International Workshop on the Algorithmic Foundations of Robotics (WAFR), 2014

Kinodynamic RRTs with Fixed Time Step and Best-Input Extension Are Not Probabilistically Complete

Tobias Kunz and Mike Stilman

Georgia Tech, Atlanta, GA 30332, USA
tobias@gatech.edu

Abstract. RRTs are a popular method for kinodynamic planning that many consider to be probabilistically complete. However, different variations of the RRT algorithm exist and not all of them are probabilistically complete. The tree can be extended using a fixed or variable time step. The input can be chosen randomly or the best input can be chosen such that the new child node is as close as possible to the sampled state according to the used distance metric. It has been shown that for finite input sets an RRT using a fixed step size with a randomly selected input is probabilistically complete. However, this variant is uncommon since it is less efficient. We prove that the more common variant of choosing the best input in combination with a fixed time step is not probabilistically complete.

1 Introduction

Rapidly-Exploring Random Trees (RRTs) as introduced by LaValle and Kuffner [11,13,14,15,16] are a popular method for geometric and kinodynamic planning. Many, e.g. [4,5,7], consider RRTs to be a synonym for probabilistic completeness. However, this is not necessarily the case. Kinodynamic RRTs [13,14,15,16] only have the property of probabilistic completeness under a set of assumptions, which depend on implementation details that are left open by the RRT algorithm. These details govern how the time step and the input are chosen to extend the tree from the selected node. While it has been shown that the RRT algorithm for kinodynamic planning is probabilistically complete with a fixed time step and a random control input [15, 16], we now describe that a more commonly used variant is not probabilistically complete in the general case. This variant uses a fixed time step and chooses the best control input for the extension of the tree from the selected node. This variant is for example used in [1, 2, 4, 6, 8, 18].

Even though we prove this variant to not be probabilistically complete in general, it could potentially be made probabilistically complete by introducing additional requirements on the system dynamics and/or the used distance metric. In fact, one of the goals of this paper is to spur further research on the exact conditions under which RRTs are probabilistically complete.

1.1 Problem Formulation

In this analysis, consider a system with differential constraints given as

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) \tag{1}$$

with state $\boldsymbol{x} \in \mathcal{X}$ and input $\boldsymbol{u} \in \mathcal{U}$.

The set of all collision-free states is given as $\mathcal{X}_{\text{free}} \subseteq \mathcal{X}$. An initial state $\boldsymbol{x}_{\text{init}} \in \mathcal{X}_{\text{free}}$ and a goal set $\mathcal{X}_{\text{goal}} \subseteq \mathcal{X}_{\text{free}}$ are given. We want to find a duration T and an input trajectory $\boldsymbol{u}(t)$ such that the differential constraints of Eq. 1 are satisfied for all $0 \leq t \leq T$, the trajectory is collision free with $x(t) \in \mathcal{X}_{\text{free}}$ for all $0 < t < T$, $\boldsymbol{x}(0) = \boldsymbol{x}_{\text{init}}$ and $\boldsymbol{x}(T) \in \mathcal{X}_{\text{goal}}$.

1.2 Kinodynamic RRT Algorithm

A distance function $\rho : \mathcal{X} \times \mathcal{X} \to [0, \infty)$ is given, which establishes a concept of closeness between states and is used by the RRT algorithm to extend the tree. Most commonly the Euclidean distance is used.

