Rapidly-Exploring Random Trees: Progress and Prospects

Steven M. LaValle, *Iowa State University, Ames, IA, USA*
James J. Kuffner, Jr., *University of Tokyo, Bunkyo-ku, Toyko, Japan*

We present our current progress on the design and analysis of path planning algorithms based on Rapidly-exploring Random Trees (RRTs). The basis for our methods is the incremental construction of search trees that attempt to rapidly and uniformly explore the state space, offering benefits that are similar to those obtained by other successful randomized planning methods; however, RRTs are particularly suited for problems that involve differential constraints. Basic properties of RRTs are established, including convergence to a uniform coverage of nonconvex spaces. Several planners based on RRTs are discussed and compared. Experimental results are presented for planning problems that involve holonomic constraints for rigid and articulated bodies, manipulation, nonholonomic constraints, kinodynamic constraints, kinematic closure constraints, and up to twelve degrees of freedom. Key open issues and areas of future research are also discussed.

1 Introduction

Given the vast, growing collection of applications that involve the design of motion strategies, the successes of motion planning algorithms have just begun to scratch its surface. The potential for automating motions is now greater than ever as similar problems continue to emerge in seemingly disparate areas. The traditional needs of roboticists continue to expand in efforts to automate mobile robots, manipulators, humanoids, spacecraft, etc. Researchers in computer graphics and virtual reality have increasing interests in automating the animations of life-like characters or other moving bodies. In the growing field of computational biology, many geometric problems have arisen, such as studying the configuration spaces of flexible molecules for protein-ligand docking and drug design. Virtual prototyping is a rapidly-expanding area that allows the evaluation of proposed mechanical designs in simulation, in efforts to avoid the costs of constructing physical prototypes. Motion planning techniques have already been applied to assembly problems in this area [11]. As the power and generality of planning techniques increase, we expect that more complicated problems that include differential constraints can be solved, such as the evaluation of vehicle performance and safety through dynamical simulation conducted by a virtual "stunt driver."

As we approach applications of increasing difficulty, it becomes clear that planning algorithms need to handle problems that involve a wide variety of models, high degrees of freedom, complicated geometric constraints, and finally, differential constraints. Although existing algorithms address some of these concerns, there is relatively little work that addresses all of them simultaneously. This provides the basis for the work presented in this paper, which presents randomized, algorithmic techniques for path planning that are particular suited for problems that involve differential constraints.

We present an overview of our recent progress and plans for future research based on our development of Rapidly-exploring Random Trees (RRTs) [32]. The results and discussion presented here summarize and extend the work presented in [33, 23]. RRTs build on ideas from optimal control theory [7], nonholonomic planning (see [29] for an overview), and randomized path planning [2, 24, 39]. The basic idea is to use control-theoretic representations, and incrementally grow a search tree from an initial state by applying control inputs over short time intervals to reach new states. Each vertex in the tree represents a state, and each directed edge represents an input that was applied to reach the new state from a previous state. When a vertex reaches a desired goal region, an open-loop trajectory from the initial state is represented by the tree.

For problems that involve low degrees of freedom, classical dynamic programming ideas can be employed to yield numerical optimal control solutions for a broad class of problems [7, 25, 31]. Since control theorists have traditionally preferred feedback solutions, the representation takes the form of a mesh over which cost-to-go values are defined using interpolation, enabling inputs to be selected over any portion of the state space. If open-loop solutions are the only requirement, then each cell in the mesh could be replaced by a vertex that represents a single state within the cell. In this case, control-theoretic numerical dynamic programming technique can oftne be reduced to the construction of a tree grown from an initial state. This idea has been proposed in path planning literature for nonholonomic [6] planning and kinodynamic planning in [15]. Because these methods are based on dynamic programming and systematic exploration of a grid or mesh, their application is limited to problems with low degrees of freedom.

We would like to borrow some of the ideas from numerical optimal control techniques, while weakening the requirements enough to obtain methods that can apply to problems with high degrees of freedom. As is common in most of path planning research, we forego trying to obtain optimal solutions, and attempt to find solutions that are "good enough," as long as they satisfy all of the constraints. This avoids the use of dynamic programming and systematic exploration of the space; however, a method is needed to guide the search in place of dynamic programming.

Inspired by the success of randomized path planning techniques and Monte-Carlo techniques in general for addressing high-dimensional problems, it is natural to consider adapting existing planning techniques to our problems of interest. The primary difficulty with existing techniques is that, although powerful for standard path planning, they do not naturally extend to general problems that involve differential constraints. The randomized potential field method [5], while efficient for holonomic planning, depends heavily on the choice of a good heuristic potential function, which could become a daunting task when confronted with obstacles, and differential constraints. In the probabilistic roadmap approach [2, 24], a graph is constructed in the configuration space by generating random configurations and attempting to connect pairs of nearby configurations with a local planner that will connect pairs of configurations. For planning of holonomic systems or steerable nonholonomic systems (see [29] 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 many problems with differential constraints. This has motivated our development of RRTs.

2 Problem Formulation

The class of problems considered in this paper can be formulated in terms of the following six components:

1. **State Space:** A topological space, X
2. **Boundary Values:** $x_{init} \in X$ and $X_{goal} \subset X$
3. **Collision Detector:** A function, $D : X \rightarrow \{true, false\}$, that determines whether global constraints are satisfied from state x. This could be a binary-valued or real-valued function.
4. **Inputs:** A set, U, which specifies the complete set of controls or actions that can affect the state.
5. **Incremental Simulator:** Given the current state, $x(t)$, and inputs applied over a time interval, $\{u(t')|t \leq t' \leq t + \Delta t\}$, compute $x(t + \Delta t)$.
6. **Metric:** A real-valued function, $\rho : X \times X \rightarrow [0, \infty)$, which specifies the distance between pairs of points in X.

Path planning will generally be viewed as a search in a state 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}. It is assumed that a complicated set of global constraints is imposed on X, and any solution path must keep the state within this set. A collision detector reports whether a given state, x, satisfies the global constraints. We generally use the notation X_{free} to refer to the set of all states that satisfy the constraints. Local, differential constraints are effected through the definition of a set of inputs (or controls) and an incremental simulator. Taken together, these two components specify possible changes in state. The incremental simulator can be considered as the response of a discrete-time system (or a continuous-time system that is approximated in discrete time). Finally, a metric is defined to indicate the closeness of pairs of points in the state space. This metric will be used in Section 3, when the RRT is introduced.

Basic (Holonomic) Path Planning Path planning can generally be viewed as a search in a configuration space, $\mathcal{C}$, in which each $q \in \mathcal{C}$ specifies the position and orientation of one or more geometrically-complicated bodies in a 2D or 3D world [26, 36]. The path planning task is to compute a continuous path from an initial configuration, q_{init}, to a goal configuration, q_{goal}. Thus, $X = \mathcal{C}$, $x_{init} = q_{init}$, $x_{goal} = q_{goal}$, and $X_{free} = \mathcal{C}_{free}$, which denotes the set of configurations for which these bodies do not collide with any static obstacles in the world. The obstacles are modeled completely in the world, but an explicit representation of X_{free} is not available. However, using a collision detection algorithm, a given configuration can be tested. (To be more precise, we usually employ a distance-computation algorithm that indicates how close the geometric bodies are to violating the constraints in the world. This can be used to ensure that intermediate configurations are collision free when discrete jumps are made by the incremental simulator.) The set, U, of inputs is the set of all velocities $\dot{x}$ such that $\|\dot{x}\| \leq c$ for some positive constant c. The incremental simulator produces a new state by integration to obtain, $x_{new} = x + u\Delta t$, for any given input $u \in U$.

Nonholonomic Path Planning Nonholonomic planning [27] addresses problems that involve nonintegrable constraints on the state velocities, in addition to the components that appear in the basic path planning problems. These constraints often arise in many contexts such as wheeled-robot systems [9, 28, 50], and manipulation by pushing [1, 37]. A recent survey appears in [29]. The constraints often appear in the implicit form $h_i(\dot{q}, q) = 0$ for some i from 1 to $k < N$ (N is the dimension of $\mathcal{C}$). By the implicit function theorem, the constraints can also be expressed in control-theoretic form, $\dot{q} = f(q, u)$, in which u is an input chosen from a set of inputs U. Using our general notion, x replaces q to obtain $\dot{x} = f(x, u)$. This form is often referred to as the *state transition equation* or *equation of motion.* Using the state transition equation, an incremental simulator can be constructed by numerical integration (using, for example Runge-Kutta techniques).

