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, e.g., in terms of a given cost function, have been established so far. The purpose of this paper is to fill this gap, by designing efficient incremental sampling-based algorithms with provable optimality properties. The first contribution of this paper is a negative result: it is proven that, under mild technical conditions, the cost of the best path returned by RRT converges almost surely to a non-optimal value, as the number of samples increases. Second, a new algorithm is considered, called the Rapidly-exploring Random Graph (RRG), and it is shown that the cost of the best path returned by RRG converges to the optimum almost surely. Third, a tree version of RRG is introduced, called RRT*, 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

Even though modern robots may posses significant differences in sensing, actuation, size, application, or workspace, the motion planning problem, i.e., the problem of planning a dynamically feasible trajectory through a complex environment cluttered with obstacles, 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]–[5].

Motion planning has been a highly active area of research since the late 1970s. Early approaches to the problem has mainly focused on the development of *complete* planners (see, e.g., [6]), which find a solution if one exists and return failure otherwise. However, it was established as early as 1979 that even a most basic version of the motion planning problem, called the piano mover's problem, is known to be PSPACE-hard [7], which strongly suggests that complete planners are doomed to suffer from computational complexity.

Tractable algorithms approach the motion planning problem by relaxing the completeness requirement to, for instance, *resolution completeness*, which amounts to finding a solution, if one exists, when the resolution parameter of the algorithm is set fine enough. Most motion planning methods that are based on gridding or cell decomposition fall into this category. A

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

more recent line of research that has achieved a great success has focused on the construction of paths connecting randomly-sampled points. Algorithms such as Probabilistic RoadMaps (PRM) [8] have been shown to be *probabilistically complete*, i.e., such that the probability of finding a solution, if one exists, approaches one as the number of samples approaches infinity.

The PRM algorithm constructs a graph of feasible paths off-line, and is primarily aimed at multiple-query applications, in which several motion-planning problems need to be solved in the same environment. Incremental sampling-based algorithms have been developed for single-query, real-time applications; among the most influential of these, one can mention Rapidly-exploring Random Trees (RRT) [9], and the algorithm in [10]. These algorithms have been shown to be probabilistically complete, with an exponential decay of the probability of failure. Moreover, RRTs were demonstrated in various robotic platforms in major robotics events (see, e.g., [11]).

A class of incremental sampling-based motion planning algorithms that is worth mentioning at this point is the Rapidly-exploring Random Graphs (RRGs), which were proposed in [12] to find feasible trajectories that satisfy specifications other than the usual "avoid all the obstacles and reach the goal region". More generally, RRGs can handle specifications given in the form of deterministic μ-calculus, which includes the widely-used Linear Temporal Logic (LTL). RRGs incrementally build a graph of trajectories, since specifications given in μ-calculus, in general, require infinite-horizon looping trajectories, which are not included in trees.

To address the challenges posed by real-time applications, state-of-the-art motion planning algorithms, such as RRTs, are tailored to return a feasible solution quickly, paying almost no attention to the "quality" of the solution. On the other hand, in typical implementations [11], the algorithm is not terminated as soon as the first feasible solution is found; rather, all the available computation time is used to search for an improved solution, with respect to a performance metric such as time, path length, fuel consumption, etc. A shortcoming of this approach is that there is no guarantee that the computation will eventually converge to optimal trajectories. In fact, despite the clear practical need, there has been little progress in characterizing optimality properties of sampling-based motion planning algorithms, even though the importance of these problems was emphasized in early seminal papers such as [9].

Yet, the importance of the quality of the solution returned by the planners has been noticed, in particular, from the point of view of incremental sampling-based motion planning. In [13], 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 [14], 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 optimal solution.

To the best of the authors' knowledge, this paper provides the first thorough analysis of optimality properties of incremental sampling-based motion planning algorithms. In particular, it is shown that the probability that the RRT converges to an optimal solution, as the number of samples approaches infinity, is zero under some reasonable technical assumptions. In fact, the RRT algorithm almost always converges to a nonoptimal solution. Second, it is shown 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, a novel variant of the RRG algorithm is introduced, 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 as it discovers new lower-cost paths reaching the nodes that are already in the tree. Finally, it is shown that the asymptotic computational complexity of the RRG and RRT* algorithms is essentially the same as that of RRTs.

