Rapidly-Exploring Random Trees: A New Tool for Path Planning

Steven M. LaValle
Department of Computer Science
Iowa State University
Ames, IA 50011 USA
lavalle@cs.iastate.edu

Abstract

We introduce the concept of a Rapidly-exploring Random Tree (RRT) as a randomized data structure that is designed for a broad class of path planning problems. While they share many of the beneficial properties of existing randomized planning techniques, RRTs are specifically designed to handle nonholonomic constraints (including dynamics) and high degrees of freedom. An RRT is iteratively expanded by applying control inputs that drive the system slightly toward randomly-selected points, as opposed to requiring point-to-point convergence, as in the probabilistic roadmap approach. Several desirable properties and a basic implementation of RRTs are discussed. To date, we have successfully applied RRTs to holonomic, nonholonomic, and kinodynamic planning problems of up to twelve degrees of freedom.

1 Introduction

Over the past decade, several randomized approaches have been proposed and successfully applied to the general problem of path planning in a high-dimensional configuration space. Two of the more popular approaches include the randomized potential field algorithm (e.g., [2]) and the probabilistic roadmap algorithm (e.g., [1, 4]). Given these successes, and the fact that there is little hope of ever obtaining an efficient, general path planning algorithm, it is natural to ask: Why do we need *another* randomized path planning technique?

The primary difficulty with existing techniques is that, although powerful for standard path planning, they do not naturally extend to general nonholonomic planning problems. Using state-space representations, this class of problems includes kinodynamic planning [3], which is an extremely general and important area in robotics, virtual prototyping, and many other applications. The randomized potential field method depends heavily on the choice of a good heuristic potential function, which becomes a daunting task when confronted with obstacles, kinematic differential constraints, and dynamical constraints. In the probabilistic roadmap approach, a graph is constructed in the configuration space by generating random configurations and attempting to connect pairs of nearby configurations with a

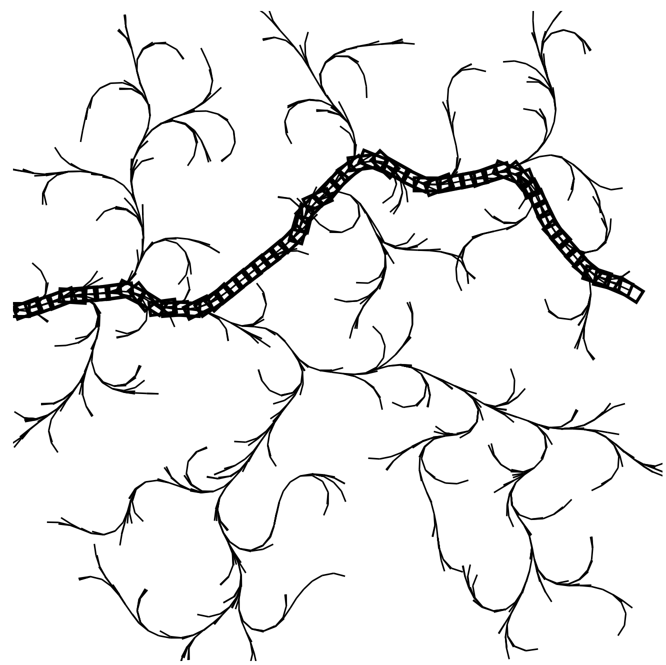


Figure 1: A 2D projection of a 5D RRT for a kinodynamic car.

local planner that will connect pairs of configurations. For planning of holonomic systems or steerable nonholonomic systems (see [6] and references therein), the local planning step might be efficient; however, in general the connection problem can be as difficult as designing a nonlinear controller, particularly for complicated nonholonomic and dynamical systems. The probabilistic roadmap technique might require the connections of thousands of configurations or states to find a solution, and if each connection is akin to a nonlinear control problem, it seems impractical for many nonholonomic (and kinodynamic) problems that arise in robotics and related areas.

In this paper, we introduce a randomized data structure for path planning that is designed for problems that have nonholonomic constraints. This leads to the introduction of a Rapidly-exploring Random Tree (RRT), which is defined in Section 2. An RRT includes some of the same desirable properties as a probabilistic roadmap. Both are designed with as few heuristics and arbitrary parameters as possible. This tends to lead to better performance analysis and consistency of behavior. It also facilitates the adaptation of the methods to related applications. The unique advantage of RRTs is that they can be directly applied to nonholonomic and kinodynamic planning. This advantage stems from the fact that RRTs do not require any connections to be made between pairs of configurations (or states), while probabilistic roadmaps typically require tens of thousands of connections. As discussed shortly, RRTs might be more efficient than a basic probabilistic roadmap for holonomic path planning.