Kinodynamic[1] Path Planning For kinoydynamic planning, constraints on both velocity and acceleration exist, yielding implicit equations of the form $h_i(\ddot{q}, \dot{q}, q) = 0$ [10, 12, 15, 14, 13, 16, 17, 19, 42, 45]. A state, $x \in X$, is defined as $x = (q, \dot{q})$, for $q \in \mathcal{C}$. Using the state space representation, this can be simply written as a set of m implicit equations of the form $G_i(x, \dot{x}) = 0$, for $i = 1, \ldots, m$ and $m < 2N$. Once again, the implicit function theorem is applied to obtain a state transition equation, and an incremental simulator. The collision detection component may also include global constraints on the velocity, since $\dot{q}$ is part of the state vector. For example, the collision detector might test whether a mobile robot is satisfying a maximum speed constraint.

Other Problems A variety of other problems fit within our problem formulation, and can be approached using the techniques in this paper. In general, any open-loop trajectory design problem can formulated because the models are mostly borrowed from control theory. For example, the planner might be used

[1]In nonlinear control literature, kinodynamic planning is encompassed by the definition of nonholonomic planning.

to compute a strategy that controls an electrical circuit, or an economic system. In some applications, a state transition equation might not be known, but this does not present a problem. For example, a physical simulator might be developed by engineers for simulating a proposed racing car design. The software might simply accept control inputs at some sampling rate, and produce new states. This could serve directly as the incremental simulator for our approach. Other minor variations of the formulation can be considered. Time-varying problems can be formulated by augmenting the state space with a time. State-dependent inputs sets can also be considered. For example, a robot engaged in a grasping task might have different inputs available than while navigating. Depending on the state, different decisions would have to be made. Problems that involve kinematic closure constraints can also be addressed; an example is shown in Figure 17.

3 Rapidly-Exploring Random Trees

The Rapidly-exploring Random Tree (RRT) was introduced in [32] as an efficient data structure and sampling scheme to quickly search high-dimensional spaces that have both algebraic constraints (arising from obstacles) and differential constraints (arising from nonholonomy and dynamics). The key idea is to bias the exploration toward unexplored portions of the space by sampling points in the state space, and incrementally "pulling" the search tree toward them. Recently, at least two other randomized path planning techniques have been proposed that generate a search tree: the Ariadne's clew algorithm [39] and the planners in [20, 53]. Intuitively, these planners attempt to "push" the search tree away from previously-constructed vertices, contrasting the RRT, which uses the surrounding space to "pull" the search tree, ultimately leading to uniform coverage of the state space. Furthermore, to the best of our knowledge, a randomized search tree approach has not been proposed previously for nonholonomic or kinodynamic planning. Perhaps the most related approaches are [50, 48], in which the probabilistic roadmap method is combined with nonholonomic steering techniques to plan paths for wheeled mobile robot systems.

```
BUILD_RRT($x_{init}$)
1  $\mathcal{T}$.init($x_{init}$);
2  for $k = 1$ to $K$ do
3      $x_{rand} \leftarrow$ RANDOM_STATE();
4      EXTEND($\mathcal{T}, x_{rand}$);
5  Return $\mathcal{T}$
```

```
EXTEND($\mathcal{T}, x$)
1  $x_{near} \leftarrow$ NEAREST_NEIGHBOR($x, \mathcal{T}$);
2  if NEW_STATE($x, x_{near}, x_{new}, u_{new}$) then
3      $\mathcal{T}$.add_vertex($x_{new}$);
4      $\mathcal{T}$.add_edge($x_{near}, x_{new}, u_{new}$);
5      if $x_{new} = x$ then
6          Return Reached;
7      else
8          Return Advanced;
9  Return Trapped;
```

Figure 1: *The basic RRT construction algorithm.*

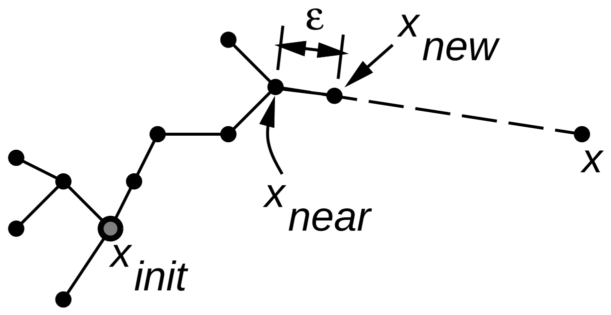

Figure 2: *The EXTEND operation.*

The basic RRT construction algorithm is given in Figure 1. A simple iteration in performed in which each step attempts to extend the RRT by adding a new vertex that is biased by a randomly-selected state. The EXTEND function, illustrated in Figure 2, selects the nearest vertex already in the RRT to the given sample state. The "nearest" vertex is chosen according to the metric, ρ. The function NEW_STATE makes a motion toward x by applying an input $u \in U$ for some time increment Δt. This input can be chosen at random, or selected by trying all possible inputs and choosing the

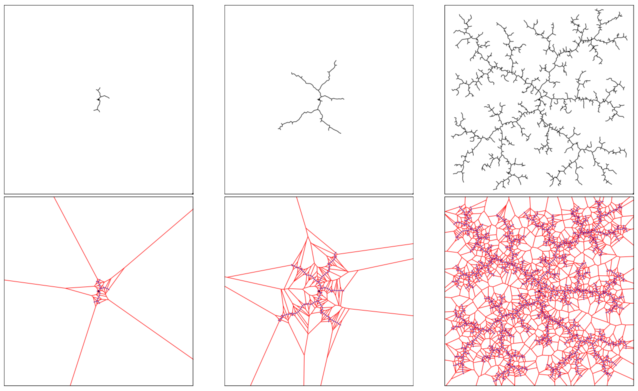

Figure 3: *An RRT is biased by large Voronoi regions to rapidly explore, before uniformly covering the space.*

one that yields a new state as close as possible to the sample, x (if U is infinite, then an approximation or analytical technique can be used). In the case of holonomic planning, the optimal value for u can be chosen easily by a simple vector calculation. NEW_STATE also implicitly uses the collision detection function to determine whether the new state (and all intermediate states) satisfy the global constraints. For many problems, this can be performed quickly ("almost constant time") using incremental distance computation algorithms [18, 34, 40] by storing the relevant invariants with each of the RRT vertices. If NEW_STATE is successful, the new state and input are represented in x_{new} and u_{new}, respectively. Three situations can occur: *Reached*, in which the new vertex reaches the sample x (for the nonholonomic planning case, we might instead have a threshold, $\|x_{new} - x\| < \epsilon$ for a small $\epsilon > 0$); *Advanced*, in which a new vertex $x_{new} \neq x$ is added to the RRT; *Trapped*, in which NEW_STATE fails to produce a state that lies in X_{free}. The top row of Figure 3 shows an RRT for a holonomic planning problem, constructed in a 2D square space. The lower figure shows the Voronoi diagram of the RRT vertices; note that the probability that a vertex is selected for extension is proportional to the area of its Voronoi region. This biases the RRT to rapidly explore. In Section 4 it is shown that RRTs also arrive at a uniform coverage of the space, which is also a desirable property of the probabilistic roadmap planner.

4 Analysis of RRTs

This section provides some analysis of RRTs, and indicates several open problems for future investigation. A key result shown so far is that the distribution of the RRT vertices converges to the sampling distribution, which is usually uniform. This currently has been shown for holonomic planning in a nonconvex state space. We have also verified the results through simulations and chi-square tests. We have generally had many experimental successes, indicated in Section 6, that far exceed our current analysis capabilities. Considerable effort remains to close the gap between our experimental success, and the analysis that supports the success.

The limiting distribution of vertices Let $D_k(x)$ denote a random variable whose value is the distance of x to the closest vertex in G, in which k is the number of vertices in an RRT. Let d_k denote the value of D_k. Let ϵ denote the incremental distance traveled in the EXTEND procedure (the RRT step size).

Consider the case of a holonomic planning problem, in which $\dot{x} = u$ (the incremental simulator permits motion in any direction). The first lemma establishes that the RRT will (converging in probability) come arbitrarily close to any point in a convex space.

Lemma 1 *Suppose X_{free} is a convex, bounded, open, n-dimensional subset of an n-dimensional state space. For any $x \in X_{free}$ and positive constant $\epsilon > 0$, $\lim_{k\to\infty} P[d_k(x) < \epsilon] = 1$.*

The next lemma extends the result from convex spaces to nonconvex spaces.

Lemma 2 *Suppose X_{free} is a nonconvex, bounded, open, n-dimensional connected component of an n-dimensional state space. For any $x \in X_{free}$ and positive real number $\epsilon > 0$, then $\lim_{n\to\infty} P[d_n(x) < \epsilon] = 1$.*

For holonomic path planning, this immediately implies the following:

Theorem 3 *Suppose x_{init} and x_{goal} lie in the same connected component of a nonconvex, bounded,* *open, n-dimensional connected component of an n-dimensional state space. The probability that an RRT constructed from x_{init} will find a path to x_{goal} approaches one as the number of RRT vertices approaches infinity.*

This establishes probabilistic completeness, as defined in [26], of the basic RRT.

