Anytime Computation of Time-Optimal Off-Road Vehicle Maneuvers using the RRT*

Jeong hwan Jeon Sertac Karaman Emilio Frazzoli

***Abstract*— Incremental sampling-based motion planning algorithms such as the Rapidly-exploring Random Trees (RRTs) have been successful in efficiently solving computationally challenging motion planning problems involving complex dynamical systems. A recently proposed algorithm, called the RRT*, also provides asymptotic optimality guarantees, i.e., almost-sure convergence to optimal trajectories (which the RRT algorithm lacked) while maintaining the computational efficiency of the RRT algorithm. In this paper, time-optimal maneuvers for a high-speed off-road vehicle taking tight turns on a loose surface are studied using the RRT* algorithm. Our simulation results show that the aggressive skidding maneuver, usually called the trail-braking maneuver, naturally emerges from the RRT* algorithm as the minimum-time trajectory. Along the way, we extend the RRT* algorithm to handle complex dynamical systems, such as those that are described by nonlinear differential equations and involve high-dimensional state spaces, which may be of independent interest. We also exploit the RRT* as an anytime computation framework for nonlinear optimization problems.**

I. Introduction

There have been attempts to analyze and reproduce specialized human driving techniques, e.g., a minimum-time lane change, a minimum lap-time trajectory, a trail-braking maneuver, using optimal control theory [1], usually based on numerical optimization methods [2]–[5]. Although earlier work focused on posing the problem as an optimization over a sequence of steady-state trim conditions, more recently transient phases of extreme operating conditions were also taken into account using high-fidelity modeling [3].

Although these approaches are successful in describing and realizing certain aspects of the said vehicle maneuvers, the computation is usually carried out offline with careful transcription to numerical optimization formulations. Most algorithms of this class must be started with a feasible initial solution, which is usually hard to generate in the first place. Moreover, some numerical optimization methods suffer from local optimality, except for few unrealistic problem instances. While handling differential constraints is efficient in the most nonlinear programming methods (e.g., in a collocation-based algorithm), imposing geometrical constraints in configuration space, e.g., road boundaries, turns out to be challenging [6].

In this paper, we use motion planning methods to generate optimal trajectories for minimum-time maneuvering of high-speed off-road vehicles. Given an initial state, a goal set, an obstacle set, and a description of the system dynamics, the motion planning problem is to find a control input that drives the system from the initial state to the goal set while avoiding collision with obstacles. An algorithm that solves this problem is said to be *complete*, if it returns such a control input when one exists and returns failure otherwise.

This problem of navigating through a complex environment is one of the fundamental problems in robotics [7] with applications including, but not limited to, autonomous driving [8], manipulation planing [9], logistics [10], and medical surgery [11]. The motion planning problem also has several applications outside the domain of robotics, ranging from verification to computational biology [12]–[14].

Although the motion planning problem is interesting from a practical perspective, the problem is known to be computationally challenging. In fact, a simple version of the problem, referred to as the *Piano Mover's Problem*, was proven to be PSPACE-hard [15], which implies that complete algorithms are doomed to suffer from computational complexity.

Designing computationally-efficient motion planning algorithms that can handle systems with high-dimensional state spaces has been a long-standing challenge for several decades. Particularly tailored for this setting, sampling-based algorithms such as the Probabilistic RoadMaps (PRMs) [16] have achieved great success in computationally challenging instances of the motion planning problem. Later, incremental sampling-based algorithms such as the Rapidly-Exploring Random Trees (RRTs) [17] have been proposed to address planning for systems also with differential constraints. Most sampling-based methods, including the RRT, achieve the computational efficiency by relaxing the completeness of requirements to *probabilistic completeness*, meaning that the probability of finding a solution, if one exists, converges to one as the number of samples approaches infinity.

Although the RRT algorithm efficiently finds solutions in many challenging problems, the solutions provably remain suboptimal, even if the algorithm is provided with infinite computation time [18]. As a modification, the RRT* algorithm was proposed to ensure asymptotic optimality of the solutions while maintaining the probabilistic completeness and the computational efficiency of the RRT [18]. Subsequently, preliminary work on extending the RRT* algorithm to handle systems with differential constraints has appeared in [19], which discussed applications to systems such as the double integrator and Dubins' vehicle dynamics.

This paper focuses on using the RRT* algorithm to solve the optimal kinodynamic motion planning with probabilistic guarantees for complex dynamical systems with high-dimensional state spaces. Systems of this nature include those that have high maneuvering capabilities, such as race

The authors are with the Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA 02139, USA {jhjeon, sertac, frazzoli}@mit.edu

cars and aerobatic airplanes. Earlier work in [19] had assumed that the existence of a "steering" function that can find a control input that exactly connects an initial state to a final state, which is usually non-trivial to construct for such systems. Our first contribution is to extend the RRT* algorithm by relaxing this assumption to allow "approximate" steering functions. Second, we interpret the RRT* as an anytime computation framework for optimization problems with complex differential and geometric constraints. Third, using the resulting algorithm, we numerically analyze time-optimal maneuvers for a high-speed off-road vehicle.

Even though our current implementation is not amenable to real-time computation on commercially-available personal computers such as laptops, our results are promising in that they may provide guidelines in implementing the RRT* algorithm on future or dedicated parallel-computation devices towards realizing high-speed robotic race cars.