2 Rapidly-Exploring Random Trees

Path planning will generally be viewed as a search in a metric space, X, for a continuous path from an initial state, x_{init} to a goal region $X_{goal} \subset X$ or goal state x_{goal}. We use the term *state space* to indicate a greater generality than is usually considered in path planning. For a standard problem, $X = \mathcal{C}$, which is the configuration space of a rigid body or system of bodies in a 2D or 3D world [5]. For a kinodynamic planning problem, $X = T(\mathcal{C})$, which is the tangent bundle of the configuration space [7] (a state encodes both configuration and velocity). Many other interpretations of X are possible.

It is assumed that a fixed *obstacle region*, $X_{obs} \subset X$ must be avoided, and that an explicit representation of X_{obs} is not available. One can only check whether a given state lies in X_{obs}. States in X_{obs} could correspond to velocity bounds, configurations at which a robot is in collision with an obstacle in the world, or several other interpretations, depending on the application. A Rapidly-exploring Random Tree (RRT) will be constructed so that all of its vertices are states in X_{free}, the complement of X_{obs}. Furthermore, each edge of the RRT will correspond to a path that lies entirely in X_{free}.

A *state transition equation* of the form $\dot{x} = f(x, u)$ is defined to express the nonholonomic constraints. The vector u is selected from a set, U, of *inputs*. The vector $\dot{x}$ denotes the derivative of state with respect to time. This control-theoretic representation is powerful enough to encode virtually any kinematic and dynamical model. By integrating f over a fixed time interval, Δt, the next state, x_{new} can be determined for a given initial state, x, and input $u \in U$. Using Euler integration, $x_{new} \approx x + f(x, u)\Delta t$; however, it is usually preferable to use a higher-order integration technique, such as Runge-Kutta. Let NEW_STATE$(x, u, \Delta t)$ denote an algorithm that returns x_{new}.

For holonomic planning, one can define $f(x, u) = u$, and $\|u\| \leq 1$, which implies that any bounded velocity can be achieved. After integrating f over Δt, a new state can be obtained that moves the system in any direction relative to x. For a nonholonomic problem, the next state is constrained due to the choice of f.

For a given initial state, x_{init}, an RRT, $\mathcal{T}$, with K vertices is constructed as shown below:

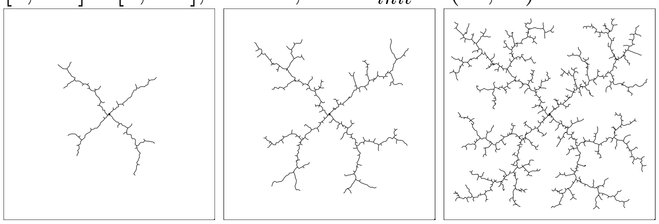
GENERATE_RRT$(x_{init}, K, \Delta t)$
1 $\mathcal{T}$.init(x_{init});
2 **for** $k = 1$ **to** K **do**
3 $\quad x_{rand} \leftarrow$ RANDOM_STATE();
4 $\quad x_{near} \leftarrow$ NEAREST_NEIGHBOR$(x_{rand}, \mathcal{T})$;
5 $\quad u \leftarrow$ SELECT_INPUT(x_{rand}, x_{near});
6 $\quad x_{new} \leftarrow$ NEW_STATE$(x_{near}, u, \Delta t)$;
7 $\quad \mathcal{T}$.add_vertex(x_{new});
8 $\quad \mathcal{T}$.add_edge(x_{near}, x_{new}, u);
9 Return $\mathcal{T}$

Let ρ denote a distance metric on the state space. The first vertex of $\mathcal{T}$ is $x_{init} \in X_{free}$. In each iteration, a random state, x_{rand}, is selected from X (it is assumed that X is bounded). Step 4 finds the closest vertex to x_{rand} in terms of ρ. Step 5 selects an input, u, that minimizes the distance from x_{near} to x_{rand}, and ensures that the state remains in X_{free}. Collision detection can be performed by an incremental method such as Mirtich's V-Clip. NEW_STATE is called on each input to evaluate a potential new state (if U is not finite, it can be discretized, or an alternative optimization procedure can be used). The new state, x_{new}, which is obtained by applying u, is added as a vertex to $\mathcal{T}$. An edge from x_{near} to x_{new} is also added, and the input u is recorded with the edge (because this input must be applied to reach x_{new} from x_{near}).

3 Nice Properties of RRTs

This section presents several properties of RRTs, which make them ideally suited for a wide variety of practical planning problems. The key advantages of RRTs are: 1) the expansion of an RRT is heavily biased toward unexplored portions of the state space; 2) the distribution of vertices in an RRT approaches the sampling distribution, leading to consistent behavior; 3) an RRT is probabilistically complete under very general conditions; 4) the RRT algorithm is relatively simple, which facilitates performance analysis (this is also a preferred feature of probabilistic roadmaps); 5) an RRT always remains connected, even though the number of edges is minimal; 6) an RRT can be considered as a path planning module, which can be adapted and incorporated into a wide variety of planning systems; 7) entire path planning algorithms can be constructed without requiring the ability to steer the system between two prescribed states, which greatly broadens the applicability of RRTs.

