R3T: Rapidly-exploring Random Reachable Set Tree for Optimal Kinodynamic Planning of Nonlinear Hybrid Systems

R3T: Rapidly-exploring Random Reachable Set Tree for Optimal Kinodynamic Planning of Nonlinear Hybrid Systems

Albert Wu, Sadra Sadraddini, Russ Tedrake

***Abstract*— We introduce R3T, a reachability-based variant of the rapidly-exploring random tree (RRT) algorithm that is suitable for (optimal) kinodynamic planning in nonlinear and hybrid systems. We developed tools to approximate reachable sets using polytopes and perform sampling-based planning with them. This method has a unique advantage in hybrid systems: different dynamic modes in the reachable set can be explicitly represented using multiple polytopes. We prove that under mild assumptions, R3T is probabilistically complete in kinodynamic systems, and asymptotically optimal through rewiring. Moreover, R3T provides a formal verification method for reachability analysis of nonlinear systems. The advantages of R3T are demonstrated with case studies on nonlinear, hybrid, and contact-rich robotic systems.**

I. Introduction

Sampling-based motion planning algorithms such as probabilistic road-maps (PRMs) [1] and rapidly-exploring random trees (RRTs) [2]–[5] have been proven powerful in a broad range of planning problems. However, a number of limitations exist in these methods.

The bases of RRTs are rapid exploration of the state space and connection of new states to explored states. When kinodynamic constraints are present, one needs to solve the expensive two point boundary value problem of finding an admissible trajectory to perform connection. For linear systems, the problem is manageable and implementations exist [6], [7]. A widely-adopted alternative for general systems is to simulate trajectories forward, then expand the explored states with the nearest produced point to the sample state [3], [8]. Nevertheless, this method may not be probabilistically complete in kinodynamic settings [9]. Existing RRT approaches also tend to perform poorly in hybrid systems, such as those in contact-based robotics. Crucial maneuvers with particular mode sequences may be hard to explore. Difficulties associated with the choice of extension strategy and distance metrics exacerbate this issue [10]–[12].

Many variants of RRT have been developed to improve planning performance in kinodynamic and hybrid systems. In these systems, the nearest state in the tree to the sample might not be associated with the nearest reachable state to the sample (see Fig. 1). Reachability guided methods, such as reachability guided RRT (RG-RRT) [13], address this problem by exploring reachable states in advance and then growing the tree to the nearest reachable state to the sample.

The authors are with the Computer Science and Artificial Intelligence Laboratory (CSAIL) at Massachusetts Institute of Technology, Cambridge, MA 02139. This research was partially funded by Department of the Navy, Office of Naval Research Award No.: N00014-17-1-2699 {wualbert,sadra,russt@mit.edu }

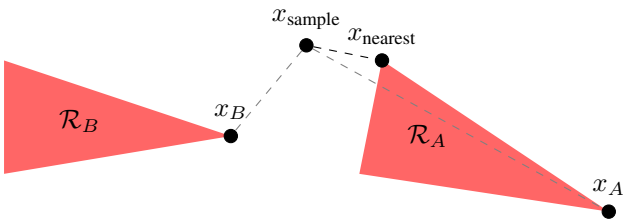


Fig. 1: RRT extension and system reachability. While x_B is closer to x_{sample} than x_A, the reachable set of x_A, $\mathcal{R}_A$, has states closer to x_{sample} than any point in $\mathcal{R}_B$. A reachability guided algorithm would extend the tree to x_{nearest}, whereas traditional RRT would either extend in $\mathcal{R}_B$ or fail.

However, this requires searching over a much larger number of points. A number of algorithms in the spirit of RG-RRT have been developed. Environment-guided RRT (EG-RRT) [14] combines RG-RRT with a sampling strategy biased toward more promising parts of the state-space; planning with motion cones [15] uses the notion of reachability in configuration space for in-hand manipulation.

In this paper, we take a formal approach to reachability-guided sampling-based planning. We developed a method to represent and plan on the whole reachable set. Our contributions are as follows.

- A framework to represent the forward (or backward) reachable set of a state as (the union of) polytope(s) using the linearized local dynamics. (Sec. III).
- ***R**apidly-exploring **R**andom **R**eachable Set **T**ree (R3T)* sampling-based planning algorithm guided by (polytopic) reachable sets (Sec. IV). The algorithm is probabilistically complete in kinodynamic settings with approximated reachable sets. R3T*, the version with rewiring (Sec. IV-E), retains the asymptotic optimality of traditional RRT*s. We demonstrate the benefits of planning with reachable sets using kinodynamic and hybrid systems (Sec. V).
- As a consequence of R3T, a tool for approximate (subject to linearization errors) reachability verification, as opposed to RRT-based falsification [16], [17].

A video summary of this paper is available on YouTube[1].

II. Problem Statement and Approach

First, we provide the necessary background on polytopes. An *H-polyhedron* is a set defined by a finite number of linear inequalities $\mathbb{H} = \{x \in \mathbb{R}^n | Hx \leq h\}$, where $H \in \mathbb{R}^{q \times n}, h \in \mathbb{R}^q$, and the inequality vector is interpreted element-wise.