To the authors' knowledge, the algorithms considered in this paper are the first computationally efficient incremental sampling-based motion planning algorithms with asymptotic optimality guarantees. Indeed, the results in this paper 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 [15], [16].

The paper is organized as follows. 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 and directions for future work are given in Section VII.

Due to space limitations, results are stated without formal proofs. An extended version of this paper, including proofs of the major results, technical discussions, and extensive experimental results, is available [17]. An implementation of the RRT* algorithm in the C language is available at `http://ares.lids.mit.edu/software`.

The focus of this paper is on the basic problem of navigating through a connected bounded subset of a d-dimensional Euclidean space. However, the proposed algorithms also extend to systems with differential constraints, as shown in [18].

II. Preliminary Material

A. Notation

A sequence on a set A, denoted as $\{a_i\}_{i\in\mathbb{N}}$, is a mapping from $\mathbb{N}$ to A with $i \mapsto a_i$. Given $a, b \in \mathbb{R}$, the closed interval between a and b is denoted by $[a, b]$. The Euclidean norm is denoted by $\|\cdot\|$. Given a set $X \subset \mathbb{R}^d$, the closure of X is denoted by $\mathrm{cl}(X)$, the Lebesgue measure of X, i.e., its volume, is denoted by $\mu(X)$. The closed ball of radius $r > 0$ centered at $x \in \mathbb{R}^d$ is defined as $\mathcal{B}_{x,r} := \{y \in \mathbb{R}^d \mid \|y - x\| \leq r\}$. 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 $\sigma : [0, s] \to X$, where s is the length of the path defined in the usual way. Given two paths in X, $\sigma_1 : [0, s_1] \to X$, and $\sigma_2 : [0, s_2] \to X$, with $\sigma_1(s_1) = \sigma_2(0)$, their concatenation is denoted by $\sigma_1|\sigma_2$, i.e., $\sigma = \sigma_1|\sigma_2 : [0, s_1 + s_2] \to X$ with $\sigma(s) = \sigma_1(s)$ for all $s \in [0, s_1]$, and $\sigma(s) = \sigma_2(s - s_1)$ for all $s \in [s_1, s_1 + s_2]$. The set of all paths in X with nonzero length is denoted by Σ_X. The straight continuous path between $x_1, x_2 \in \mathbb{R}^d$ is denoted by $\mathtt{Line}(x_1, x_2)$.

Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space. A random variable is a measurable function from Ω to $\mathbb{R}$; an extended random variable can also take the values $\pm\infty$. A sequence $\{\mathcal{Y}_i\}_{i\in\mathbb{N}}$ of random variables is said to converge surely to a random variable $\mathcal{Y}$ if $\lim_{i\to\infty} \mathcal{Y}_i(\omega) = \mathcal{Y}(\omega)$ for all $\omega \in \Omega$; the sequence is said to converge almost-surely if $\mathbb{P}(\{\lim_{i\to\infty} \mathcal{Y}_i = \mathcal{Y}\}) = 1$.

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., $X \setminus X_{\mathrm{obs}}$, 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 $\sigma : [0, s] \to X_{\mathrm{free}}$ is feasible if and only if $\sigma(0) = x_{\mathrm{init}}$ and $\sigma(s) \in X_{\mathrm{goal}}$.

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

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

Let $c : \Sigma_{X_{\mathrm{free}}} \to \mathrm{R}_{>0}$ 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.

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* $\sigma^* : [0, s] \to \mathrm{cl}(X_{\mathrm{free}})$ *such that (i)* $\sigma^*(0) = x_{\mathrm{init}}$ *and* $\sigma^*(s) \in X_{\mathrm{goal}}$*, and (ii)* $c(\sigma^*) = \min_{\sigma \in \Sigma_{\mathrm{cl}(X_{\mathrm{free}})}} c(\sigma)$*. 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 $\texttt{Sample} : \mathbb{N} \to X_{\text{free}}$ returns independent identically distributed (i.i.d.) samples from X_{free}.