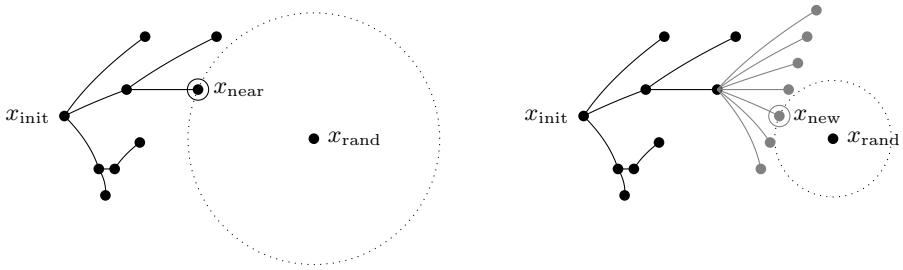
Algorithm 1 shows the construction of an RRT. Lavalle and Kuffner introduced different variants of the RRT algorithm. All RRT variants grow a tree from $\boldsymbol{x}_{\text{init}}$ by sampling the state space (line 4) and then selecting the node in the tree closest to the sampled state according to the provided distance function (line 5). This is visualized in Fig. 1(a). The NewState function (line 6) extends the tree from the selected node by applying some input $\boldsymbol{u} \in \mathcal{U}$ for some time step Δt. Variants of the RRT algorithm differ in how Δt and $\boldsymbol{u}$ are chosen.

Algorithm 1: BuildRRT($x_{\text{init}}, \mathcal{X}_{\text{goal}}$)

```
1  V ← {x_init};
2  E ← ∅;
3  while V ∩ X_goal = ∅ do
4  |   x_rand ← SampleState();
5  |   x_near ← NearestNeighbor(V, x_rand);
6  |   (x_new, u_new, Δt) ← NewState(x_near, x_rand);
7  |   if CollisionFree(x_near, x_new, u_new, Δt) then
8  |   |   V ← V ∪ {x_new};
9  |   |   E ← E ∪ {(x_near, x_new, u_new, Δt)};
10 return (V, E);
```

Early work on RRTs [13, 14] used a fixed time step Δt and chose the best input $\boldsymbol{u}$. Each input $\boldsymbol{u}$ is associated with a successor state, in which the system will end up when applying the input for a fixed time Δt from the current node. "Best input" refers to the input whose successor state is closest to the sampled state. This is visualized in Fig. 1(b) and formalized in Algorithm 2.

The Simulate function used in Algorithm 2 returns a successor state by simulating the system forward by a given time step Δt using a given constant input $\boldsymbol{u}$. I. e. it returns $\boldsymbol{x}(\Delta t)$, such that the differential equation $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u})$ with the initial condition $\boldsymbol{x}(0) = \boldsymbol{x}_{\text{near}}$ is satisfied.



(a) Select nearest node (b) Select best input

Fig. 1. Visualization of best-input RRT variant. The shown system is a double integrator with $\dot{x}_1 = x_2$, $\dot{x}_2 = u$ and finite input set $\mathcal{U}$.

Algorithm 2: NewState($x_{\text{near}}, x_{\text{rand}}$)
(using fixed time step and best-input extension)

1 $\boldsymbol{u}_{\text{new}} \leftarrow \arg\min_{\boldsymbol{u} \in \mathcal{U}} \{\rho(\text{Simulate}(\boldsymbol{x}_{\text{near}}, \boldsymbol{u}, \Delta t), \boldsymbol{x}_{\text{rand}})\}$;
2 $\boldsymbol{x}_{\text{new}} \leftarrow \text{Simulate}(\boldsymbol{x}_{\text{near}}, \boldsymbol{u}_{\text{new}}, \Delta t)$;
3 **return** $(x_{\text{new}}, u_{\text{new}}, \Delta t)$;

If $\mathcal{U}$ is finite, the best input in line 1 of Algorithm 2 can be chosen by forward simulating all inputs and evaluating all resulting successor states. If $\mathcal{U}$ is continuous, this is not possible. Instead an analytical method must be used for an exact solution. However, often the best input is approximated instead by choosing the best one out of a finite number of sampled inputs.

Later, [15, 16] generalized the RRT algorithm and gave choices for the implementation of the NewState function. The time step Δt can either be fixed or variable and either the best or a random input $\boldsymbol{u}$ can be chosen. Algorithm 1 is general enough to allow all these variations. However, when the time step is fixed, the algorithm and data structures can be simplified by leaving out Δt.