[1] https://youtu.be/E8TICePNqE0

A bounded H-polyhedron is called an *H-polytope* [18]. An *AH-polytope* is a set in the form of an affine transformation of an H-polytope $\mathbb{H} \subset \mathbb{R}^p$ [19], [20] $\mathbb{A}(\bar{x}, G, H, h) := \bar{x} + G\mathbb{H}$, where $G \in \mathbb{R}^{n\times p}$ is the linear transformation and $\bar{x} \in \mathbb{R}^n$ is the offset term. In general, an AH-polytope can be transformed into an H-polytope, but its number of hyperplanes may be exponentially large [20]. As such, we do not use the H-polytope representation in this paper.

Consider the following continuous-time hybrid system:

$$\dot{x} = f(x, u, \sigma), (x, u) \notin \mathbb{G}, \tag{1a}$$

$$(\sigma, x)^+ = r(x, u, \sigma), (x, u) \in \mathbb{G}, \tag{1b}$$

where $x \in X \subset \mathbb{R}^n$ is the continuous system state, $u \in U \subset \mathbb{R}^m$ is the control input, $\sigma \in \Sigma$ is the system mode with Σ being a finite set, and $\mathbb{G}$ is the (zero-measured) set of guards where mode transitions happen. For instance, the guard may represent the ground in a hopping robot.

Problem 1. *Given system* (1)*, an initial state x_0 and a goal state x_G, find a trajectory $\zeta : [0, T] \to (x, u)$ respecting* (1)*, $x(0) = x_0, x(T) = x_G$, $x(t) \in X$, $u(t) \in U, \forall t \in [0, T]$, and T is finite. If optimality is desired, find the optimal trajectory such that it minimizes $J = \int_0^T c(x, u)dt$, where $c : X \times U \to \mathbb{R}$ is the running cost.*

Using a time step $\tau \in \mathbb{R}_+$ and a time-integration scheme such as time-stepping ([21]), we bring (1) into the following discrete time form:

$$x^+ = F_i(x, u), (x, u) \in \mathbb{S}_i, \tag{2}$$

where $\mathbb{S}_i, i = 1, \cdots, N$, are interior-disjoint sets corresponding to N modes described by a number of inequalities $S_i(x, u) \leq 0$. The constraint $(x, u) \in X \times U$ is also included in each $\mathbb{S}_i, i = 1, \cdots, N$.

The continuous-time trajectory between two consecutive states in (2) is often approximated by a straight-line. This approximation is valid in continuous systems with small time steps, and is also reasonable in certain classes of (1), where discrete jumps only occur in the mode and not in the state. Examples include soft contact models where impacts are modeled with springs instead of instantaneous bouncing.

III. Polytopic Reachable Set Approximations

In this section, we develop a framework to approximate the forward (or backward) reachable sets of (2).

Definition 1. *The forward reachable set of state $\bar{x}$ is defined as the set of all states that can be reached from $\bar{x}$ in less than time τ using valid control inputs*

$$R(\bar{x}) := \{x \in X | \exists [0, \tau] \to U, x(0) = \bar{x}, (1)\}. \tag{3}$$

While R3T supports planning with exact $R(\bar{x})$ (see Sec. V-B), finding an explicit representation for $R(\bar{x})$ is generally difficult. However, an approximation can be found with time discretization and linearization. We linearize the dynamics and the constraints of each mode of the system. Given $\eta = (\bar{x}, \bar{u})$, the linearized dynamics is $x^+ = A_i^\eta x + B_i^\eta u + c_i^\eta$,

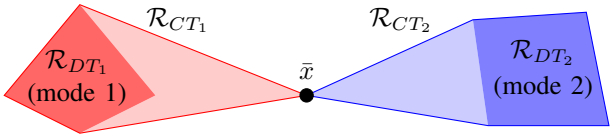


Fig. 2: Schematic illustration of the reachable sets of a hybrid system. $\mathcal{R}_{DT_1}$ and $\mathcal{R}_{DT_2}$ represent discrete time reachable sets of $\bar{x}$ in 2 different modes. $\mathcal{R}_{CT_1}$ (red) and $\mathcal{R}_{CT_2}$ (blue), the continuous-time reachable set of each mode, is computed by taking the convex hull of $\bar{x}$ and the respective $\mathcal{R}_{DT}$.

where $A_i^\eta = \frac{\partial F_i}{\partial x}|_\eta$, $B_i^\eta = \frac{\partial F_i}{\partial u}|_\eta$, and $c_i^\eta = F_i(\bar{x}, \bar{u}) - A_i^\eta \bar{x} - B_i^\eta \bar{u}$. The set of constraints $S_i(x, u) \leq 0$ is linearized to obtain the following polyhedral set in $X \times U$:

$$D_i^\eta x + E_i^\eta u \leq \zeta_i^\eta, \tag{4}$$

where $D_i^\eta = \frac{\partial S_i}{\partial x}|_\eta, E_i^\eta = \frac{\partial S_i}{\partial u}|_\eta$, and $\zeta_i^\eta = D_i^\eta \bar{x} + E_i^\eta \bar{u} - S_i(\bar{x}, \bar{u})$. Note that (4) is an H-polytope if X and U are bounded sets, which is often the case. An approximation of the reachable set in discrete time is the following set:

$$\mathcal{R}_{DT}(\bar{x}) = \bigcup_{i=1}^{N} (A_i^\eta \bar{x} + c_i^\eta) + B_i^\eta \{u \mid E_i^\eta u \leq \zeta_i^\eta - D_i^\eta \bar{x}\}, \tag{5}$$