This paper is organized as follows. Sections II and III introduce the optimal motion planning problem and the RRT* algorithm, respectively. In Section IV, several extensions of the RRT* algorithm are introduced to handle systems with differential constraints and high-dimensional state spaces. Implementation details for an off-road rally car dynamics are provided in Section V, and simulation results are discussed in Section VI. The paper is concluded in Section VII.

II. Problem Description

Let $X \subset \mathbb{R}^n$ and $U \subset \mathbb{R}^m$ be compact sets. Let $z_0 \in X$ and consider the following time-invariant dynamical system:

$$\dot{x}(t) = f(x(t), u(t)), \qquad x(0) = z_0, \tag{1}$$

where $x(t) \in X$, $u(t) \in U$, and f is a continuously differentiable function with respect to both of its variables. Let $\mathcal{X}$ and $\mathcal{U}$ denote the set of all essentially bounded measurable functions defined from $[0, T]$ to X and U, respectively, for all real numbers $T > 0$. The sets $\mathcal{X}$ and $\mathcal{U}$ are called the *trajectories* and the *controls*, respectively.

Let $X_{\rm obs}$, $X_{\rm goal} \subset X$, called the *obstacle region* and the *goal region*, respectively, be open sets, and define the *obstacle-free region* as $X_{\rm free} := X \setminus X_{\rm obs}$. Let $g : X \to \mathbb{R}$ be a Lipschitz continuous function that is bounded away from zero, i.e., $\inf_{z \in X} g(z) > 0$. Using a *cost functional* g that associates each trajectory $x \in \mathcal{X}$ with a non-zero cost can be defined as the line integral of $g(x(t))$ over the interval $[0, T]$ that x is defined, i.e., $J_g(x) := \int_0^T g(x(t))dt$.

Problem 1 (Optimal Kinodynamic Motion Planning) *Given a state space X, an obstacle region $X_{\rm obs}$, a goal region $X_{\rm goal}$, an initial state $z_0 \in X$, a dynamical system described by a differential equation as in Equation (1), and a cost functional $J_g : \mathcal{X} \to \mathrm{R}_{>0}$, find a control $u \in \mathcal{U}$ with domain $[0, T]$ such that the unique trajectory $x \in \mathcal{X}$ that satisfies Equation (1) for all $t \in [0, T]$,*

- *avoids collision, i.e., $x(t) \notin X_{\rm obs}, \forall t \in [0, T]$,*
- *reaches the goal region, i.e., $x(T) \in X_{\rm goal}$,*
- *and minimizes the cost functional $J_g(x)$.*

III. RRT* Algorithm

The RRT* algorithm [18] is an incremental sampling-based motion planning algorithm for planning in configuration spaces, and extended to handle more complex dynamics in [19]. In this section, the RRT* algorithm is introduced as described in [19] after slight modifications.

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

Sampling: The sampling procedure $\texttt{Sample} : \mathbb{N} \to X_{\rm free}$ returns independent and identically distributed (i.i.d.) samples from the obstacle-free set. For simlicity, we assume that the sampling distribution is uniform, even though our results hold for a large class of sampling strategies.

Distance metric: 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 $X_{\rm free}$ 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 metric 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 $X_{\rm free}$, 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 $\mathrm{Reach}(z, l) : \{z' \in X : \texttt{dist}(z, z') \leq l\}$. Given z and n, the distance threshold $l(n)$ is chosen in such a way that the set $\mathrm{Reach}(z, l(n))$ contains a ball of volume $\gamma \log(n)/n$, where γ is an appropriate constant, and finally $\texttt{NearVertices}(G, z, n) = V \cap \mathrm{Reach}(z, l(n))$.

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

The RRT* algorithm is given in Algorithms 1 and 2. Initialized with the tree that includes $z_{\rm init}$ being its only vertex, RRT* 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. The algorithm first extends the nearest vertex towards the sample (Lines 2-4). The trajectory that extends the nearest vertex $z_{\rm nearest}$ towards the sample is denoted as $x_{\rm new}$. The final state on the trajectory $x_{\rm new}$ is denoted as $z_{\rm new}$. If $x_{\rm new}$ is collision free, $z_{\rm new}$ 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_{near} of near-by vertices around z_{new} (Line 8). Then, among the vertices in Z_{near}, the vertex that can be steered to z_{new} exactly with minimum cost is chosen as the parent (Lines 9-14). 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 15-20) as follows. For any vertex z_{near} in Z_{near}, the algorithm attempts to steer z_{new} towards z_{near} (Line 16); if the steering procedure can connect z_{new} and z_{near} with a collision-free trajectory that incurs cost less than the current cost of z_{near} (Line 17), then z_{new} is made the new parent of z_{near} (Lines 18-20), i.e., the vertex z_{near} is "rewired".

The RRT* algorithm guarantees asymptotic optimality, i.e., almost-sure convergence to optimal trajectories, while ensuring effective use of computational resources [18], [19].

