Gordon, and S. Thrun. Anytime A* with provable bounds on sub-optimality. In *Advances in Neural Information Processing Systems*, 2004.
 - [49] M. Likhachev, D. Ferguson, G. Gordon, A. Stentz, and S. Thrun. Anytime search in dynamic graphs. *Artificial intelligence Journal*, 172(14):1613–1643, 2008.
 - [50] D. Stentz. The focussed D* algorithm for real-time replanning. In *International Joint Conference on Artificial Intelligence*, 1995.
 - [51] M. Likhachev and D. Ferguson. Planning long dynamically-feasible maneuvers for autonomous vehicles. *International Journal of Robotics Research*, 28(8):933–945, 2009.
 - [52] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel. *Experimental Robotics*, chapter Path Planning for Autonomous Driving in Unknown Environments, pages 55–64. Springer, 2009.
 - [53] M. Penrose. *Random Geometric Graphs*. Oxford University Press, 2003.
 - [54] J. Dall and M. Christensen. Random geometric graphs. *Physical Review E*, 66(1):016121, Jul 2002.
 - [55] S.M. LaValle, M.S. Branicky, and S.R. Lindemann. On the relationship between classical grid search and probabilistic roadmaps. *International Journal of Robotics Research*, 23(7–8):673–692, 2004.

- [56] H. Samet. *Applications of Spatial Data Structures: Computer Graphics, Image Processing and Gis*. Addison-Wesley, 1989.
- [57] H. Samet. *Design and Analysis of Spatial Data Structures*. Addison-Wesley, 1989.
- [58] A. Amentov and S. M. LaValle. Efficient nearest neighbor searching for motion planning. In *IEEE International Conference on Robotics and Automation*, 2002.
- [59] T. H. Cohen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. MIT Press, 1990.
- [60] H. Edelsbrunner. *Algorithms in Computational Geometry*. Springer-Verlag, 1987.
- [61] K. L. Clarkson. A randomized algorithm for closest-point queries. *SIAM Journal of Computation*, 17:830–847, 1988.
- [62] S. Arya, D. M. Mount, R. Silverman, and A. Y. Wu. An optimal algorithm for approximate nearest neighbor search in fixed dimensions. *Journal of the ACM*, 45(6):891–923, November 1999.
- [63] E. Plaku and L. E. Kavraki. Quantitative analysis of nearest-neighbors search in high-dimensional sampling-based motion planning. In *Workshop on Algorithmic Foundations of Robotics (WAFR)*, 2008.
- [64] D. T. Lee and C. K. Wong. Worst-case analysis for region and partial region searches in multidimensional binary search trees and quad trees. *Acta Informatica*, 9:23–29, 1977.
- [65] P. Chanzy, L. Devroye, and C. Zamora-Cura. Analysis of range search for random k-d trees. *Acta Informatica*, 37:355–383, 2001.
- [66] S. Arya and D. M. Mount. Approximate range searching. *Computational Geometry: Theory and Applications*, 17:135–163, 2000.
- [67] S. Arya, T. Malamatos, and D. M. Mount. Space-time tradeoffs for approximate spherical range counting. In *Symposium on Discrete Algorithms*, 2005.
- [68] S. LaValle and J. Kuffner. Space filling trees. Technical Report CMU-RI-TR-09-47, Carnegie Mellon University, The Robotics Institute, 2009.
- [69] N.C. Rowe and R.S. Alexander. Finding optimal-path maps for path planning across weighted regions. *The International Journal of Robotics Research*, 19:83–95, 2000.
- [70] H. A. David and H. N. Nagaraja. *Order Statistics*. Wiley, 2003.
- [71] G. Grimmett and D. Stirzaker. *Probability and Random Processes*. Oxford University Press, Third edition, 2001.
- [72] M. Mitzenmacher and E. Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005.
- [73] S. Muthukrishnan and G. Pandurangan. The bin-covering technique for thresholding random geometric graph properties. In *Proceedings of the sixteenth annual ACM-SIAM symposium on discrete algorithms*, 2005.

APPENDIX

Proof of Theorem 10

Since the feasibility problem is assumed to admit a solution, the cost of the optimal path, c^* , is a finite real number. First, consider the following technical lemma.

Lemma 24 *The probability that the RRT constructs an optimal path at a finite iteration $i \in \mathbb{N}$ is zero, i.e.,*

$$\mathbb{P}(\cup_{i \in \mathbb{N}} \{\mathcal{Y}_i^{\text{RRT}} = c^*\}) = 0.$$

Proof: Let B_i denote the event the RRT has a path that has cost exactly equal to c^* at the end of iteration i , i.e., $B_i = \{\mathcal{Y}_i^{\text{RRT}} = c^*\}$. Let B denote the event that the RRT finds a path that costs exactly c^* at some finite iteration i . Then, B can be written as $B = \cup_{i \in \mathbb{N}} B_i$. Notice that $B_i \subseteq B_{i+1}$; thus, we have that $\lim_{i \rightarrow \infty} \mathbb{P}(B_i) = \mathbb{P}(B)$, by monotonicity of measures. Notice also that by Assumptions 7 and 8, $\mathbb{P}(B_i) = 0$ for all $i \in \mathbb{N}$, since the probability that the set $\bigcup_{j=1}^i \{\text{Sample}(j)\}$ of points contains a point from a zero-measure set is zero. Hence, $\mathbb{P}(B) = 0$. ■

Let C_i denote the event that x_{init} is chosen to be extended by the Nearest procedure at iteration i . The following lemma

directly implies a necessary condition for asymptotic optimality: with probability one, the initial state, x_{init} , must be chosen for extension infinitely often.

Lemma 25 *The following inequality holds:*

$$\mathbb{P}\left(\left\{\lim_{i \rightarrow \infty} \mathcal{Y}_i^{\text{RRT}} = c^*\right\}\right) \leq \mathbb{P}\left(\limsup_{i \rightarrow \infty} C_i\right).$$

Proof: Denote by $\psi(x)$ a minimum-cost path starting from a given state $x \in X_{\text{free}}$ and reaching one of the states in the goal region⁵, and define a sequence $\{c_i\}_{i \in \mathbb{N}}$ of random variables as follows:

$$c_i = \min \left\{ \text{Cost}(\text{Line}(x_{\text{init}}, x) \mid \psi(x)) \mid x \in \mathcal{V}_i^{\text{RRT}}, (x_{\text{init}}, x) \in \mathcal{E}_i^{\text{RRT}} \right\}.$$

Essentially, c_i denotes the minimum cost that is incurred by following the line segment that connects x_{init} to one of its children and following an optimal path afterwards. Notice that c_i is strictly greater than the optimal cost c^* for all $i \in \mathbb{N}$, unless the root node, x_{init} , has a child node that is a part of an optimal path.

