A Rapidly-Exploring Random Trees Motion Planning Algorithm for Hybrid Dynamical Systems*

Nan Wang and Ricardo G. Sanfelice†

October 28, 2022

Abstract

This paper proposes a rapidly-exploring random trees (RRT) algorithm to solve the motion planning problem for hybrid systems. At each iteration, the proposed algorithm, called HyRRT, randomly picks a state sample and extends the search tree by flow or jump, which is also chosen randomly when both regimes are possible. Through a definition of concatenation of functions defined on hybrid time domains, we show that HyRRT is probabilistically complete, namely, the probability of failing to find a motion plan approaches zero as the number of iterations of the algorithm increases. This property is guaranteed under mild conditions on the data defining the motion plan, which include a relaxation of the usual positive clearance assumption imposed in the literature of classical systems. The motion plan is computed through the solution of two optimization problems, one associated with the flow and the other with the jumps of the system. The proposed algorithm is applied to a walking robot so as to highlight its generality and computational features.

1 Introduction

Motion planning consists of finding a state trajectory and associated inputs, connecting the initial and final state while satisfying the dynamics of the

1

system as well as a given safety criterion. Motion planning problems for purely continuous-time systems and purely discrete-time systems have been well studied in the literature; see, e.g., [1]. In recent years, various planning algorithms have been developed to solve motion planning problems, from graph search algorithms [2] to artificial potential field methods [3]. A main drawback of graph search algorithms is that the number of vertices grows exponentially as the dimension of states grows, which makes computing motion plans inefficient for high-dimensional systems. The artificial potential field method suffers from getting stuck at local minimum. Arguably, the most successful algorithm to solve motion planning problems for purely continuous-time systems and purely discrete-time systems is the sampling-based RRT algorithm [4]. This algorithm incrementally constructs a tree of state trajectories toward random samples in the state space. Similar to graph search algorithms, RRT suffers from the curse of dimensionality, but, in practice, achieves rapid exploration in solving high-dimensional motion planning problems [5]. Compared with the artificial potential field method, RRT is probabilistically complete [6], which means that the probability of failing to find a motion plan converges to zero, as the number of samples approaches infinity.

While RRT algorithms have been used to solve motion planning problems for purely continuous-time systems [6] and purely discrete-time systems [7], fewer efforts have been devoted to applying RRT-type algorithms to solve motion planning problems for systems with combined continuous and discrete behavior. In [8], a hybrid RRT algorithm is proposed for motion planning problems for a special class of hybrid systems, which follows the classical RRT scheme but does not establish key properties of the algorithm, such as probabilistic completeness.

This paper focuses on motion planning problems for hybrid systems modeled as hybrid equations [9]. In this modeling framework, differential and difference equations with constraints are used to describe the continuous and discrete behavior of the hybrid system, respectively. This general hybrid system framework can capture most hybrid systems emerging in robotic applications, not only the class of hybrid systems considered in [8], but also systems with memory states, timers, impulses, and constraints. For this broad class of hybrid systems, a motion planning algorithm is proposed in this paper. Following [6], the proposed algorithm, called HyRRT, incrementally constructs search trees, rooted in the initial state set and toward the random samples. At first, HyRRT draws samples from the state space. Then, it selects the vertex such that the state associated with this vertex has minimal distance to the sample. Next, HyRRT propagates the state tra-

jectory from the state associated with the selected vertex. Following [10], it is established that, under certain assumptions, HyRRT is probabilistically complete. To the authors' best knowledge, HyRRT is the first RRT-type algorithm for systems with hybrid dynamics that is probabilistically complete. The proposed algorithm is applied to a walking robot example so as to assess its capabilities.

The remainder of the paper is structured as follows. Section 2 presents notation and preliminaries. Section 3 presents the problem statement and introduction of application. Section 4 presents the HyRRT algorithm. Section 5 presents the analysis of the probabilistic completeness of HyRRT algorithm. Section 6 presents the illustration of HyRRT in the example. Due to space constraints, proofs will be published elsewhere.

2 Notation and Preliminaries

2.1 Notation

The real numbers are denoted as $\mathbb{R}$ and its nonnegative subset is denoted as $\mathbb{R}_{\geq 0}$. The set of nonnegative integers is denoted as $\mathbb{N}$. The notation $\text{int}\,I$ denotes the interior of the interval I. The notation $\overline{S}$ denotes the closure of the set S. The notation ∂S denotes the boundary of the set S. Given sets $P \subset \mathbb{R}^n$ and $Q \subset \mathbb{R}^n$, the Minkowski sum of P and Q, denoted as $P + Q$, is the set $\{p + q : p \in P, q \in Q\}$. The notation $|\cdot|$ denotes the Euclidean norm. The notation $\text{rge}\,f$ denotes the range of the function f. Given a point $x \in \mathbb{R}^n$ and a subset $S \subset \mathbb{R}^n$, the distance between x and S is denoted $\text{dist}(x, S) := \inf_{s \in S} |x - s|$. The notation $\mathbb{B}$ denotes the closed unit ball of appropriate dimension in the Euclidean norm.

2.2 Preliminaries

A hybrid system $\mathcal{H}$ with inputs is modeled as [9]

$$\mathcal{H} : \begin{cases} \dot{x} = f(x, u) & (x, u) \in C \\ x^+ = g(x, u) & (x, u) \in D \end{cases} \tag{1}$$

where $x \in \mathbb{R}^n$ is the state, $u \in \mathbb{R}^m$ is the input, $C \subset \mathbb{R}^n \times \mathbb{R}^m$ represents the flow set, $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ represents the flow map, $D \subset \mathbb{R}^n \times \mathbb{R}^m$ represents the jump set, and $g : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ represents the jump map, respectively. The continuous evolution of x is captured by the flow map f. The discrete evolution of x is captured by the jump map g. The flow set C

collects the points where the state can evolve continuously. The jump set D collects the points where jumps can occur.