1.3 Probabilistic Completeness of Kinodynamic RRTs

An algorithm is probabilistically complete if the probability that an existing solution is found converges to 1 as the number of iterations grows to infinity [17].

It has been shown in [15, 16] that if $\mathcal{U}$ is finite, Δt is fixed and $\boldsymbol{u}$ is chosen at random, the RRT algorithm is probabilistically complete. However, choosing $\boldsymbol{u}$ randomly may not result in the RRT exploring the state space rapidly.

	Fixed Δt	Variable Δt
Random $\boldsymbol{u}$	Probabilistically complete (if $\mathcal{U}$ finite) [15, 16]	?
Best $\boldsymbol{u}$	Not probabilistically complete [this paper]	?

Table 1. Probabilistic completeness of different kinodynamic RRT variants

In contrast, the preliminary RRT variant introduced in [13, 14] also uses a fixed time step Δt but chooses the best input $\boldsymbol{u}$. The very first paper on RRTs [13] but none of the later papers [14, 15, 16] claimed this variant to be "probabilistically complete under very general conditions". We show that this variant is not probabilistically complete.

Restricting the RRT to a fixed time step renders the algorithm unable to find solutions that do not consist of Δt long segments of constant input. However, even if a solution with Δt long segments of constant input exists, the RRT with best-input extension might never find it.

This section up until here is summarized in Table 1.

As mentioned in Sec. 1.2 the best input out of a continuous input set is often approximated in practice by sampling a finite set of inputs and choosing the best input out of the finite set. This approximation may render the RRT algorithm probabilistically complete because of the added randomness. However, an algorithm that is probabilistically complete only thanks to approximation errors is likely to not be very efficient.

1.4 RRTs Using Steering Methods

A steering method is able to exactly connect any two states $x_1, x_2 \in \mathcal{X}$ with $\|x_1 - x_2\| < \epsilon$ for some $\epsilon > 0$ while ignoring obstacles. Computationally efficient steering methods are not available for general dynamical systems. They are only available for a few simple systems, e.g. Dubin's car [3, 17] and a set of double integrators [12]. A steering method in combination with a collision checker yields what is called a local planner in the probabilistic roadmap literature [10].

To be generally applicable, kinodynamic RRTs as introduced in [13,14,15,16] do not require a steering method. Instead, they only rely on an incremental simulator that can simulate the system forward for a given input and time step. However, there are RRT algorithms that make use of a steering method. These are not the topic of this paper. However, we want to briefly mention them in this section to make the differences clear and to emphasize that the negative result on probabilistic completeness presented in this paper does not apply to those.

A steering method usually returns a trajectory that minimizes some cost, e. g. time. When using a steering method, the distance function used by the RRT is also based on this steering method by defining the distance as the optimal cost to move between two states ignoring obstacles. Karaman and Frazzoli [9] proved that an RRT* using an optimal steering method in combination with a distance function based on that steering method is probabilistically complete. Since an RRT* uses the same vertices as an RRT, the RRT algorithm is also probabilistically complete under these assumptions.

Geometric RRT planners [11] that use a Euclidean distance function and connect configurations with a straight line in configuration space can also be viewed as using a steering method and fit into the framework assumed in [9]. The straight line is the trajectory that minimizes path length and the distance function returns that path length.

Whereas RRT planners using steering methods have been most successful in practice and come with guarantees on probabilistic completeness, not requiring a steering method was one of the selling points when the RRT was initially introduced.

2 Proof

We demonstrate that a kinodynamic RRT with fixed time steps and best-input extension is not generally probabilistically complete. The proof uses a counter example.

The RRT variant we are considering here selects both the node and the input by evaluating closeness to the sampled state according to the provided distance metric ρ. In order for a node to get selected it must be the closest one to the sample. The same goes for the input: In order to be selected, the successor state resulting from the input must be the closest one to the sample among all the successor states resulting from applying inputs from the current node. Even though for every node and for every input there exist states such that the considered node or the considered successor state is closest, in order for a specific input to be selected for extension from a specific node, more is required: (1) The specific node must be the closest to the sample *and* (2) among all the successor states resulting from applying inputs from the specific node, the state resulting from the specific input must be the closest. We provide an example case in which there is no state that could be sampled that satisfies both requirements.