which is a union of AH-polytopes. It is possible that some of the polytopes in (5) are empty—not every mode is attainable at a given state. Following the straight line continuous-time approximation discussed earlier, the polytopic reachable set in continuous time $\mathcal{R}_{CT}$ is approximated by taking the convex hull of $\mathcal{R}_{DT}$ with $\bar{x}$. The convex-hull of a point and an AH-polytope is still an AH-polytope [20]. We have

$$\mathcal{R}_{CT}(\bar{x}) = \bigcup_{i=1}^{N} \mathbb{A}(\bar{x}, [B_i^\eta, A_i^\eta \bar{x} + c_i^\eta - \bar{x}], [E_i^\eta, D_i^\eta \bar{x} - \zeta_i^\eta], 0),$$

where $[\cdot, \cdot]$ stands for concatenating matrices horizontally. This construction of reachable sets is illustrated in Fig. 2.

IV. Sampling-based Planning with Reachable Sets

Algorithm 1 provides an overview of R3T. First, the tree is initialized with the start state. The tree is then expanded through sampling the state space and performing the `Extend` routine. Whenever a new node is added, R3T may check for more optimal paths through `Rewire` if `optimality` is flagged true. R3T then checks whether the goal is reachable from the new node with `ExtendToPoint`. If it is the case, a path is found and R3T terminates.

A. Guiding Tree Growth with Random Sampling

R3T's extension routine is entirely based on the reachable set. First, the nearest reachable set in the tree $R_c \in T$ to the random state sample x_s is found. If $x_s \in R_c$, the tree is extended toward x_s. Otherwise, extension is performed through finding the nearest state $x_c \in R_c$. This sampling method favors states that are feasible and prevents sample rejection, similar to RG-RRT [13]. Moreover, the extension

Algorithm 1 R3T

Require: $\mathcal{R}$, $\mathcal{S}$, x_0 and x_g ▷ reachable set oracle, sampling function, start state, goal state
Require: `optimality`, i_{max}, ϵ ▷ whether to rewire, max iterations, tolerance for reaching x_g

T.add($x_0, \mathcal{R}(x_0)$) ▷ initialize tree with start state
$i \leftarrow 0$ ▷ reset iterations count
if $x_g \in \mathcal{R}(x_0)$ **then**
  $g \leftarrow$ ExtendToPoint($\mathcal{R}(x_0), x_g$)
  if $g \neq \emptyset$ **then**
    T.add(g)
    return BuildPath(T, g) ▷ found path to goal
while $i < i_{max}$ **do**
  $x_s \leftarrow \mathcal{S}()$ ▷ sample the state space
  $R_c, x_c \leftarrow$ FindNearest(T, x_s)
  $x_n \leftarrow$ Extend(R_c, x_c)
  if $x_n \neq \emptyset$ **then** ▷ successful extension
    T.add($x_n, \mathcal{R}(x_n)$) ▷ add new node
    if optimality==True **then**
      Rewire($T, \mathcal{R}(x_n)$) ▷ optional rewiring
    if $x_g \in \mathcal{R}(x_n)$ **then**
      $g \leftarrow$ ExtendToPoint($\mathcal{R}(x_n), x_g$)
      if $g \neq \emptyset \wedge$ Dist(g, x_g) $< \epsilon$ **then**
        T.add(g)
        return BuildPath(g) ▷ found path
  $i \leftarrow i + 1$
return $\emptyset$ ▷ R3T failed

routine of R3T avoids an explicit distance metric if at least one reachable set contains x_s.

B. Finding the Nearest Reachable Set

`FindNearest` finds the nearest reachable set R_c to a sample state x_s, and the nearest state $x_c \in R_c$ to x_s. The nearest state is x_s if $x_s \in R_c$. Otherwise, an explicit distance metric is necessary to find the nearest reachable set. This is the only part of R3T that relies on an explicit distance metric. L_2 distance was used in Sec. V. `FindNearest` is critical to R3T's performance as it is called on every extension. In this paper, the method based on axis-aligned bounding boxes from [22] was used.

C. Computing Reachable Sets

In general, the reachable set oracle $\mathcal{R}(\cdot)$ computes reachable sets with AH-polytope approximations (Sec. III). Some systems such as Dubins car (Sec. V-B) have exploitable properties that allow system-specific reachable set computation.

D. Extension

`Extend` routine is trivial if R is a conservative approximation, as the target state must be reachable. Otherwise, Algorithm 2 can be used to ensure the extended path is indeed reachable. We emphasize that x_n may be obtained by a more precise simulation than when generating R.

For the polytopic reachable set approximation discussed in Sec. III, `CalcInput` can be implemented using (6).