Given a flow set C, the set $U_C := \{u \in \mathbb{R}^m : \exists x \in \mathbb{R}^n \text{ such that } (x, u) \in C\}$ includes all possible input values that can be applied during flows. Similarly, given a jump set D, the set $U_D := \{u \in \mathbb{R}^m : \exists x \in \mathbb{R}^n \text{ such that } (x, u) \in D\}$ includes all possible input values that can be applied at jumps. These sets satisfy $C \subset \mathbb{R}^n \times U_C$ and $D \subset \mathbb{R}^n \times U_D$. Given a set $K \subset \mathbb{R}^n \times U_\star$, where $\star$ is either C or D, we define $\Pi_\star(K) := \{x : \exists u \in U_\star \text{ s.t. } (x, u) \in K\}$ as the projection of K onto $\mathbb{R}^n$, and define $C' := \Pi_C(C)$ and $D' := \Pi_D(D)$.

In addition to ordinary time $t \in \mathbb{R}_{\geq 0}$, we employ $j \in \mathbb{N}$ to denote the number of jumps of the evolution of x and u for $\mathcal{H}$ in (1), leading to hybrid time (t, j) for the parameterization of its solutions and inputs. The domain of a solution to $\mathcal{H}$ is given by a hybrid time domain. A hybrid time domain is defined as a subset E of $\mathbb{R}_{\geq 0} \times \mathbb{N}$ that, for each $(T, J) \in E$, $E \cap ([0, T] \times \{0, 1, ..., J\})$ can be written as $\cup_{j=0}^{J}([t_j, t_{j+1}], j)$ for some finite sequence of times $0 = t_0 \leq t_1 \leq t_2 \leq ... \leq t_{J+1} = T$. A hybrid arc $\phi : \operatorname{dom} \phi \to \mathbb{R}^n$ is a function on a hybrid time domain that, for each $j \in \mathbb{N}$, $t \mapsto \phi(t, j)$ is locally absolutely continuous on each interval $I^j := \{t : (t, j) \in \operatorname{dom} \phi\}$ with nonempty interior. The definition of solution pair to a hybrid system is given as follows. For more details, see [9].

Definition 2.1. *(Solution pair to a hybrid system) Given a pair of functions $\phi : \operatorname{dom} \phi \to \mathbb{R}^n$ and $u : \operatorname{dom} u \to \mathbb{R}^m$, (ϕ, u) is a solution pair to (1) if $\operatorname{dom}(\phi, u) := \operatorname{dom} \phi = \operatorname{dom} u$ is a hybrid time domain, $(\phi(0, 0), u(0, 0)) \in \overline{C} \cup D$, and the following hold:*

1) For all $j \in \mathbb{N}$ such that I^j has nonempty interior,

    a) the function $t \mapsto \phi(t, j)$ is locally absolutely continuous,

    b) $(\phi(t, j), u(t, j)) \in C$ for all $t \in \operatorname{int} I^j$,

    c) the function $t \mapsto u(t, j)$ is Lebesgue measurable and locally bounded,

    d) for almost all $t \in I^j$, $\dot{\phi}(t, j) = f(\phi(t, j), u(t, j))$.

2) For all $(t, j) \in \operatorname{dom}(\phi, u)$ such that $(t, j+1) \in \operatorname{dom}(\phi, u)$,

$$(\phi(t, j), u(t, j)) \in D \quad \phi(t, j+1) = g(\phi(t, j), u(t, j)).$$

HyRRT requires concatenating solution pairs. The concatenation operation of solution pairs is defined next.

Definition 2.2. *(Concatenation operation) Given two functions $\phi_1 : \text{dom}\,\phi_1 \to \mathbb{R}^n$ and $\phi_2 : \text{dom}\,\phi_2 \to \mathbb{R}^n$, where $\text{dom}\,\phi_1$ and $\text{dom}\,\phi_2$ are hybrid time domains, ϕ_2 can be concatenated to ϕ_1 if ϕ_1 is compact and $\phi : \text{dom}\,\phi \to \mathbb{R}^n$ is the concatenation of ϕ_2 to ϕ_1, denoted $\phi = \phi_1|\phi_2$, namely,*

1) *$\text{dom}\,\phi = \text{dom}\,\phi_1 \cup (\text{dom}\,\phi_2 + \{(T, J)\})$, where $(T, J) = \max \text{dom}\,\phi_1$ and the plus sign denotes Minkowski addition;*

2) *$\phi(t, j) = \phi_1(t, j)$ for all $(t, j) \in \text{dom}\,\phi_1 \backslash \{(T, J)\}$ and $\phi(t, j) = \phi_2(t - T, j - J)$ for all $(t, j) \in \text{dom}\,\phi_2 + \{(T, J)\}$.*

In the main result of this paper, the following definition of closeness between hybrid arcs is used; see [9].

Definition 2.3. *((τ, ϵ)-closeness of hybrid arcs) Given $\tau, \epsilon > 0$, two hybrid arcs ϕ_1 and ϕ_2 are (τ, ϵ)-close if*

1. *for all $(t, j) \in \text{dom}\,\phi_1$ with $t + j \leq \tau$, there exists s such that $(s, j) \in \text{dom}\,\phi_2$, $|t - s| < \epsilon$, and $|\phi_1(t, j) - \phi_2(s, j)| < \epsilon$;*

2. *for all $(t, j) \in \text{dom}\,\phi_2$ with $t + j \leq \tau$, there exists s such that $(s, j) \in \text{dom}\,\phi_1$, $|t - s| < \epsilon$, and $|\phi_2(t, j) - \phi_1(s, j)| < \epsilon$.*

3 Problem Statement and Applications

The motion planning problem for hybrid systems studied in this paper is formulated as follows.