Steering: Given two points $x, y \in X$, the function $\texttt{Steer} : (x, y) \mapsto z$ 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 $\texttt{Steer}$ will be such that z minimizes $\|z - y\|$ while at the same time maintaining $\|z - x\| \leq \eta$, for a prespecified $\eta > 0$, i.e., $\texttt{Steer}(x, y) = \operatorname{argmin}_{z \in \mathbb{R}^d, \|z-x\| \leq \eta} \|z - y\|$.

Nearest Neighbor: Given a graph $G = (V, E) \in \mathcal{G}_{X_{\text{free}}}$ and a point $x \in X_{\text{free}}$, the function $\texttt{Nearest} : (G, x) \mapsto v$ 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., [9] for alternative choices), i.e., $\texttt{Nearest}(G = (V, E), x) = \operatorname{argmin}_{v \in V} \|x - v\|$.

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 $\texttt{Near} : (G, x, n) \mapsto V'$ returns a set V' of vertices such that $V' \subseteq V$. The $\texttt{Near}$ 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 $\texttt{Nearest}$ procedure, there are many ways to define the $\texttt{Near}$ procedure, each of which leads to different algorithmic properties. For technical reasons to become clear later, we define $\texttt{Near}(G, x, n)$ 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 $\texttt{ObstacleFree}(x, x')$ returns $\texttt{True}$ 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 $\texttt{Extend}$ 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 the vertices returned by the $\texttt{Near}$ procedure, producing a graph in general. In both cases, all the extensions resulting in collision-free trajectories 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 $\quad G \leftarrow (V, E);$
4 $\quad x_{\text{rand}} \leftarrow \texttt{Sample}(i); i \leftarrow i + 1;$
5 $\quad (V, E) \leftarrow \texttt{Extend}(G, x_{\text{rand}});$

Algorithm 2: $\texttt{Extend}_{RRT}$

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

Algorithm 3: $\texttt{Extend}_{RRG}$

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

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.

Sets of vertices and edges of the graphs maintained by the RRT and the RRG algorithms 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 [9]) *If there exists a feasible solution to Problem 1, then* $\lim_{i \to \infty} \mathbb{P}\left(\{\mathcal{V}_i^{\text{RRT}} \cap X_{\text{goal}} \neq \emptyset\}\right) = 1$.

An *attraction sequence* [9] is defined as a finite sequence $\mathcal{A} = \{A_1, A_2, \ldots, A_k\}$ 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 $\|x - y\| \leq \|x - z\|$ Given an attraction sequence $\mathcal{A}$ of length k, let p_k denote $\min_{i \in \{1,2,\ldots,k\}}\left(\frac{\mu(A_i)}{\mu(X_{\text{free}})}\right)$.

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

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

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, it is shown 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-optimal* solution with probability one. On the contrary, it is subsequently shown that the RRG algorithm converges to an optimal 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, $\mathcal{Y}_i^{\text{RRT}}(\omega) \geq c^*$ 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 $\lim_{i \to \infty} \mathcal{Y}_i^{\text{RRT}}(\omega)$ exists. The same argument also holds for the sequence $\{\mathcal{Y}_i^{\text{RRG}}\}_{i \in \mathbb{N}}$.

1) Almost Sure Nonoptimality 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., $X_{\text{opt}} = \cup_{\sigma^* \in \Sigma^*} \cup_{\tau \in [0, s^*]} \{\sigma^*(\tau)\}$. Consider the following assumptions.

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 $\{\texttt{Sample}(i)\}_{i \in \mathbb{N}}$ 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: $c(\sigma_1) \leq c(\sigma_1 | \sigma_2)$.*

Assumption 7 rules out trivial cases, in which the RRT algorithm can sample exactly an optimal path with non-zero probability. 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.,*

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

whenever $d \geq 2$.

The key idea in proving this result is that the probability of extending a node on an optimal path (e.g., the root node) goes to zero very quickly, in such a way that any such node will only have a finite number of children, almost surely. Because of Assumptions 7 and 8, this implies the result.

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

It is interesting to note that, 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 [14]. In fact, running multiple instances of the RRT algorithm amounts to drawing multiple samples of $\mathcal{Y}_\infty^{RRT}$.