The next step is to characterize the limiting distribution of the RRT vertices. Let $\mathbf{X}$ denote a vector-valued random variable that represents the sampling process used to construct an RRT. This reflects the distribution of samples that are returned by the RANDOM_STATE function in the EXTEND algorithm. Usually, $\mathbf{X}$ is characterized by a uniform probability density function over X_{free}; however, we will allow $\mathbf{X}$ to be characterized by any smooth probability density function. Let $\mathbf{X}_k$ denote a vector-valued random variable that represents the distribution of the RRT vertices.

Theorem 4 *$\mathbf{X}_k$ converges to $\mathbf{X}$ in probability.*

We now consider the more general case. Suppose that motions obtained from the incremental simulator are locally constrained. For example, they might arise by integrating $\dot{x} = f(x, u)$ over some time Δt. Suppose that the number of inputs to the incremental simulator is finite, Δt is constant, no two RRT vertices lie within a specified $\epsilon > 0$ of each other according to ρ, and that EXTEND chooses the input at random. It may be possible eventually to remove some of these restrictions; however, we have not yet pursued this route. Suppose x_{init} and x_{goal} lie in the same connected component of a nonconvex, bounded, open, n-dimensional connected component of an n-dimensional state space. In addition, there exists a sequence of inputs, u_1, u_2, $\ldots$, u_k, that when applied to x_{init} yield a sequence of states, $x_{init} = x_0$, x_1, x_2, $\ldots$, $x_{k+1} = x_{goal}$. All of these states lie in the same open connected component of X_{free}.

The following establishes the probabilistic completeness of the nonholonomic planner.

Theorem 5 *The probability that the RRT initialized at x_{init} will contain x_{goal} as a vertex approaches one as the number of vertices approaches infinity.*

Other Properties Several other properties of RRTs are briefly discussed as part of our ongoing research. The analysis obtained thus far is helpful in establishing several useful properties of RRTs; however, considerable open questions remain. One of the key difficulties at present is characterizing the rate of convergence of RRT-based planners. In numerous simulation experiments we have observed fast convergence toward solutions, but it has been challenging to obtain theoretical bounds on performance that match our observations.

Consider the holonomic planning case. We have already established that the RRT vertices converge to the sampling distribution, but what about the "rapidly-exploring" property? It was argued using Voronoi diagrams in Section 3, that RRT growth is strongly biased toward unexplored portions of the state space. It would be valuable to have precise bounds on this rapid exploration. Suppose that X is a disc with infinite radius (this can be defined more precisely using limits) in the plane, and that x_{init} is at the center. We expect the RRT in this case to rapidly advance in a small number of directions, as opposed to trying to fill the space near x_{init}. The "branches" of the RRT can be roughly characterized by counting the number of vertices in the convex hull of the RRT. We have observed experimentally that after the first few iterations, there are only a few vertices in the hull, regardless of the number of RRT vertices. Figure 4.a shows a typical result, in which the RRT has three major branches, each roughly 120 degrees apart. We speculate that the number of major branches is linear in the dimension of the space, and that the expected number of vertices in the convex hull is bounded by no more than 7 for the planar case.

Another observation that we have made through simulation experiments is that the paths in an RRT, while jagged, are not too far from the shortest path (recall Figure 3). This is not true for paths generated by a simpler technique, such as Brownian motion. For paths in the plane, we have performed repeated experiments that compare the distance of randomly-chosen RRT vertices to the root by following the RRT path to the Euclidean distance to the root. Experiments were performed in a square region in the plane. The expected

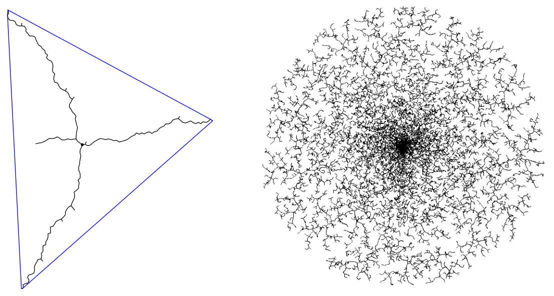

Figure 4: *a) The convex hull of an RRT in an "infinitely" large disc; b) a 2D RRT that was constructed using biased sampling.*

ratio of RRT-path distance to Euclidean distance is consistently between 1.3 and 1.7. It would be useful to establish a theoretical ratio bound on the path length in comparison to the optimal length.

5 Designing Path Planners

Sections 3 and 4 introduced the basic RRT and analyzed its exploration properties. Now the focus is on developing path planners using RRTs. We generally consider the RRT as a building block that can be used to construct an efficient planner, as opposed to a path planning algorithm by itself. For example, one might use an RRT to escape local minima in a randomized potential field path planner. In [51], an RRT was used as the local planner for the probabilistic roadmap planner. We present several alternative RRT-based planners in this section. The recommended choice depends on several factors, such as whether differential constraints exist, the type of collision detection algorithm, or the efficiency of nearest neighbor computations.

Single-RRT Planners In principle, the basic RRT can be used in isolation as a path planner because its vertices will eventually cover a connected component of X_{free}, coming arbitrarily close to any specified x_{goal}. The problem is that without any bias toward the goal, convergence might be slow. An improved planner, called RRT-GoalBias, can be obtained by replacing RANDOM_STATE in Figure 2 with a function that tosses a biased coin to determine what should be returned. If the coin toss yields "heads", then x_{goal} is returned; otherwise, a random state is returned. Even with a small probability of returning heads (such as 0.05), RRT-GoalBias usually converges to the goal much faster than the basic RRT. If too much bias is introduced; however, the planner begins to behave like a randomized potential field planner that is trapped in a local minimum. An improvement called RRT-GoalZoom replaces RANDOM_STATE with a decision, based on a biased coin toss, that chooses a random sample from either a region around the goal or the whole state space. The size of the region around the goal is controlled by the closest RRT vertex to the goal at any iteration. The effect is that the focus of samples gradually increases around the goal as the RRT draws nearer. This planner has performed quite well in practice; however, it is still possible that performance is degraded due to local minima. In general, it seems best to replace RANDOM_STATE with a sampling scheme that draws states from a nonuniform probability density function that has a "gradual" bias toward the goal. Figure 4.b shows an example of an RRT that was constructed by sampling states from a probability density that assigns equal probability to concentric circular rings. There are still many interesting research issues regarding the problem of sampling. It might be possible to use some of the sampling methods that were proposed to improve the performance of probabilistic roadmaps [2, 8].

One more issue to consider is the size of the step that is used for RRT construction. This could be chosen dynamically during execution on the basis of a distance computation function that is used for collision detection. If the bodies are far from colliding, then larger steps can be taken. Aside from following this idea to obtain an incremental step, how far should the new state, x_{new} appear from x_{near}? Should we try to connect x_{near} to x_{rand}? Instead of attempting to extend an RRT by an incremental step, EXTEND can be iterated until the random state or an obstacle is reached, as shown in the CONNECT algorithm description in Figure 5. CONNECT can replace EXTEND, yielding an RRT that grows very quickly, if permitted by

```
CONNECT($\mathcal{T}, x$)
1  repeat
2      $S \leftarrow$ EXTEND($\mathcal{T}, x$);
3  until not ($S = Advanced$)
4  Return $S$;
```

Figure 5: *The CONNECT function.*

collision detection constraints and the differential constraints. One of the key advantages of the CONNECT function is that a long path can be constructed with only a single call to the NEAREST_NEIGHBOR algorithm. This advantage motivates the choice of a greedier algorithm; however, if an efficient nearest-neighbor algorithm [3, 21] is used, as opposed to the obvious linear-time method, then it might make sense to be less greedy. After performing dozens of experiments on a variety of problems, we have found CONNECT to yield the best performance for holonomic planning problems, and EXTEND seems to be the best for nonholonomic problems. One reason for this difference is that CONNECT places more faith in the metric, and for nonholonomic problems it becomes more challenging to design good metrics.

Bidirectional Planners Inspired by classical bidirectional search techniques [43], it seems reasonable to expect that improved performance can be obtained by growing two RRTs, one from x_{init} and the other from x_{goal}; a solution is found if the two RRTs meet. For a simple grid search, it is straightforward to implement a bidirectional search; however, RRT construction must be biased to ensure that the trees meet well before covering the entire space, and to allow efficient detection of meeting.