Problem 1. *Given a hybrid system $\mathcal{H}$ with input $u \in \mathbb{R}^m$ and state $x \in \mathbb{R}^n$, the initial state set $X_0 \subset \mathbb{R}^n$, the final state set $X_f \subset \mathbb{R}^n$, and the unsafe set $X_u \subset \mathbb{R}^n \times \mathbb{R}^m$, find a pair $(\phi, u) : \text{dom}(\phi, u) \to \mathbb{R}^n \times \mathbb{R}^m$, namely, a motion plan, such that for some $(T, J) \in \text{dom}(\phi, u)$, the following hold:*

1) *$\phi(0, 0) \in X_0$, namely, the initial state of the solution belongs to the given initial state set X_0;*

2) *(ϕ, u) is a solution pair to $\mathcal{H}$ as defined in Definition 2.1;*

3) *(T, J) is such that $\phi(T, J) \in X_f$, namely, the solution belongs to the final state set at hybrid time (T, J);*

4) *$(\phi(t, j), u(t, j)) \notin X_u$ for each $(t, j) \in \text{dom}(\phi, u)$ such that $t + j \leq T + J$, namely, the solution pair does not intersect with the unsafe set before its state trajectory reaches the final state set.*

Therefore, given sets X_0, X_f and X_u, and a hybrid system $\mathcal{H}$ with data (C, f, D, g), a motion planning problem $\mathcal{P}$ is formulated as $\mathcal{P} = (X_0, X_f, X_u, (C, f, D, g))$.

There are some interesting special cases of Problem 1. For example, when $D = \emptyset$ $(C = \emptyset)$ and C (D) is nonempty, then $\mathcal{P}$ denotes the motion planning problem for purely continuous-time (discrete-time, respectively) systems under constraints. Therefore, Problem 1 covers the motion planning problems for purely continuous-time and purely discrete-time system studied in [6] and [1]. Moreover, note that the unsafe set X_u can be used to constrain both states and inputs.

Problem 1 is illustrated in the following example.

Example 3.1. *(Walking robot) The state x of the compass model of a walking robot is composed of the angle vector θ and the velocity vector ω [11]. The angle vector θ contains the planted leg angle θ_p, the swing leg angle θ_s, and the torso angle θ_t. The velocity vector ω contains the planted leg angular velocity ω_p, the swing leg angular velocity ω_s, and the torso angular velocity ω_t. The input u is the input torque, where u_p is the torque applied on the planted leg from the ankle, u_s is the torque applied on the swing leg from the hip, and u_t is the torque applied on the torso from the hip. The continuous dynamics of $x = (\theta, \omega)$ comes from the Lagrangian method and is given by $\dot{\theta} = \omega, \dot{\omega} = D_f(\theta)^{-1}(-C_f(\theta, \omega)\omega - G_f(\theta) + Bu) =: \alpha(x, u)$ where D_f and C_f are the inertial and Coriolis matrices, respectively, and B is the actuator relationship matrix. In [12], the input torques that produce an acceleration a for a special state x are determined by a function μ, defined as $\mu(x, a) := B^{-1}(D_f(\theta)a + C_f(\theta, \omega)\omega + G_f(\theta))$. By applying $u = \mu(x, a)$ to $\dot{\omega} = \alpha(x, u)$, we obtain $\dot{\omega} = a$. Then, the flow map f is defined as*

$$f(x, a) := \begin{bmatrix} \omega \\ a \end{bmatrix} \quad \forall (x, a) \in C.$$

Flow is allowed when only one leg is in contact with the ground. To determine if the biped has reached the end of a step, a function h is defined as $h(x) := \phi_s - \theta_p$ for all $x \in \mathbb{R}^6$ where ϕ_s denotes the step angle. The condition $h(x) \geq 0$ indicates that only one leg is in contact with the ground. Thus, the flow set is given as $C := \{(x, a) \in \mathbb{R}^6 \times \mathbb{R}^3 : h(x) \geq 0\}$. Furthermore, a step occurs when the change of h is such that θ_p is approaching ϕ_s, and h equals zero. Thus, the jump set D is defined as $D := \{(x, a) \in \mathbb{R}^6 \times \mathbb{R}^3 : h(x) = 0, \omega_p \geq 0\}$.

Following [11], when a step occurs, the swing leg becomes the planted leg, and the planted leg becomes the swing leg. The function Γ is defined to swap

6

angles and velocity variables as $\theta^+ = \Gamma(\theta)$. The angular velocities after a step are determined by a contact model denoted as $\Omega(x) := (\Omega_p(x), \Omega_s(x), \Omega_t(x))$, where Ω_p, Ω_s, and Ω_t are the angular velocity of the planted leg, swing leg, and torso, respectively. Then, the jump map g is defined as

$$g(x, a) := \begin{bmatrix} \Gamma(\theta) \\ \Omega(x) \end{bmatrix} \quad \forall (x, a) \in D. \tag{2}$$

