49th IEEE Conference on Decision and Control
December 15-17, 2010
Hilton Atlanta Hotel, Atlanta, GA, USA

Optimal Kinodynamic Motion Planning using Incremental Sampling-based Methods

Sertac Karaman Emilio Frazzoli

Abstract— **Sampling-based algorithms such as the Rapidly-exploring Random Tree (RRT) have been recently proposed as an effective approach to computationally hard motion planning problem. However, while the RRT algorithm is known to be able to find a feasible solution quickly, there are no guarantees on the quality of such solution, e.g., with respect to a given cost functional. To address this limitation, the authors recently proposed a new algorithm, called RRT*, which ensures asymptotic optimality, i.e., almost sure convergence of the solution returned by the algorithm to an optimal solution, while maintaining the same properties of the standard RRT algorithm, both in terms of computation of feasible solutions, and of computational complexity. In this paper, the RRT* algorithm is extended to deal with differential constraints. A sufficient condition for asymptotic optimality is provided. It is shown that the RRT* algorithm equipped with any local steering procedure that satisfies this condition converges to an optimal solution almost surely. In particular, simple local steering procedures are provided for a Dubins' vehicle as well as a double integrator. Simulation examples are also provided for these systems comparing the RRT and the RRT* algorithms.**

I. Introduction

Motion planning is a fundamental problem that is embedded and essential in almost all robotics applications [1], ranging from health care [2] to industrial production [3], and from autonomous urban navigation [4]–[6] to military logistics [7]. Moreover, motion planning has many applications outside the domain of robotics, including, for instance, verification, drug design, computer animation, etc. [8].

Informally speaking, given a robot with a description of its dynamics, an initial state, a final state, a set of obstacles, and a goal region, the motion planning problem is to find a sequence of inputs that drives the system from its initial condition to the goal region, while avoiding collision with obstacles. Even though a vast set of applications make this problem interesting, it was shown as early as 1979 that a basic version of the problem, called the piano mover's problem, is PSPACE-hard [9]. Yet, many approaches, such as cell decomposition [10] and potential fields [11] have been proposed to tackle challenging instances of the problem. More recently, with the advent of sampling-based methods [12], [13], several real-world motion planning problems were solved in real-time in online settings [4].

Sampling-based approaches to motion planning randomly sample a set of states from the state-space and check their connectivity by methods that do not require explicit construction of obstacles in the state-space, which provides considerable savings in computation time. The connectivity of these samples provides a strong hypothesis on the connectivity of the obstacle-free portion of the state-space, and, in particular, the connectivity of the initial and goal states. Even though sampling-based methods do not provide completeness guarantees, they are probabilistically complete in the sense that the probability of finding a feasible solution, if one exists, approaches one as the number of samples increase.

One such sampling-based planner is the Rapidly-Exploring Random Tree (RRT) algorithm, first proposed by LaValle and Kuffner in [14]. The algorithm has received considerable attention in the literature since its introduction. Recently, the RRT algorithm and its variants were successfully demonstrated on different robotic platforms in major robotics events (see, e.g., [4], [15], [16]).

In many real-time applications, it is highly desirable to find a feasible solution quickly and improve the "quality" of solution in the remaining time until the execution of the motion plan. In fact, many field implementations of the RRT algorithm do not stop sampling once a feasible trajectory is found, but keep searching for an better solution [4], [15]. However, the quality of the solution returned by the RRT, or by algorithms designed to provably improve the quality of the solution given more computation time, has remained largely open despite the increasing recent interest [15], [17], [18].

An important step towards efficiently optimizing using randomized planners was taken in [19]. In particular, the authors showed that the RRT algorithm converges to a non-optimal solution with probability one. Furthermore, they introduced a new algorithm, called RRT*, and proved that it is globally asymptotically optimal for systems without differential constraints, while maintaining the same probabilistic completeness and computational efficiency of the baseline RRT.

In this paper, we extend the RRT* algorithm to handle systems with differential constraints. The main contribution of this paper is the identification of a set of sufficient conditions to ensure asymptotic optimality of the RRT* algorithm for systems with differential constraints. These sufficient conditions are satisfied by locally controllable systems (in particular, controllable linear systems) and locally optimal steering methods. Simulation examples that involve systems with nonholonomic dynamics are also provided.

This paper is organized as follows. In Section II, we provide a formal definition of the optimal kinodynamic motion planning problem. In Section III, we provide the RRT* algorithm extended to handle systems with differential

The authors are with the Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA, 02139.

constraints. In Section IV, we provide the sufficient conditions for asymptotic optimality and state our main result. We consider the Dubins vehicle and double integrator examples in Section V and provide the related simulation results in Section VI. We conclude the paper in Section VII.

II. Problem Definition

Let $X \subset \mathbb{R}^n$ and $U \subset \mathbb{R}^m$ be compact sets and consider the dynamical system

$$\dot{x}(t) = f(x(t), u(t)), \quad x(0) = x_0 \tag{1}$$

where $x(t) \in X \subseteq \mathbb{R}^d$, $u(t) \in U \subseteq \mathbb{R}^m$, for all t, $x_0 \in X$, and f is a smooth (continuously differentiable) function of its variables. Let us denote the set of all essentially bounded measurable functions defined from $[0, T]$ to X, for any $T \in \mathbb{R}_{>0}$, by $\mathcal{X}$ and define $\mathcal{U}$ similary. The functions in $\mathcal{X}$ and $\mathcal{U}$ are called *trajectories* and *controls*, respectively.