Figure 5 shows the RRT_BIDIRECTIONAL algorithm, which may be compared to the BUILD_RRT algorithm of Figure 1. RRT_BIDIRECTIONAL divides the computation time between two processes: 1) exploring the state space; 2) trying to grow the trees into each other. Two trees, $\mathcal{T}_a$ and $\mathcal{T}_b$ are maintained at all times until they become connected and a solution is found. In each iteration, one tree is extended, and an attempt is made to connect the nearest vertex of

```
RRT_BIDIRECTIONAL($x_{init}, x_{goal}$)
1  $\mathcal{T}_a$.init($x_{init}$); $\mathcal{T}_b$.init($x_{goal}$);
2  for $k = 1$ to $K$ do
3      $x_{rand} \leftarrow$ RANDOM_STATE();
4      if not (EXTEND($\mathcal{T}_a, x_{rand}$) = Trapped) then
5          if (EXTEND($\mathcal{T}_b, x_{new}$) = Reached) then
6              Return PATH($\mathcal{T}_a, \mathcal{T}_b$);
7      SWAP($\mathcal{T}_a, \mathcal{T}_b$);
8  Return Failure
```

Figure 6: *A bidirectional RRT-based planner.*

the other tree to the new vertex. Then, the roles are reversed by swapping the two trees. Growth of two RRTs was also proposed in [33] for kinodynamic planning; however, in each iteration both trees were incrementally extended toward a random state. The current algorithm attempts to grow the trees into each other half of the time, which has been found to yield much better performance.

Several variations of the above planner can also be considered. Either occurrence of EXTEND may be replaced by CONNECT in RRT_BIDIRECTIONAL. Each replacement makes the operation more aggressive. If the EXTEND in Line 4 is replaced with CONNECT, then the planner aggressively explores the state space, with the same tradeoffs that existed for the single-RRT planner. If the EXTEND in Line 5 is replaced with CONNECT, the planner aggressively attempts to connect the two trees in each iteration. This particular variant was very successful at solving holonomic planning problems. For convenience, we refer to this variant as RRT-ExtCon, and the original bidirectional algorithm as RRT-ExtExt. Among the variants discussed thus far, we have found RRT-ExtCon to be most successful for holonomic planning [23], and RRT-ExtExt to be best for nonholonomic problems. The most aggressive planner can be constructed by replacing EXTEND with CONNECT in both Lines 4 and 5, to yield RRT-ConCon. We are currently evaluating the performance of this variant.

Through extensive experimentation over a wide variety of examples, we have concluded that, when applicable, the bidirectional approach is much more efficient than a single RRT approach. One shortcoming of using the bidirectional approach for nonholonomic and kinodynamic planning problems is the need to make a connection between a pair of vertices, one from each RRT. For a planning problem that involves reaching a goal region from an initial state, no connections are necessary using a single-RRT approach. The gaps between the two trajectories can be closed in practice by applying steering methods [29], if possible, or classical shooting methods, which are often used for numerical boundary value problems.

Other Approaches If a dual-tree approach offers advantages over a single tree, then it is natural to ask whether growing three or more RRTs might be even better. These additional RRTs could be started at random states. Of course, the connection problem will become more difficult for nonholonomic problems. Also, as more trees are considered, a complicated decision problem arises. The computation time must be divided between attempting to explore the space and attempting to connect RRTs to each other. It is also not clear which connections should be attempted. Many research issues remain in the development of this and other RRT-based planners.

It is interesting to consider the limiting case in which a new RRT is started for every random sample, x_{rand}. Once the single-vertex RRT is generated, the CONNECT function from Figure 5 can be applied to every other RRT. To improve performance, one might only consider connections to vertices that are within a fixed distance of x_{rand}, according to the metric. If a connection succeeds, then the two RRTs are merged into a single graph. The resulting algorithm simulates the behavior of the probabilistic roadmap approach to path planning [24]. Thus, the probabilistic roadmap can be considered as an extreme version of an RRT-based algorithm in which a maximum number of separate RRTs are constructed and merged.

6 Implementations and Experiments

In this section, results for four different types of problems are summarized: 1) holonomic planning, 2) nonholonomic planning, 3) kinodynamic planning, and 4) planning for systems with closed kinematic chains. Presently, we have constructed two planning systems based on RRTs. One is written in Gnu C++ and LEDA, and experiments were conducted on a 500Mhz Pentium III PC running Linux. This implementation is very general, allowing many planning variants and models to be considered; however, it is limited to planar obstacles and robots, and performs naive collision detection. The software can be obtained from *http://janowiec.cs.iastate.edu/~lavalle/rrt/*. The second implementation is written in SGI C++ and SGI's OpenInventor library, and experiments were conducted on a 200MHz SGI Indigo2 with 128MB. This implementation considers 3D models, and was particularly designed inclusion in a software platform for automating the motions of digital actors [22]. Currently, we are constructing a third implementation, which is expected to be general-purpose, support 3D models, and be based on freely-available collision detection and efficient nearest neighbor libraries.

Holonomic planning experiments Through numerous experiments, we have found RRT-based planners to be very efficient for holonomic planning. Note that our planners attempt to find a solution without performing precomputations over the entire state space, and are therefore suited for single-query path planning problems. The probabilistic roadmap approach performs significant precomputation, which is motivated by its design for multiple-query path planning (i.e., the same environment is used to solve numerous problems). Thus, straightforward performance comparisons between the probabilistic roadmap approach and RRT-based approaches is not possible, as they are designed with different intentions. It is easy to construct single-query examples on which an RRT-based planner will be superior by finding a solution well before covering the entire state space, and it is easy to construct mutiple-query problems in which the probabilistic roadmap approach will be superior by relying repeatedly on its precomputed roadmap.

Most of the experiments in this section were conducted on the 200MHz SGI Indigo2. The CONNECT function is most effective when one can expect relatively open spaces for the majority of the planning queries. We first performed hundreds of experiments on over a dozen examples for planning the motions of rigid objects in 2D, resulting in 2D and 3D configuration spaces. Path smoothing was performed on the final paths to reduce jaggedness. Figure 7 depicts a computed solution for a 3D model of a grand piano moving from one room to another amidst walls and low obstacles. Several tricky rotations are required of the piano in order to solve this query. Manipulation planning experiments have been conducted for a model of a 6-DOF Puma industrial manipulator arm. Combined with an inverse kinematics algorithm, the RRT-ExtCon planner facilitates a task-level control mechanism for planning manipulation motions by computing three motions for a high-level request to move an object: 1) move the arm to grasp an object; 2) move the object to a target location; 3) release the object and return the arm to its rest position. Several snapshots of a path to move a book from the middle shelf to the bottom shelf of a desk is shown in Figure 8.