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 $\quad V' \leftarrow V' \cup \{z_{\text{new}}\}$;
7 $\quad z_{\min} \leftarrow z_{\text{nearest}}$; $J_{\min} \leftarrow \texttt{Cost}(z_{\text{new}})$;
8 $\quad Z_{\text{near}} \leftarrow \texttt{NearVertices}(G, z_{\text{new}}, |V|)$;
9 $\quad$ **for** *all* $z_{\text{near}} \in Z_{\text{near}}$ **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** $x_{\text{near}}(T_{\text{near}}) = z_{\text{new}}$ *and* $\texttt{ObstacleFree}(x_{\text{near}})$ *and* $\texttt{Cost}(z_{\text{near}}) + J(x_{\text{near}}) < J_{\min}$ **then**
12 $\quad\quad\quad J_{\min} \leftarrow \texttt{Cost}(z_{\text{near}}) + J(x_{\text{near}})$;
13 $\quad\quad\quad z_{\min} \leftarrow z_{\text{near}}$;
14 $\quad E' \leftarrow E' \cup \{(z_{\min}, z_{\text{new}})\}$;
15 $\quad$ **for** *all* $z_{near} \in Z_{\text{near}} \setminus \{z_{\min}\}$ **do**
16 $\quad\quad (x_{\text{near}}, u_{\text{near}}, T_{\text{near}}) \leftarrow \texttt{Steer}(z_{\text{new}}, z_{\text{near}})$;
17 $\quad\quad$ **if** $x_{\text{near}}(T_{\text{near}}) = z_{\text{near}}$ *and* $\texttt{ObstacleFree}(x_{\text{near}})$ *and* $\texttt{Cost}(z_{\text{new}}) + J(x_{\text{near}}) < \texttt{Cost}(z_{\text{near}})$ **then**
18 $\quad\quad\quad z_{\text{parent}} \leftarrow \texttt{Parent}(z_{\text{near}})$;
19 $\quad\quad\quad E' \leftarrow E' \setminus \{(z_{\text{parent}}, z_{\text{near}})\}$;
20 $\quad\quad\quad E' \leftarrow E' \cup \{(z_{\text{new}}, z_{\text{near}})\}$;
21 **return** $G' = (V', E')$

IV. Extensions

This section discusses several extensions to the RRT* algorithm to allow planning for systems with complex differential constraints involving high-dimensional state spaces.

A. Task Space Planning

To effectively deal with high-dimensional state spaces, the RRT* algorithm operates in a task space, which has a smaller dimension when compared to the state space (see [20] for a discussion on task spaces). More precisely, let $T \subset \mathbb{R}^{m'}$ be a compact set. Recall that the set U of inputs is a subset of the m-dimensional Euclidean space. Usually, m' is much smaller than m. For practical purposes, it is tacitly assumed that there exists a surjective mapping $\mathcal{T} : X_{\text{free}} \to T$ that maps each collision-free state to its equivalent in the task space. Moreover, the Sample procedure returns i.i.d. random samples from the task space, and the steering procedure operates in the task space as explained below.

B. Steering Procedure

In the description of the RRT* algorithm, it is assumed that for any given $z_1, z_2 \in X_{\text{free}}$, the $\texttt{Steer}(z_1, z_2)$ procedure returns an *optimal* trajectory that starts from z_1 and reaches z_2 exactly, when such a trajectory exists.

Finding a trajectory that connects z_1 and z_2 in this manner may be computationally challenging, since it amounts to solving a two-point boundary value problem for an ordinary differential equation. For certain dynamical systems, e.g., single integrator, a double integrator, or a curvature-constrained car (i.e., Dubins' vehicle) [21], analytical solutions to this boundary value problem do exist [21], [22]. However, an analytical solution to this problem is not available for most dynamical systems [23].

In what follows, we provide the implementation details of a steering procedure that is approximate in the following sense: the trajectory $x : [0, T] \to X_{\text{free}}$ generated by the $\texttt{Steer}(z_1, z_2)$ procedure is such that (i) x starts at z_1, i.e., $x(0) = z_1$, (ii) reaches a neighborhood of z_2 in the task space, i.e., there exists some $\delta \geq 0$ such that $\mathcal{T}(x(T)) \in B(\mathcal{T}(z_2); \delta)$[1], and (iii) has cost $c^* + \epsilon$, where c^* is the cost of the minimum-time trajectory that starts from z_1 and reaches z_2. Here, the parameters δ and ϵ are bounds on the connection error and sub-optimality of the trajectory, respectively.

Our steering function is based on numerical methods for solving differential equations [24] described in detail below.

1) Piecewise-constant Input: The steering function considers only constant inputs. More precisely, the trajectory $x : [0, T] \to X_{\text{free}}$ returned by $\texttt{Steer}(z_1, z_2)$ is such that $\dot{x} = f(x(t), \bar{u})$ and $x(0) = z_1$ for some $\bar{u} \in U$.

2) Shooting Method: The steering procedure proposed in this paper is based on the shooting method [24], which can be used with either the bisection method or the false position method in order to determine the constant input value for local steering. If the method does not converge within a given number of iterations, the connection is regarded as infeasible.

Before providing the shooting method, let us note the following definitions. Given a constant input $\bar{u} \in U$ and an initial state $\bar{z} \in X$, let $x(t; \bar{z}, \bar{u})$ denote the resulting trajectory of the dynamical system when started from initial state

[1] In the sequel, for a subset A of the d-dimensional Euclidean space, the set $B(a, r) \subset A$ denotes the closed ball of radius $r \in \mathbb{R}_{\geq 0}$ centered at $a \in A$, i.e., $B(a, r) := \{a' \in A \mid \|a' - a\| \leq r\}$.