Let X_{obs} and X_{goal}, called the *obstacle region* and the *goal region*, respectively, be open subsets of X. Let X_{free}, also called the *free space*, denote the set defined as $X \setminus X_{\text{obs}}$.

Informally speaking, the *kinodynamic motion planning* problem is to find a control $u : [0, T] \to U$, such that the unique trajectory $x(t)$ that satisfies Equation (1) reaches the goal region while avoiding the obstacle region.

Problem 1 (Kinodynamic motion planning) *Given the domain X, obstacle region X_{obs}, goal region X_{goal}, and a smooth function f that describes the system dynamics, find a control $u \in \mathcal{U}$ with domain $[0, T]$ for some $T \in \mathbb{R}_{>0}$ such that the corresponding unique trajectory $x \in \mathcal{X}$, with $\dot{x}(t) = f(x(t), u(t))$ for all $t \in [0, T]$,*

- *avoids the obstacles, i.e., $x(t) \in X_{\text{free}}$ for all $t \in [0, T]$,*
- *and reaches the goal region, i.e., $x(T) \in X_{\text{goal}}$.*

In this paper, we consider the *optimal kinodynamic motion planning problem*, which is to solve the kinodynamic motion planning problem while minimizing a given cost functional. As it is general practice in optimal control [20], we assume that this cost functional is the line integral of a Lipschitz continuous function $g : X \to \mathbb{R}_{>0}$, where g is bounded away from zero in X[1].

Problem 2 (Optimal kinodynamic motion planning) *Given the domain X, obstacle region X_{obs}, goal region X_{goal}, and a smooth function f that describes the system dynamics, find a control $u \in \mathcal{U}$ with domain $[0, T]$ for some $T \in \mathbb{R}_{>0}$ such that the unique corresponding trajectory $x \in \mathcal{X}$, with $\dot{x}(t) = f(x(t), u(t))$ for all $t \in [0, T]$,*

- *avoids the obstacles, i.e., $x(t) \in X_{\text{free}}$ for all $t \in [0, T]$,*
- *reaches the goal region, i.e., $x(T) \in X_{\text{goal}}$,*
- *and minimizes the cost functional $J(x) = \int_0^T g(x(t))\, dt$.*

[1]More general formulations of optimal control problems also consider the input u as a part of the cost functional and/or a terminal cost function, $h : X \to \mathbb{R}_{>0}$, which incurs an extra additive term $g(x(T))$ in the cost functional. Our results can be extended to these cases, although we avoid this discussion here for the sake of simplicity.

III. RRT* Algorithm

Before providing the details of RRT*, let us outline the primitive procedures that the algorithm relies on.

Sampling: The sampling procedure $\texttt{Sample} : \mathbb{N} \to X_{\text{free}}$ returns independent and identically distributed samples from the obstacle-free space. To simplify theoretical arguments we assume that the sampling distribution is uniform, even though our results hold for a large class of sampling strategies.

Distance function: Let $\texttt{dist} : X \times X \to \mathbb{R}_{\geq 0}$ be a function that returns the optimal cost of a trajectory between two states, assuming no obstacles. In other words, $\texttt{dist}(z_1, z_2) = \min_{T \in \mathbb{R}_{\geq 0}, u:[0,T] \to \mathcal{U}} J(x)$, s.t. $\dot{x}(t) = f(x(t), u(t))$ for all $t \in [0, T]$, and $x(0) = z_1$, $x(T) = z_2$.

Nearest Neighbor: Given a graph $G = (V, E)$ on X_{free} and a state $z \in X$, the procedure $\texttt{Nearest}(G, z)$ returns the vertex $v \in V$ that is closest to z, according to the distance function defined above, i.e., $\texttt{Nearest}(G, z) = \arg\min_{v \in V} \texttt{dist}(v, z)$.

Near-by Vertices: Given a graph $G = (V, E)$ on X_{free}, a state $z \in X$, and a number $n \in \mathbb{N}$, the `NearVertices` procedure returns all the vertices in V that are near z, where the nearness is parametrized by n. More precisely, for any $z \in X$, let $\text{Reach}(z, l) : \{z' \in X : \texttt{dist}(z, z') \leq l \vee \texttt{dist}(z', z) \leq l\}$. Given z and n, the distance threshold $l(n)$ is chosen in such a way that the set $\text{Reach}(z, l(n))$ contains a ball of volume $\gamma \log(n)/n$, where γ is an appropriate constant, and finally $\texttt{NearVertices}(G, z, n) = V \cap \text{Reach}(z, l(n))$.

Local Steering: Given two states $z_1, z_2 \in X$, the `Steer` procedure returns the optimal trajectory starting at z_1 and ending at z_2 in some local neighborhood. In other words, there exists a $\bar{\epsilon} > 0$ such that `Steer` procedure returns a trajectory $x : [0, T] \to X$, with $x(0) = z_1, x(T) = z_2$, the input $u : [0, T] \to U$ that drives the system along the trajectory x, and the time T, such that $J(\texttt{Steer}(z_1, z_2)) = \texttt{dist}(z_1, z_2)$, for all $\|z_1 - z_2\| \leq \bar{\epsilon}$.