To gain a better understanding of RRTs, consider the special case in which X is a bounded, convex region in the plane. Assume that a holonomoic model is used, implying that $f = u$ and $U = \{u \in \Re^2 \mid \|u\| \leq 1\}$. Let ρ represent the Euclidean metric. The frames below show the construction of an RRT for the case of $X = [0, 100] \times [0, 100]$, $\Delta t = 1$, and $x_{init} = (50, 50)$:



The RRT quickly expands in a few directions to quickly explore the four corners of the square. Although the construction method is simple, it is no easy task to find a method that yields such desirable behavior. Consider, for example, a naive random tree that is constructed incrementally by selecting a vertex at random, an input at random, and then applying the input to generate a new vertex. Although one might intuitively expect the tree to "randomly" explore the space, there is actually a very strong bias toward places already explored (our simulation experiments yielded an extremely high density of vertices near x_{init}, with little other exploration). A random walk also suffers from a bias toward places already visited. An RRT works in the opposite manner by being biased toward places not yet visited. This can be seen by considering the Voronoi diagram of the RRT vertices. Larger Voronoi regions occur on the "frontier" of the tree. Since vertex selection is based on nearest neighbors, this implies that vertices with large Voronoi regions are more likely to be selected for expansion. On average, an RRT is constructed by iteratively breaking large Voronoi regions into smaller ones.

Based on simulation experiments, such as the one shown above, we have concluded that the generated paths are not far from optimal and that the vertices will eventually become uniformly distributed. Even though the paths appear jagged, note that no spiraling occurs. Based on several experiments in 2D, convex spaces, the optimal path to the root in comparison to the path in the RRT, differ on average by a factor of 1.3 to 2.0. Uniformity of the RRT vertices was repeatedly confirmed by the passing of several Chi-square tests, which are typically used to evaluate random number generators.

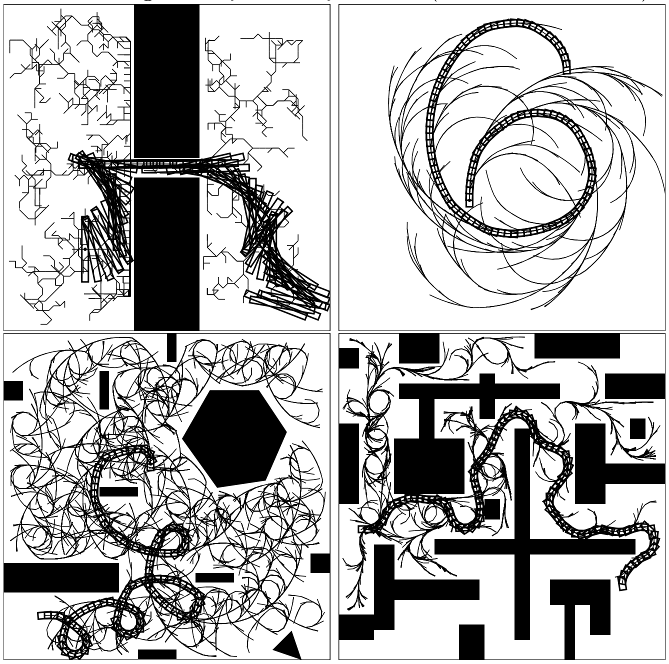
It is not difficult to prove that the vertices will become uniformly distributed. As the RRT initially expands, the vertices are clearly not uniformly distributed; however, the probability that a randomly-chosen point lies within Δt of a vertex of the tree eventually approaches one. In this case, the random sample will be added as a vertex to the tree. If the samples are generated uniformly, the vertices in the tree will become uniform. This result is independent of the initial vertex location (also confirmed by our experiments)! In general, if the points x_{rand} are sampled from any smooth probability density function, $p(x)$, the vertices of the RRT will distributed according to $p(x)$. This property is very useful for generating biasing schemes. A crucial piece of analysis that remains open is the rate of convergence.

For interesting planning problems, X will be nonconvex. In this case, the RRT vertices will still become uniformly distributed; however, one would expect the rate of convergence to be slower. This leads to a probabilistically complete [4] holonomic planner. Ideal performance could be obtained by defining a metric, ρ, that yields the length of the shortest path between two states, but determining this metric is as difficult as solving the path planning problem. All randomized path planning methods suffer from the difficulty of determining or estimating the ideal metric. In the case of nonholonomic systems, the resulting RRT remains probabilistically complete under fairly general conditions; however, convergence issues become even more important. For kinodynamic planning, the ideal metric (or pseudometric, due to asymmetry) would be one that gives the cost of the optimal trajectory between any two states. Once again, determining this metric is as hard as solving the original problem. Thus, we (and others) are forced to use simple metrics, hoping that convergence will be fast in practice.