The system used as counter example is described in Sec. 2.1. In Sec. 2.2 we present a possible intermediate tree and in Sec. 2.4 we demonstrate that the considered RRT variant cannot explore the full reachable state space from that intermediate tree because there exists a node and an input such that no sampled state results in selecting both of them. Sec. 2.3 provides some background of Voronoi regions, which are used in the proof in Sec. 2.4.

2.1 Counter Example

Consider the following system with a 2-dimensional state vector $[x_1, x_2]$, a scalar input u and no obstacles.

$$\dot{x}_1 = u \tag{2}$$

$$\dot{x}_2 = u^2 - 3 \tag{3}$$

$$\text{with} \quad |u| \leq 1 \tag{4}$$

Note that $-4 \leq \dot{x}_2 \leq -2$ and thus the system is always moving in negative direction along the x_2 axis. The set of possible successor states after a time step of Δt from the current state is a segment of a parabola. Fig. 2 shows the set of states reachable within $3\Delta t$ from some initial state $\boldsymbol{x}_{\text{init}}$ assuming constant input during a fixed time step Δt.

Observe that the fact that the system is restricted to always move in the negative direction of the x_2 axis makes it *impossible to revisit an earlier state.* Also, all states at $t = \Delta t$ and $t = 2\Delta t$ are only reachable at one specific point in time.

Our counter example uses a Euclidean distance for the RRT algorithm.

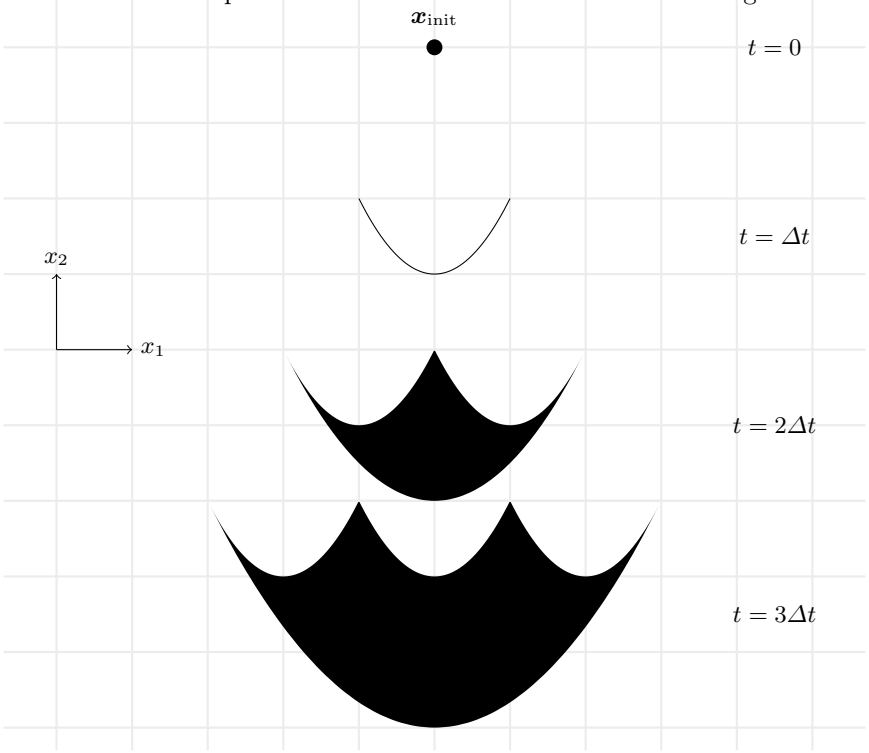


Fig. 2. States reachable from $\boldsymbol{x}_{\text{init}}$

2.2 Intermediate Tree

A probabilistically complete algorithm must be able to explore the whole reachable space from any intermediate tree that the algorithm might produce. Fig. 3 shows what the RRT could look like after two extensions from the initial state. The new nodes $\boldsymbol{x}_a$ and $\boldsymbol{x}_b$ sit at the ends of the parabola segment that represents the reachable space at time Δt.