Collision Check: Given a trajectory $x : [0, T] \to X$, the `ObstacleFree` procedure returns true iff x lies entirely in the obstacle-free space, i.e., $x(t) \in X_{\text{free}}$ holds for all $t \in [0, T]$.

The RRT* algorithm is given in Algorithms 1 and 2. Initialized with the tree that includes x_{init} as its only vertex and no edges, the algorithm iteratively builds a tree of collision-free trajectories by first sampling a state from the obstacle-free space (Line 4) and then extending the tree towards this sample (Line 5), at each iteration. The cost of the unique trajectory from the root vertex to a given vertex z is denoted as $\texttt{Cost}(z)$.

The `Extend` procedure is formalized in Algorithm 2. Notice that the algorithm first extends the nearest vertex towards the sample (Lines 2-4). The trajectory that extends the nearest vertex z_{near} towards the sample is denoted as x_{new}. The final state on the trajectory x_{new} is denoted as z_{new}. If x_{new} is collision free, z_{new} is added to the tree (Line 6) and its parent is decided as follows. First, the `NearVertices` procedure is invoked to determine the set

Z_{nearby} of near-by vertices around z_{new} (Line 8). Then, among the vertices in Z_{nearby}, the vertex that can be steered to z_{new} exactly incuring minimum cost to get to z_{new} is chosen as the parent (Lines 9-15). Once the new vertex z_{new} is inserted into the tree together with the edge connecting it to its parent, the extend operation also attempts to connect z_{new} to vertices that are already in the tree (Lines 16-21). That is, for any vertex z_{near} in Z_{nearby}, the algorithm attempts to steer z_{new} towards z_{near} (Line 17); if the steering procedure can exactly connect z_{new} and z_{near} with a collision-free trajectory that incurs cost less than the current cost of z_{near} (Line 18), then z_{new} is made the new parent of z_{near} (Lines 19-21), i.e., the vertex z_{near} is "rewired".

Algorithm 1: The RRT* Algorithm

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

Algorithm 2: The Extend Procedure

1 $V' \leftarrow V$; $E' \leftarrow E$;
2 $z_{\text{nearest}} \leftarrow \texttt{Nearest}(G, z)$;
3 $(x_{\text{new}}, u_{\text{new}}, T_{\text{new}}) \leftarrow \texttt{Steer}(z_{\text{nearest}}, z)$;
4 $z_{\text{new}} \leftarrow x_{\text{new}}(T_{\text{new}})$;
5 **if** $\texttt{ObstacleFree}(x_{\text{new}})$ **then**
6 $\quad V' := V' \cup \{z_{\text{new}}\}$;
7 $\quad z_{\min} \leftarrow z_{\text{nearest}}$; $c_{\min} \leftarrow \texttt{Cost}(z_{\text{new}})$;
8 $\quad Z_{\text{nearby}} \leftarrow \texttt{NearVertices}(G, z_{\text{new}}, |V|)$;
9 $\quad$ **for** *all* $z_{\text{near}} \in Z_{\text{nearby}}$ **do**
10 $\quad\quad (x_{\text{near}}, u_{\text{near}}, T_{\text{near}}) \leftarrow \texttt{Steer}(z_{\text{near}}, z_{\text{new}})$;
11 $\quad\quad$ **if** $\texttt{ObstacleFree}(x_{near})$ *and* $x_{\text{near}}(T_{\text{near}}) = z_{\text{new}}$ **then**
12 $\quad\quad\quad$ **if** $\texttt{Cost}(z_{\text{near}}) + J(x_{\text{near}}) < c_{\min}$ **then**
13 $\quad\quad\quad\quad c_{\min} \leftarrow \texttt{Cost}(z_{\text{near}}) + J(x_{\text{near}})$;
14 $\quad\quad\quad\quad z_{\min} \leftarrow z_{\text{near}}$;
15 $\quad E' \leftarrow E' \cup \{(z_{\min}, z_{\text{new}})\}$;
16 $\quad$ **for** *all* $z_{near} \in Z_{\text{nearby}} \setminus \{z_{\min}\}$ **do**
17 $\quad\quad (x_{\text{near}}, u_{\text{near}}, T_{\text{near}}) \leftarrow \texttt{Steer}(z_{\text{new}}, z_{\text{near}})$;
18 $\quad\quad$ **if** $x_{\text{near}}(T_{\text{near}}) = z_{\text{near}}$ *and* $\texttt{ObstacleFree}(x_{\text{near}})$ *and* $\texttt{Cost}(z_{\text{near}}) > \texttt{Cost}(z_{\text{new}}) + J(x_{\text{near}})$ **then**
19 $\quad\quad\quad z_{\text{parent}} \leftarrow \texttt{Parent}(z_{\text{near}})$;
20 $\quad\quad\quad E' \leftarrow E' \setminus \{(z_{\text{parent}}, z_{\text{near}})\}$;
21 $\quad\quad\quad E' \leftarrow E' \cup \{(z_{\text{new}}, z_{\text{near}})\}$;
22 **return** $G' = (V', E')$

IV. Asymptotic Optimality

In this section, we provide a set of sufficient conditions to guarantee asymptotic optimality of the RRT* algorithm for systems with differential constraints.