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

Assumption 11 (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 12 (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 13 (Obstacle Spacing) *There exists a constant $\delta \in \mathbb{R}_+$ such that for any point $x \in X_{\text{free}}$, there exists $x' \in X_{\text{free}}$, 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 12 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 11 and 12. Assumption 13 is a rather technical assumption, which ensures existence of some free space around the optimal trajectories to allow convergence. For simplicity, it is assumed that the sampling is uniform, although the results can be directly 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 14 *Let Assumptions 11, 12, and 13 hold, and assume that Problem 1 admits a feasible solution. Then, the cost of the minimum-cost path in the RRG converges to the optimal cost almost-surely, i.e.,*

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

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

This result relies on the fact that a random geometric graph with n vertices formed by connecting each vertex with vertices within a distance of $d_n = \gamma'\,(\log n \,/\, n)^{1/d}$ will result in a connected graph almost surely as $n \to \infty$, whenever γ' is larger than a certain lower bound γ_1 [19]. In fact, the bound on γ' is a *tight threshold* in the sense that there exists an upper bound $\gamma_2 < \gamma_1$ such that, if $\gamma' < \gamma_2$, then the resulting graph will be disconnected almost surely [19]. This result strongly suggests that shrinking the ball in the Near procedure faster than the rate proposed will not yield an asymptotically optimal algorithm. The authors have experienced this fact in simulation studies: setting γ to around one third of γ_L does not seem to provide the asymptotic optimality property. On the other hand, as it will be shown in the next section, if the size of the same ball is reduced slower than the proposed rate, then the asymptotic complexity of the resulting algorithm will not be the same as the RRT. Hence, scaling r_n with $(\log n \,/\, n)^{1/d}$ in the Near procedure, surprisingly, achieves the perfect balance between asymptotic optimality and low computational complexity, since relevant results in the theory of random geometric graphs and lower bounds on nearest neighbor computation strongly suggest that a different rate will lose either the former or the latter while failing to provide an extra benefit in any of the two.

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.

Consider first the computational complexity of the RRT and the RRG algorithms in terms of the number of calls to the primitive procedures introduced in Section III. 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.

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 15 *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., $\limsup_{i \to \infty} \mathbb{E}\left[\frac{\mathcal{O}_i^{\text{RRG}}}{\log(\mathcal{N}_i)}\right] \leq \phi$, where $\phi \in \mathbb{R}_{>0}$ is a constant that depends only on the problem instance.*

However, some primitive procedures clearly take more computation time to execute than others. For a meaningful comparison, one should also evaluate the time required to execute each primitive procedure in terms of the number of simple operations (also called steps) 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 thats the RRT and the RRG algorithms have the same asymptotic computational complexity in terms of the number of steps that they perform.

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.

Second, consider the computational complexity of the Nearest procedure. The problem of finding a nearest neighbor is widely studied, e.g., in the computer graphics literature. Even though algorithms that achieve sub-linear time complexity are known [20], lower bounds suggest that nearest neighbor computation requires at least logarithmic time [21]. In fact, assuming that the Nearest procedure computes an approximate nearest neighbor (see, e.g., [21] for a formal definition) using the algorithm given in [21], which is optimal in fixed dimensions from a computational complexity point of view closely matching a lower bound for tree-based algorithms, the Nearest 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 executed by the RRT algorithm in iteration i.

Lemma 16 *Assuming that* Nearest *is implemented using the algorithm given in [21], 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* $\phi_{RRT} \in \mathbb{R}_{>0}$ *such that* $\liminf_{i \to \infty} \mathbb{E}\left[\frac{\mathcal{M}_i^{\text{RRT}}}{\log(\mathcal{N}_i)}\right] \geq \phi_{RRT}$.

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 [20].

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. In fixed dimensions, computing ε-approximate solutions can, in fact, be done in logarithmic time using polynomial space, in the worst case [22].

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 15, the discussion above implies the following lemma.

Lemma 17 *Assuming that the* Near *procedure is implemented using the algorithm given in [22], 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* $\phi_{RRG} \in \mathbb{R}_{>0}$ *such that* $\limsup_{i \to \infty} \mathbb{E}\left[\frac{\mathcal{M}_i^{\text{RRG}}}{\log(\mathcal{N}_i)}\right] \leq \phi_{RRG}$.