If the algorithm was probabilistically complete, it would still be able to explore the whole reachable space. However, we show that given this tree configuration, the RRT is never going to explore the state space areas shown in gray, even though they are reachable by the system. The parabola segment at $t = \Delta t$ is never explored except its endpoints. The unexplored space at $t = 2\Delta t$ and $t = 3\Delta t$ is just the result of the unexplored parabola segment at $t = \Delta t$, since getting there requires moving through a state in the interior of the parabola segment. Also, note that the unexplored space at $t = 2\Delta t$ and $t = 3\Delta t$ does not play a role for our proof, since the inability of the RRT to explore the interior of the parabola segment is enough for it to not be probabilistically complete. Part of the unexplored space at $t = 3\Delta t$ could potentially still be explored at $t = 4\Delta t$, since it overlaps with the reachable space at $t = 4\Delta t$, which is not shown in the figure.

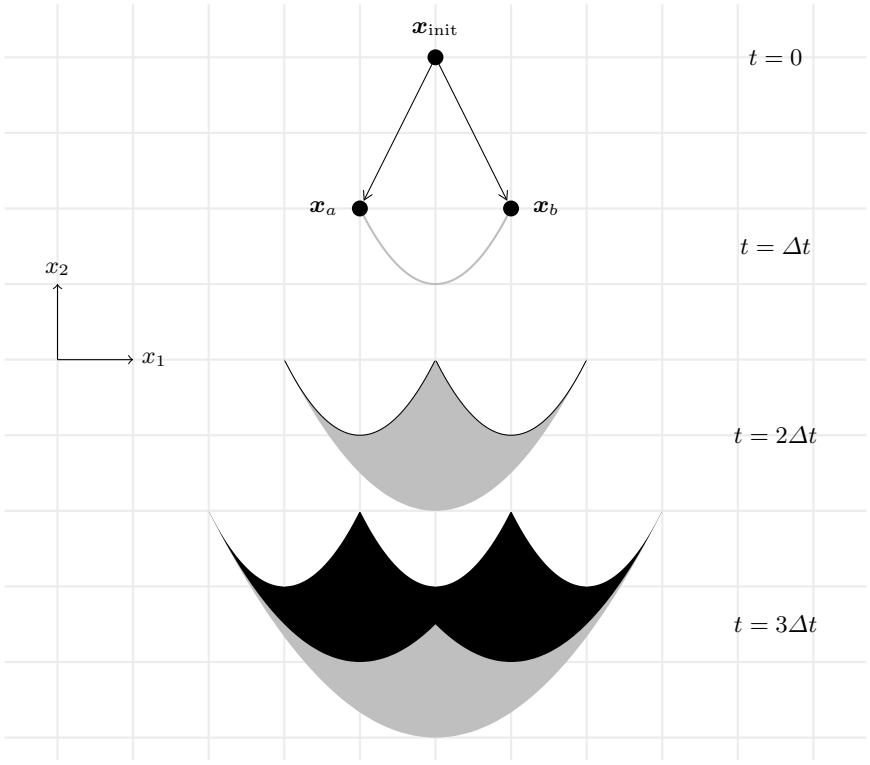


Fig. 3. RRT after two extensions. Gray areas of the state space are never explored.

2.3 Background: Voronoi Regions of Sites

Even though Voronoi regions are a well-known concept, we are going to review them in this section since our proof in the next section uses the less common concept of a Voronoi region of an infinite set of points instead of only Voronoi regions of single points.