Algorithm 3: The `Steer` procedure based on shooting

1 $\bar{u} \leftarrow 0$;
2 **for** $i = 1$ *to* M **do**
3 $\quad \bar{t} \leftarrow \inf_{t \in [0,\infty)} \{t \,|\, x(t; z_1, \bar{u}) \in \mathrm{Term}(\mathcal{T}(z_2))\}$;
4 $\quad \bar{z} \leftarrow x(\bar{t}; z_1, \bar{u})$;
5 $\quad$ **if** $\mathcal{T}(\bar{z}) \in B(\mathcal{T}(z_2); \delta)$ **then**
6 $\quad\quad$ **return** $x([0, \bar{t}]; z_1, \bar{u})$;
7 $\quad$ **else**
8 $\quad\quad \bar{u} \leftarrow \bar{u} + \texttt{Bisect}(z_1, z_2, \bar{z}, \bar{u})$;

$\bar{z}$ under constant input $\bar{u}$, i.e., $\dot{x}(t; \bar{z}, \bar{u}) = f(x(t; \bar{z}, \bar{u}), \bar{u})$ for all $t \in [0, \infty)$ and $x(0, \bar{z}, \bar{u}) = \bar{z}$. Given some $\bar{t} \in \mathbb{R}_{\geq 0}$, let $x([0, \bar{t}], \bar{z}, \bar{u})$ denote the restriction of the trajectory $x(\cdot; \bar{z}, \bar{u})$ to the interval $[0, \bar{t}]$. Let $\texttt{Term} : T \rightarrow \{\texttt{false}, \texttt{true}\}$ denote a termination condition that associates a terminal condition with each task space state. Finally, let $\texttt{Bisect} : (z_1, z_2, z_3, u) \rightarrow u'$ be a bisection algorithm that returns an increment for the input so as to steer the terminal state of the dynamical system towards z_2, when the dynamical system is started at state z_1 and the current constant input u under consideration steers the system to state z_3.

The shooting method is given in Algorithm 3. The algorithm initially starts with the zero input (Line 1). The state where the trajectory of the system starting from z_1 enters $\texttt{Term}(\mathcal{T}(z_2))$, i.e., the terminal set associated with z_2, is determined in Lines 3-4. This state is denoted as $\bar{z}$. If $\bar{z}$ is within the δ-neighborhood of $\mathcal{T}(z_2)$, then the algorithm returns the trajectory that reaches $\bar{z}$ (Line 6). Otherwise, a new input is selected using the `Bisect` function (Line 8), and the procedure is continued until the maximum number of iterations (denoted as M in Algorithm 3) is reached.

3) Repropagation: Since the steering function is approximate, i.e., can only reach a neighborhood of the final state, the rewiring procedure of the RRT* cannot be implemented directly as errors induced by the steering function in the rewiring step causes inconsistency in the states of descendant nodes. To handle this issue, corresponding descendant nodes are re-simulated using the stored sequences of inputs.

Before presenting the repropagation algorithm, let us note the following definitions. Recall that (V, E) denotes the graph maintained by the RRT* algorithm. Given a vertex $z \in V$ in the tree, let $\texttt{Children}(z)$ denote the set of all children vertices of z, i.e., $\texttt{Children}(z) := \{z' \in V \,|\, (z, z') \in E\}$. Given an edge $e = (z_1, z_2) \in E$, let $\texttt{Input}(e)$ denote the input that drives the dynamical system from state z_1 to state z_2, and $\texttt{Time}(e)$ denote the time it takes for this trajectory to reach z_2 starting from z_1. Clearly, `Children`, `Input`, and `Time` functions can be populated incrementally as the RRT* algorithm proceeds. Hence, these functions can be evaluated quickly, e.g., without re-running the `Steer` procedure.

The repropagation procedure is given in Algorithm 4, called whenever the extension towards an existing node $z \in V$ reaches $\bar{z} \neq z$. This procedure recursively calculates the new state, denoted by z_{new}, for all the descendants of z.

Clearly the re-propagation may render some marginally-safe trajectories collide with adjacent obstacles. However, the authors have observed in experiments that the tree quickly recovers the lost edges with better trajectories.

Algorithm 4: The $\texttt{Repropagate}(z, \bar{z})$ procedure

1 **for** *all* $z' \in \texttt{Children}(z)$ **do**
2 $\quad z'_{\mathrm{new}} \leftarrow x(\texttt{Time}((z, z')); \bar{z}, \texttt{Input}((z, z')))$;
3 $\quad \texttt{Repropagate}(z', z'_{\mathrm{new}})$;
4 $V \leftarrow V \setminus \{z\}$;
5 $V \leftarrow V \cup \{z_{\mathrm{new}}\}$;

C. Conditional Activation of the RRT*

In order to find a feasible solution quickly, we run the RRT algorithm until the algorithm returns a feasible solution. Once a feasible solution is obtained, the RRT* algorithm is run as is. More precisely, until a feasible solution is found, the `NearVertices` procedure returns the empty set.

D. Branch-and-Bound

Branch-and-bound algorithm [25] is a widely-known technique in combinatorial optimization. It is often used in robotics in conjunction with graph search algorithms [26], [27]. A recent application of the branch-and-bound on the RRT* can be found in [28] as well.