A particular motion planning problem for the walking robot is to generate a walking gait. The final state set is defined as $X_f = \{(\phi_s, -\phi_s, 0, 0.1, 0.1, 0)\}$ so that after the impact, the walking robot starts the next walking cycle. The initial state set is chosen as $X_0 = \{x_0 \in \mathbb{R}^6 : x_0 = g(x_f, 0), x_f \in X_f\}$. In setting X_0, the input argument of g can be set arbitrarily because input does not affect the value of g; see (2). In practice, there are constraints on the acceleration of the planted leg, swinging leg, and the torso, respectively. To capture these, the unsafe set is defined as $X_u = \{(x, a) \in \mathbb{R}^6 \times \mathbb{R}^3 : a_1 \notin [a_1^{\min}, a_1^{\max}]$ or $a_2 \notin [a_2^{\min}, a_2^{\max}]$ or $a_3 \notin [a_3^{\min}, a_3^{\max}]$ or $(x, a) \in D\}$, where $a_1^{\min}$, $a_2^{\min}$, and $a_3^{\min}$ are the lower bounds of a_1, a_2, and a_3, respectively, and $a_1^{\max}$, $a_2^{\max}$, and $a_3^{\max}$ are the upper bounds of a_1, a_2, and a_3, respectively.
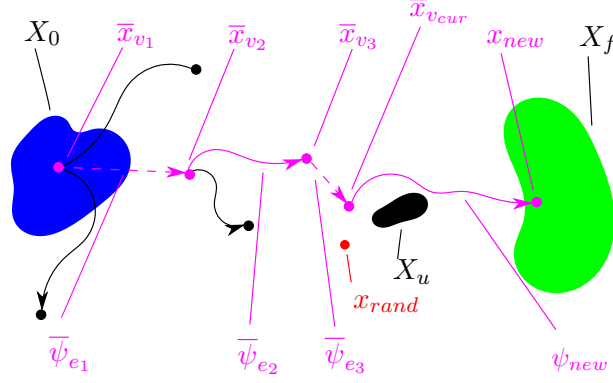
In the forthcoming Example 6.1, we employ HyRRT to solve this motion planning problem formulated in Example 3.1.

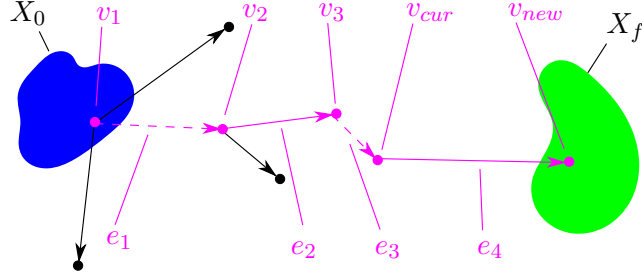# 4   HyRRT: A Motion Planning Algorithm for Hybrid Systems

4.1   Overview

HyRRT searches for a motion plan by incrementally constructing a search tree. The search tree is modeled by a directed tree. A directed tree $\mathcal{T}$ is a pair $\mathcal{T} = (V, E)$, where V is a set whose elements are called vertices and E is a set of paired vertices whose elements are called edges. The edges in the directed tree are directed, which means the pairs of vertices that represent edges are ordered. The set of edges E is defined as $E \subseteq \{(v_1, v_2) : v_1 \in V, v_2 \in V, v_1 \neq v_2\}$. The edge $e = (v_1, v_2) \in E$ represents an edge from v_1 to v_2. A path in $\mathcal{T} = (V, E)$ is a sequence of vertices $p = (v_1, v_2, ..., v_k)$ such that $(v_i, v_{i+1}) \in E$ for all $i = 1, 2, ..., k - 1$.

Each vertex in the search tree $\mathcal{T}$ is associated with a state value of $\mathcal{H}$. Each edge in the search tree is associated with a solution pair to $\mathcal{H}$ that connects the state values associated with their endpoint vertices. The state

(a) States and solution pairs.

(b) Search tree associated with the states and solution pairs in Figure 1(a).

Figure 1: The association between states/solution pairs and the vertices/edges in the search tree. The blue region denotes X_0, the green region denotes X_f, and the black region denotes X_u. The dots and lines between dots in Figure 1(b) denote the vertices and edges associated with the states and solution pairs in Figure 1(a). The path $p = (v_1, v_2, v_3, v_{cur}, v_{new})$ in the search graph in Figure 1(b) represents the solution pair $\tilde{\psi}_p = \overline{\psi}_{e_1} | \overline{\psi}_{e_2} | \overline{\psi}_{e_3} | \psi_{new}$ in Figure 1(a).

value associated with vertex $v \in V$ is denoted as $\overline{x}_v$ and the solution pair associated with edge $e \in E$ is denoted as $\overline{\psi}_e$, as shown in Figure 1. The solution pair that the path $p = (v_1, v_2, ..., v_k)$ represents is the concatenation of all those solutions associated with the edges therein, namely,

$$\tilde{\psi}_p := \overline{\psi}_{(v_1, v_2)} | \overline{\psi}_{(v_2, v_3)} | \; ... \; | \overline{\psi}_{(v_{k-1}, v_k)} \tag{3}$$

where $\tilde{\psi}_p$ denotes the solution pair associated with the path p. For the notion of concatenation, see Definition 2.2. An example of the path p and its associated solution pair $\tilde{\psi}_p$ is shown in Figure 1.

The proposed HyRRT algorithm requires a library of possible inputs. The input library $(\mathcal{U}_C, \mathcal{U}_D)$ includes the input signals that can be applied during flows (collected in $\mathcal{U}_C$) and the input values that can be applied at jumps (collected in $\mathcal{U}_D$).

Next, we introduce the main steps executed by HyRRT. Given the motion planning problem $\mathcal{P} = (X_0, X_f, X_u, (C, f, D, g))$ and the input library $(\mathcal{U}_C, \mathcal{U}_D)$, HyRRT performs the following steps:

Step 1: Sample a finite number of points from X_0 and initialize a search tree $\mathcal{T} = (V, E)$ by adding vertices associated with each sampling point.

Step 2: Randomly select one regime among flow regime and jump regime for the evolution of $\mathcal{H}$.

Step 3: Randomly select a point x_{rand} from C' (D') if the flow (jump, respectively) regime is selected in Step 2.

Step 4: Find the vertex associated with the state value that has minimal Euclidean distance to x_{rand}, denoted v_{cur}, as is shown in Figure 1(b).