Algorithm 2 `Extend` with approximated $\mathcal{R}(\cdot)$

Require: R, x ▷ Approximated reachable set, target state
Require: τ ▷ Time horizon

$u \leftarrow$ CalcInput(R, x) ▷ Control input to get to x
$x_n \leftarrow \int_{t=0, x(0)=R.x}^{t=\tau} f(R.x, u)dt$ ▷ Simulate trajectory using u. $R.x$ is the state from which R is generated.
if CollisionFree($R.x, x_n$) **then**
  return x_n
return $\emptyset$

Algorithm 3 `ExtendToPoint` with input enumeration

Require: R, x ▷ Approximated reachable set, target state
Require: $\mathcal{U}, \tau$ ▷ Possible control inputs, time horizon

$\mathbf{u} \leftarrow$ Sample($\mathcal{U}$) ▷ Sample control inputs
for $u_i \in \mathbf{u}$ **do**
  $\mathbf{x_i} \leftarrow \int_{t=0, x(0)=R.x}^{t=\tau} f(R.x, u_i)dt$ ▷ Simulate trajectory using u_i. $R.x$ is the state from which R is generated.
  if Dist($x, \mathbf{x_i}$) $\approx 0 \wedge$ CollisionFree($R.x, x$) **then**
    return x
return $\emptyset$

$$u = (B^\eta)^\dagger (x_c - x - A^\eta x - c^\eta)|_{x=R_c.x}, \tag{6}$$

where $(.)^\dagger$ stands for Moore-Penrose inverse. Equation (6) is implemented in all subsequent experiments. For proving probabilistic completeness, another `CalcInput` is proposed and discussed in Section IV-H.

A variant of `Extend`, `ExtendToPoint`, has an additional constraint that Dist(x_n, x_c) is small. `ExtendToPoint` is used in rewiring and goal checking. This routine is also trivial if the R is a conservative approximation. If R is an over-approximation, solving the two-point boundary value from $R.x$ to x_n problem is necessary. Algorithm 3 provides an approach where the solution is obtained through sampling the input space. Note that by keeping a reachable set approximation, we can weed out most unreachable states with little computation. Generally, the explicit trajectory between tree nodes only needs to be calculated during the `BuildPath` routine. Even with rewiring, the algorithm can maintain optimality as long as the cost-to-go is consistent. If there are obstacles in addition to the system dynamics, the collision checking routine `CollisionFree` is performed during extension. Standard collision checking routines used by other RRT algorithms may be applied.

E. Rewiring

R3T may maintain asymptotic optimality (Sec. IV-I) through a rewiring procedure similar to [8]. `Rewire` consists of finding the best parent of a newly added node, and using the new node as a potential parent. Algorithm 4 describes `Rewire` in detail.

Algorithm 4 `Rewire`

Require: T, R ▷ R3T, newly added reachable set
Require: $\mathcal{C}$ ▷ Cost-to-go function
$\mathbf{R}_R \leftarrow \texttt{Contains}(T.\mathbf{R}, R.x)$ ▷ Find all reachable sets containing R
for $R_i \in \mathbf{R}_R$ **do**
  if $\mathcal{C}(R.x) > \mathcal{C}(R_i.x) + \mathcal{C}(R_i.x, R.x)$ **then**
    $x' \leftarrow \texttt{ExtendToPoint}(R_i, R.x)$
    if $x' \neq \emptyset \wedge \texttt{Dist}(x', R.x) \approx 0$ **then**
      $R.x$.parent $\leftarrow R_i.x$ ▷ Rewire $R.x$ with R_i
$\mathbf{x}_R \leftarrow \texttt{Contains}(R, T.\mathbf{x})$ ▷ Find all explored states contained in R
for $x_j \in \mathbf{x}_R$ **do**
  if $\mathcal{C}(x_j) > \mathcal{C}(R.x) + \mathcal{C}(R.x, x_j)$ **then**
    $x'_j \leftarrow \texttt{ExtendToPoint}(R, x_j)$
    if $x'_j \neq \emptyset \wedge \texttt{Dist}(x'_j, x_j) \approx 0$ **then**
      x_j.parent $\leftarrow R.x$ ▷ Rewire x_j with R
return

F. Goal Checking

Whenever a new reachable set is added to the tree, `ExtendToPoint` is performed to see whether the goal state is in the new reachable set. If it is, the goal node is added to the tree, and the algorithm terminates.

G. Optimizations

Information from the reachable sets can be exploited to accelerate R3T. As discussed in Sec. III, taking the convex hull of the discrete time reachable set and the originating state allows choosing a coarse τ. Another useful technique is to exploit deterministic dynamics. If the input matrix $B = \mathbf{0}$ in some dynamic modes, the control inputs do not affect state evolution, and the node can be explored further cheaply through simulating forward dynamics until $B \neq \mathbf{0}$. Both techniques were implemented in Sec. V.

H. Correctness and Probabilistic Completeness (PC) of R3T

The correctness of R3T follows by construction. Algorithms 2 and 3 ensure feasibility of the solution path whether the exact or approximated reachable sets are used.

Assume there exists a *robustly feasible* (see [4]) solution $\mathcal{P}$ to Problem 1. We consider Lipschitz continuous systems with 1 dynamic mode. We require the support of the sampling function $\mathcal{S}$ to contain $\cup\{R(x) \mid \forall x \in \mathcal{P}\}$.