Consider k subsets $S_i \subset \mathcal{X}$ with $i = 1 \dots k$ such that $\forall i \neq j : S_i \cap S_j = \emptyset$. The sets S_i are called *sites*. The Voronoi region of site S_i is the set of all points that is closer to S_i according to our distance metric ρ than to any other site. Or more formally

$$\mathrm{Vor}(S_i) = \{x \in \mathcal{X} \mid \exists p \in S_i \ \forall j = 1 \dots k \ \forall q \in S_j \colon \rho(x,p) \leq \rho(x,q)\} \qquad (5)$$

Note that in the common case all sites S_i only contain a single point, but we are also going to make use of a site S_i containing infinitely many points. Also note that $\mathrm{Vor}(S_i)$ does not only depend on S_i but on all S_j with $j = 1 \dots k$. A Voronoi diagram is a tuple $(\mathrm{Vor}(S_i))_{i \in \{1 \dots k\}}$ of all the k Voronoi regions.

2.4 Non-Exploration of Parabola Segment

We now look closer at $t = \Delta t$ to determine why the interior of the parabola segment is not explored by the RRT algorithm. As mentioned in Sec. 2.1, because the example system is constrained to always move in negative x_2 direction, the states on the parabola segment can only be reached at time $t = \Delta t$. Thus, the parabola segment can only be explored by extending the tree from the root node.

To extend the tree to the parabola segment, the random sample of the RRT algorithm must fall in the Voronoi region of the root node. The Voronoi regions of the three tree nodes (which are the Voronoi sites here) are shown in Fig. 4. The root node's Voronoi region is shaded with lines.

Now assume the RRT samples somewhere in the root node's Voronoi region and thus selects the root node as the nearest neighbor for extension. The next step is to choose the input to use to extend the tree from the root node. The RRT variant we are considering chooses the input such that the distance of the new child node to the sampled state is minimized. Similar to the way the closest node to the sample gets picked by the RRT algorithm, now the closest successor state of the selected node gets picked. We will now look at Voronoi regions of different successor states of the root node. We consider three sites and their Voronoi regions. Two sites are defined to be the two end points of the parabola segment and the third site is the entire rest, the interior, of the parabola segment. Note that the latter Voronoi site consists of infinitely many states. The three Voronoi regions of those sites are shown in Fig. 5. The Voronoi region of the interior of the parabola segment is shaded with dots.

For the RRT to explore the interior of the parabola segment, the sampled state must lie in both, the Voronoi region of the root node and the Voronoi region of the interior of the parabola segment. However, as Fig. 6 shows, the two Voronoi regions don't overlap. Thus, the RRT cannot explore the interior of the parabola segment and the algorithm is not probabilistically complete.

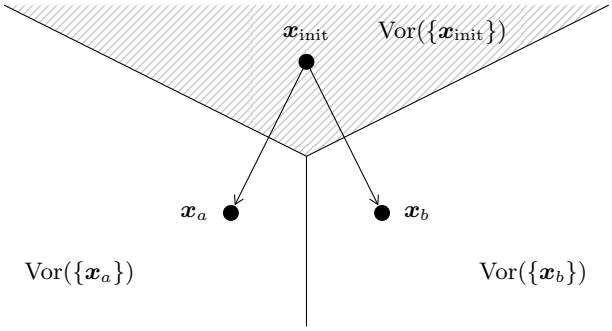


Fig. 4. Voronoi diagram of the three tree nodes, i. e. of the three Voronoi sites $S_1 = \{\boldsymbol{x}_{\text{init}}\}$, $S_2 = \{\boldsymbol{x}_a\}$ and $S_3 = \{\boldsymbol{x}_b\}$. The root node's Voronoi region is shaded with lines.