An *admissible heuristic*, or a *cost-to-go function* is a function that maps each state $z \in X_{\mathrm{free}}$ to a non-negative real number that is less than or equal to the cost of the optimal trajectory that reaches the goal starting from z (taking the obstacles into account). A cost-to-go function that is closer to the optimal cost leads to effective pruning, but most often such functions are computationally very expensive to get. In practice, even loose approximations enable significant pruning of the search tree. For instance, the Euclidean distance from z to X_{goal} divided by the maximum speed can be used as a cost-to-go function for minimum-time problems.

In the rest of the paper we assume that a cost-to-go function is available and implemented into the $\texttt{CostToGo}(z)$ procedure. Let x_{soln} be the minimum-cost trajectory in the tree reaching the goal region, then any vertex z with $\texttt{Cost}(z) + \texttt{CostToGo}(z) > J(x_{soln})$ can be removed.

E. Reachability

Systems subject to non-honomic differential constraints and input saturation have smaller reachable sets (e.g., in terms of Lebesgue measure) when compared to holonomic dynamical systems. Motivated by the assumption that sampling is relatively cheaper than edge expansion, reachability information had been used to improve the RRT algorithm [29]. Clearly, the RRT* attempts to expand more edges when compared to the RRT. Moreover, the cost of edge expansion becomes even larger with higher fidelity trajectory simulation. Therefore, the knowledge of the reachable set must be useful for the RRT*, especially for systems with a high-dimensional state space. In general, computing reachable sets exactly is known to be computationally quite challenging [30]. However, conservative approximations thereof usually can be computed rather easily.

V. Application to High-speed Off-road Vehicles

In this section, a nonlinear single-track vehicle dynamics is considered to simulate a rally car driving on loose surface. Specifically, a set of cornering maneuvers for various road curvatures are generated. The single-track model is less general than a full-body dynamics with suspensions, but it sufficiently captures the longitudinal load transfer phenomenon that takes an important role in cornering maneuvers [4].

A. Vehicle Dynamics

Let m and I_z denote the mass and the inertia of the vehicle. Let I_i, r_i, and ω_i ($i \in \{F, R\}$), denote the moment of inertia, the radius, and the angular velocity, respectively, where the subscripts F and R denote the front or rear wheels. Let x and y denote the position of the vehicle's center of gravity and ψ denote the yaw angle, in the inertial reference frame. Let v denote the speed of the vehicle. Let f_{Fx} and f_{Fy} denote the longitudinal and lateral forces acting on the front wheel, and f_{Rx} and f_{Ry} denote those acting on the rear wheel. Let β denote the side-slip angle. Let δ and T_i ($i \in \{F, R\}$) denote the steering angle and torques acting on each wheel, respectively. See Figure 1 for a depiction of these variables.

Then, the equations of motion can be written as follows:

$$\begin{aligned}
m\ddot{x} &= f_{Fx}\cos(\psi+\delta) - f_{Fy}\sin(\psi+\delta) \\
&\quad + f_{Rx}\cos\psi - f_{Ry}\sin\psi, \\
m\ddot{y} &= f_{Fx}\sin(\psi+\delta) + f_{Fy}\cos(\psi+\delta) \\
&\quad + f_{Rx}\sin\psi + f_{Ry}\cos\psi, \\
I_z\ddot{\psi} &= (f_{Fy}\cos\delta + f_{Fx}\sin\delta)l_F - f_{Ry}l_R, \\
I_F\dot{\omega}_F &= T_F - f_{Fx}r_F, \quad I_R\dot{\omega}_R = T_R - f_{Rx}r_R.
\end{aligned}$$

The tire force f_{ij} depends on the normal force f_{iz} and the friction coefficient μ_{ij} determined by Pacejka's Magic Formula [31]. The equations are

$$\begin{aligned}
f_{ij} &= \mu_{ij} f_{iz}, \quad (i = F, R,\ j = x, y); \\
\mu_{ij} &= -\frac{s_{ij}}{s_i}\mu_i(s_i), \quad (i = F, R,\ j = x, y); \\
\mu_i(s_i) &= D_i \sin(C_i \arctan(B_i s_i)), \quad (i = F, R).
\end{aligned}$$

The slip ratio, denoted by s_i, is calculated as follows:

$$\begin{aligned}
s_i &= \sqrt{s_{ix}^2 + s_{iy}^2}, \quad (i = F, R); \\
s_{ix} &= \frac{V_{ix} - \omega_i r_i}{\omega_i r_i}, \quad s_{iy} = \frac{V_{iy}}{\omega_i r_i}, \quad (i = F, R).
\end{aligned}$$

The velocities at the wheels are calculated by

$$\begin{aligned}
V &= \sqrt{\dot{x}^2 + \dot{y}^2}, \quad \beta = \arctan\frac{\dot{y}}{\dot{x}} - \psi, \\
V_{Fx} &= V\cos(\beta - \delta) + \dot{\psi} l_F \sin\delta, \\
V_{Fy} &= V\sin(\beta - \delta) + \dot{\psi} l_F \cos\delta, \\
V_{Rx} &= V\cos\beta, \quad V_{Ry} = V\sin\beta - \dot{\psi} l_R.
\end{aligned}$$

Finally, on a flat surface, normal forces are calculated by

$$\begin{aligned}
f_{Fz} &= \frac{l_R mg - hmg\mu_{Rx}}{(l_F + l_R) + h(\mu_{Fx}\cos\delta - \mu_{Fy}\sin\delta - \mu_{Rx})} \\
f_{Rz} &= mg - f_{Fz}
\end{aligned}$$

where h is the height of the vehicle's center of gravity.