Based on our preliminary experiments, it appears that RRTs might be faster than the basic probabilistic roadmap approach for holonomic planning problems. An RRT is minimal in the sense that it is always able to maintain a connected structure with the fewest edges. A probabilistic roadmap often suffers in performance because many extra edges are generated in attempts to form a connected roadmap. RRTs also require single nearest-neighbor queries, while probabilistic roadmaps require more-expensive k-nearest neighbor queries. Collision detection is a key bottleneck in path planning, and an RRT is completely suited for incremental collision detection. This allows the fastest-avaliable collision detection algorithms to be applied for every collision check. For these reasons and our preliminary observations from experimentation, it appears that an RRT-based planner may generally yield better performance than a probabilistic roadmap-based planner; however, it is difficult to make a conclusive experimental comparison.

4 Examples

Several illustrative examples of RRTs are presented here. In a related paper [7], we presented an RRT-based planner that computes collision-free kinodynamic trajectories that fire thrusters for hovercrafts and satellites in cluttered 2D and 3D environments. Several complicated problems were solved, including uncontrollable systems and a 3D rigid body with dynamics (a 12D state space).



Each example above shows a 2D rigid body that moves in a 2D environment. The projection of the RRT into the plane is shown, along with a computed path for the robot. In the upper left, a solution to a tightly-constrained 3D holonomic planning problem is shown. In the upper right, an RRT is shown for car that is only allowed to move forward and turn right in varying degrees. The lower left shows a computed solution for an uncontrollable car in a cluttered environment. The car is only capable of moving forward and turning left in three different increments (it cannot even move straight). Figure 1 shows an RRT and a computed trajectory for a 5DOF dynamical model of a car. A solution path for this same model in a cluttered environment is shown above in the lower right. The current implementation neglects many efficiency issues; nevertheless, the computational performance is encouraging so far.

5 Research Issues

Although our experiments with RRTs have been successful, many challenging issues remain. Efficient nearest-neighbor techniques are needed, which has been a topic of active interest in computational geometry. There are a variety of ways to embed an RRT into a planner. Efficient planners can be designed by generating multiple RRTs (for example, one rooted at x_{init} and another rooted at x_{goal}). An RRT could replace the random walk stage in a randomized potential field approach. For some problems, it might be preferable to obtain multiple, homotopically-distinct paths. In this case, an RRT could be converted into a cyclic graph. Within a homotopy class, the solution quality can be generally improved by employing variational techniques. Also, there are many issues involved in biasing the samples, x_{rand}. For example, a bias can be given that slightly prefers a goal state (if the artificial bias is too strong, the RRT could suffer the same pitfalls as a potential field method). Significant theoretical analysis of RRTs also remains. It would be particularly valuable to determine bounds on the convergence rate and on solution quality with respect to the optimal solution.

At the present time, we believe we have barely scratched the surface of potential applications of RRTs. By allowing dynamics to be considered directly, robot planning problems for numerous navigation, manipulation, and locomotion tasks can be approached. Automotive engineers can evaluate virtual prototypes to determine whether a proposed vehicle is likely to roll over sideways, or can perform high-speed lane changes. Similar problems can be imagined in the design of spacecraft, aircraft, and underwater vehicles. Researchers in computational fluid dynamics can study the effects of flow fields on movable bodies. In computer graphics, dynamical motions of simulated machines and digital actors can be automated.

Acknowledgments I thank Bruce Donald for his encouragement to explore kinodynamic planning, James Kuffner for his suggestions and experimental work using RRTs, and Jim Bernard for providing a dynamical vehicle model. Steve LaValle is partially supported by an NSF CAREER award.

References

[1] N. M. Amato and Y. Wu. A randomized roadmap method for path and manipulation planning. In *IEEE Int. Conf. Robot. & Autom.*, pages 113–120, 1996.

[2] J. Barraquand and J.-C. Latombe. Robot motion planning: A distributed representation approach. *Int. J. Robot. Res.*, 10(6):628–649, December 1991.

[3] B. R. Donald, P. G. Xavier, J. Canny, and J. Reif. Kinodynamic planning. *Journal of the ACM*, 40:1048–66, November 1993.

[4] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Trans. Robot. & Autom.*, 12(4):566–580, June 1996.

[5] J.-C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Boston, MA, 1991.

[6] J. P. Laumond, S. Sekhavat, and F. Lamiraux. Guidelines in nonholonomic motion planning for mobile robots. In J.-P. Laumond, editor, *Robot Motion Plannning and Control*, pages 1–53. Springer-Verlag, Berlin, 1998.

[7] S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. In *Proc. IEEE Int'l Conf. on Robotics and Automation*, 1999. To appear.