Recall, from the proof of Lemma 24, that B denotes the event $\cup_{i \in \mathbb{N}} \{\mathcal{Y}_i^{\text{RRT}} = c^*\}$. Let A_i denote the event that c_i decreases at iteration i , i.e., $A_i = \{c_i < c_{i-1}\}$. First, notice that, conditioning on the event B^c , we have that the convergence of the cost of the best path in the RRT to the optimum cost (i.e., the event $\{\lim_{i \rightarrow \infty} \mathcal{Y}_i^{\text{RRT}} = c^*\}$) implies that c_i must decrease infinitely often, i.e., A_i must hold infinitely often. More precisely, we have that $\{\lim_{i \rightarrow \infty} \mathcal{Y}_i^{\text{RRT}} = c^*\} \cap B^c \subseteq \limsup_{i \rightarrow \infty} A_i$.

Noting that $\mathbb{P}(B^c) = 1$ (by Lemma 24), it follows that

$$\begin{aligned} \mathbb{P}\left(\left\{\lim_{i \rightarrow \infty} \mathcal{Y}_i^{\text{RRT}} = c^*\right\}\right) &= \mathbb{P}\left(\left\{\lim_{i \rightarrow \infty} \mathcal{Y}_i^{\text{RRT}} = c^*\right\} \cap B^c\right) \\ &\leq \mathbb{P}\left(\limsup_{i \rightarrow \infty} A_i\right). \end{aligned}$$

Notice that, by Assumption 9, for A_i to occur infinitely often, C_i must also occur infinitely often, i.e., $\limsup_{i \rightarrow \infty} A_i \subseteq \limsup_{i \rightarrow \infty} C_i$. Thus, $\mathbb{P}(\limsup_{i \rightarrow \infty} A_i) \leq \mathbb{P}(\limsup_{i \rightarrow \infty} C_i)$, which implies the lemma. ■

To prove the theorem, it is shown that RRT fails to satisfy this necessary condition:

Lemma 26 *The probability that the RRT algorithm extends its root node infinitely many times is zero, i.e.,*

$$\mathbb{P}\left(\limsup_{i \rightarrow \infty} C_i\right) = 0.$$

Proof: Let $\{\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_K\}$ be a finite set of cones, such that (i) each cone is centered at x_{init} and has aperture at most $\pi/3$, and (ii) collectively the cones cover the ball of radius η around x_{init} . Defined for all $k \in \{1, 2, \dots, K\}$ and all $j \in \mathbb{N}$, let $D_{k,j}$ be the event that a node from inside cone \mathcal{C}_k is connected to x_{init} for the j th time. Clearly, for

⁵If there is no such path, then let Λ be a symbol for “unfeasible path,” and set $\psi(x) = \Lambda$, with $\text{Cost}(\Lambda) = +\infty$.

C_i to occur infinitely often, $D_{k,j}$ must occur for infinitely many j , for at least one k . More formally, $\limsup_{i \rightarrow \infty} C_i \subseteq \cup_{k=1}^K \limsup_{j \rightarrow \infty} D_{k,j}$, and that

$$\begin{aligned}\mathbb{P}(\limsup_{i \rightarrow \infty} C_i) &\leq \mathbb{P}(\cup_{k=1}^K \limsup_{j \rightarrow \infty} D_{k,j}) \\ &\leq \sum_{k=1}^K \mathbb{P}(\limsup_{j \rightarrow \infty} D_{k,j}).\end{aligned}$$

The next step is to show that $\mathbb{P}(\limsup_{j \rightarrow \infty} D_{k,j}) = 0$ holds for all $k \in \{1, 2, \dots, K\}$, which implies the lemma. Let k be any index from $\{1, 2, \dots, K\}$. Let $V_{k,j}$ denote the set of vertices inside cone \mathcal{C}_k right after the connection of j th node to x_{init} from inside the cone \mathcal{C}_k . Let $r_{\min,k,j}$ be the distance of the node that is inside $V_{k,j}$ and has minimum distance to x_{init} among all the vertices in $V_{k,j}$, i.e.,

$$r_{\min,k,j} = \min_{v \in V_{k,j}} \|v - x_{\text{init}}\|.$$

This distance is a random variable for fixed values of k and j . Let $f_{r_{\min,k,j}}$ denote its probability density function. Also, given two points $x, y \in X$, let $\mathcal{D}_{x,y}$ denote the set of all points in X that are closer to y than they are to x , i.e.,

$$\mathcal{D}_{x,y} = \{z \in X \mid \|x - z\| > \|y - z\|\}.$$

For any $j \in \mathbb{N}_{>0}$, conditioning on $r_{\min,k,j} = z$, the probability that a node from \mathcal{C}_k is connected to x_{init} for the j th time is upper bounded by the measure of the region $\mathcal{C}_k \setminus \mathcal{D}_{x_{\text{init}},y}$, where y is any point with distance z to x_{init} (see Figure 9). Hence, $\mathbb{P}(D_{k,j} \mid r_{\min,k,j} = z) \leq \alpha \mu(\mathcal{C}_k \setminus \mathcal{D}_{x_{\text{init}},y})$ for some constant $\alpha \in \mathbb{R}_{>0}$ by Assumption 8. Furthermore, since \mathcal{C}_k has aperture at most $\pi/3$, the region $\mathcal{C}_k \setminus \mathcal{D}_{x_{\text{init}},y}$ is included within the ball of radius z centered at x_{init} (also illustrated in Figure 9). Hence,

$$\begin{aligned}\mathbb{P}(D_{k,j}) &= \int_{z=0}^{\infty} \mathbb{P}(D_{k,j} \mid r_{\min,k,j} = z) f_{r_{\min,k,j}}(z) dz \\ &\leq \int_{z=0}^{\infty} \alpha \mu(\mathcal{C}_k \setminus \mathcal{D}_{x_{\text{init}},y}) f_{r_{\min,k,j}}(z) dz \\ &\leq \int_{z=0}^{\infty} \alpha \mu(\mathcal{B}_{x_{\text{init}},z}) f_{r_{\min,k,j}}(z) dz.\end{aligned}$$