Let $\mathcal{B}_\epsilon(z)$ denote the closed ball centered at z, i.e., $\mathcal{B}_\epsilon(z) = \{z' \in X \mid \|z' - z\| \leq \epsilon\}$. Given two states $z, z' \in X$, let $\mathcal{X}_{z,z'}$ denote the set of all trajectories that start from z and reach z' through X. Note that the trajectories do not need to avoid the obstacles. Given a state $z \in Z$ and a constant $\epsilon > 0$, let $\mathcal{R}_\epsilon(z)$ denote the set of all states in X that are reachable from z with a path x that does not leave the ϵ-ball centered at z. More precisely, $\mathcal{R}_\epsilon(z) = \{z' \in Z \mid$ there exists $x \in \mathcal{X}_{z,z'}$ such that $x(t) \in \mathcal{B}_\epsilon(z)$ for all $t \in [0, T]\}$. $\mathcal{R}_\epsilon(z)$ is called the ϵ-reachable set of z. Any element of $\mathcal{R}_\epsilon(z)$ is said to be ϵ-reachable from z.

The following assumption characterizes local controllability of the system in a weak sense.

Assumption 3 (Weakened Local Controllability) [2] *There exist constants $\alpha, \bar{\epsilon} \in \mathbb{R}_{>0}$, $p \in \mathbb{N}$, such that for any $\epsilon \in (0, \bar{\epsilon})$, and any state $z \in X$, the set $\mathcal{R}_\epsilon(z)$ of all states that can be reached from z with a path that lies entirely inside the ϵ-ball centered at z, contains a ball of radius $\alpha\epsilon^p$.*

Assumption 3 holds, in particular, for locally controllable systems, hence, controllable linear systems. Moreover, many non-holonomic systems, including the Dubins vehicle, also satisfy this weakened version of the local controllability assumption.

Recall that a trajectory x is collision-free if it lies inside the obstacle-free space, i.e., for all $t \in [0, T]$, we have that $x(t) \in X_{\text{free}}$. Generalizing this notion, we say that a trajectory x is ϵ-collision-free, for $\epsilon \in \mathbb{R}_{>0}$, if for all states along the trajectory are at least ϵ away from the obstacle region, or, in other words, the ϵ-ball around any state along x lies entirely inside X_{free}, i.e., for all $t \in [0, T]$, there holds $\mathcal{B}_\epsilon(x(t)) \subset X_{\text{free}}$. Finally, we can state the assumption on the environment, i.e., the obstacle region, to ensure that there exists an optimal trajectory with enough free space around it to allow almost-sure convergence.

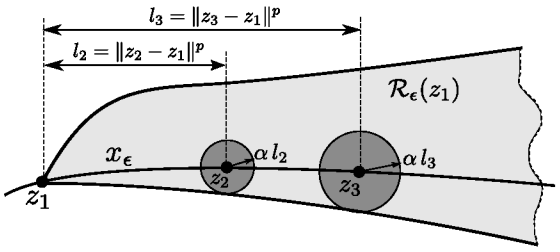
Assumption 4 (ϵ-collision-free Approximate Trajectories) *There exists an optimal feasible trajectory $x^* : [0, T^*] \to X_{\text{free}}$, constants $\bar{\epsilon}, \alpha \in \mathbb{R}_{>0}$, $p \in \mathbb{N}$, and a continuous function $q : \mathbb{R}_{>0} \to \mathcal{X}$ with $\lim_{\epsilon \downarrow 0} q(\epsilon) = x^*$ such that for all $\epsilon \in (0, \bar{\epsilon})$ the following hold for the path $x_\epsilon = q(\epsilon) : [0, T_\epsilon] \to X_{\text{free}}$:*

- *x_ϵ is an ϵ-collision-free path that starts from z_{init} and reaches the goal, i.e., $x_\epsilon(0) = z_{\text{init}}$ and $x_\epsilon(T) \in X_{\text{goal}}$,*
- *for any $t_1 < t_2$ such that $t_1 < t_2$, let $z_1 = x_\epsilon(t_1)$ and $z_2 = x_\epsilon(t_2)$, then the ball of radius $\alpha\|z_1 - z_2\|^p$ centered at z_2 is ϵ-reachable from z_1, i.e., $\mathcal{B}_{\alpha\|z_1 - z_2\|^p}(z_2) \subset \mathcal{R}_\epsilon(z_1)$ for some $p \geq 1$.*

Assumption 4 ensures the existence of an optimal path, which has enough obstacle-free space around it so as to allow almost-sure convergence. In particular, the function q ensures the existence of a class of paths, each of which starts from

[2] This assumption should not be confused with the weak local controllability assumption, e.g., in [21]. Indeed, neither assumption implies the other. However, any system that is locally controllable [21] satisfies Assumption 3.

the initial state, x_{init}, and reach the goal region. For any $\epsilon \leq \bar{\epsilon}$, this class includes a path, $r(\epsilon)$, which is ϵ-collision free. Moreover, each path $q(\epsilon)$ is such that for any two states z_1 and z_2 on this path, that correspond to time instance t_1 and t_2 with $t_1 < t_2$, we have that the $\alpha\|z_1 - z_2\|^p$-ball around z_2 is ϵ-reachable from z_1 (see Figure 1). Finally, as ϵ is varied from $\bar{\epsilon}$ to zero, $q(\epsilon)$ continuously turns the paths to the optimal path x^*.



(a)

Fig. 1. An illustration of an approximate path. The approximate path is denoted as x_ϵ. Two points along the path, namely z_2 and z_3 are shown along with their local neighborhood that has to lie inside the ϵ-reachable set of z_1 (shaded region).