Step 5: Randomly select an input signal (value) from $\mathcal{U}_C$ ($\mathcal{U}_D$) if the flow (jump, respectively) regime is selected. Then, compute a solution pair starting from $\overline{x}_{v_{cur}}$ with the selected input applied, denoted $\psi_{new} = (\phi_{new}, u_{new})$. Denote the final state of ϕ_{new} as x_{new}, as is shown in Figure 1(a). If ψ_{new} does not intersect with X_u, add a vertex v_{new} associated with x_{new} to V and an edge (v_{cur}, v_{new}) associated with ψ_{new} to E. Then, go to **Step 2**.

9

4.2 HyRRT Algorithm

Following the overview in Section 4.1, the proposed algorithm is given in Algorithm 1. The inputs of Algorithm 1 are the problem $\mathcal{P} = (X_0, X_f, X_u, (C, f, D, g))$, the input library $(\mathcal{U}_C, \mathcal{U}_D)$, a parameter $p_n \in (0, 1)$, which tunes the probability of proceeding with the flow regime or the jump regime, an upper bound $K \in \mathbb{N}_{>0}$ for the number of iterations to execute, and two tunable sets $X_c \supset \overline{C'}$ and $X_d \supset D'$, which act as constraints in finding a closest vertex to x_{rand}. Each function in Algorithm 1 is defined next.

4.2.1 $\mathcal{T}.init(X_0)$

The function call $\mathcal{T}.init$ is used to initialize a search tree $\mathcal{T} = (V, E)$. It randomly selects a finite number of points from X_0. For each sampling point x_0, a vertex v_0 associated with x_0 is added to V. At this step, no edge is added to E.

4.2.2 $x_{rand} \leftarrow random\_state(S)$

The function call $random\_state$ randomly selects a point from the set $S \subset \mathbb{R}^n$. It is designed to select from $\overline{C'}$ and D' separately depending on the value of r rather than to select from $\overline{C'} \cup D'$. The reason is that if $\overline{C'}$ (D') has zero measure while D' ($\overline{C'}$) does not, the probability that the point selected from $\overline{C'} \cup D'$ lies in $\overline{C'}$ (D', respectively) is zero, which would prevent establishing probabilistic completeness.

4.2.3 $v_{cur} \leftarrow nearest\_neighbor(x_{rand}, \mathcal{T}, \mathcal{H}, flag)$

The function call $nearest\_neighbor$ searches for a vertex v_{cur} in the search tree $\mathcal{T} = (V, E)$ such that its associated state value has minimal distance to x_{rand}. This function is implemented as follows.

- When $flag = flow$, the following optimization problem is solved over X_c.

  Problem 2. *Given a hybrid system $\mathcal{H} = (C, f, D, g)$, $x_{rand} \in \overline{C'}$, and a search tree $\mathcal{T} = (V, E)$, solve*

$$\underset{v \in V}{\arg\min} \quad |\overline{x}_v - x_{rand}|$$
$$s.t. \quad \overline{x}_v \in X_c.$$

- When $flag = jump$, the following optimization problem is solved over X_d.

Problem 3. *Given a hybrid system $\mathcal{H} = (C, f, D, g)$, $x_{rand} \in D'$, and a search tree $\mathcal{T} = (V, E)$, solve*

$$\underset{v \in V}{\arg\min} \quad |\overline{x}_v - x_{rand}|$$
$$s.t. \quad \overline{x}_v \in X_d.$$

The data of Problem 2 and Problem 3 comes from the arguments of the *nearest_neighbor* function call. This optimization problem can be solved by traversing all the vertices in $\mathcal{T} = (V, E)$.

4.2.4 $return \leftarrow new\_state(x_{rand}, v_{cur}, (\mathcal{U}_C, \mathcal{U}_D), \mathcal{H}, X_u,$
$\qquad x_{new}, \psi_{new})$

If $\overline{x}_{v_{cur}} \in \overline{C'} \backslash D'$ ($\overline{x}_{v_{cur}} \in D' \backslash \overline{C'}$), the function call *new_state* generates a new solution pair ψ_{new} to hybrid system $\mathcal{H}$ starting from $\overline{x}_{v_{cur}}$ by applying a input signal $\tilde{u}$ (an input value u_D) randomly selected from $\mathcal{U}_C$ ($\mathcal{U}_D$, respectively). If $\overline{x}_{v_{cur}} \in \overline{C'} \cap D'$, then this function generates ψ_{new} by randomly selecting flows or jump. The final state of ψ_{new} is denoted as x_{new}.

Note that the choices of inputs are random. Some RRT variants choose the optimal input that drives x_{new} closest to x_{rand}. However, [13] proves that such a choice makes the RRT algorithm probabilistically incomplete. After ψ_{new} and x_{new} are generated, the function *new_state* checks if there exists $(t, j) \in \text{dom } \psi_{new}$ such that $\psi_{new}(t, j) \in X_u$. If so, then ψ_{new} intersects with the unsafe set and *new_state* returns *false*. Otherwise, this function returns *true*.

4.2.5 $v_{new} \leftarrow \mathcal{T}.add\_vertex(x_{new})$ **and** $\mathcal{T}.add\_edge(v_{cur}, v_{new}, \psi_{new})$

The function call $\mathcal{T}.add\_vertex(x_{new})$ adds a new vertex v_{new} associated with x_{new} to $\mathcal{T}$ and returns v_{new}. The function call $\mathcal{T}.add\_edge(v_{cur}, v_{new}, \psi_{new})$ adds a new edge $e_{new} = (v_{cur}, v_{new})$ associated with ψ_{new} to $\mathcal{T}$.