Note that $\mu(\mathcal{B}_{x_{\text{init}},z}) \leq \zeta_d z^d$, where d is, recall, the dimensionality of the smallest Euclidean space containing X , and ζ_d is the volume of the unit ball in this space. Hence,

$$\mathbb{P}(D_{k,j}) \leq \alpha \zeta_d \int_{z=0}^{\infty} z^d f_{r_{\min,k,j}}(z) dz = \alpha \zeta_d \mathbb{E}[(r_{\min,k,j})^d],$$

where the last equality follows merely by the definition of expectation. By the order statistics of the minimum distance, we have that $\mathbb{E}[(r_{\min,k,j})^d]$ evaluates to $\frac{\beta}{j^d}$ for some constant $\beta \in \mathbb{R}_{>0}$, under Assumption 8 (see, e.g., [70]). Hence, we have that, for all k , $\sum_{j=1}^{\infty} \mathbb{P}(D_{k,j}) < \infty$, since $d \geq 2$. Then, by the Borel-Cantelli Lemma [71], one can conclude that the probability that $D_{k,j}$ occurs for infinitely many j is zero, i.e., $\mathbb{P}(\limsup_{j \rightarrow \infty} D_{k,j}) = 0$, which implies the claim. ■

The theorem is immediate from Lemmas 25 and 26.

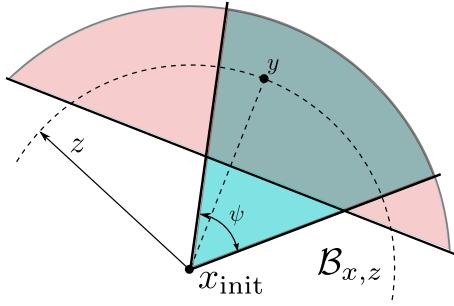


Fig. 9. Illustration of intersection of \mathcal{C}_k (shaded in blue) of aperture ψ at least $\pi/3$ with the region $\mathcal{D}_{x_{\text{init}},y}$. The ball $\mathcal{B}_{x_{\text{init}},z}$ of radius z that includes $\mathcal{C}_k \setminus \mathcal{D}_{x_{\text{init}},y}$ is also shown with dotted lines in the figure.

Proof of Theorem 15

In order to prove that $\mathcal{Y}_i^{\text{RRG}}$ converges to c^* almost surely, it will be shown that $\sum_{i=0}^{\infty} \mathbb{P}(\mathcal{Y}_i^{\text{RRG}} > c^* + \varepsilon)$ is finite for all $\varepsilon > 0$ (see, for instance, [71] for justification of this implication).

Recall that the volume of the unit ball in d dimensions is denoted by ζ_d . Recall also that r_i denotes the radius of the ball with volume $\gamma \frac{\log i}{i}$ used in the Near procedure and that we have $r_i = \left(\frac{\gamma \log i}{\zeta_d i}\right)^{1/d}$. Let $\sigma^* : [0, s^*] \rightarrow \text{cl}(X_{\text{free}})$ be an optimal path with length s^* . Recall that i denotes the iteration number (see Algorithm 1). The proof is first outlined in the following section, and then described in detail.

Outline of the proof: First, construct a sequence $\{X_i\}_{i \in \mathbb{N}}$ of subsets of X_{free} such that (i) the sequence is monotonically non-decreasing in the partial subset ordering, i.e., $X_i \subseteq X_{i+1}$ for all $i \in \mathbb{N}$, (ii) for all large enough $i \in \mathbb{N}$, any point in X_i is at least a certain fraction of r_i away from obstacles, i.e., $\|x - y\| \geq \lambda_i$ for all $x \in X_i$ and all $y \in X_{\text{obs}}$, where $\lambda_i = \alpha r_i$ for some constant α (see Figures 10(a) and 10(b)), in particular, the set X_i is included in X_{free} , and (iii) the sequence converges to X_{free} in the sense that $\cup_{i \in \mathbb{N}} X_i = X_{\text{free}}$.

Second, construct a sequence $\{\sigma_i\}_{i \in \mathbb{N}}$ of paths such that (i) for all $i \in \mathbb{N}$, the path σ_i lies completely inside X_i , and (ii) the sequence of paths converges to the optimal path σ^* , i.e., $\lim_{i \rightarrow \infty} \|\sigma_i - \sigma^*\| = 0$. This can be done by dividing σ^* into segments and constructing an approximation to any segment that lies outside X_i (see Figure 10(c)). Note that, since for all large enough i , any point in X_i is at least a certain fraction of r_i away from the obstacles, in particular, any point on the path σ_i is at least a certain fraction of r_i away from the obstacles, for all such $i \in \mathbb{N}$.

Third, for any $i \in \mathbb{N}$, construct a set B_i of overlapping balls, each with radius q_i and centered at a point on the path σ_i , such that the balls in B_i collectively “cover” the path σ_i (see Figure 10(d)). Moreover, the radius q_i is chosen in such a way that for all large enough i , we have that q_i is less than r_i . Hence, for all such large enough i , all the balls in B_i completely lie inside the obstacle-free region.

Finally, compute the probability of the event that at least one ball in B_i contains no node of the RRG in iteration i ; this event is denoted by A_i . Subsequently, one can show that A_i can not occur infinitely often as i approaches infinity, which implies that its complement, A_i^c , must occur infinitely often

(with probability one). That is, one can conclude that the RRG will include a node in each ball in the set B_i of balls for infinitely many i 's, with probability one. Joining the vertices in subsequent balls in B_i , one can generate a path, whose cost is shown to be close to c^* . Moreover, as i approaches infinity, the costs of such paths converge c^* . Appropriately choosing q_i guarantees that the RRG algorithm joins the points in subsequent balls by an edge, i.e., the path is included in the RRG. Hence, the result follows.

The assumptions on the cost function are mostly used to make the convergence arguments possible. The assumptions on the obstacles (i.e., environment), on the other hand, are primarily used for ensuring that the paths in the sequence $\{\sigma_i\}_{i \in \mathbb{N}}$ are collision-free.

The proof technique is based on the bin-covering technique [72], which is widely used for analyzing almost-sure properties of random geometric graphs (see, e.g., [73]). Similar methods were also used for analyzing PRMs (see [33] and the references therein).