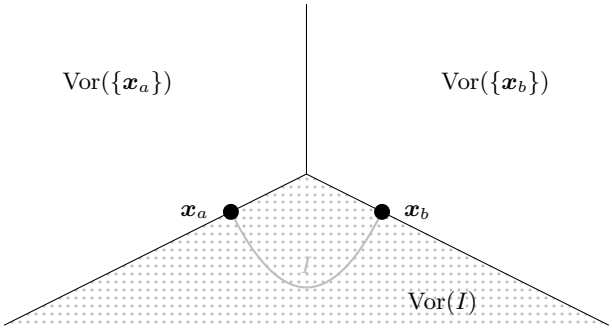


Fig. 5. Voronoi diagram of the successor states of the root node. Three Voronoi sites are considered: the two endpoints of the parabola segment, $S_1 = \{\boldsymbol{x}_a\}$ and $S_2 = \{\boldsymbol{x}_b\}$, and its interior $S_3 = I$. The Voronoi region of the interior I of the parabola segment is shaded with dots.

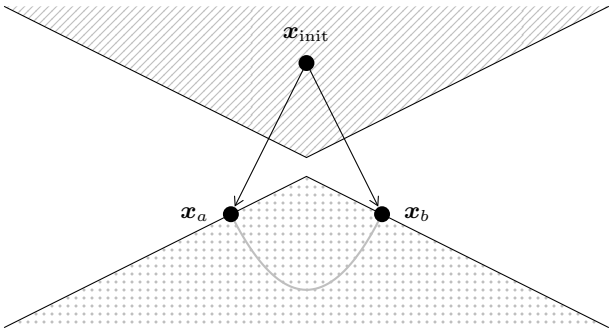


Fig. 6. Combining the two Voronoi diagrams from Fig. 4 and 5: Voronoi regions of the root node (lines) and the interior of the parabola segment (dots). They do not overlap.

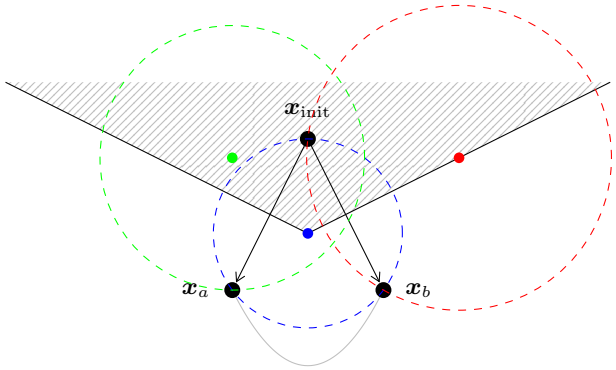


Fig. 7. Points within the root node's Voronoi region are closer to either one of the endpoints of the parabola segment than to its interior.

Fig. 7 provides a slightly different illustration of the same fact that every sample in the root node's Voronoi region is closer to one of the endpoints of the parabola segment than to its interior. The figure shows three exemplary points within the root node's Voronoi region. The dashed circles around them show that the closest point on the parabola segment is always one of the endpoints.

2.5 Discrete Inputs

Above proof can easily be extended to the discrete case. For example we can replace the entire interior of the parabola by a single input that leads to a state in the center of the parabola segment. This means Eq. 4 is replaced by $u \in \{-1, 0, 1\}$. The voronoi regions for this discrete counter example are shown in Fig. 8. Similarly to the continuous case, the zero-input state shown in gray will never be explored.

However, in the discrete-input case the RRT algorithm can be easily adapted to be probabilistically complete by making sure that no input is applied to the same node twice [17]. This forces the RRT to eventually try to expand all inputs of a node. This adaption is not possible in the continuous-input case.

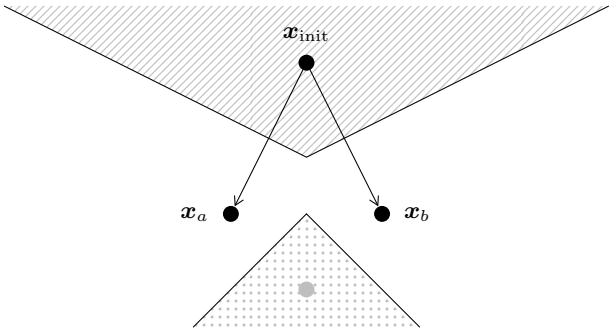


Fig. 8. Discrete case: Voronoi regions of the root node (lines) and the zero-input state (dots)

3 Conclusion

We showed that a common variant of kinodynamic RRTs is not probabilistically complete. This contradicts general perception that RRTs are inherently probabilistically complete. Instead, probabilistic completeness depends on the implementation details of the RRT, the specific problem and/or the chosen distance metric. Whether the RRT variant considered here can be made probabilistically complete by introducing constraints on the problem or distance metric is left open for further research. The question whether kinodynamic RRTs with a variable time step are probabilistically complete is also left open.