Let us note the following remarks regarding assumption 4. First, the assumption guarantees the existence of an optimal trajectory that can be approximated. Due to the differential constraints of the dynamical system, merely allowing spacing between obstacles does not guarantee this condition. Second, the existence of the constants α and p also places extra assumptions on the dynamical system (namely, its controllability properties) as the dynamical system may not allow such trajectories.

Let $\mathcal{Y}_i^{RRT^*}$ denote the cost of the best path in the RRT* at the end of iteration i. Let c^* denote the cost of an optimal solution to Problem 2. The main result can be stated as follows.

Theorem 5 *Let Assumptions 3 and 4 hold and assume that Problem 1 admits a solution. Then, the cost of the best path in the RRT* converges to the optimal cost c^* almost surely, i.e.,* $\mathbb{P}\left(\left\{\lim_{i\to\infty} \mathcal{Y}_i^{RRT^*} = c^*\right\}\right) = 1$.

The proof of this theorem is similar to the proof of optimality in [19]. A sketch of the proof is provided in the appendix.

Using an analysis similar to the one provided in [19], it can also be shown that Algorithm 1 shares the theoretical properties of the RRT algorithm in terms of finding a feasible solution. More precisely, Algorithm 1 is probabilistically complete and the probability of failure decays to zero exponentially fast with the number of samples. We leave an analysis of the complexity of the algorithm to future work.

V. Example Systems

In this section, we consider three dynamical systems, each widely used in various robotics applications, and provide and simple and intuituve steering procedures that satisfy the assumptions put forward in the previous section. The dynamical systems discussed in this section all satisfy Assumptions 3, but we omit the proof of this statement due to space limitations. Moreover, optimal steering laws are available for the class of problems we will consider, i.e., minimum-time problems, in which $g(x) = 1$ in the cost functional $J(x) = \int_0^T g(x(t))\, dt$.

A. Dubins' vehicle

1) System dynamics: Dynamics of the Dubins' vehicle is described by the following nonlinear differential equation:

$$\begin{aligned} \dot{x}_D &= v_D \cos(\theta_D) \\ \dot{y}_D &= v_D \sin(\theta_D) \\ \dot{\theta}_D &= u_D, \qquad |u_D| \leq \frac{v_D}{\rho}, \end{aligned} \tag{2}$$

where x_D and y_D are the cartesian coordinates of the vehicle, θ_D is the heading angle, u_D is the steering input, v_D is the speed, ρ is the minimum turning radius. We will assume that the speed v_D of the vehicle as well the minimum turning radius ρ are constants. The variables x_D, y_D, and θ_D constitute the state variables; the steering input u_D, on the other hand, describes the input to the system. Hence, The state-space X and the input space U are, then, subsets of the 3-dimensional and the 1-dimensional Euclidean space, respectively. More precisely, we have that $z = (x_D, y_D, \theta_D) \in X$ and $w = (v_D) \in U$. Notice that the system is under-actuated.

2) Steering procedure: Given two states $z_1, z_2 \in X$ for the Dubins' vehicle, it is well known that the optimal path(s) to drive the system from z_1 to z_2 can be parameterized by six families of canonical paths [22]. Indeed, if the steering procedure is designed to choose the minimum-length path among these six canonical paths, the steering procedure is optimal globally. However, in the rest of this section, we outline a simpler steering procedure, which also satisfies the conditions. This "locally optimal" steering procedure is a simplified version of the globally optimal procedure given in [22].

(a) (b) (c) (d)

Fig. 2. Four Dubins curves that describe the steering procedure. The figures (a), (b), (c), and (d) illustrate RSR, RSL, LSL, LSR paths, respectively. The local steering procedure picks the trajectory with minimum length among these three curves (in this case, the curve in Figure (a) has minimal length).

It was shown in [22] that the optimal path between two states that the optimal path that drives a vehicle described by the dynamics given in Equation 2 is either one of six different types of paths. This class of paths consists of paths that either turn with maximum steering or go straight. This six different path types are denoted as RSL, LSR, RSR, LSL, RLR, and LRL, where each letter L means left, R means right, and S means straight. When the system is executing, for instance, the RSL path, it first turns right with minimum turning

radius, then proceeds straight, and finally turns left. Other path types are defined similarly. Given two configurations, any minimal path of this type, if it exists, is unique. The main result of Dubins in [22] is that the optimal path that drives the system from an initial state to a final state is one of these six paths.

For the steering procedure of this section, we restrict ourselves to only four types of paths, namely RSL, LSR, RSR, LSL (excluding RLR and LRL), which are all illustrated in Figure 2.

B. 2D Double Integrator

1) Dynamics: The double integrator dynamics is described by the following differential equation with four states:

$$\dot{x} = v_x, \quad \dot{v_x} = a_x, \quad \dot{y} = v_y, \quad \dot{v_y} = a_y,$$

where $|a_x| \leq 1$ and $|a_y| \leq 1$.

2) Steering procedure: We use the steering procedure put forward in [23]. Informally speaking, the procedure works as follows. Let $z = (x, y, v_x, v_y)$ and $z' = (x', y', v'_x, v'_y)$ be two states. The `Steer` procedure first computes the time optimal control for both of the axes individually, i.e., computes the optimal control to reach from (x, v_x) to (x', v'_x) and also from (y, v_y) to (y', v'_y). Let t_x be the time that the first optimal control reaches (x', v'_x) and let t_y be defined similarly. Without loss of any generality assume $t_x \leq t_y$. Then, it fixes the control for the y-axis, and computes a control for the x-axis that reaches (x', v'_x) in exactly t_y amount of time by computing the time optimal trajectory when $|a_y| \leq \bar{a}$ using a binary search over $\bar{a}$ (see [23] for details).