Theorem 1. *R3T with exact reachable sets is PC.*

Proof: Leveraging the results and notations from [23], we only need to show the probability of a successful propagation into $\mathcal{B}_{\kappa\delta}(x_1)$ from $x'_0 \in \mathcal{B}_\delta(x_0)$ is nonzero. Since the path is robustly feasible, $\mathcal{B}_{\kappa\delta}(x_1) \cap R(x'_0)$ has nonzero volume and nonzero probability of being sampled. □

For R3T with approximated reachable sets $\bar{R}(\cdot)$ as discussed in Algorithm 3, we add the following assumptions:

Assumption 1. *There exists $\mathcal{M}_x : \bar{R}(x) \to R(x)$ defining the following relationship: for $\forall y \in R(x), \exists \bar{y} \in \bar{R}(x)$ such that if $\bar{y}$ is sampled, y is extended to from x. An alternative assumption is every feasible input $u \in U$ and time $t \in [0, \tau]$ has a nonzero probability of being sampled.*

Assumption 2. *In the case where a state is contained in multiple $\bar{R}(x)$, we break ties randomly. For convenience, we assume the tie breaking probability is uniform.*

Theorem 2. *R3T with reachable set approximations satisfying Assumptions 1 and 2 is still PC.*

Proof: By Assumption 1, $\exists \bar{\mathcal{B}}_{\kappa\delta}(x_1) := \{y \mid \mathcal{M}_{x'_0}(y) = x, x \in \mathcal{B}_{\kappa\delta}(x_1) \cap R(x'_0)\} \neq \emptyset$. Denote the probability of sampling in $\bar{\mathcal{B}}_{\kappa\delta}(x_1)$ as p, $0 < p \leq 1$. There may be tree nodes $Z = \{z_1, z_2, \ldots, z_m\} \subset T$ such that $\bar{\mathcal{B}}_{\kappa\delta}(x_1) \cap \bar{R}(z_i) \neq \emptyset$. By Assumption 2, the probability of choosing x'_0 is at least $\frac{1}{m+1}$. The probability of propagating from x'_0 into $\bar{\mathcal{B}}_{\kappa\delta}(x_1)$ is $\frac{p}{m+1} \geq \frac{p}{|\mathcal{T}|}$, where $|\mathcal{T}|$ is the tree size. Suppose initially $|T| = k$ and n extensions are performed. The probability of *failing* to propagate into $\bar{\mathcal{B}}_{\kappa\delta}(x_1)$ is upper bounded by $\prod_{i=0}^{n-1}(1 - \frac{p}{k+i})$. For $0 < a < 1$,

$$\ln(1-a) = -a - \frac{a^2}{2(1-\epsilon)^2} \leq -a, \ 0 < \epsilon < a. \quad (7)$$

The logarithm of the failure probability after infinite steps is

$$\lim_{n\to\infty} \sum_{i=0}^{n-1} \ln(1 - \frac{p}{k+i}) \leq \lim_{n\to\infty} \sum_{i=0}^{n-1} -\frac{p}{k+i} = -\infty, \quad (8)$$

thus

$$\lim_{n\to\infty} \prod_{i=0}^{n-1} (1 - \frac{p}{k+i}) = 0. \quad (9)$$

In virtue of (9), after infinite steps, the propagation to $\bar{\mathcal{B}}_{\kappa\delta}(x_1)$ succeeds almost surely. □

To achieve PC with polytopic reachable set $\bar{R}(\cdot)$, consider an alternative `CalcInput` satisfying Assumption 1. Observe that extending to $x_s \in \bar{R}(x_c)$ is a polytopic constraint on the time step $\bar{t} = \beta\tau$ and reparameterized input $v := \beta u$.

$$x_s = (A^\eta x + c^\eta)\beta + (B^\eta)v + x_c, v \in \beta U, \beta \in [0, 1]. \quad (10)$$

Since x_s may be any state in $\bar{R}(\cdot)$, if a random polytope sampler such as [24] is used to choose $u, \bar{t}$ for Algorithm 2, all feasible $u, \bar{t}$ can be chosen and the setup is PC.

I. Asymptotic Optimality with Rewiring

Definition 2. *An R3T tree T is "optimal" if the edges connecting the root x_r and all $x_t \in T$ form a path no more costly than any other path from x_r to x_t via waypoints $\{x_i\} \in T$, where each $x_{i+1} \in R(x_i)$.*

Theorem 3. *Given an R3T tree T constructed with the rewiring procedure, the tree is optimal.*

Proof: We prove by induction. Suppose we start from an optimal tree T, and a new node x_n is added. If a suboptimal path appears after adding x_n, it must contain x_n. The rewiring procedure checks for better paths involving $\{x \mid x_n \in R(x) \wedge x \in T\}$ and $\{x \mid x \in R(x_n) \wedge x \in T\}$, which covers all path segments involving x_n with duration

$\leq \tau$. Therefore, $T \cup \{x_n\}$ is still optimal. The "base case" of a tree with 1 node is optimal, so the proof is complete. □

With PC, all possible paths will be explored by R3T given a long enough running time. We therefore speculate that R3T posses the asymptotic optimality given in [4], [25]. While we do not bound the complexity of rewiring, empirical evidence suggests that rewiring is beneficial in practice.

V. Case Studies

We implemented R3T, RG-RRT [13], and RRT [3] for testing. Our RRT and RG-RRT implementations sample 3 evenly-spaced inputs for tree extension using their respective strategies. All scripts are available on GitHub[2]. All tests were performed on a personal computer with i7-7820HQ CPU.

A. Pendulum Swing-Up

We consider a single-link torque-limited pendulum system with damping. The pendulum was started at rest, and the goal is to swing up the pendulum to rest at $\theta = \pi$. The tolerance for reaching the goal is $||x - x_g||_2 \leq 0.05$. For R3T and RG-RRT, a reachable set time horizon of 0.2s was used. The time step size for RRT and forward dynamics was 0.01s. 10 consecutive planning tries were done, and the results are summarized in Table I.

TABLE I: Path Planning Statistics with Pendulum.