Finally, by Lemmas 16 and 17, we conclude that the RRT and the RRG algorithms have the same asymptotic computational complexity as stated in the following theorem.

Theorem 18 *Under the assumptions of Lemmas 16 and 17, there exists a constant* $\phi \in \mathbb{R}_{>0}$ *such that*

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

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 $\text{Line}(x, x') : [0, s] \to X_{\text{free}}$ 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 $\text{Cost}(v)$. Initially, $\text{Cost}(x_{\text{init}})$ is set to zero.

Algorithm 4: Extend$_{\text{RRT}^*}$

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** $\text{ObstacleFree}(x_{\text{nearest}}, x_{\text{new}})$ **then**
5      $V' \leftarrow V' \cup \{x_{\text{new}}\}$;
6      $x_{\text{min}} \leftarrow x_{\text{nearest}}$;
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** $\text{ObstacleFree}(x_{\text{near}}, x_{\text{new}})$ **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                 $x_{\text{min}} \leftarrow x_{\text{near}}$;
13     $E' \leftarrow E' \cup \{(x_{\text{min}}, x_{\text{new}})\}$;
14   **for** *all* $x_{near} \in X_{\text{near}} \setminus \{x_{\text{min}}\}$ **do**
15   **if** $\text{ObstacleFree}(x_{\text{new}}, x_{\text{near}})$ *and* $\text{Cost}(x_{\text{near}}) > \text{Cost}(x_{\text{new}}) + c(\text{Line}(x_{\text{new}}, x_{\text{near}}))$ **then**
16             $x_{\text{parent}} \leftarrow \text{Parent}(x_{\text{near}})$;
17             $E' \leftarrow E' \setminus \{(x_{\text{parent}}, x_{\text{near}})\}$;
            $E' \leftarrow E' \cup \{(x_{\text{new}}, x_{\text{near}})\}$;

18 **return** $G' = (V', E')$

Similar 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. The following lemma is the equivalent of Lemma 3.

Lemma 19 *For all* $i \in \mathbb{N}$ *and all* $\omega \in \Omega$, $\mathcal{V}_i^{\text{RRT}^*}(\omega) = \mathcal{V}_i^{\text{RRG}}(\omega)$, *and* $\mathcal{E}_i^{\text{RRT}^*}(\omega) \subseteq \mathcal{E}_i^{\text{RRG}}(\omega)$.

From Lemma 19 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 20 *If there exists a feasible solution to Problem 1, then $\lim_{i\to\infty} \mathbb{P}\left(\{\mathcal{V}_i^{\mathrm{RRT}^*} \cap X_{\mathrm{goal}} \neq \emptyset\}\right) = 1$. Moreover, if an attraction sequence $\mathcal{A}$ of length k exists, then $\mathbb{P}\left(\{\mathcal{V}_i^{\mathrm{RRT}^*} \cap X_{\mathrm{goal}} = \emptyset\}\right) \leq e^{-\frac{1}{2}(i\,p_k - 2\,k)}$.*

C. Asymptotic Optimality

Let $\mathcal{Y}_i^{\mathrm{RRT}^*}$ 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 21 *Let Assumptions 11, 12, and 13 hold. Then, the cost of the minimum cost path in the RRT* converges to c^* almost surely, i.e., $\mathbb{P}\left(\{\lim_{i\to\infty} \mathcal{Y}_i^{\mathrm{RRT}^*} = c^*\}\right) = 1$.*

D. Computational Complexity

Let $\mathcal{M}_i^{\mathrm{RRT}^*}$ be the number of steps performed by the RRT* algorithm in iteration i. The following theorem follows from Lemma 19 and Theorem 18.

Theorem 22 *Under the assumptions of Theorem 18, there exists a constant ϕ such that $\limsup_{i\to\infty} \mathbb{E}\left[\frac{\mathcal{M}_i^{\mathrm{RRT}^*}}{\mathcal{M}_i^{\mathrm{RRT}}}\right] \leq \phi$.*

VI. SIMULATIONS

This section presents simulation examples. A thorough simulation study of the algorithms can be found in [17].