We assume that both input signals, namely the steering angle δ and the engine/brake torques T_i are limited to a range, i.e., $\delta \in [\delta_{\min}, \delta_{\max}]$ and $T_i \in [T_{i\min}, T_{i\max}]$, and vehicle parameters are set identical to those given in [4].

B. Implementation details

In this subsection, several technical details are described in applying the RRT* algorithm and its extensions to the nonlinear single-track vehicle dynamics.

1) Cost Functional: As the RRT* cost functional for each trajectory, the total travel time is used since we aim to acquire the minimum-time cornering maneuver. Edges in the tree are constructed by 200-Hz forward simulation of the dynamics.

2) Sampling Strategy: The full state space consists of 8 variables x, y, $\dot{x}$, $\dot{y}$, ψ, $\dot{\psi}$, ω_F, and ω_R, and sampling happens uniformly at a 4-dimensional task space (x, y, V, ψ) to enable the false position method with piecewise-constant inputs. The position (x, y) is sampled on free space, and the velocity and the yaw angle are sampled within ranges $V \in [V_{min}, V_{max}]$ and $\psi \in [\psi_{min}, \psi_{max}]$. The yaw angle ψ_t of the road tangential guides the ψ range such that $\psi \in [\psi_t - \Delta, \psi_t + \Delta]$ where Δ is chosen sufficiently large.

3) Distance Metric: In searching the nearest neighbor or near-by vertices within a ball, a metric is necessary for distance evaluation between two vertices. We define the distance metric as the Euclidean distance divided by the average speed. More precisely, given two states $z_i = (x_i, y_i, \dot{x}_i, \dot{y}_i, \psi_i, \dot{\psi}_i, \omega_{Fi}, \omega_{Ri}) \in X$ for $i \in \{1, 2\}$, the distance function is computed as

$$\mathrm{dist}(z_1, z_2) = \frac{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}}{(\sqrt{\dot{x}_1^2 + \dot{y}_1^2} + \sqrt{\dot{x}_2^2 + \dot{y}_2^2})/2}.$$

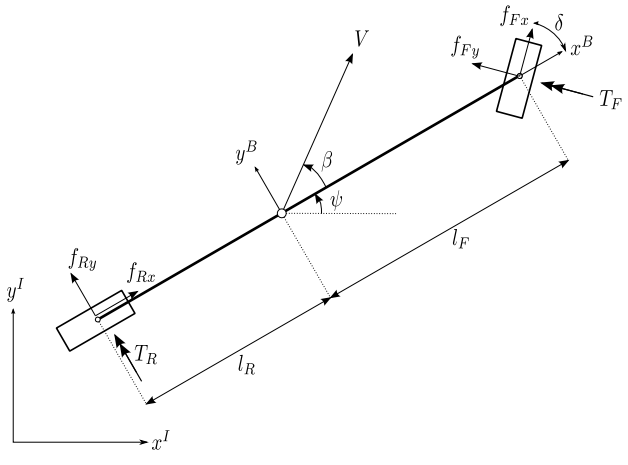
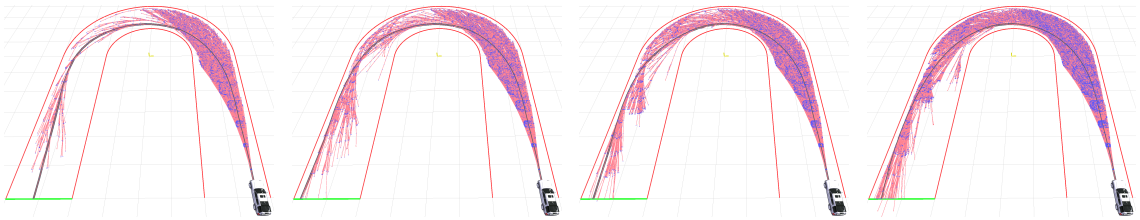


Fig. 1. The Single-Track Vehicle Model



(a) 11793 Nodes (b) 20946 Nodes (c) 30520 Nodes (d) 59542 Nodes

Fig. 2. The RRT* Tree for 180-deg Turning

The metric is consistent with the cost functional. Moreover our choice is consistent with the discussion in [17] that the optimal cost-to-go is likely to be a good choice.

4) Reachability: Exact estimation of the reachable set is a computationally intensive task. However, most often computationally-efficient conservative estimates of the reachable set of a dynamical system are available. In the sequel, an estimate of the reachable set is said to be conservative if it includes the reachable set itself. For a more precise definition, let $T \in \mathbb{R}_{>0}$ and $z \in X$. Given a dynamical system of the form in Equation (1), an initial state z, and a control input $u \in \mathcal{U}$, recall that $x(t; z, u)$ denotes the unique solution of the differential equation with initial state z and input u. Then, the T-reachable set of a dynamical system described by this differential equation is defined as the set of all states that are reachable from z by some admissible control input before time T, i.e., $\mathcal{R}_T(z) = \{z' \in Z \,|\, \exists u \in \mathcal{U}, t \in [0, T] \text{ such that } x(t; z, u) = z'\}$.