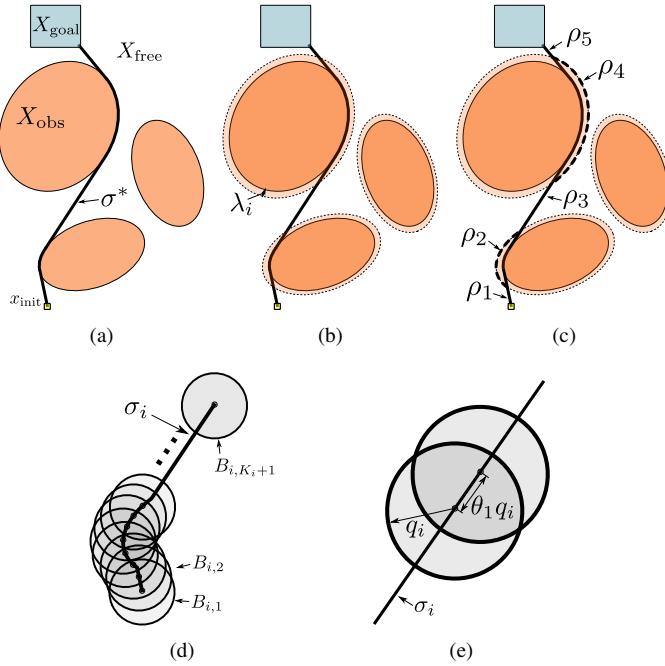


Fig. 10. A region with obstacles is illustrated in Figure 10(a). An optimal path is also shown and denoted as σ^* . In Figure 10(b), the region that lies outside the light orange shaded shaded area (enclosed by the dotted lines) illustrates the set X_i . Notice that the dotted lines, hence all the points in the set X_i , have distance at least λ_i to any point in the obstacle region. In Figure 10(c), the optimal path is divided into parts and the segments $\rho_1, \rho_2, \dots, \rho_5$ are constructed so that ρ_1, ρ_3 , and ρ_5 (solid lines) lie inside X_i , whereas ρ_2 and ρ_4 (dotted lines) lie inside $X_{\text{free}} \setminus X_i$. Note that these path segments together make up the path σ_i , i.e., $\sigma_i = \rho_1 | \rho_2 | \dots | \rho_5$. In Figure 10(d), the tiling of σ_i with overlapping balls in B_i are illustrated. Finally, in Figure 10(e), this tiling is shown in detail.

Construction of the sequence $\{X_i\}_{i \in \mathbb{N}}$: Let $\theta_1 \in (0, 1)$ be a constant. First, define a sequence $\{\lambda_i\}_{i \in \mathbb{N}}$ of real numbers as $\lambda_i = \min \left\{ \delta, \frac{1+\theta_1}{2+\theta_1} r_i \right\}$, where δ is given by Assumption 14. Construct a sequence $\{X_i\}_{i \in \mathbb{N}}$ of subsets of X_{free} as follows (see Figure 10(b)):

$$X_i := X_{\text{free}} \setminus \{x \in X_{\text{free}} \mid \exists x' \in X_{\text{obs}}, \|x - x'\| < \lambda_i\}.$$

Denote the boundary of X_i by ∂X_i . The precise value of the constant θ_1 will become clear shortly.

Construction of the sequence $\{\sigma_i\}_{i \in \mathbb{N}}$: Assume, without loss of any generality, that the beginning and the end of the optimal path σ^* are at least δ away from the obstacles, i.e., for $\tau \in \{0, s^*\}$, it is the case that $\|\sigma^*(\tau) - x\| \geq \delta$ for all $x \in X_{\text{obs}}$.

Construct a sequence $\{\sigma_i\}_{i \in \mathbb{N}}$ of paths, where $\sigma_i : [0, s_i] \rightarrow X_i$, as follows (see Figure 10(c) for an illustration). First, break up the optimal path σ^* into a minimum number of segments such that each segment completely lies either inside X_i or inside $\text{cl}(X_{\text{free}}) \setminus X_i$. For this purpose, define a sequence $\{\tau_0, \tau_1, \dots, \tau_{m+1}\}$ of scalars as follows:

- $\tau_0 = 0$ and $\tau_{m+1} = s^*$,
- for all $j \in \{1, 2, \dots, m\}$, $\tau_j \in \partial X_i$, and
- for all $\{0, 1, \dots, m\}$, either $\sigma^*(\tau') \in X_i$ for all $\tau \in (\tau_j, \tau_{j+1})$ or $\sigma^*(\tau') \notin X_i$ for all $\tau \in (\tau_j, \tau_{j+1})$.

Second, for each segment of σ^* defined by its end points $\sigma^*(\tau_j)$ and $\sigma^*(\tau_{j+1})$, construct a path that lies in X_i and agrees with this segment of σ^* in its endpoints. More precisely, define a set $\{\rho_0, \rho_1, \dots, \rho_m\}$ of paths as follows: (i) if $\sigma^*(\tau') \in X_i$ for $\tau' \in (\tau_j, \tau_{j+1})$, then let ρ_j be the j th segment of σ^* , i.e., $\rho_j(\tau' - \tau_j) = \sigma^*(\tau')$ for all $\tau' \in [\tau_j, \tau_{j+1}]$, (ii) if, on the other hand, $\sigma^*(\tau') \notin X_i$ for all $\tau' \in (\tau_j, \tau_{j+1})$, then let ρ_i be a minimum-cost continuous path in X_i starting from $\sigma^*(\tau_j)$ and ending in $\sigma^*(\tau_{j+1})$. Such a path always exists by the construction of X_i and Assumption 14. Finally, define σ_i as the concatenation of path segments $\rho_0, \rho_1, \dots, \rho_m$, i.e., define $\sigma_i := \rho_0 | \rho_1 | \dots | \rho_m$.

Two important properties of the sequence of paths $\{\sigma_i\}_{i \in \mathbb{N}}$ follow. First, each path σ_i is at least λ_i away from the obstacles. More precisely $\|\sigma_i(\tau) - x\| \geq \lambda_i$ for all $i \in \mathbb{N}$, $\tau \in [0, s_i]$, and $x \in X_{\text{obs}}$. This claim follows from Assumption 14 and the fact that $\lambda_i \leq \delta$ for all i . The second property, regarding the costs of the paths in $\{\sigma_i\}_{i \in \mathbb{N}}$, is stated in the following lemma:

Lemma 27 *Under Assumptions 12 and 13, $\lim_{i \rightarrow \infty} c(\sigma_i) = c(\sigma^*)$ holds.*

Proof: First, for all $i \in \mathbb{N}$, σ_i is continuous by construction. Second, note that $X_i \subseteq X_{i+1}$ for all $i \in \mathbb{N}$, and that $\cup_{i \in \mathbb{N}} X_i = X_{\text{free}}$. Let m_i be the number of segments of σ^* that lie in $X_{\text{free}} \setminus X_i$. By Assumption 13, the cost of each such segment is bounded by $\kappa \lambda_i$. Then, by Assumption 12, $|c(\sigma_i) - c(\sigma^*)| \leq m_i \kappa \lambda_i$. Finally, since m_i is a monotonically non-increasing function of i , and λ_i converges to zero as i approaches infinity, the lemma follows. ■

Construction of the sequence $\{B_i\}_{i \in \mathbb{N}}$ of sets of balls: Next, for each $i \in \mathbb{N}$, construct a set B_i of equal-radius overlapping balls that collectively “cover” the path σ_i .