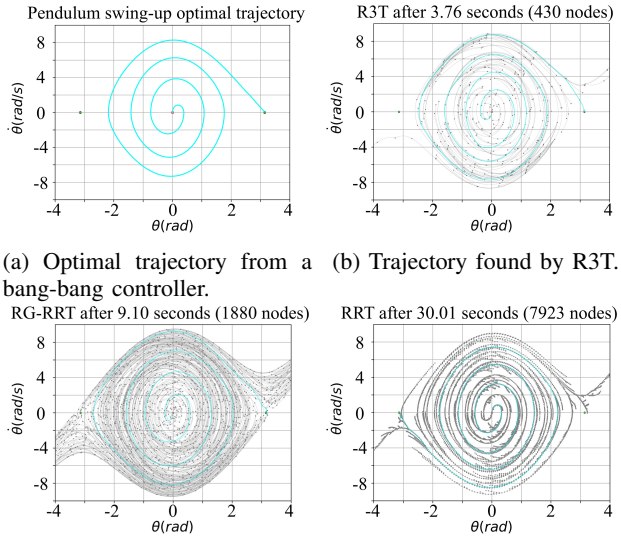
	R3T		RG-RRT		RRT	
	Time(s)	Nodes	Time(s)	Nodes	Time(s)	Nodes
Mean	5.92	559	21.4	4352	45.8	11134
Median	4.07	472	6.6	1381	31.0	8349
Max	17.26	1218	110.8	22600	97.5	19360
Min	2.14	284	1.5	336	28.0	7677
S.D.	5.09	326	33.8	4353	23.6	4273

R3T significantly outperformed RG-RRT and RRT in both runtime and nodes explored. This is attributed to the large simulation timestep allowed by planning with reachable sets. Moreover, by using the convex hull technique discussed in Sec. IV-G, choosing a large simulation time step does not cause R3T to "overlook" the goal states, unlike in RG-RRT and RRT where the exploration often passed by the goal but not terminate. One way to mitigate this pitfall is to add a possibly expensive routine in RG-RRT and RRT to check if the goal lies on the path between two nodes. Fig. 4 shows the reachable sets explored by R3T as time progresses.

B. Dubins Car

To demonstrate asymptotic optimality of R3T, we consider the "Dubins car" problem. The problem is to find a path subject to a car's kinematic constraints. The problem can be simplified to finding a path that consists of curvature-limited Dubins curves [26]. Observing that the reachable set from each state is just a linear transformation of a "base" reachable set, a fixed-horizon exact reachable set was precomputed through simulating the system dynamics forward. Fig. 5a and 5b shows the solution improvements as time progressed. Fig. 5c plots the path length against time. As expected from asymptotic optimality, the cost is monotonically decreasing.

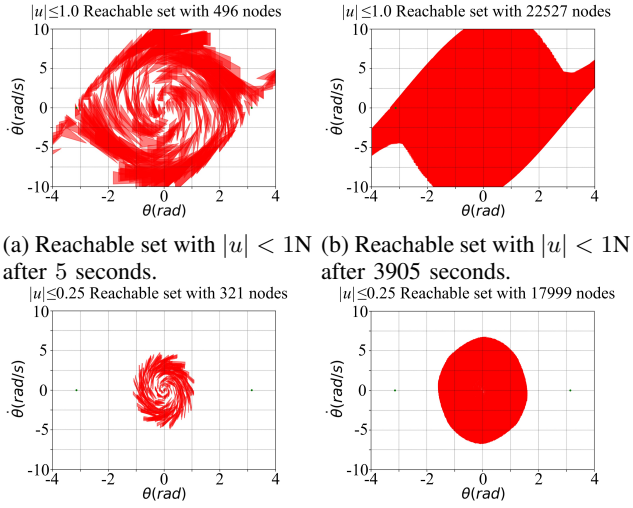
[2]https://github.com/wualbert/r3t



(a) Optimal trajectory from a bang-bang controller. (b) Trajectory found by R3T.

(c) Trajectory found by RG-RRT. (d) Trajectory found by RRT.

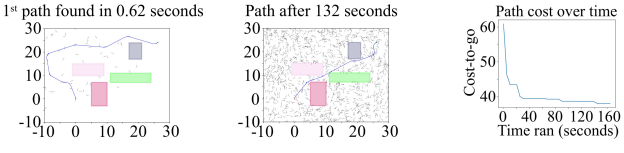
Fig. 3: Pendulum swing-up trajectories found by various algorithms. R3T found a solution significantly faster with fewer nodes explored.



(a) Reachable set with $|u| < 1$N after 5 seconds. (b) Reachable set with $|u| < 1$N after 3905 seconds.

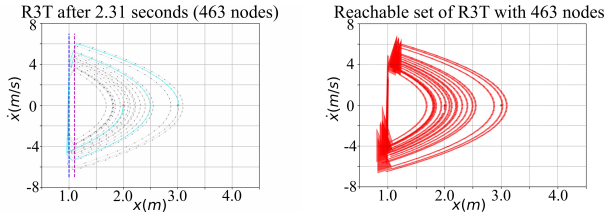
(c) Reachable set with $|u| < 0.25$N after 5 seconds. (d) Reachable set with $|u| < 0.25$N after 3905 seconds.

Fig. 4: Polytopic approximation of the pendulum reachable set as explored by R3T. Swing-up is impossible with the input limit and the damping coefficient in 4c and 4d. As the runtime increased, nearly all feasible states were covered, proving probabilistic completeness empirically.