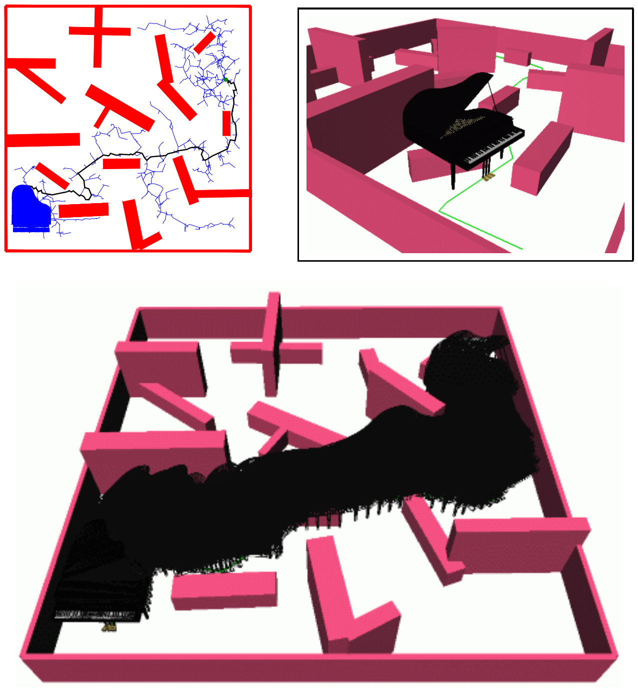

Figure 7: *Moving a Piano*

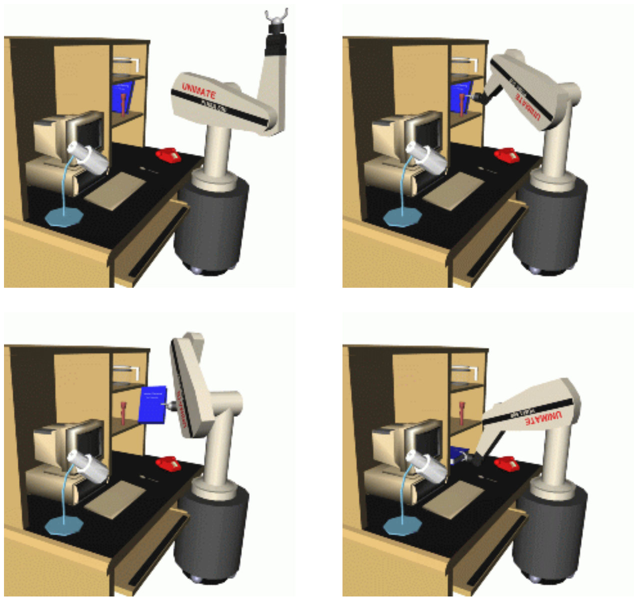

Figure 8: *A 6-DOF Puma robot moving a book*

Figure 9 shows a human character playing chess. Each of the motions necessary to reach, grasp, and reposition a game piece on the virtual chess board were generated using the RRT-ExtCon planner in an average of 2 seconds on the 200 MHz SGI Inidigo2. The human arm is modeled as a 7-DOF kinematic chain, and the entire scene contains over 8,000 triangle primitives. The 3D collision checking software used for these experiments was the RAPID library based on OBB-Trees developed by the University of North Carolina [35]. The speed of the planner allows for the user to interact with the character in real-time, and even engage in an interactive game of "virtual chess". The planner can also handle more complicated queries with narrow passages in X_{free}, such as the assembly maintenance scene depicted in Figure 10. Here, the task is to grasp the tool from within the box and place it inside the tractor wheel housing. Solving this particular set of queries takes an average of 80 seconds on the SGI Indigo2, and about 15 seconds on a high-end SGI (R10000 processor). The scene contains over 13,000 triangles.

The final holonomic planning example, shown in Figure 11, was solved using the RRT-ExtExt planner.

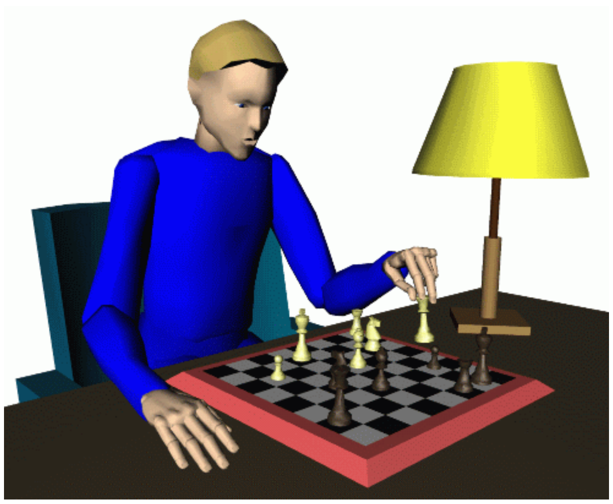

Figure 9: *Playing a game of virtual chess*

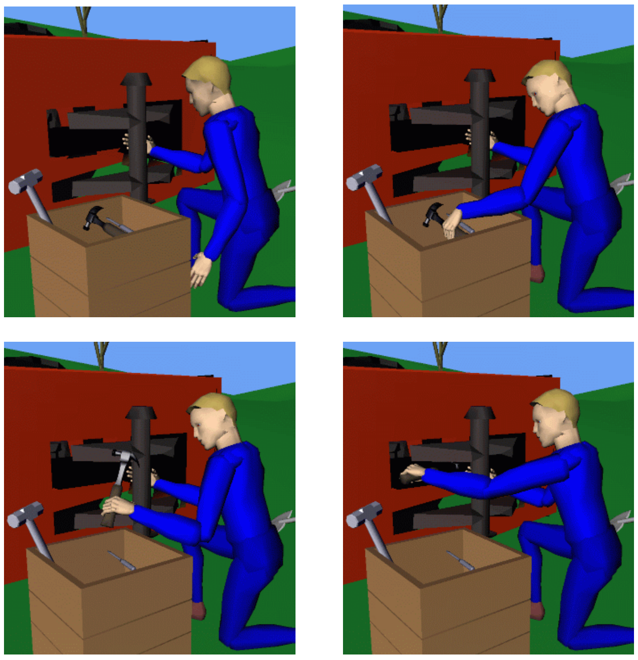

Figure 10: *A path planning problem that involves finding and using a hammer in a virtual world.*

This problem was presented in [8] as a test challenge for randomized path planners due to the narrow passageways that exist in the configuration space when the "U"-shaped object passes through the center of the world. In the example shown, the RRT does not explore to much of the surrounding space (some of this might be due to the lucky placement of the corridor in the center of the world). One average, about 1500 nodes are generated, and the problem is solved in two seconds on a PC using naive collision checking.

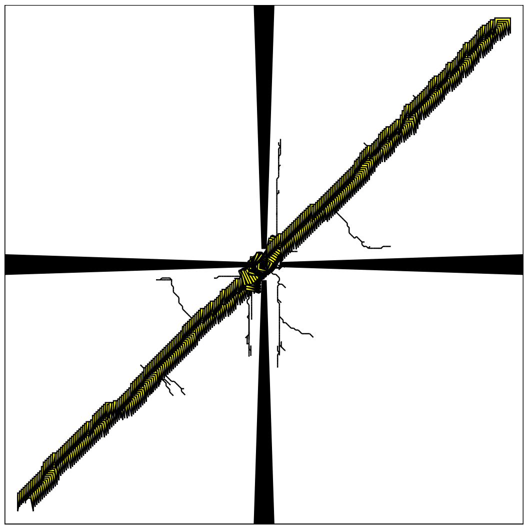

Figure 11: *A narrow-corridor example, designed for evaluating randomized path planners.*

Nonholonomic planning experiments Several nonholonomic planning examples are shown in Figures 12 and 13. These examples were computed using the RRTExtExt planner, and the average computation times were less than five seconds on the PC using naive collision detection. The four examples in Figure 12 involve car-like robots that moves at constant speed under different nonholonomic models. A 2D projection of the RRTs is shown for each case, along with the computed path from an initial state to a goal state. The top two pictures show paths computed for a 3-DOF model, using the standard kinematics for a car-like robot [26].

In the first example, the car is allowed to move in both forward and reverse. In the second example, the car can move forward only. In the first example in the second row in Figure 12, the car is only allowed to turn left in varying degrees! The planner is still able to overcome this difficult constraint and bring the robot to its goal. The final example uses a 4-DOF model,

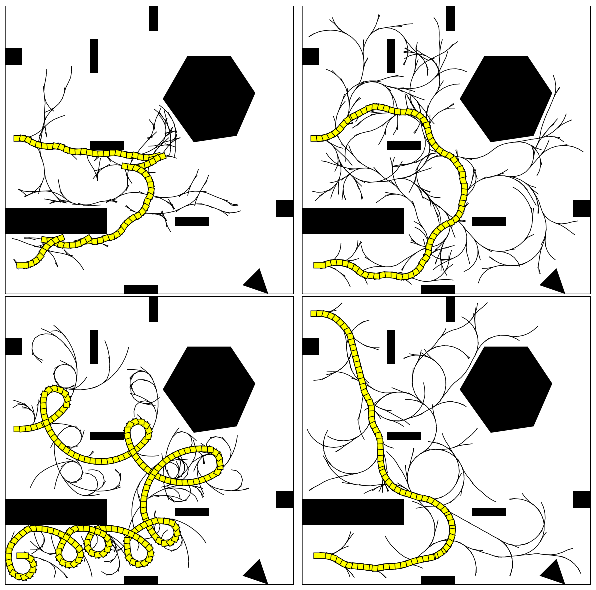

Figure 12: *Several car-like robots.*

which results in continuous curvature paths [47]. The variable x_4 represents the orientation of the wheels, and u_2 represents a change in the steering angle.

The final nonholonomic planning problem involves the 4-DOF car pulling three trailers, resulting in a 7-DOF system. The kinematics are given in [41]. The goal is to pull the car with trailers out of one stall, and back it into another. The RRTs shown correspond to one of the best executions; in other iterations the exploration was much slower due to metric problems.

Kinodynamic planning experiments Several kinodynamic planning experiments have been performed for both non-rotating and rotating rigid objects in 2D and 3D worlds with velocity and acceleration bounds obeying L_2 norms. For the 2D case, controllability issues were studied recently in [38]. The dynamic models were derived from Newtonian mechanics of rigid bodies in non-gravity environments, except for the last example, which involves a "lunar lander" in an environment with gravity. All experiments utilized a simple metric on X based on a weighted Euclidean distance for position coordinates and their derivatives, along with a weighted metric on unit quaternions for rotational coordinates and their derivatives.