C. A simple 3D airplane model

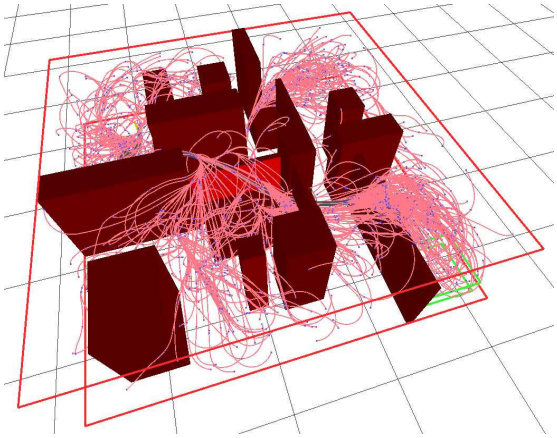
1) Dynamics: In the next section, we also present results for a simplified 3D airplane model, which consists of a Dubins' car on the plane and a double integrator for the altitude dynamics. This model has five states in total.

2) Steering procedure: The `Steer` procedure independently determines the trajectory on the plane, and the trajectory on the vertical axis. If the amount of time required to execute the former is more than the amount of time required to execute the latter, then the `Steer` procedure uses binary search as in the previous section to find a trajectory for the altitude.

VI. Simulations

This section is devoted to simulation examples involving dynamical systems described in the previous section.

First, a system with the Dubins' vehicle dynamics was considered. RRT and RRT* algorithms were also run in an environment including obstacles and tree maintained by the RRT* algorithm was shown in Figures 3(a)-3(c) in different stages; for comparison, the tree maintained by the RRT algorithm is shown in Figure 3(d). The RRT* algorithm achieved the tree in Figures 3(a) and 3(c) in 0.1 and 5 seconds of computation time, respectively.



(a)

Fig. 4. RRT* algorithm after 5000 iterations in an environment with obstacles. The example took around 7 seconds to compute.

Second, a system with a 2D double integrator dynamics was considered. The RRT* algorithm is shown in different stages in an environment with obstacles in Figures 3(e)-3(g). The tree maintained by the RRT algorithm, on the other hand, is shown in Figure 3(h). The RRT* algorithm achieved the tree shown in Figures 3(e) and 3(g) in around 0.2 and 10 seconds, respectively. The RRT algorithm achieved the tree shown in Figure 3(h) in around 2 seconds of computation time. The cost of the best path in the tree by the RRT* shown in Figure 3(g) was 27 time units, as opposed to the cost of the best path in the tree maintained by the RRT shown in Figure 3(h), which was 55 time units.

Third, the simple 3D airplane model is considered. The tree maintained by the RRT* algorithm is shown in an environment cluttered with obstacles resembling an urban city environment is shown in Figure 4.

The simulations clearly illustrate the difference between the quality of the solutions returned by the RRT and RRT* algorithms. Most often, RRT* is able to provide significant improvement in quality with no substantial extra computational effort. The RRT algorithm, on the other hand, generally gets stuck with the first solution found and is unable to improve the solution; such behavior is well known empirically and is at the basis of the negative result in [19], on the almost-sure non-optimality of RRT. In most of the cases considered in this paper, the RRT* algorithm is able to get close to the optimal solution within reasonable computation time (all the simulations presented in this paper took no more than 10 seconds to compute on a laptop computer).

VII. Conclusion

In this paper, the optimal kinodynamic motion planning problem was considered. The RRT* algorithm was extended to handle kinodynamic constraints. In particular, sufficient conditions on the system dynamics, local steering function, and free-space to guarantee asymptotic optimality were provided. The effectiveness of the algorithm was also shown