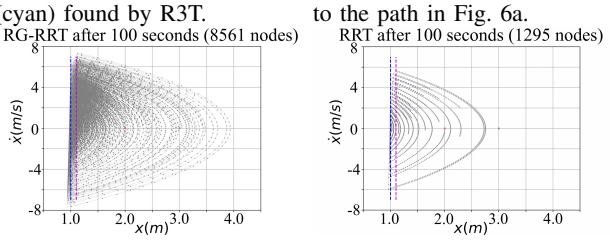
(a) First path found. Cost-to-go is 60.9. (b) Path after 132s. Cost-to-go is 38.6. (c) Cost-to-go plotted over time.

Fig. 5: Results of R3T on Dubins car. Figures 5a, 5b are snapshots of the explored states (grey) and solution (blue).



(a) Nodes (grey) and trajectory (cyan) found by R3T.

(b) Reachable set corresponding to the path in Fig. 6a.



(c) Instance where RG-RRT failed. With a small goal region, RG-RRT failed to acknowledge the goal between two nodes.

(d) Instance where RRT failed. To match the simulation granularity of R3T, RRT needed a much smaller time step.

Fig. 6: 1D hopper trajectories found by various algorithms. R3T finds a solution consistenly and quickly. RG-RRT and RRT each suffer from different failure modes.

C. 1D-Hopper

The 1D hopping robot is used to compare R3T and other method's ability to plan on hybrid systems. Our model has 2 states, 2 continuous dynamic modes (flight, soft ground contact), and 1 discrete dynamic mode (inelastic collision with the ground). For our tests, the hopper should hop from 2m to 3m. The tolerance for reaching the goal is $||s - s_g||_2 \leq 0.05$. For R3T and RG-RRT, a reachable set time horizon of 0.04s was used. The time step size for RRT and forward dynamics was 0.01s. 10 consecutive planning tries were done with a maximum runtime of 100s. The statistics on successful tries of finding the first path are summarized in Table II.

TABLE II: Path Planning Statistics with 1D Hopper.

	R3T		RG-RRT		RRT	
	Time(s)	Nodes	Time(s)	Nodes	Time(s)	Nodes
Mean	2.39	530	47.23	4288		
Median	2.18	497	35.54	3403		
Max	5.01	1021	96.49	8006	N/A	N/A
Min	0.37	75	19.74	2194		
S.D.	1.28	265	30.92	2432		
Fails	0		2		10	

Fig. 6 shows the trees explored by each algorithm. R3T is the only algorithm that found a path consistently and quickly. Two properties from planning with reachable sets contribute to this result. Since the 1D hopper has no control input during flight phase, R3T can exploit this property using methods described in Sec. III. In addition, maintaining reachable sets allows for more accurate distance-to-goal calculation and goal identification, as discussed in Sec. V-A.

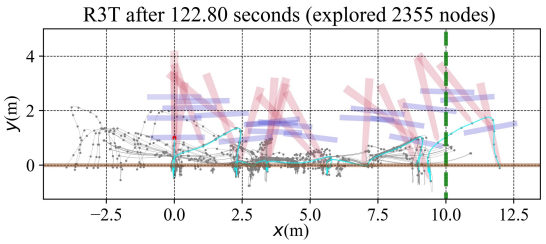


Fig. 7: R3T solution trajectory (cyan) for the 2D hopper. The tree explored is in grey. Snapshots of the hopper configurations (red for leg, purple for body) are shown.

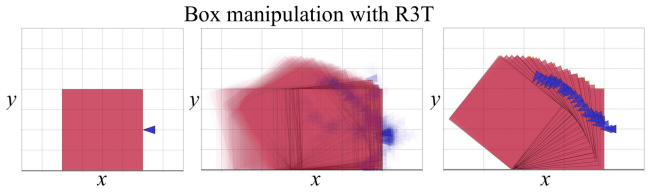


Fig. 8: The robot finger positions are shown in blue. [Left] Initial pose. [Middle] Reachable poses explored by R3T. [Right] A trajectory where the finger flips the box by tilting its center of mass to the left of the ground contact point.

D. 2D-Hopper

The 2D hopper robot [27], [28] has 10 states, 2 control inputs, and 2 contact modes [28]. The task is to make the robot hop from $x = 0$ to $x = 10$. A body-attitude controller was used in flight and the leg was modeled as a constant-k spring during compression. R3T was used to plan push-off and hip torque during contact. 10 consecutive runs were performed. R3T found a path in all runs using a median time of 106.3s and 2163 nodes. Fig. 7 shows a successful run.

E. Box Manipulation with a Robot Finger

We consider a contact-rich task of a robot finger manipulating a rigid box. two corner-ground and one finger-box contact points are modeled. The gravity is downward. Using Coulomb friction model, each contact point has 4 modes of sticking, sliding in each direction, and no contact. We use the hard contact model with time-stepping [21] and use local piecewise affine approximations in 8 states, 2 inputs, $4^3 = 64$ modes combinations in the state-space. The R3T algorithm was able to identify a broad range of mode sequences that correspond to rapid state exploration. Preliminary results on flipping the box by $90°$ are shown in Fig. 8.