Recall the constant $\theta_1 \in (0, 1)$, introduced above. The amount of overlap between two consecutive balls will be parametrized by this constant θ_1 . Denote the radius of each ball in B_i by q_i , which we define as $q_i := \frac{\lambda_i}{(1+\theta_1)}$ for reasons to become clear shortly.

Construct the set $B_i = \{B_{i,1}, B_{i,2}, \dots, B_{i,K_i+1}\}$ of balls recursively as follows (see Figures 10(d) and 10(e)):

- Let $B_{i,1}$ be centered at $\sigma_i(0)$.
- For all $k > 1$, let $B_{i,k}$ centered at $\sigma_i(\tau)$, where τ is such that the centers of $B_{i,k}$ and $B_{i,k-1}$ are exactly $\theta_1 q_i$ apart from each other.
- Let K_i be the number of balls that can possibly be generated in this manner and, finally, let B_{i,K_i+1} be the ball centered at $\sigma_i(s_i)$.

Now, consider paths that can be constructed by connecting points from balls in B_i in a certain way. More precisely, for all $k \in \{1, 2, \dots, K_i + 1\}$, let x_k be a point from the ball $B_{i,k}$. Consider a path σ'_i that is constructed by joining the line segments connecting two consecutive points in $\{x_1, x_2, \dots, x_{K_i+1}\}$, i.e., $\sigma'_i = \text{Line}(x_1, x_2) | \text{Line}(x_2, x_3) | \dots | \text{Line}(x_{K_i}, x_{K_i+1})$. With an abuse of notation, let Σ'_i denote the set of all such paths.

First, notice that, by the choice of the radius q_i of the balls in B_i , any path $\sigma'_i \in \Sigma'_i$ is collision-free.

Lemma 28 *Let Assumption 14 hold. For any path $\sigma'_i \in \Sigma'_i$, where $\sigma'_i : [0, s'_i] \rightarrow X$, $\sigma'_i(\tau) \in X_{\text{free}}$ for all $\tau \in [0, s'_i]$.*

Proof: Let $\{x_1, x_2, \dots, x_{K_i+1}\}$ be the set of points that forms $\sigma'_i = \text{Line}(x_1, x_2) | \dots | \text{Line}(x_{K_i}, x_{K_i+1})$. It will be shown that each segment $\text{Line}(x_k, x_{k+1})$ is obstacle-free, i.e., lies in X_{free} , which implies the lemma. Let y_k denote the center of the ball $B_{i,k}$. Note that x_k and y_k have distance at most q_i . Similarly, noting that x_{k+1} has distance at most q_i to y_{k+1} and that y_k and y_{k+1} are at most $\theta_1 q_i$ apart from each other, one can conclude that x_{k+1} has distance at most $(1+\theta_1)q_i$ to y_k . Indeed, y_k has distance at most $(1+\theta_1)q_i = \lambda_i$ to any point in the convex combination $[x_k, x_{k+1}]$. Finally, recall that, as a consequence of Assumption 14, any point with distance at most λ_i from y_k lies in X_{free} , which completes the proof. ■

Second, notice that the cost of σ'_i is “close” to that of σ_i . Indeed, the following lemma holds.

Lemma 29 *Let Assumptions 12 and 13 hold. Let $\{\sigma'_i\}_{i \in \mathbb{N}}$ be any sequence of paths such that $\sigma'_i \in \Sigma'_i$ for all $i \in \mathbb{N}$, then $\lim_{i \rightarrow \infty} |c(\sigma'_i) - c(\sigma_i)| = 0$.*

Proof: The distance between a path $\sigma'_i \in \Sigma'_i$ and the path σ_i approaches zero as i approaches infinity. Moreover, by Assumptions 12 and 13, $|c(\sigma'_i) - c(\sigma_i)|$ converges to zero. Hence, the result follows from Lemma 27. ■

Let us define $\varepsilon_i = \sup_{\sigma'_i \in \Sigma'_i} |c(\sigma'_i) - c^*|$. Hence, Lemma 29 establishes that $\lim_{i \rightarrow \infty} \varepsilon_i = 0$.

Third, according to our choice of q_i , any path in Σ'_i is “constructable” by the RRG in the following sense.

Lemma 30 *For all $i \in \mathbb{N}$ and for all $k \in \{1, 2, \dots, K_i\}$, $\|x_{k+1} - x_k\| \leq r_i$, where $x_k \in B_{i,k}$.*

Proof: Recall that the radius of each ball in B_i is $q_i = \frac{\lambda_i}{(1+\theta_1)}$. Given any two points x_k and x_{k+1} , both x_k and x_{k+1} have distances at most q_i to the centers of the balls $B_{i,k}$ and $B_{i,k+1}$, respectively. Note also that the centers of these balls have distance at most $\theta_1 q_i$ to each other. Using

the triangle inequality together with the definitions of λ_i and q_i , one obtains $\|x_{k+1} - x_k\| \leq (2 + \theta_1)q_i = \frac{2+\theta_1}{1+\theta_1}\lambda_i \leq r_i$. ■ Intuitively, Lemma 30 establishes that if each ball in B_i contains at least one node of the RRG by the end of iteration i , then the RRG algorithm will indeed connect these vertices with edges and construct a path σ'_i from Σ'_i .

The probability that a path from Σ'_i is constructed: Let $A_{i,k}$ be the event that the RRG has no node inside the ball $B_{i,k}$ at the end of iteration i . Moreover, let A_i be the event that at least one of the balls in B_i contains no node of the RRG at the end of iteration i , i.e., $A_i = \bigcup_{k=1}^{K_i+1} A_{i,k}$.