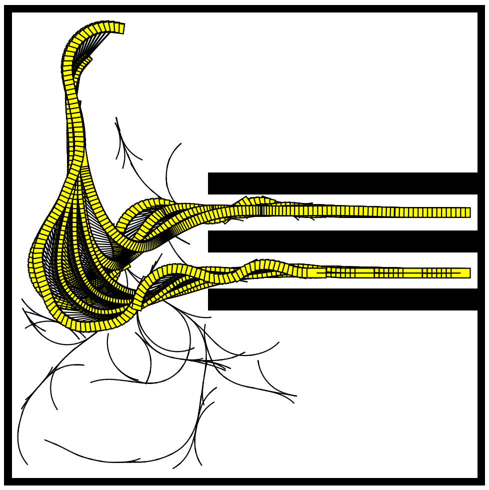

Figure 13: *A nonholonomic planning problem that involves a car pulling three trailers. There are seven degrees of freedom.*

All experiments were performed using variants of the RRT_BIDIRECTIONAL planner. Other kinodynamic planning experiments are presented in [33].

First, we consider the case of a rigid object with two unilateral thrusters each producing a torque of opposite sign, such as the one described in [38]. Each thruster provides a line of force fixed in the body frame that restricts its motion to a plane. This model is similar to that of a hovercraft, navigating with drift. The state space of this system has 6 degrees of freedom, but only 3 controls: translate forward, rotate clockwise, and rotate counter-clockwise are provided. Figure 14 shows the result obtained after 13,600 nodes. The total computation time for this example was 4.2 minutes.

We next consider the case of a fully-orientable satellite model with limited translation. The satellite is assumed to have momentum wheels that enable it to orient itself along any axis, and a single pair of opposing thruster controls that allow it to translate along the primary axis of the cylinder. This model has a 12-dimensional state space. The task of the satellite, modeled as a rigid cylindrical object, is to perform a collision-free docking maneuver into the cargo bay of the space shuttle model amidst a cloud of obstacles. Figure 15 shows the candidate solution found after

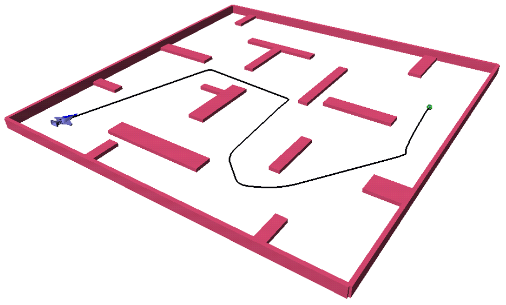

Figure 14: *A computed trajectory for the planar body with unilateral thrusters that allow it to rotate freely but translate only in the forward direction.*

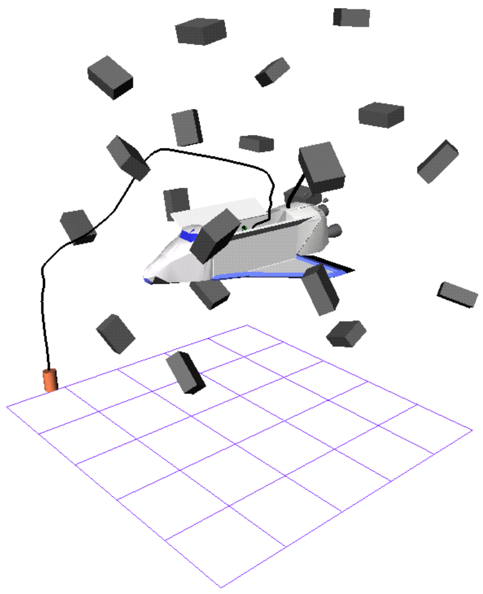

Figure 15: *The docking maneuver computed for the fully-orientable satellite model. The satellite's initial state is in the lower left corner, and the goal state is in the interior of the cargo bay of the shuttle.*

23,800 states were explored. The total computation time was 8.4 minutes.

The another kinodynamic planning example is shown in Figure 16 for a "lunar lander" that navigates in a 2D environment with gravity. There is one central thruster on the bottom of the craft, and a smaller thruster on each side to provide lateral movements.

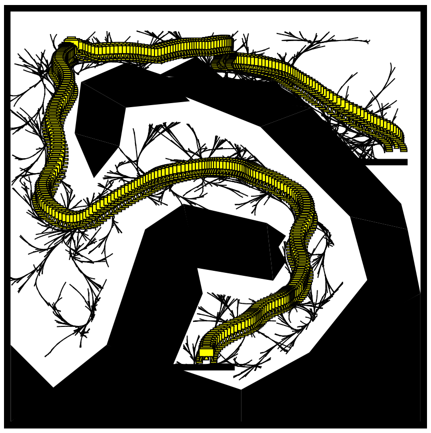

Figure 16: *A spacecraft is required to navigate through hazardous terrain and dock safely in spite of gravity.*

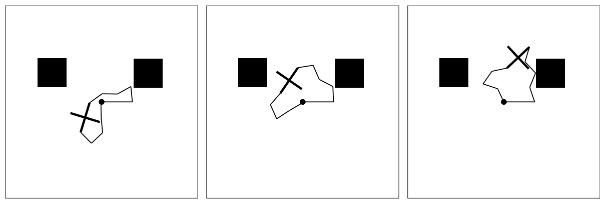

Figure 17: *Two manipulators transport a cross-shaped object while maintaining kinematic closure.*

Planning for closed kinematic chains Figure 17 shows a problem that involves a kinematic closure constraint that must be maintained in addition to performing holonomic path planning. Many more examples and experiments are discussed in [52]. In the initial state, the closure constraint is satisfied. The incremental simulator performs local motions that maintain with closure constraint within a specified tolerance.

7 Discussion

We have presented a general framework for developing randomized path planning algorithms based on the concept of Rapidly-exploring Random Trees (RRTs). After extensive experimentation, we are satisfied with the results obtained to date. There is, however, significant room for improvement given the complexity of problems that arise in many applications. To date, we believe we have presented the first randomized path planning techniques that are particularly designed for handling differential constraints (without necessarily requiring steering ability). RRTs have also led to very efficient planners for single-query holonomic path planning.

Several issues and topics are mentioned below, which are under current investigation.

Designing Metrics The primary drawback with the RRT-based methods is the sensitivity of the performance on the choice of the metric, ρ. All of the results presented in Section 6 were obtained by assigning a simple, weighted Euclidean metric for each model (the same metric was used for different collections of obstacles). Nevertheless, we observed that the computation time varies dramatically for some problems as the metric is varied. This behavior warrants careful investigation into the effects of metrics. This problem might seem similar to the choice of a potential function for the randomized potential field planer; however, since RRTs eventually perform uniform exploration, the performance degradation is generally not as severe as a local minimum problem. Metrics that would fail miserably as a potential function could still yield good performance in an RRT-based planner.

In general, we can characterize the ideal choice of a metric (technically this should be called a pseudometric due to the violation of some metric properties). Consider a cost or loss functional, L, defined as

$$L = \int_0^T l(x(t), u(t))dt + l_f(x(T)).$$

As examples, this could correspond to the distance traveled, the energy consumed, or the time elapsed during the execution of a trajectory. The optimal cost to go from x to x' can be expressed as

$$\rho^*(x, x') = \min_{u(t)} \left\{ \int_0^T l(x(t), u(t))dt + l_f(x(T)) \right\}.$$

Ideally, ρ^* would make an ideal metric because it indicates "closeness" as the ability to bring the state from x to x' while incurring little cost. For holonomic planning, nearby states in terms of a weighted Euclidean metric are easy to reach, but for nonholonomic problems, it can be difficult to design a good metric. The ideal metric has appeared in similar contexts as the nonholonomic metric (see [29]), the value function [49], and the cost-to-go function [4, 30]. Of course, computing ρ^* is as difficult as solving the original planning problem! It is generally useful, however, to consider ρ^* because the performance of RRT-based planners seems to generally degrade as ρ and ρ^* diverge. An effort to make a crude approximation to ρ^*, even if obstacles are neglected, will probably lead to great improvements in performance.

Efficient Nearest-Neighbors One of the key bottlenecks in construction of RRTs so far has been nearest neighbor computations. To date, we have only implemented the naive approach in which every vertex is compared to the sample state. Fortunately, the development of efficient nearest-neighbor for high-dimensional problems has been a topic of active interest in recent years (e.g., [3, 21]). Techniques exist that can compute nearest neighbors (or approximate nearest-neighbors) in near-logarithmic time in the number of vertices, as opposed to the naive method which takes linear time. Implementation and experimentation with nearest neighbor techniques is expected to dramatically improve performance. Three additional concerns must be addressed: 1) any data structure that is used for efficient nearest neighbors must allow incremental insertions to be made efficiently due to the incremental construction of an RRT, and 2) the method must support whatever metric, ρ, is chosen, and 3) simple adaptations must be made to account for the topology of the state space (especially in the case of S^1 and P^3, which arise from rotations).