VI. Conclusion

We introduced the R3T algorithm, a variant of RRT which takes advantage of reachable sets. We proposed a framework for R3T planning in nonlinear hybrid systems using local linearization, proved probabilistic completeness of R3T in kinodynamic systems, and presented a rewiring procedure that provides asymptotic optimality. Case studies showed R3T outperforms previous methods in speed and nodes explored and R3T can be applied to planning for complex hybrid systems.

REFERENCES

[1] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[2] J. J. Kuffner Jr and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *ICRA*, vol. 2, 2000.

[3] S. M. LaValle and J. J. Kuffner Jr, "Randomized kinodynamic planning," *The international journal of robotics research*, vol. 20, no. 5, pp. 378–400, 2001.

[4] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The international journal of robotics research*, vol. 30, no. 7, pp. 846–894, 2011.

[5] M. Elbanhawi and M. Simic, "Sampling-based robot motion planning: A review," *Ieee access*, vol. 2, pp. 56–77, 2014.

[6] D. J. Webb and J. Van Den Berg, "Kinodynamic rrt*: Asymptotically optimal motion planning for robots with linear dynamics," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 5054–5061.

[7] G. Goretkin, A. Perez, R. Platt, and G. Konidaris, "Optimal sampling-based planning for linear-quadratic kinodynamic systems," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 2429–2436.

[8] S. Karaman and E. Frazzoli, "Optimal kinodynamic motion planning using incremental sampling-based methods," in *49th IEEE conference on decision and control (CDC)*. IEEE, 2010, pp. 7681–7687.

[9] T. Kunz and M. Stilman, *Kinodynamic RRTs with Fixed Time Step and Best-Input Extension Are Not Probabilistically Complete*. Cham: Springer International Publishing, 2015, pp. 233–244. [Online]. Available: https://doi.org/10.1007/978-3-319-16595-0_14

[10] M. S. Branicky, M. M. Curtiss, J. A. Levine, and S. B. Morgan, "Rrts for nonlinear, discrete, and hybrid planning and control," in *42nd IEEE International Conference on Decision and Control (IEEE Cat. No. 03CH37475)*, vol. 1. IEEE, 2003, pp. 657–663.

[11] L. Liu, K. Yin, M. van de Panne, T. Shao, and W. Xu, "Sampling-based contact-rich motion control," in *ACM Transactions on Graphics (TOG)*, vol. 29, no. 4. ACM, 2010, p. 128.

[12] A. Shkolnik, M. Levashov, I. R. Manchester, and R. Tedrake, "Bounding on rough terrain with the littledog robot," *The International Journal of Robotics Research*, vol. 30, no. 2, pp. 192–215, 2011.

[13] A. Shkolnik, M. Walter, and R. Tedrake, "Reachability-guided sampling for planning under differential constraints," in *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 2859–2865.

[14] L. Jaillet, J. Hoffman, J. Van den Berg, P. Abbeel, J. M. Porta, and K. Goldberg, "Eg-rrt: Environment-guided random trees for kinodynamic motion planning with uncertainty and obstacles," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2011, pp. 2646–2652.

[15] N. Chavan-Dafle, R. Holladay, and A. Rodriguez, "In-hand manipulation via motion cones," *arXiv preprint arXiv:1810.00219*, 2018.

[16] E. Plaku, L. E. Kavraki, and M. Y. Vardi, "Hybrid systems: From verification to falsification," in *International Conference on Computer Aided Verification*. Springer, 2007, pp. 463–476.

[17] R. Geraerts and M. H. Overmars, "Reachability analysis of sampling based planners," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*. IEEE, 2005, pp. 404–410.

[18] G. M. Ziegler, *Lectures on polytopes*. Springer Science & Business Media, 2012, vol. 152.

[19] Z. Han and B. H. Krogh, "Reachability analysis of large-scale affine systems using low-dimensional polytopes," in *International Workshop on Hybrid Systems: Computation and Control*. Springer, 2006, pp. 287–301.

[20] S. Sadraddini and R. Tedrake, "Linear encodings for polytope containment problems," *arXiv preprint arXiv:1903.05214*, 2019.

[21] D. E. Stewart, "Rigid-body dynamics with friction and impact," *SIAM review*, vol. 42, no. 1, pp. 3–39, 2000.

[22] A. Wu, S. Sadraddini, and R. Tedrake, "The nearest polytope problem: Algorithms and application to controlling hybrid systems," in *2020 American Control Conference*. IEEE, 2020.

[23] M. Kleinbort, K. Solovey, Z. Littlefield, K. E. Bekris, and D. Halperin, "Probabilistic completeness of rrt for geometric and kinodynamic planning with forward propagation," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. x–xvi, April 2019.

[24] H. O. Mete and Z. B. Zabinsky, "Pattern hit-and-run for sampling efficiently on polytopes," *Operations Research Letters*, vol. 40, no. 1, pp. 6 – 11, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167637711001271

[25] K. Solovey, L. Janson, E. Schmerling, E. Frazzoli, and M. Pavone, "Revisiting the Asymptotic Optimality of RRT*," pp. 0–6, 2019. [Online]. Available: http://arxiv.org/abs/1909.09688

[26] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[27] M. H. Raibert, "Hopping in legged systems — modeling and simulation for the two-dimensional one-legged case," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SMC-14, no. 3, pp. 451–463, May 1984.

[28] R. Tedrake, "Applied optimal control for dynamically stable legged locomotion," Ph.D. dissertation, Massachusetts Institute of Technology, 2004.