4.3 Solution Checking during HyRRT Construction

When the function call *extend* returns *Reached* or *Advanced*, a solution checking function is employed to check if a path in $\mathcal{T}$ can be used to construct a motion plan to the given motion planning problem. If this function finds a path $p = ((v_0, v_1), (v_1, v_2), ..., (v_{n-1}, v_n)) =: (e_0, e_1, ..., e_{n-1})$ in $\mathcal{T}$ such that

1) $\overline{x}_{v_0} \in X_0$ and 2) $\overline{x}_{v_n} \in X_f$, then the solution pair $\tilde{\psi}_p$ is a motion plan to the given motion planning problem. In practice, item 2) is too restrictive. Given $\epsilon > 0$ representing the tolerance with this condition, we implement item 2) as $\text{dist}(\overline{x}_{v_n}, X_f) \leq \epsilon$.

Algorithm 1 HyRRT algorithm

Input: $X_0, X_f, X_u, \mathcal{H} = (C, f, D, g), (\mathcal{U}_C, \mathcal{U}_D), p_n \in (0, 1), K \in \mathbb{N}_{>0}$

1: $\mathcal{T}.init(X_0)$;
2: **for** $k = 1$ to K **do**
3:     randomly select a real number r from $[0, 1]$;
4:     **if** $r \leq p_n$ **then**
5:         $x_{rand} \leftarrow random\_state(\overline{C'})$;
6:         $extend(\mathcal{T}, x_{rand}, (\mathcal{U}_C, \mathcal{U}_D), \mathcal{H}, X_u, flow)$;
7:     **else**
8:         $x_{rand} \leftarrow random\_state(D')$;
9:         $extend(\mathcal{T}, x_{rand}, (\mathcal{U}_C, \mathcal{U}_D), \mathcal{H}, X_u, jump)$;
10:     **end if**
11: **end for**
12: **return** $\mathcal{T}$;

$extend(\mathcal{T}, x, (\mathcal{U}_C, \mathcal{U}_D), \mathcal{H}, X_u, flag)$

1: $v_{cur} \leftarrow nearest\_neighbor(x, \mathcal{T}, \mathcal{H}, flag)$;
2: **if** $new\_state(x, v_{cur}, (\mathcal{U}_C, \mathcal{U}_D), \mathcal{H}, X_u, x_{new}, \psi_{new})$ **then**
3:     $v_{new} \leftarrow \mathcal{T}.add\_vertex(x_{new})$;
4:     $\mathcal{T}.add\_edge(v_{cur}, v_{new}, \psi_{new})$;
5:     **if** $x_{new} == x$ **then**
6:         **return** *Reached*;
7:     **else**
8:         **return** *Advanced*;
9:     **end if**
10: **end if**
11: **return** *Trapped*;

---

5 Probabilistic Completeness Analysis

This section analyzes the probabilistic completeness property of HyRRT algorithm. Probabilistic completeness means that the probability that the planner fails to return a motion plan, if it exists, approaches zero as the

number of samples approaches infinity. Section 5.1 presents the preliminaries to establish the probabilistic completeness. Section 5.2 presents our main result showing that the HyRRT algorithm is probabilistically complete under certain assumptions.

5.1   Preliminaries about Probabilistic Completeness

The following defines the clearance of a motion plan.

Definition 5.1. *(clearance of a solution pair) Given a motion plan $\psi = (\phi, u)$ to the motion planning problem $\mathcal{P} = (X_0, X_f, X_u, (C, f, D, g))$, the clearance of $\psi = (\phi, u)$ is equal to the maximal $\delta_{clear} > 0$ if the following hold:*

1) *For all $(t, j) \in \operatorname{dom} \psi$ such that I^j has nonempty interior, $(\phi(t, j) + \delta_{clear}\mathbb{B}, u(t, j) + \delta_{clear}\mathbb{B}) \subset C$;*

2) *For all $(t, j) \in \operatorname{dom} \psi$ such that $(t, j+1) \in \operatorname{dom} \psi$, $(\phi(t, j) + \delta_{clear}\mathbb{B}, u(t, j) + \delta_{clear}\mathbb{B}) \subset D$;*

3) *For all $(t, j) \in \operatorname{dom} \psi$, $(\phi(t, j) + \delta_{clear}\mathbb{B}, u(t, j) + \delta_{clear}\mathbb{B}) \cap X_u = \emptyset$.*

The following assumption is imposed on the input library.

Assumption 5.2. *The input library $(\mathcal{U}_C, \mathcal{U}_D)$ is such that*

1) *Each input signal in $\mathcal{U}_C$ is constant and $\mathcal{U}_C$ includes all possible input signals such that their time domains are subsets of the interval $[0, T_m]$ for some $T_m > 0$ and their images belong to U_C. In other words, there exists $T_m > 0$ such that $\mathcal{U}_C = \{\tilde{u} : \operatorname{dom} \tilde{u} = [0, T] \subset [0, T_m], \tilde{u}$ is constant and $\operatorname{rge} \tilde{u} \in U_C\}$;*

2) *$\mathcal{U}_D = U_D$.*

The following assumption is imposed on the random selection in HyRRT.

Assumption 5.3. *The probability distributions of the random selection in the function calls T.init, random_state, and new_state are the uniform distribution.*

The following assumptions are imposed on the flow map f and the jump map g of the hybrid system $\mathcal{H}$ in (1).