It is appropriate to explain how the main result of this theorem is related to the sequence $\{A_i\}_{i \in \mathbb{N}}$ of events. Recall that ε_i was defined as $\varepsilon_i = \sup_{\sigma'_i \in \Sigma'_i} |c(\sigma'_i) - c^*|$ and that it was already established that ε_i converges to zero as i approaches infinity. Hence, given any ε , one can find a number $i_1 \in \mathbb{N}$ such that for all $i \geq i_1$, $\varepsilon_i < \varepsilon$. Recall also that r_i converges to zero as i approaches infinity. Hence, there exists some $i_2 \in \mathbb{N}$ such that $r_i \leq \delta$ for all $i \geq i_2$. Let $i_0 = \max\{i_1, i_2\}$ and note the following upper bound:

$$\begin{aligned} & \sum_{i=0}^{\infty} \mathbb{P}(\{\mathcal{Y}_i^{\text{RRG}} > c^* + \varepsilon\}) \\ &= \sum_{i=0}^{i_0-1} \mathbb{P}(\{\mathcal{Y}_i^{\text{RRG}} > c^* + \varepsilon\}) + \sum_{i=i_0}^{\infty} \mathbb{P}(\{\mathcal{Y}_i^{\text{RRG}} > c^* + \varepsilon\}) \\ &\leq i_0 + \sum_{i=i_0}^{\infty} \mathbb{P}(\{\mathcal{Y}_i^{\text{RRG}} > c^* + \varepsilon_i\}) \leq i_0 + \sum_{i=i_0}^{\infty} \mathbb{P}(A_i). \end{aligned}$$

To complete the proof, it will be established that $\sum_{i=0}^{\infty} \mathbb{P}(A_i) < \infty$, which implies that $\sum_{i=0}^{\infty} \mathbb{P}(\{\mathcal{Y}_i^{\text{RRG}} > c^* + \varepsilon\}) \leq \infty$ for all $\varepsilon > 0$, which in turn implies the result.

To formally show this claim, define the following sequence of events. For all $i \in \mathbb{N}$, let C_i be the event that for any point $x \in X_{\text{free}}$ the RRG algorithm includes a node v such that $\|x - v\| \leq \eta$ and that the line segment $\text{Line}(x, v)$ joining v and x is obstacle-free. Under the assumptions of Theorem 6, the following lemma holds:

Lemma 31 *Let the assumptions of Theorem 6 and Assumption 14 hold, then there exist constants $a, b \in \mathbb{R}_{>0}$ such that $\mathbb{P}(C_i^c) \leq ae^{-bi}$ for all $i \in \mathbb{N}$.*

Proof: The set X_{free} can be partitioned into finitely many convex sets such that each partition is bounded by a ball of radius η , since X_{free} is a bounded subset of \mathbb{R}^d . Since, by Theorem 6, the probability that each such ball does not include a node of the RRG decays to zero exponentially as i approaches infinity, the probability that at least one such ball does not contain a node of the RRG also decays to zero exponentially (this follows from the union bound). Moreover, if each partition contains a node of the RRG, then any point inside the partition can be connected to the associated node of the RRG with an obstacle-free line segment of length less than η , since each partition is convex and is bounded by a ball of radius η . ■

The following lemma constitutes another important step in proving the main result of this theorem.

Lemma 32 Let $\gamma > 2^d(1 + 1/d)\mu(X_{\text{free}})$, and $\theta_2 \in (0, 1)$ be a constant. Then, $\sum_{i=1}^{\infty} \mathbb{P}(A_i \mid \cap_{j=\lfloor \theta_2 i \rfloor}^i C_j) < \infty$.

Proof: It is desired to compute $\mathbb{P}(A_i \mid \cap_{j=\lfloor \theta_2 i \rfloor}^i C_j)$ under the theorem's assumptions. Since $\cap_{j=\lfloor \theta_2 i \rfloor}^i C_j$ is given, it can be assumed that, from iteration $\lfloor \theta_2 i \rfloor$ to iteration i , the RRG has the following coverage property: for every point $x \in X_{\text{free}}$, the RRG includes a node v that is at most η away from x and the line segment between x and v lies in the obstacle-free region.

Recall that the length of σ_i was s_i . A bound on the number of balls in B_i with respect to i , s_i , and other constants of the problem can be derived as follows. Note that the segment of σ_i that starts at the center of $B_{i,k}$ and ends at the center of $B_{i,k+1}$ has length at least $\theta_1 q_i$ (recall that distance between the centers of two consecutive balls in B_i is $\theta_1 q_i$ by construction), except for the last segment, which has length less than $\theta_1 q_i$. Thus, for all $i \geq i_2$,

$$\begin{aligned} |B_i| = K_i + 1 &\leq \frac{s_i}{\theta_1 q_i} = \frac{(2 + \theta_1)s_i}{\theta_1 r_i} \\ &= \frac{2 + \theta_1}{\theta_1} s_i \left(\frac{\gamma}{\zeta_d}\right)^{1/d} \left(\frac{i}{\log i}\right)^{1/d} \end{aligned}$$

Also, given $\cap_{j=\lfloor \theta_2 i \rfloor}^i C_j$, the probability that a single ball, say $B_{i,1}$, does not contain a node of the RRG at the end of iteration i can be upper-bounded as follows:

$$\begin{aligned} \mathbb{P}(A_{i,1} \mid \cap_{j=\lfloor \theta_2 i \rfloor}^i C_j) &\leq \left(1 - \frac{\mu(B_{i,1})}{\mu(X_{\text{free}})}\right)^{i-\theta_2 i} \\ &= \left(1 - \frac{\gamma}{(2 + \theta_1)^d \mu(X_{\text{free}})} \frac{\log i}{i}\right)^{(1-\theta_2)i} \end{aligned}$$

Using the fact that $(1 - 1/f(i))^r \leq e^{-r/f(i)}$, the right-hand side can further be bounded to obtain the following inequality:

$$\mathbb{P}(A_{i,1} \mid \cap_{j=\lfloor \theta_2 i \rfloor}^i C_j) \leq e^{-\frac{(1-\theta_2)\gamma}{(2+\theta_1)^d \mu(X_{\text{free}})} \log i} = i^{-\frac{(1-\theta_2)\gamma}{(2+\theta_1)^d \mu(X_{\text{free}})}}$$

As a result of the discussion above, we have that

$$\begin{aligned} \mathbb{P}(A_i \mid \cap_{j=\lfloor \theta_2 i \rfloor}^i C_j) &\leq \mathbb{P}\left(\cup_{k=1}^{|B_i|} A_{i,k} \mid \cap_{j=\lfloor \theta_2 i \rfloor}^i C_j\right) \\ &\leq \sum_{k=1}^{|B_i|} \mathbb{P}(A_i \mid \cap_{j=\lfloor \theta_2 i \rfloor}^i C_j) = |B_i| \mathbb{P}(A_{i,1} \mid \cap_{j=\lfloor \theta_2 i \rfloor}^i C_j) \\ &\leq \frac{2s_i \mu(X_{\text{free}})^{1/d}}{\gamma^{1/d}} \left(\frac{i}{\log i}\right)^{1/d} i^{-\frac{(1-\theta_2)\gamma}{(2+\theta_1)^d \mu(X_{\text{free}})}} \\ &= \frac{2s_i \mu(X_{\text{free}})^{1/d}}{\gamma^{1/d}} \frac{1}{(\log i)^{1/d}} i^{-\left(\frac{(1-\theta_2)\gamma}{\mu(X_{\text{free}})(2+\theta_1)^d} - \frac{1}{d}\right)}. \end{aligned}$$