Below, we provide a procedure for computing a conservative estimate $\hat{\mathcal{R}}_T(z)$ of the reachable set of the dynamical system described in Section V-A. This estimate assumes that bounds on the acceleration of the car as well the yaw rate are known, and denoted by $\dot{v}_{\max}$ and $\dot{\psi}_{\max}$, respectively. For notational convenience, we describe computing the complement of $\hat{\mathcal{R}}_T(z)$. Given two states $z_1, z_2 \in X$, we have $z_2 \notin \hat{\mathcal{R}}_t(z_1)$ whenever at least one of the following holds:

- average yaw rate is greater than the maximum allowed, i.e., $|(\psi_2 - \psi_1)/(\text{dist}(z_1, z_2))| \geq \dot{\psi}_{\max}$;
- average acceleration is greater than the maximum allowed, i.e., $|(v_2 - v_1)/(\text{dist}(z_1, z_2))| \geq \dot{v}_{\max}$;
- z_2 is "behind" z_1, i.e., $(x_2, y_2) \in H_{(x_1,y_1),\psi_1}$, where $H_{(x_1,y_1),\psi}$ is the half-space with normal vector $(\cos(\psi_1), \sin(\psi_1))$ that has (x_1, y_1) on its boundary.

This estimate of the reachable set is a conservative estimate for all small T. That is, for any $z \in X$, there exists some $T \in (0, \infty)$ such that $\mathcal{R}_t(z) \subseteq \hat{\mathcal{R}}_t(z)$ for all $t \in [0, T]$.

5) Branch-and-Bound: For $\texttt{CostToGo}(z)$ in the branch-and-bound, the minimum distance from each node to the goal region is divided by the maximum achievable velocity so that the function never overestimates the actual cost.

VI. Simulation Results

The algorithm is evaluated in 90-, 150-, 180-, and 270-degree turns. The time to complete the maneuver reaching the end of the road is used as the cost function.

Figure 2 shows of the RRT* tree (projected on the x-y coordinate space) for minimum-time 180-deg cornering. The solid black line represents the solution trajectory of the center of gravity. The branch-and-bound procedure makes the part of the tree close to the finish line sparse due to pruning.

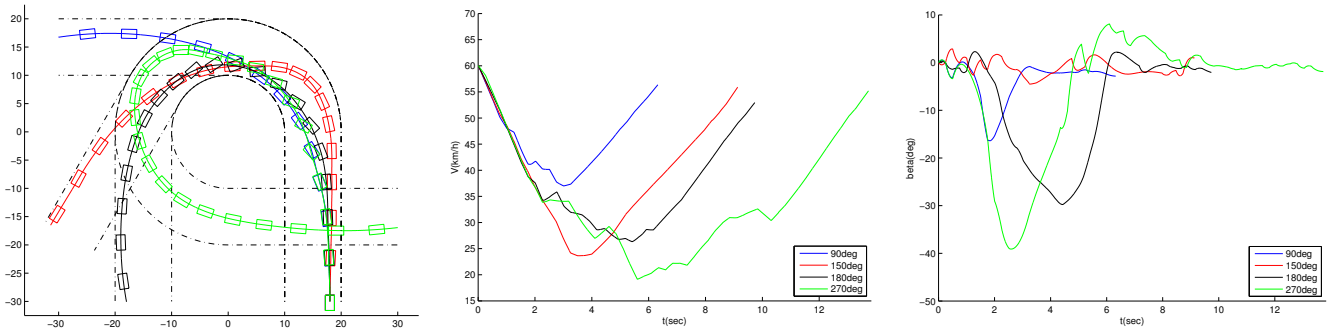
Figure 3 is a comparison of the trajectories, the speeds, and the vehicle slip angles for several turning angles. Plots show some regional non-smoothness since the algorithm did not grow the total number of vertices in the tree more than 60,000. It is noticeable that the time-optimal solution of the 150-deg turning does not involve much skidding while solutions for the other angles heavily involve the skidding regime. We observe that the characteristics of the optimal maneuver depends on the road shape and other conditions. Roughly speaking, Trail-Braking maneuvers by inputs parametrization in [4] can be characterized as two synchronized V-shapes in the speed and the slip angle. Our time-optimal 180-deg turning maneuver shows the synchronization of two V-shapes, meaning that the time-optimal maneuver for 180-deg turning with an initial speed 60 km/h on a loose surface with a friction coefficient $\mu = 0.52$ is indeed the trail-braking maneuver. For other road conditions and initial speeds, we expect other turning angles would generate the trail-braking-like trajectories also as the time-optimal solution.

The videos for the simulations can be found on our website at `http://ares.lids.mit.edu/rrtstar/`.

VII. Conclusion

This paper extended the application domain of the RRT* to systems with complex differential and geometric constraints with high-dimensional state spaces. This approach was used for generating aggressive skidding maneuvers as minimum-time solutions for high-speed off-road vehicle cornering on loose surfaces with a low friction coefficient.

There are several directions for future work. On one hand, from the theoretical point of view, a rigorous analysis of the convergence properties of the RRT* algorithms in the presence of numerical errors in the steering procedure will be included in future work. From a practical perspective, on the other hand, efficient implementations of the sampling strategies, steering procedure, and cost estimation are sought, in order to enable real-time computations. Finally, the algorithm is currently applicable to deterministic models of the vehicle dynamics, which may not be realistic in the scenario of interest. Future work will also involve extending the current approach to handle uncertainty and modeling errors.