The RRT and the RRT* algorithms are run in a square environment with obstacles and the cost function is set to the Euclidean path length. The trees maintained by the algorithms at different stages are shown in Figure 1. The figure illustrates that the RRT algorithm does not considerably improve the solution, whereas the RRT* algorithm converges towards an optimal solution by finding a feasible solution of the homotopy class that the optimal path lies in. An important difference between the RRT and the RRT* algorithms is the ability of the latter to efficiently consider different homotopy classes. Thus, in an environment cluttered with obstacles, the cost of first feasible solution found by the RRT or the RRT* algorithms can be drastically higher than the optimal cost. Although the RRT* algorithm efficiently improves the solution over time, the RRT algorithm tends to get stuck with the first solution found. In fact, Monte-Carlo runs of both algorithms, as shown in Figure 2.(a)-(b), illustrate that generally the RRT does not improve the first solution found, whereas the RRT* algorithm improves the solution significantly within the first few thousand iterations, for this particular scenario. Moreover, the cost of the best path in RRT seems to have high variance, while after a few thousand iterations the costs of the best path in the RRT* is almost the same in all runs, as expected from the theoretical results presented in the previous sections. Finally, the relative complexity of the two algorithms is demonstrated in Monte-Carlo runs in Figure 2.(c). Notice that the ratio of the running time of the algorithms up until a certain iteration converges to a constant as the number of iterations increases. Note that the convergence to this constant is achieved when the free space is "fully explored", i.e., almost uniformly filled with the nodes of the trees. However, before then the complexity of the RRT* is much lower than the complexity in the limit value. In fact, the average amount of time that the RRT* algorithm takes for finding a feasible solution was found to be no more than five times that of the RRT, in this particular scenario. Moreover, the first solution found by the RRT* generally costs considerably less than that found by the RRT.