Note that $\sum_{i=1}^{\infty} \mathbb{P}(A_i \mid \cap_{j=\lfloor \theta_2 i \rfloor}^i C_j) < \infty$, for all choices of γ such that $\frac{(1-\theta_2)\gamma}{\mu(X_{\text{free}})(2+\theta_1)^d} - \frac{1}{d} \geq 1$ holds, i.e., whenever $\gamma > 2^d(1 + 1/d)\mu(X_{\text{free}})$, noting that for any such choice of γ , there exists $\theta_1 > 0$ and $\theta_2 > 0$ satisfying the former inequality. Hence, the lemma follows. ■

Lemma 33 Let $\theta_2 \in (0, 1)$ be a constant. Then, $\sum_{i=1}^{\infty} \mathbb{P}\left(\left(\cap_{j=\lfloor \theta_2 i \rfloor}^i C_j\right)^c\right) < \infty$.

Proof: Note the following inequalities:

$$\begin{aligned} \sum_{i=1}^{\infty} \mathbb{P}\left(\left(\cap_{j=\lfloor \theta_2 i \rfloor}^i C_j\right)^c\right) &= \sum_{i=1}^{\infty} \mathbb{P}\left(\cup_{j=\lfloor \theta_2 i \rfloor}^i C_j^c\right) \\ &\leq \sum_{i=1}^{\infty} \sum_{j=\lfloor \theta_2 i \rfloor}^i \mathbb{P}(C_j^c), \leq \sum_{i=1}^{\infty} \sum_{j=\lfloor \theta_2 i \rfloor}^i a' e^{-b' j}, \end{aligned}$$

in which the right-hand side is finite for all $\theta_2 \in (0, 1)$, and where the second inequality follows from Lemma 31. ■

Finally, the relationship of Lemmas 32 and 33 to the main result can be pointed out. Note the following inequality:

$$\begin{aligned} \mathbb{P}(A_i \mid \cap_{j=\lfloor \theta_2 i \rfloor}^i C_j) &= \frac{\mathbb{P}\left(A_i \cap (\cap_{j=\lfloor \theta_2 i \rfloor}^i C_j)\right)}{\mathbb{P}(\cap_{j=\lfloor \theta_2 i \rfloor}^i C_j)} \\ &\geq \mathbb{P}\left(A_i \cap (\cap_{j=\lfloor \theta_2 i \rfloor}^i C_j)\right) \\ &= 1 - \mathbb{P}\left(A_i^c \cup (\cap_{j=\lfloor \theta_2 i \rfloor}^i C_j)^c\right) \\ &\geq 1 - \mathbb{P}(A_i^c) - \mathbb{P}\left((\cap_{j=\lfloor \theta_2 i \rfloor}^i C_j)^c\right) \\ &= \mathbb{P}(A_i) - \mathbb{P}\left((\cap_{j=\lfloor \theta_2 i \rfloor}^i C_j)^c\right). \end{aligned}$$

Hence, taking infinite sums of both sides with respect to n and rearranging, the following inequality is established:

$$\sum_{i=1}^{\infty} \mathbb{P}(A_i) \leq \sum_{i=1}^{\infty} \mathbb{P}(A_i \mid \cap_{j=\lfloor \theta_2 i \rfloor}^i C_j) + \sum_{i=1}^{\infty} \mathbb{P}\left((\cap_{j=\lfloor \theta_2 i \rfloor}^i C_j)^c\right)$$

By Lemmas 32 and 33, the right-hand side is finite, whenever $\gamma > 2^d(1 + 1/d)\mu(X_{\text{free}})$, which implies that, under the same conditions, $\sum_{i=1}^{\infty} \mathbb{P}(A_i) < \infty$, which in turn implies the result.

Proof of Lemma 16

The proof of this lemma employs results from the Random Geometric Graph (RGG) theory (see, e.g., [53]). First, some of the relevant results in the theory of RGGs will be introduced, and then the proof of this lemma will be given.

An RGG $G(V_n, r_n) = (V_n, E_n)$ in X_{free} with n vertices is formed as follows: (i) V_n is a set of n vertices that are sampled identically and independently from X_{free} according to a distribution with a continuous density function $f(x)$, (ii) two vertices $v, v' \in V_n$ are connected with an edge if and only if the distance between them is less than r_n , i.e., $\|v-v'\| \leq r_n$.

The following lemma is a special case of Proposition 3.1 in [53].

Lemma 34 (see [53]) If r_n satisfies $\lim_{n \rightarrow \infty} r_n = 0$, then

$$\lim_{n \rightarrow \infty} \frac{\mathbb{E}[|E_n|]}{n^2(r_n)^d} = \frac{1}{2} \zeta_d \int_{x \in \mathbb{R}^d} f(x)^2 dx.$$

Note that, if $f(x)$ is the uniform density function, then $\int_{x \in \mathbb{R}^d} f(x)^2 dx = 1/\mu(X_{\text{free}})$.

Let V_n° denote the number of vertices that are isolated, i.e., those that do not have any neighbors. The following lemma characterizes the “density” of the isolated vertices in the limit.

Lemma 35 If $\zeta_d(r_n)^d = \gamma \frac{\log n}{n}$ for some $\gamma \in \mathbb{R}_{>0}$, then the number of isolated vertices, $|V_n^0|$, satisfies the following:

$$\frac{\mathbb{E}[|V_n^0|]}{n} \leq \left(1 - \gamma \frac{\log n}{n}\right)^{n-1}.$$

Thus, $\lim_{n \rightarrow \infty} \frac{\mathbb{E}[|V_n^0|]}{n} = 0$; moreover, $\sum_{n=1}^{\infty} \frac{\mathbb{E}[|V_n^0|]}{n}$ is finite whenever $\gamma > 1$.

Proof: Reference [53] does not consider the case when $n(r_n)^d \rightarrow \infty$ as $n \rightarrow \infty$. However, following the reasoning in the proof for the case when $n(r_n)^d \rightarrow \rho \in \mathbb{R}_{>0}$ (see the proof of Proposition 3.3 in [53]), it is possible construct a proof for this lemma. Let us sketch this proof.