Variational Optimization Due to randomization, it is obvious that the generated trajectories are not optimal, even within their homotopy class. For randomized approaches to holonomic planning, it is customary to perform simple path smoothing to partially optimize the solution paths. Simple and efficient techniques can be employed in this case; however, in the presence of differential constraints, the problem becomes slightly more complicated. In general, variational techniques from classical optimal control theory can be used to optimize trajectories produced by our methods. For many problems, a trajectory that is optimal over the homotopy class that contains the original trajectory can be obtained. These techniques work by iteratively making small perturbations to the trajectory by slightly varying the inputs and verifying that the global constraints are not violated. Since variational techniques require a good initial starting trajectory, they can be considered as complementary to the RRT-based planners. In other words, the RRT-based planners can produce good guesses for variational optimization techniques. The bidirectional planner could be adapted to general trajectories in multiple homotopy classes. In combination with variational techniques, it might be possible to develop an RRT-based planner that produces trajectories that improve over time, ultimately converging probabilistically to a globally-optimal trajectory.

Collision Detection For collision detection in our previous implementations, we have not yet exploited the fact that RRTs are based on incremental motions. Given that small changes usually occur between configurations, a data structure can be used that dramatically improves the performance of collision detection and distance computation [18, 34, 40, 44]. For pairs of convex polyhedral bodies, the methods proposed in [34, 40] can compute the distance between closest pairs of points in the world in "almost constant time." It is expected that these methods could dramatically improve performance. For holonomic planning, it seems best to take the largest step possible given the distance measurement (a given distance value can provide a guarantee that the configuration can change by a prescribed amount without causing collision). This might, however, counteract the performance benefits of the incremental distance computation methods. Further research is required to evaluate the tradeoffs.

Applications in Robotics Most of the problems considered thus far in our experiments are motivated by robotics applications. We are currently working towards developing a kinodynamic planner for determining open-loop trajectories for basic manipulation and motor tasks for humanoid robots, such as the Honda humanoid. For example, a task might be to apply a specified force to a nail using a hammer. An RRT-based kinodynamic planner could be used to compute automatically a trajectory for the robot arm and hand that would arrive at a state that produces the desired force. A system of this complexity involves more degrees of freedom than we have considered at this point, and several of the issues mentioned above will have to be investigated further to achieve this goal. We are also generally interested in evaluating the RRT-based planners on as many models and problems as possible. It is hoped that some day it might be possible to change the incremental simulator (and corresponding state transition equation) for nonholonomic and kinodynamic problems as easily as one presently changes the obstacles in a holonomic path planning method.

Applications in Virtual Prototyping Since RRTs produce open-loop trajectories, their potential for application seems greatest when feedback is not necessary at any stage. In virtual prototyping, mechanical designs are evaluated through simulation in a virtual environment, without requiring the design of a feedback motion strategy. We are presently experimenting with vehicle dynamics models to answer questions such as, "Can this car perform a rapid lane change without losing control?" or "Is it possible that this minivan could flip over sideways through some combination of inputs?". An RRT-based planner can be used as a kind of virtual "stunt driver" that tries to determine potential flaws early in the design process. In addition to cars, we can imagine applications to aircraft, spacecraft, hovercrafts, submarines, and a wide variety of mechanical machinery.

Applications in Computer Graphics There are two avenues to pursue in computer graphics. RRT-based planning methods have been used so far for the automation of digital actors in a virtual environment [22]. In industries such as entertainment and education, we expect to observe an increasing need to automate the motions of a variety of geometric bodies.

In addition to using RRTs for planning, they might also be useful directly as a modeling tool. The images in Figure 3 are similar to those produced by Diffusion Limited Aggregation (DLA) [46], which has been applied to construct a wide variety of images for problems such as crystal growth, erosion patterns, natural plants, etc. DLAs are grown by incrementally accumulating points that become "stuck" in the existing structure after performing a random walk that starts at infinity. Both DLAs and RRTs can be considered as stochastic fractals. Given the efficiency of the RRT construction method, particularly in light of efficient nearest-neighbor algorithms, it could become a useful modeling tool for generating fractal-based scenes.

Acknowledgments

This work has benefitted greatly from discussions with Nancy Amato, Jim Bernard, Francesco Bullo, Peng Cheng, Bruce Donald, David Hsu, Yan-Bin Jia, Lydia Kavraki, Jean-Claude Latombe, Jean-Paul Laumond, Kevin Lynch, Ahmad Masoud, and Jeff Yakey. The authors thank Valerie Boor for supplying the problem shown in Figure 11. LaValle is supported in part by NSF CAREER Award IRI-9875304. Kuffner is supported in part by a jointly-funded NSF-JSPS (Japan) Post-Doctoral Fellowship. Both authors are also supported in part by a grant from Honda Research.

References

[1] P. K. Agarwal, J.-C. Latombe, R. Motwani, and P. Raghavan. Nonholonomic path planning for pushing a disk among obstacles. In *IEEE Int. Conf. Robot. & Autom.*, 1997.

[2] 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.

[3] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu. An optimal algorithm for approximate nearest neighbor searching. *Journal of the ACM*, 45:891–923, 1998.

[4] T. Başar and G. J. Olsder. *Dynamic Noncooperative Game Theory.* Academic Press, London, 1982.

[5] J. Barraquand, B. Langlois, and J. C. Latombe. Numerical potential field techniques for robot path planning. *IEEE Trans. Syst., Man, Cybern.*, 22(2):224–241, 1992.

[6] J. Barraquand and J.-C. Latombe. Nonholonomic multibody mobile robots: Controllability and motion planning in the presence of obstacles. In *IEEE Int. Conf. Robot. & Autom.*, pages 2328–2335, 1991.

[7] R. E. Bellman. *Dynamic Programming.* Princeton University Press, Princeton, NJ, 1957.

[8] V. Boor, N. H. Overmars, and A. F. van der Stappen. The gaussian sampling strategy for probabilistic roadmap planners. In *IEEE Int. Conf. Robot. & Autom.*, pages 1018–1023, 1999.

[9] L.G. Bushnell, D.M. Tilbury, and S.S. Sastry. Steering three-input nonholonomic systems: The fire-truck example. *Int. Journal of Robotics Research*, 14(3), 1995.

[10] J. Canny, A. Rege, and J. Reif. An exact algorithm for kinodynamic planning in the plane. *Discrete and Computational Geometry*, 6:461–484, 1991.

[11] H. Chang and T. Y. Li. Assembly maintainability study with motion planning. In *IEEE Int. Conf. Robot. & Autom.*, pages 1012–1019, 1995.

[12] C. Connolly, R. Grupen, and K. Souccar. A hamiltonian framework for kinodynamic planning. In *Proc. of the IEEE International Conf. on Robotics and Automation (ICRA'95)*, Nagoya, Japan, 1995.

[13] B. Donald and P. Xavier. Provably good approximation algorithms for optimal kinodynamic planning for cartesian robots and open chain manipulators. *Algorithmica*, 14(6):480–530, 1995.

[14] B. Donald and P. Xavier. Provably good approximation algorithms for optimal kinodynamic planning: Robots with decoupled dynamics bounds. *Algorithmica*, 14(6):443–479, 1995.

[15] B. Donald, P. Xavier, J. Canny, and J. Reif. Kinodynamic motion planning. *Journal of the ACM*, 40(5):1048–1066, November 1993.

[16] P. Ferbach. A method of progressive constraints for nonholonomic motion planning. In *Proc. of the IEEE International Conf. on Robotics and Automation (ICRA'96)*, pages 1637–1642, Minneapolis, MN, April 1996.

[17] Th. Fraichard and C. Laugier. Kinodynamic planning in a structured and time-varying 2d workspace. In *IEEE Int. Conf. Robot. & Autom.*, pages 2: 1500–1505, 1992.

[18] L. J. Guibas, D. Hsu, and L. Zhang. H-Walk: Hierarchical distance computation for moving convex bodies. In *Proc. ACM Symposium on Computational Geometry*, pages 265–273, 1999.

[19] G. Heinzinger, P. Jacobs, J. Canny, and B. Paden. Time-optimal trajectories for a robotic manipulator: A provably good approximation algorithm. In *Proc. of IEEE Int. Conf. on Robotics and Automation*, pages 150–155, Cincinnati, OH, 1990.

[20] D. Hsu, J.-C. Latombe, and R. Motwani. Path planning in expansive configuration spaces. *Int. J. Comput. Geom. & Appl.* To appear.

[21] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, 1998.

[22] L. Kavraki, P. Svestka, J. C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration space. *Int. Transactions on Robotics and Automation*, 12(4):566–580, 1996.

[23] J. J. Kuffner. *Autonomous Agents for Real-time Animation.* PhD thesis, Stanford University, 1999.

[24] J. J. Kuffner and S. M. LaValle. RRT-connect: An efficient approach to single-query path planning. In *Proc. IEEE Int'l Conf. on Robotics and Automation*, 2000.