Even though RRTs were initially designed for not requiring a steering method, the finding in this paper provides an argument for using RRTs with a steering method as we do in [12].

Acknowledgments

This paper is dedicated to the memory of Mike Stilman.
This work was supported in part by ONR grant N00014-14-1-0120.

References

1. Bhatia, A., Frazzoli, E.: Incremental search methods for reachability analysis of continuous and hybrid systems. In: Alur, R., Pappas, G. (eds.) Hybrid Systems: Computation and Control, Lecture Notes in Computer Science, vol. 2993, pp. 142–156. Springer (2004)
2. Cheng, P., LaValle, S.: Resolution complete rapidly-exploring random trees. IEEE Int. Conf. on Robotics and Automation (2002)
3. Dubins, L.E.: 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(3), pp. 497–516 (1957)
4. Esposito, J.M., Kim, J., Kumar, V.: Adaptive RRTs for validating hybrid robotic control systems. In: Algorithmic Foundations of Robotics VI, pp. 107–121. Springer (2005)
5. Frazzoli, E., Dahleh, M.A., Feron, E.: Real-time motion planning for agile autonomous vehicles. Journal of Guidance, Control, and Dynamics 25(1), 116–129 (2002)
6. Glassman, E., Tedrake, R.: A quadratic regulator-based heuristic for rapidly exploring state space. IEEE Int. Conf. on Robotics and Automation (2010)
7. Goerzen, C., Kong, Z., Mettler, B.: A survey of motion planning algorithms from the perspective of autonomous UAV guidance. Journal of Intelligent and Robotic Systems 57(1-4), 65–100 (2010)
8. Kalisiak, M., van de Panne, M.: RRT-blossom: RRT with a local flood-fill behavior. IEEE Int. Conf. on Robotics and Automation (2006)
9. Karaman, S., Frazzoli, E.: Optimal kinodynamic motion planning using incremental sampling-based methods. IEEE Conf. on Decision and Control (2010)
10. Kavraki, L., Svestka, P., Latombe, J.C., Overmars, M.: Probabilistic roadmaps for path planning in high-dimensional configuration spaces. IEEE Transactions on Robotics and Automation 12(4), 566–580 (1996)

11. Kuffner, J.J., LaValle, S.M.: RRT-connect: An efficient approach to single-query path planning. IEEE Int. Conf. on Robotics and Automation (2000)
12. Kunz, T., Stilman, M.: Probabilistically complete kinodynamic planning for robot manipulators with acceleration limits. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (2014)
13. LaValle, S.M.: Rapidly-exploring random trees: A new tool for path planning. Tech. Rep. 98-11, Computer Science Dept., Iowa State University (1998)
14. LaValle, S.M., Kuffner, J.J.: Randomized kinodynamic planning. IEEE Int. Conf. on Robotics and Automation (1999)
15. LaValle, S.M., Kuffner, J.J.: Rapidly-exploring random trees: Progress and prospects. In: Algorithmic and Computational Robotics: New Directions 2000 WAFR, pp. 293–308 (2000)
16. LaValle, S.M., Kuffner, J.J.: Randomized kinodynamic planning. The Int. Journal of Robotics Research 20(5), 378–400 (2001)
17. LaValle, S.M.: Planning algorithms. Cambridge University Press (2006)
18. Petti, S., Fraichard, T.: Safe motion planning in dynamic environments. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (2005)