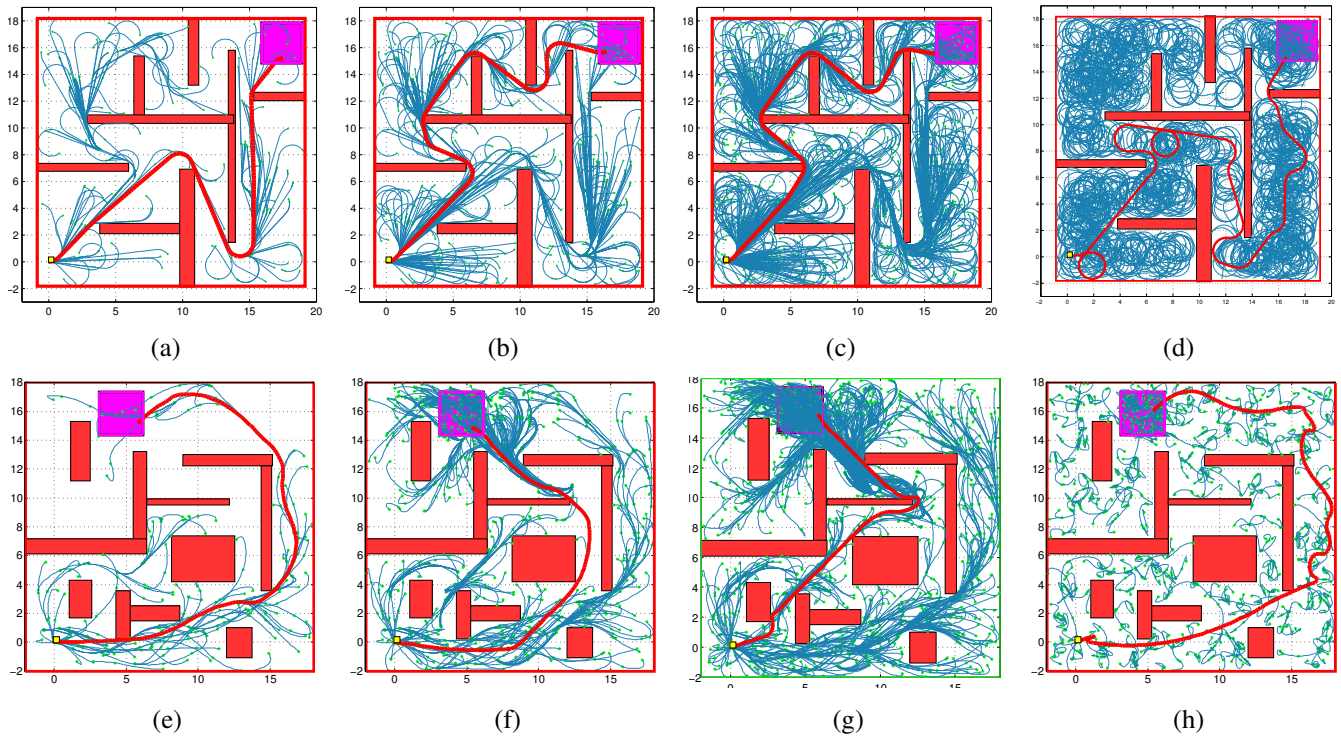


Fig. 3. RRT and RRT* algorithms were run in various environments for a dynamical system with Dubins' vehicle dynamics as well as one with a 2D double integrator dynamics. In Figures 3(a) and 3(c), the tree maintained by the RRT* algorithm is shown when including around 500 and 6500 vertices, respectively. In Figure 3(d), the tree maintained by the RRT algorithm is shown when including around 2000 vertices. In Figures 3(e) and 3(g), the tree maintained by the RRT* algorithm is shown right after 300, and 1500 iterations, respectively. The tree maintained by the RRT algorithm right after 1500 iterations is shown in Figure 3(h) for comparison.

via simulations. The computational complexity of the algorithms, e.g., with respect to that of the baseline RRT has not been explicitly addressed in this paper. We conjecture that the RRT* computational overhead per iteration is still within a constant, independent of the tree size. However, we have not analyzed this issue in detail, and will leave it to future work. Another point we are investigating is the relaxation of some of the requirements on the primitive procedures. For example, it may not be necessary to have an optimal steering procedure, but an approximately optimal procedure may suffice. Finally, future work will include applying the techniques outlined in this paper to high-dimensional dynamical systems, and practical demonstration on robotic platforms.

ACKNOWLEDGMENTS

This research was supported in part by the Michigan/AFRL Collaborative Center on Control Sciences, AFOSR grant no. FA 8650-07-2-3744.

REFERENCES

[1] S. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.

[2] R. Alterovitz, M. Branicky, and K. Goldberg. Motion planning under uncertainty for image-guided medical needle steering. *International Journal of Robotics Research*, 27:1361–1374, 2008.

[3] J.C. Latombe. *Robot Motion Planning*. Kluwer Academic, 1990.

[4] 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.

[5] M. Likhachev and D. Ferguson. Planning long dynamically-feasible maneuvers for autonomous vehicles. *International Journal of Robotics Research*, 28(8):933–945, 2009.

[6] 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.

[7] 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.

[8] J. Latombe. Motion planning: A journey of robots, molecules, digital actors, and other artifacts. *International Journal of Robotics Research*, 18(11):1119–1128, 1999.

[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] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics Research*, 5(1):90–98, 1986.

[12] L. Kavraki and J. Latombe. Randomized preprocessing of configuration space for fast path planning. In *IEEE International Conference on Robotics and Automation*, 1994.

[13] 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.

[14] S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning.

International Journal of Robotics Research, 20(5):378–400, May 2001.

[15] 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.

[16] 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.

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

[18] 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.

[19] S. Karaman and E. Frazzoli. Incremental sampling-based optimal motion planning. In *Robotics: Science and Systems (RSS)*, 2010.

[20] D. P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, 3rd edition, 2005.

[21] R. Hermann and A. J. Krener. Nonlinear controllability and observability. *IEEE Transactions on Automatic Control*, 22(5):728–740, 1977.

[22] L.E. Dubins. On the curves of minimal length on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, 79(3):497–516, 1957.

[23] E. Frazzoli, M. Dahleh, and E. Feron. Real-time motion planning for agile autonomous vehicles. *Journal of Guidance, Control, and Dynamics*, 25(1):116–129, 2002.

[24] H. K. Khalil. *Nonlinear Systems*. Prentice-Hall, Englewood Cliffs, NJ, second edition, 1995.

APPENDIX

Proof of Theorem 5 (Sketch)

The proof of this theorem is very similar to that given in [19]; however, some details require careful examination.