VII. CONCLUSIONS AND FUTURE WORK

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 optimal 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 demonstrating the effectiveness of the proposed algorithms and supporting the theoretical claims was 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 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 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.
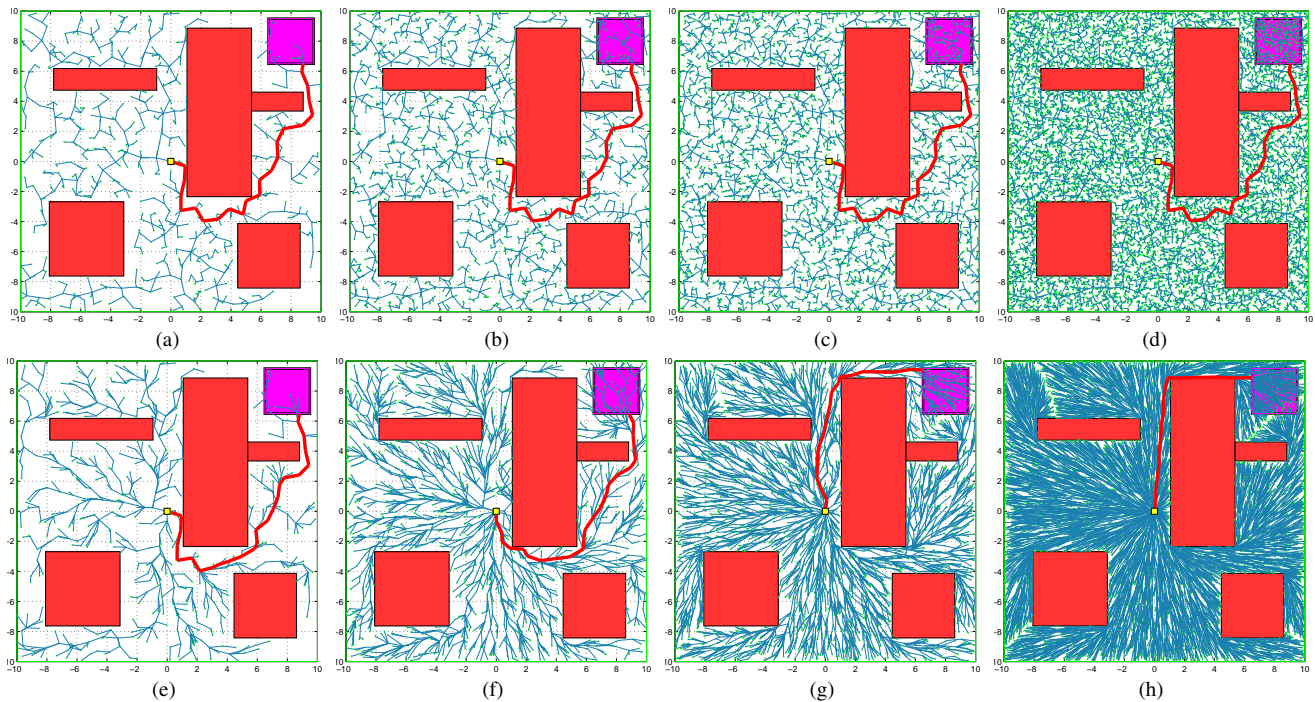
Fig. 1. A Comparison of the RRT* and RRT algorithms on a simulation example. The tree maintained by the RRT algorithm is shown in (a)-(d) in different stages, whereas that maintained by the RRT* algorithm is shown in (e)-(h). The tree snapshots (a), (e) are at 1000 iterations , (b), (f) at 2500 iterations, (c), (g) at 5000 iterations, and (d), (h) at 15,000 iterations. The goal regions are shown in magenta. The best paths that reach the target are highlighted with red.
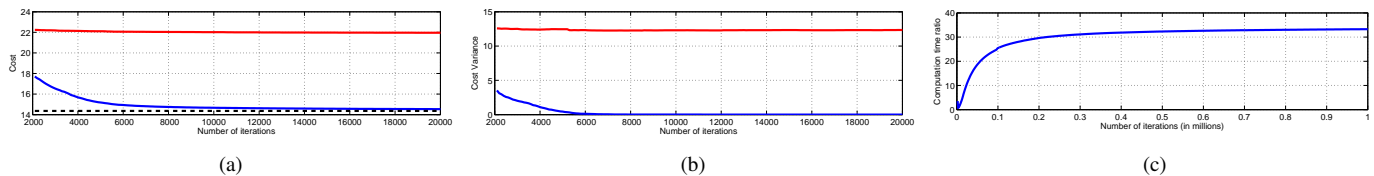

Fig. 2. The cost of the best paths in the RRT (shown in red) and the RRT* (shown in blue) plotted against iterations averaged over 500 trials in (a). The optimal cost is shown in black. The variance of the trials is shown in (b). A comparison of the running time of the RRT* and the RRT algorithms averaged over 50 trials is shown in (c); 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.

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.

[6] 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.

[7] J.H. Reif. Complexity of the mover's problem and generalizations. In *Proceedings of the IEEE Symposium on Foundations of Computer Science*, 1979.

[8] 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.

[9] S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. *International Journal of Robotics Research*, 20(5):378–400, May 2001.

[10] 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.

[11] 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.

[12] S. Karaman and E. Frazzoli. Sampling-based motion planning with deterministic μ-calculus specifications. In *IEEE Conference on Decision and Control (CDC)*, 2009.

[13] 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.

[14] D. Ferguson and A. Stentz. Anytime RRTs. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2006.

[15] M. Penrose. *Random Geometric Graphs*. Oxford University Press, 2003.

[16] J. Dall and M. Christensen. Random geometric graphs. *Physical Review E*, 66(1):016121, Jul 2002.

[17] S. Karaman and E. Frazzoli. Incremental sampling-based algorithms for optimal motion planning. http://arxiv.org/abs/1005.0416.

[18] S. Karaman and E. Frazzoli. Optimal kinodynamic motion planning using incremental sampling-based methods. In *Proceedings of the IEEE Conference on Decision and Control (CDC)*, 2010. Submitted.

[19] 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.

[20] H. Samet. *Design and Analysis of Spatial Data Structures*. Addison-Wesley, 1989.

[21] 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.

[22] S. Arya and D. M. Mount. Approximate range searching. *Computational Geometry: Theory and Applications*, 17:135–163, 2000.