Following [53] (note that the notation in [53] is slightly different from that in this paper), and after appropriate simplifications, one gets

$$\frac{\mathbb{E}[V_n^0]}{n} = \int_{x \in X_{\text{free}}} (1 - I_n(x))^{n-1} f(x) dx,$$

where $f(x)$ is the density function, which is uniform over X_{free} in the case under consideration, and $I_n(x) = \int_{x' \in \mathcal{B}_{x,r_n}} x' dx'$, which evaluates to $I_n = \gamma \frac{\log n}{n}$ whenever $(r_n)^d = \gamma \frac{\log n}{n}$ (the reader is referred to [53] for details). Hence,

$$\begin{aligned} \frac{\mathbb{E}[V_n^0]}{n} &= \int_{x \in X_{\text{free}}} \left(1 - \gamma \frac{\log n}{n}\right)^{n-1} \frac{1}{\mu(X_{\text{free}})} dx \\ &= \left(1 - \gamma \frac{\log n}{n}\right)^{n-1}, \end{aligned}$$

which converges to zero as n approaches infinity. Moreover, the sum of these terms is finite whenever $\gamma > 1$. ■

Consider the random geometric graph with the vertex set V_i that is formed with the `Sample` procedure, i.e., $V_i = \bigcup_{j=1}^i \{\text{Sample}(j)\}$. Let V_i^0 denote the set of isolated vertices and let V_i^1 denote the set of the remaining vertices, i.e., $V_i^1 = V_i \setminus V_i^0$. We will denote by E_i^0 and E_i^1 the set of all edges that connect the vertices in V_i^1 and those in V_i^0 , respectively. Hence, we have that $E_i^0 = \emptyset$ and $E_i^1 = E_n$ for all $i \in \mathbb{N}$.

Consider the RRG algorithm. Let V_i^2 denote the set of all vertices that result from extending the graph when the corresponding sample is isolated, i.e., no vertices were present within the ball of volume $\gamma \frac{\log \mathcal{N}_i}{\mathcal{N}_i}$ centered at the sample, `Sample`(i), in iteration i . Let E_i^2 denote the set of all edges that are connected to the vertices in V_i^2 .

Notice the following two facts: (i) the random geometric graph with node set V_i can be described by $G_{RGG,i} = (V_i^0 \cup V_i^1, E_i^0 \cup E_i^1)$, and (ii) the graph maintained by the RRG algorithm is a subgraph of $G_{RRG,i} = (\mathcal{V}_i^{\text{RRG}}, \mathcal{E}_i^{\text{RRG}}) = (V_i^1 \cup V_i^2, E_i^1 \cup E_i^2)$.

Notice that the number of calls to `ObstacleFree` in iteration i is exactly the number of edges created at that iteration, hence is less than or equal to $|E_i^1 \cup E_i^2| = |E_i^1| + |E_i^2|$. The following lemmas lead immediately to the result:

Lemma 36

$$\limsup_{i \rightarrow \infty} \mathbb{E} \left[\frac{|E_i^1|}{\mathcal{N}_i \log(\mathcal{N}_i)} \right] \leq \frac{1}{2} \frac{\gamma}{\mu(X_{\text{free}})}.$$

Proof: Consider the expected value of $|E_i^1|/(n^2(r_n)^d)$, when $n = \mathcal{N}_i$. Substituting $\zeta_d(r_n)^d = \gamma \frac{\log(\mathcal{N}_i)}{\mathcal{N}_i}$ and using Lemma 34, the result follows. ■

Lemma 37 $\limsup_{i \rightarrow \infty} \mathbb{E} \left[\frac{|E_i^2|}{\mathcal{N}_i \log(\mathcal{N}_i)} \right]$ is finite.

Proof: Let M_i^2 denote the number of edges added to E_i^2 at iteration i . Note that

$$\mathbb{E}[M_i^2] \leq \mathbb{E} \left[\frac{\gamma \frac{\log(\mathcal{N}_i)}{\mathcal{N}_i}}{\mu(X_{\text{free}})} |V_i^2| \right] = \frac{\gamma}{\mu(X_{\text{free}})} \mathbb{E} \left[\frac{\log(\mathcal{N}_i)}{\mathcal{N}_i} |V_i^2| \right].$$

Hence, noting $E_i^2 = \sum_{j=1}^i M_j^2$,

$$\limsup_{i \rightarrow \infty} \mathbb{E} \left[\frac{|E_i^2|}{\mathcal{N}_i \log(\mathcal{N}_i)} \right] \leq \limsup_{i \rightarrow \infty} \mathbb{E} \left[\frac{\sum_{j=1}^i \frac{\log(\mathcal{N}_j)}{\mathcal{N}_j} |V_j^2|}{\mathcal{N}_i \log(\mathcal{N}_i)} \right]$$

Consider the expectation in the right hand side. Noting that, surely, $\log(\mathcal{N}_j) \leq \log(\mathcal{N}_i)$ for all $j \leq i$, the following inequalities hold:

$$\begin{aligned} \mathbb{E} \left[\frac{\sum_{j=1}^i \frac{\log(\mathcal{N}_j)}{\mathcal{N}_j} |V_j^2|}{\mathcal{N}_i \log(\mathcal{N}_i)} \right] &\leq \mathbb{E} \left[\frac{1}{\mathcal{N}_i} \sum_{j=1}^i \frac{|V_j^2|}{\mathcal{N}_j} \right] \\ &\leq \mathbb{E} \left[\frac{1}{\mathcal{N}_i} \right]^{1/2} \mathbb{E} \left[\sum_{j=1}^i \frac{|V_j^2|}{\mathcal{N}_j} \right]^{1/2} \\ &\leq \mathbb{E} \left[\frac{1}{\mathcal{N}_i} \right]^{1/2} \left(\sum_{j=1}^i \mathbb{E} \left[\frac{|V_j^2|}{\mathcal{N}_j} \right] \right)^{1/2} \end{aligned}$$

where the second inequality follows from the Cauchy-Schwarz inequality. Notice that the first term approaches zero as $i \rightarrow \infty$, whereas, by Lemma 35, the second term is finite. Hence, it is concluded that

$$\limsup_{i \rightarrow \infty} \mathbb{E} \left[\frac{1}{\mathcal{N}_i} \right]^{1/2} \left(\sum_{j=1}^i \mathbb{E} \left[\frac{|V_j^2|}{\mathcal{N}_j} \right] \right)^{1/2} < \infty,$$

which implies the lemma. ■