By Assumption 4, there exists a family q of approximate trajectories such that $q(\epsilon)$ is ϵ-collision-free for all $\epsilon > 0$. Let $i \in \mathbb{N}$. As in [19], fix a sequence $\{\epsilon_i\}_{i\in\mathbb{N}}$ and consider the ϵ_i approximate trajectory $q(\epsilon_i)$. Tile this trajectory with overlapping balls of radius ϵ such that ball centers are separated by a distance $l^{1/p}$, where $l = \beta\,\epsilon_i$ for some constant β, independent of i. The constant p was defined in Assumption 4 (see Figure 5 for an illustration).

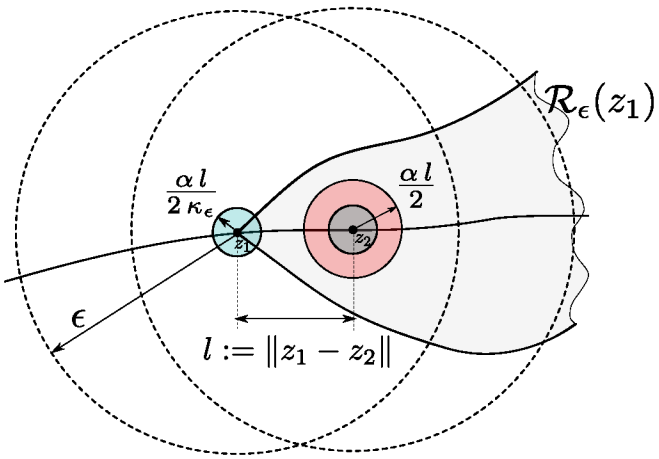
Consider two adjacent balls, which are centered at, say z_1 and z_2, respectively. By Assumption 3, the ϵ-reachable set of z_1 has nonempty interior, i.e., has positive volume in terms of the usual Lebesque measure, and hence has positive probability of being sampled at a particular iteration. Recall that, by Assumption 4, there exists a constant α such that a ball of radius $\alpha\|z_1 - z_2\|^p = \alpha\, l$ centered at z_2 is reachable from z_1 (since, by definition, $l = \|z_1 - z_2\|^p$).

By the Lipschitz continuity of f, these exists a constant κ such that any state inside a ball of radius $\frac{\alpha\, l}{2\kappa}$ can be connected to any state inside the ball of radius $\frac{\alpha\, l}{2}$ centered at z_2 [24] and, in particular, to any state that is inside a ball of radius $\frac{\alpha\, l}{2\kappa}$ (we assume without loss of generality that $\kappa > 1$, otherwise taking $\kappa = 1$ would suffice to argue similarly).

Let us note two facts at this point. First, the volume of each such ball is $V_i := \zeta_d\,(\frac{\alpha\, l}{2\kappa})^d$, where ζ_d is the volume of a unit sphere in d dimensions. Second, if the approximate trajectory had length L_i, then the number of such balls would be at most $\frac{L_i}{l^{1/p}} = \frac{L_i}{(\beta\,\epsilon_i)^{1/p}}$.

If the RRT* algorithm had sampled vertices inside all such balls of radius $\frac{\alpha\, l}{2\kappa}$ by iteration i, where i is large enough and is such that $\gamma\frac{\log i}{i} = V_i$, then the RRT* would necessarily connect the samples to one another to build path that has cost that is close to the cost of the approximately optimal trajectory $q(\epsilon)$. For this step, we also use the fact that the cost functional is the line integral of a Lipschitz function.

Let A_i denote the event that the RRT* fails to include a vertex in at least one such ball by the end of iteration i. In the limit as $i \to \infty$, $\mathbb{P}(A_i)$ can be upper-bounded by the number of balls that cover the trajectory $q(\epsilon_i)$ times the probability that one ball does not include a vertex, i.e., $\mathbb{P}(A_i) \leq \frac{L_i}{\alpha\gamma_1 l^{1/p}}\left(1 - \gamma_2\frac{\log i}{i}\right)^i$, where γ_1, γ_2 are constants. Noting that $\frac{\alpha l}{2} = \gamma\left(\frac{\log i}{i}\right)^{1/d}$, the following holds: $\mathbb{P}(A_i) \leq \frac{L_i}{\alpha\gamma_1(\frac{2}{\alpha}\frac{\log i}{i})^{1/p}}\left(1 - \gamma_2\frac{\log i}{i}\right)^i \leq \frac{L_i}{2\gamma_1\alpha^{1-1/p}} i^{-(\gamma_2 - 1/p)}$. Note that $\sum_{i=1}^{\infty}\mathbb{P}(A_i) < \infty$ for large enough γ_2, hence large enough γ. Then, by the Borel-Cantelli lemma, the event A_i can occur only finitely often with probability one. Hence, the RRT* algorithm generates an approximately optimal trajectory from $q(\epsilon_i)$ infinitely often with probability one.



(a)

Fig. 5. Illustration of ball covering of two adjacent vertices, namely z_1 and z_2, that lie on an approximately optimal ϵ-collision-free trajectory. Balls of radius ϵ are shown with dotted lines. The ϵ-reachable set of z_1 is shaded in light grey. The balls that are centered at z_1 and z_2 and have raidus $\frac{\alpha l}{2\,\kappa_\epsilon}$ are shaded in blue. The ball of radius $\frac{\alpha l}{2}$ is shaded in light red.