Assumption 5.4. *The flow map f is Lipschitz continuous. In particular, there exist $K_x^f, K_u^f \in \mathbb{R}_{>0}$ such that, for all (x_0, x_1, u_0, u_1) such that $(x_0, u_0) \in C$, $(x_0, u_1) \in C$, and $(x_1, u_0) \in C$,*

$$|f(x_0, u_0) - f(x_1, u_0)| \leq K_x^f |x_0 - x_1|$$
$$|f(x_0, u_0) - f(x_0, u_1)| \leq K_u^f |u_0 - u_1|.$$

Assumption 5.5. *The jump map g is such that there exist $K_x^g \in \mathbb{R}_{>0}$ and $K_u^g \in \mathbb{R}_{>0}$ such that, for all $(x_0, u_0) \in D$ and $(x_1, u_1) \in D$,*

$$|g(x_0, u_0) - g(x_1, u_1)| \leq K_x^g |x_0 - x_1| + K_u^g |u_0 - u_1|.$$

The following assumption assumes that the existing motion plan is away from the boundary of initial state set, final state set, and unsafe set, and uses a piecewise-constant input during flows.

Assumption 5.6. *Given a motion planning problem $\mathcal{P} = (X_0, X_f, X_u, (C, f, D, g))$, there exists a motion plan $\psi = (\phi, u)$ to $\mathcal{P}$ such that for some $\delta' > 0$*

1. *$\phi(0, 0) + \delta' \mathbb{B} \subset X_0$;*

2. *$\phi(T, J) + \delta' \mathbb{B} \subset X_f$, where $(T, J) = \max \operatorname{dom} \psi$;*

3. *for all $(t, j) \in \operatorname{dom} \psi$, $(\phi(t, j) + \delta' \mathbb{B}, u(t, j) + \delta' \mathbb{B}) \cap X_u = \emptyset$;*

4. *for all $j \in \mathbb{N}$ such that I^j has nonempty interior, $t \mapsto u(t, j)$ is piecewise constant with resolution Δt.*

5.2 Inflated Hybrid System and Main Result

In the probabilistic completeness result in [10, Theorem 2], a motion plan with positive clearance is assumed to exist. However, such assumption is restrictive for hybrid systems. Indeed, if the motion plan reaches the boundary of the flow set or of the jump set, then the motion plan has no clearance. To overcome this issue and to assure that HyRRT is probabilistically complete, the hybrid system $\mathcal{H} = (C, f, D, g)$ is modified as follows.

Definition 5.7. *(δ-inflation of hybrid system) Given a hybrid system $\mathcal{H} = (C, f, D, g)$ and $\delta > 0$, the δ-inflation of the hybrid system $\mathcal{H}$, denoted $\mathcal{H}_\delta$, is given by*

$$\mathcal{H}_\delta : \begin{cases} \dot{x} = f_\delta(x, u) & (x, u) \in C_\delta \\ x^+ = g_\delta(x, u) & (x, u) \in D_\delta \end{cases} \tag{4}$$

where

14

1) $C_\delta := \{(x,u) \in \mathbb{R}^n \times \mathbb{R}^m : \exists (y,v) \in C \text{ such that } x \in y + \delta\mathbb{B}, u \in v + \delta\mathbb{B}\}$,

2) $f_\delta(x,u) := f(x,u) \quad \forall (x,u) \in C_\delta$,

3) $D_\delta := \{(x,u) \in \mathbb{R}^n \times \mathbb{R}^m : \exists (y,v) \in D \text{ such that } x \in y + \delta\mathbb{B}, u \in v + \delta\mathbb{B}\}$,

4) $g_\delta(x,u) := g(x,u) \quad \forall (x,u) \in D_\delta$.

Note that any solution to $\mathcal{H}$ in (1) is a solution to its inflation in (4). The clearance property in Definition 5.1 is satisfied for free since items 1) and 2) therein are satisfied by constructing C_δ and D_δ, and item 3) therein is satisfied by item 3) in Assumption 5.6. Next, we state our main result.

Theorem 5.8. *Given a motion planning problem $\mathcal{P} = (X_0, X_f, X_u, (C, f, D, g))$, suppose that Assumptions 5.2, 5.3, 5.4, and 5.5 are satisfied and that there exists a motion plan (ϕ, u) to $\mathcal{P}$ satisfying Assumption 5.6 for some $\delta' > 0$. When HyRRT is used to solve the problem $\mathcal{P}_\delta = (X_0, X_f, X_u, (C_\delta, f_\delta, D_\delta, g_\delta))$, where, for some $\delta > 0$, $(C_\delta, f_\delta, D_\delta, g_\delta)$ denotes the δ-inflation of (C, f, D, g) in (4), the probability that HyRRT fails to find a motion plan $\psi' = (\phi', u')$ to $\mathcal{P}_\delta$ such that ϕ' is $(\tilde{\tau}, \tilde{\delta})$-close to ϕ after k iterations is at most $a \exp(-bk)$, for some constant $a, b \in \mathbb{R}_{>0}$, where $(T, J) = \max \operatorname{dom} \psi$, $(T', J') = \max \operatorname{dom} \psi'$, $\tilde{\tau} = \max\{T + J, T' + J'\}$, and $\tilde{\delta} = \min\{\delta, \delta'\}$.*

6 HyRRT Software Tool for Motion Planning for Hybrid Systems and Examples

Algorithm 1 leads to a software tool[1] to solve the motion planning problems for hybrid systems. This software only requires the motion planning problem data $(X_0, X_f, X_u, (C, f, D, g))$, an input library $(\mathcal{U}_C, \mathcal{U}_D)$, a tunable parameter $p_n \in (0, 1)$, an upper bound K over the iteration number and two constraint sets X_c and X_d. The tool is illustrated in Example 3.1. We have successfully applied HyRRT to other hybrid systems, including the actuated bouncing ball and a point-mass robotics manipulator.

Example 6.1. *(Walking robot system in Example 3.1, revisited) The simulation result in Figure 2 with tolerance ϵ set to 0.3 shows that HyRRT is able to solve the instance of motion planning problem for the walking robot.*

---

[1]Code at https://github.com/HybridSystemsLab/hybridRRT.