[25] R. E. Larson and J. L. Casti. *Principles of Dynamic Programming, Part II.* Dekker, New York, NY, 1982.

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

[27] J.-P. Laumond. Finding collision-free smooth trajectories for a non-holonomic mobile robot. In *Proc. Int. Joint Conf. on Artif. Intell.*, pages 1120–1123, 1987.

[28] J.-P. Laumond, P. E. Jacobs, M. Taix, and R. M. Murray. A motion planner for nonholonomic mobile robots. *IEEE Trans. Robot. & Autom.*, 10(5):577–593, October 1994.

[29] 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.

[30] S. M. LaValle. *A Game-Theoretic Framework for Robot Motion Planning.* PhD thesis, University of Illinois, Urbana, IL, July 1995.

[31] S. M. LaValle. Numerical computation of optimal navigation functions on a simplicial complex. In P. Agarwal, L. Kavraki, and M. Mason, editors, *Robotics: The Algorithmic Perspective.* A K Peters, Wellesley, MA, 1998.

[32] S. M. LaValle. Rapidly-exploring random trees: A new tool for path planning. TR 98-11, Computer Science Dept., Iowa State University. <http://janowiec.cs.iastate.edu/papers/rrt.ps>, Oct. 1998.

[33] S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. In *Proc. IEEE Int'l Conf. on Robotics and Automation*, pages 473–479, 1999.

[34] M. C. Lin and J. F. Canny. Efficient algorithms for incremental distance computation. In *IEEE Int. Conf. Robot. & Autom.*, 1991.

[35] M. C. Lin, D. Manocha, J. Cohen, and S. Gottschalk. Collision detection: Algorithms and applications. In J.-P. Laumond and M. Overmars, editors, *Algorithms for Robotic Motion and Manipulation*, pages 129–142. A K Peters, Wellesley, MA, 1997.

[36] T. Lozano-Perez. Spatial planning: A configuration space approach. *IEEE Trans. on Computers*, C-32(2):108–120, 1983.

[37] K. M. Lynch and M. T. Mason. Stable pushing: Mechanics, controllability, and planning. *Int. J. Robot. Res.*, 15(6):533–556, 1996.

[38] K.M. Lynch. Controllability of a planar body with unilateral thrusters. *IEEE Trans. on Automatic Control.* To appear.

[39] E. Mazer, J. M. Ahuactzin, and P. Bessière. The Ariadne's clew algorithm. *J. Artificial Intell. Res.*, 9:295–316, November 1998.

[40] B. Mirtich. V-Clip: Fast and robust polyhedral collision detection. Technical Report TR97-05, Mitsubishi Electronics Research Laboratory, 1997.

[41] R. M. Murray and S. Sastry. Nonholonomic motion planning: Steering using sinusoids. *Trans. Automatic Control*, 38(5):700–716, 1993.

[42] C. O'Dunlaing. Motion planning with inertial constraints. *Algorithmica*, 2(4):431–475, 1987.

[43] I. Pohl. Bi-directional and heuristic search in path problems. Technical report, Stanford Linear Accelerator Center, 1969.

[44] S. Quinlan. Efficient distance computation between nonconvex objects. In *IEEE Int. Conf. Robot. & Autom.*, pages 3324–3329, 1994.

[45] J. Reif and H. Wang. Non-uniform discretization approximations for kinodynamic motion planning. In J.-P. Laumond and M. Overmars, editors, *Algorithms for Robotic Motion and Manipulation*, pages 97–112. A K Peters, Wellesley, MA, 1997.

[46] L. Sander. Fractal growth processes. *Nature*, 322:789–793, 1986.

[47] A. Scheuer and Ch. Laugier. Planning sub-optimal and continuous-curvature paths for car-like robots. In *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, pages 25–31, 1998.

[48] S. Sekhavat, P. Svestka, J.-P. Laumond, and M. H. Overmars. Multilevel path planning for nonholonomic robots using semiholonomic subsystems. *Int. J. Robot. Res.*, 17:840–857, 1998.

[49] S. Sundar and Z. Shiller. Optimal obstacle avoidance based on the Hamilton-Jacobi-Bellman equation. *IEEE Trans. Robot. & Autom.*, 13(2):305–310, April 1997.

[50] P. Svestka and M.H. Overmars. Coordinated motion planning for multiple car-like robots using probabilistic roadmaps. In *Proc. of IEEE Int. Conf. on Robotics and Automation*, 1995.

[51] D. Vallejo, C. Jones, and N. Amato. An adaptive framework for "single shot" motion planning. Texas A&M, October 1999.

[52] J. H. Yakey. Randomized path planning for linkeages with closed kinematic chains. Master's thesis, Iowa State University, Ames, IA, 1999.

[53] Y. Yu and K. Gupta. On sensor-based roadmap: A framework for motion planning for a manipulator arm in unknown environments. In *IEEE/RSJ Int. Conf. on Intelligent Robots & Systems*, pages 1919–1924, 1998.

Appendix

Proofs of the propositions from Secton 4 are included here.

Lemma 1 Let x be any point in X_{free}, and let x_0 denote any initial RRT vertex. Let $B(x)$ denote a ball of radius ϵ, centered on x. Let $B'(x) = B(x) \cap X_{free}$. Note that $\mu(B'(x)) > 0$, in which μ denotes the volume (or measure) of a set. Initially, $d_1(x) = \rho(x, x_0)$. At each RRT iteration, the probability that the randomly-chosen point will lie in $B'(x)$ is strictly positive. Therefore, if all RRT vertices lie outside of $B(x)$, then $E[D_k] - E[D_{k+1}] > b$ for some positive real number $b > 0$. This implies that $\lim_{k\to\infty} P[d_k(x) < \epsilon] = 1$. $\triangle$

Lemma 2 Let x_0 denote any initial RRT vertex. If x_0 and x are in the same connected component of a bounded open set, then there exists a sequence, x_1, x_2, $\ldots$, x_k, of states such that a sequence of balls, $\mathcal{B} = B_1(x_1), \ldots, B_k(x_k)$, can be constructed with $B_i \cap B_{i+1} \neq \emptyset$ for each $i \in \{1, \ldots, n-1\}$, $x_0 \in B_1$, and $x \in B_k$. Let $C_i = B_i \cap B_{i+1}$. Note that $\mathcal{B}$ can be constructed so that each C_i is open, which implies that $\mu(C_i) > 0$. Lemma 1 can be applied inductively to each C_i to conclude that $\lim_{n\to\infty} P[d_n(x_i) < \epsilon] = 1$ for a point in $x_i \in C_i$. In each case, ϵ can be selected to guarantee that an RRT vertex lies in C_i. Eventually, the probability approaches one that an RRT vertex will fall into B_k. One final application of Lemma 1 implies that $P[d_n(x) < \epsilon] = 1$. $\triangle$

Theorem 4 Consider the set $Y_k = \{x \in X_{free} \mid \rho(x, v) > \epsilon \;\forall v \in V_k\}$, in which V_k is the set of RRT vertices after iteration k. Intuitively, this represents the "uncovered" portion of X_{free}. From Lemma 2, it follows that $Y_{k+1} \subseteq Y_k$ and $\mu(Y_k)$ approaches zero as k approaches infinity. Recall that the RRT construction algorithm adds a vertex to V if the sample lies within ϵ of another vertex in V (ϵ is the RRT step size). Each time this occurs, the new RRT vertex follows the same probability density as X. Because $\mu(Y_k)$ approaches zero, the probability density functions of $\mathbf{X}$ and $\mathbf{X}_k$ differ only on some set $Z_k \subseteq Y_k$. Since $\mu(Y_k)$ approaches zero as k approaches infinity, $\mu(Z_k)$ also approaches zero. Since $\mu(Z_k)$ approaches zero and the probability density function of $\mathbf{X}$ is smooth, $\mathbf{X}_k$ converges to $\mathbf{X}$ in probability. $\triangle$

Theorem 5 The argument proceeds by induction on i. Assume that the RRT contains x_i as a vertex after some finite number of iterations. Consider the Voronoi diagram associated with the RRT vertices. There exists a positive real number, c_1, such that $\mu(Vor(x_i)) > c_1$ in which $Vor(x_i)$ denotes the Voronoi region associated with x_i. If a random sample falls within $Vor(x_i)$, the vertex will be selected for extension, and a random input is applied; thus, x_i has probability $\mu(Vor(x_i))/\mu(X_{free})$ of being selected. There exists a second positive real number, c_2 (which depends on c_1), such that the probabilty that the correct input, u_i, is selected is at least c_2. If both x_i and u_i have probability of at least c_2 of being selected in each iteration, then the probability tends to one that the next step in the solution trajectory will be constructed. This argument is applied inductively from x_1 to x_k, until the final state $x_{goal} = x_{k+1}$ is reached. $\triangle$