(a) Trajectories. Vehicle is shown at every 0.5s. (b) Vehicle Speed (c) Vehicle Slip Angle

Fig. 3. Comparison Plots for 90, 150, 180, and 270-deg Turning

ACKNOWLEDGEMENTS

This research was supported in part by the US Army Research Office, MURI grant W911NF-11-1-0046, by the National Science Foundation, grant CNS-1016213, and by the STX Scholarship Foundation.

REFERENCES

[1] Arthur E. Bryson Jr. and Yu-Chi Ho. *Applied Optimal Control*. Hemisphere Publishing Corporation, 1975.

[2] J.P.M. Hendrikx, T.J.J. Meijlink, and R.F.C. Kriens. Application of optimal control theory to inverse simulation of car handling. *Vehicle System Dynamics*, 26:449–461, 1996.

[3] D. Casanova, R. S. Sharp, and P. Symonds. Minimum time manoeuvring: The significance of yaw inertia. *Vehicle System Dynamics*, 34:77–115, 2000.

[4] Efstathios Velenis, Panagiotis Tsiotras, and Jianbo Lu. Optimality properties and driver input parameterization for trail-braking cornering. *European Journal of Control*, 4:308–320, 2008.

[5] Efstathios Velenis, Emilio Frazzoli, and Panagiotis Tsiotras. Steady-state cornering equilibria and stabilization for a vehicle during extreme operating conditions. *Int. Journal of Vehicle Autonomous Systems*, 8(2-4):217–241, 2010.

[6] Timur Karatas and Francesco Bullo. Randomized searches and nonlinear programming in trajectory planning. In *Proceedings of the 40th IEEE Conference on Decision and Control*, December 2001.

[7] Jean-Claude Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

[8] J. Leonard, J. How, S. Teller, M. Berger, S. Campbell, G. Fiore, L. Fletcher, E. Frazzoli, A. Huang, S. Karaman, O. Koch, Y. Kuwata, D. Moore, E. Olson, S. Peters, J. Teo, R. Truax, M. Walter, D. Barrett, A. Epstein, K. Maheloni, K. Moyer, T. Jones, R. Buckley, M. Antone, R. Galejs, S. Krishnamurthy, and J. Williams. A perception driven autonomous urban vehicle. *Journal of Field Robotics*, September 2008.

[9] Nancy M. Amato and Yan Wu. A randomized roadmap method for path and manipulation planning. In *Proceedings of the 1996 IEEE International Conference on Robotics and Automation*, 1996.

[10] 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*, Anchorage, AK, May 2010.

[11] Russell H. Taylor and Dan Stoianovici. Medical robotics in computer-integrated surgery. *IEEE Transactions on Robotics and Automation*, 19(5):765–781, October 2003.

[12] Amit Bhatia and Emilio Frazzoli. Incremental search methods for reachability analysis of continuous and hybrid systems. In *Hybrid Systems: Computation and Control*, Lecture Notes in Computer Science, pages 142–156. Springer-Verlag, Philadelphia, PA, March 2004.

[13] Hsuan Chang and Tsai-Yen Li. Assembly maintainability study with motion planning. In *Proceedings of the 1995 IEEE International Conference on Robotics and Automation*, 1995.

[14] Lydia E. Kavraki. Geometry and the discovery of new ligands. In *Workshop on the Algorithmic Foundations of Robotics*, pages 435–448, 1996.

[15] John H. Reif. Complexity of the mover's problem and generalizations. In *Proceedings of the 20th IEEE Symposium on Foundations of Computer Science*, pages 421–427, 1979.

[16] Lydia E. Kavraki, Petr Švestka, Jean-Claude Latombe, and Mark H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, Aug 1996.

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

[18] S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *International Journal of Robotics Research*, 30(7):846–894, 2011.

[19] Sertac Karaman and Emilio Frazzoli. Optimal kinodynamic motion planning using incremental sampling-based methods. In *IEEE Conference on Decision and Control*, 2010.

[20] Alexander Shkolnik and Russ Tedrake. Path planning in 1000+ dimensions using a task-space voronoi bias. In *IEEE International Conference on Robotics and Automation*, May 2009.

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

[22] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, 1995.

[23] Arthur E. Bryson Jr. Optimal control - 1950 to 1985. *IEEE Control Systems Magazine*, 16:26–33, Jun 1996.

[24] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, 2007.

[25] Ailsa H. Land and Alison G. Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28(3):497–520, Jul 1960.

[26] Gideon Sahar and John M. Hollerbach. Planning of minimum-time trajectories for robot arms. *The International Journal of Robotics Research*, 5(3):90–100, September 1986.

[27] 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 Technology*, 17(5):1105–1118, September 2009.

[28] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller. Anytime motion planning using the RRT*. In *IEEE International Conference on Robotics and Automation*, 2011. To appear.

[29] Alexander Shkolnik, Matthew Walter, and Russ Tedrake. Reachability-guided sampling for planning under differential constraints. In *IEEE International Conference on Robotics and Automation*, May 2009.

[30] E. Asarin, T. Dang, G. Frehse, A. Girard, C. Le Guernic, and O. Maler. Recent progress in continuous and hybrid reachability analysis. In *Proceedings of the IEEE Conference on Computer Aided Control System Design*, 2006.

[31] E. Bakker, L. Nyborg, and H. B. Pacejka. Tyre modelling for use in vehicle dynamics studies. In *SAE International Congress and Exposition*, 1987.