In this simulation, the constraint set X_c is chosen as $\{(x,a) \in \mathbb{R}^6 \times \mathbb{R}^3 : h(x) \geq -s\}$ and X_d as $\{(x,a) \in \mathbb{R}^6 \times \mathbb{R}^3 : h(x) = 0, \omega_p \geq -s\}$ with a tunable parameter s set to 0, 0.3, 0.5, 1, and 2, such that $C = X_c|_{s=0} \subsetneq X_c|_{s=0.3} \subsetneq X_c|_{s=0.5} \subsetneq X_c|_{s=1} \subsetneq X_c|_{s=2}$ and $D = X_d|_{s=0} \subsetneq X_d|_{s=0.3} \subsetneq X_d|_{s=0.5} \subsetneq X_d|_{s=1} \subsetneq X_d|_{s=2}$.

The simulation is implemented in MATLAB and processed by a 3.5 GHz Intel Core i5 processor. The simulation takes $71.5/85.3/99.4/167.7/242.8$ seconds with s set to $0/0.3/0.5/1.0/2.0$, respectively. The simulation takes at least 71.5 seconds to finish. Compared with the forward/backward propagation algorithm based on breadth-first search which takes 1608.2 seconds to solve the same problem, the improvement provided by the rapid exploration is significant: 95.5% computation time improvement. It is also observed that as the sets X_c and X_d grow, HyRRT considers more vertices in solving Problems 2 and 3 leading to higher computation time.
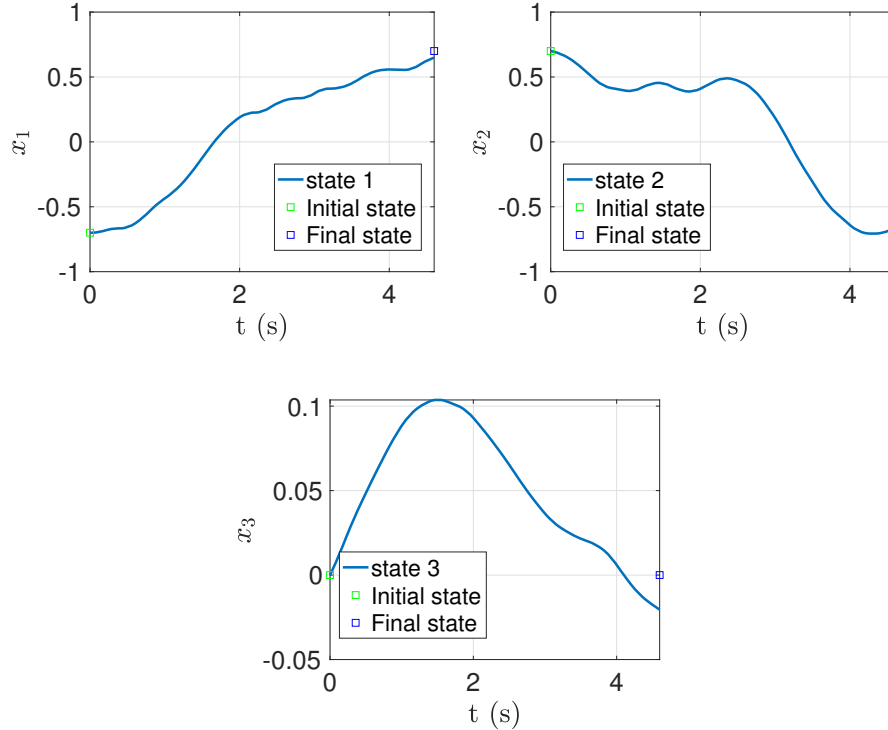

Figure 2: Selected state trajectories of the generated motion plan for the walking robot system. In each figure above, the green and blue squares denote the corresponding initial and final state components, respectively.

The software tool also succeeds in finding motion plans for the actuated

bouncing ball and point-mass robotics manipulator systems.

7 Conclusion and Future Work

In this paper, a HyRRT algorithm is proposed to solve motion planning problems for hybrid systems. The proposed algorithm is illustrated in the walking robot example and the results show its capacity to solve the problem. In addition, this paper provides a result showing HyRRT algorithm is probabilistically complete under mild assumptions. Future research direction includes the optimal motion planning.

References

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

[2] M. Likhachev, D. I. Ferguson, G. J. Gordon, A. Stentz, and S. Thrun, "Anytime dynamic A*: An anytime, replanning algorithm,," in *ICAPS*, vol. 5, 2005, pp. 262–271.

[3] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in *Autonomous Robot Vehicles*. Springer, 1986, pp. 396–404.

[4] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[5] P. Cheng, "Sampling-based motion planning with differential constraints," Tech. Rep., 2005.

[6] 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.

[7] 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.

[8] M. S. Branicky, M. M. Curtiss, J. Levine, and S. Morgan, "Sampling-based planning and control," in *Proceedings of the 12th Yale Workshop on Adaptive and Learning Systems, New Haven, CT*. Citeseer, 2003.

[9] R. G. Sanfelice, "Hybrid feedback control," 2021.

[10] 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, 2018.

[11] J. W. Grizzle, G. Abba, and F. Plestan, "Asymptotically stable walking for biped robots: Analysis via systems with impulse effects," *IEEE Transactions on Automatic Control*, vol. 46, no. 1, pp. 51–64, 2001.

[12] B. E. Short and R. G. Sanfelice, "A hybrid predictive control approach to trajectory tracking for a fully actuated biped," in *2018 Annual American Control Conference (ACC)*.   IEEE, 2018, pp. 3526–3531.

[13] T. Kunz and M. Stilman, "Kinodynamic RRTs with fixed time step and best-input extension are not probabilistically complete," in *Algorithmic Foundations of Robotics XI*.   Springer, 2015, pp. 233–244.