RRT-Connect: An Efficient Approach to Single-Query Path Planning

James J. Kuffner, Jr.
Computer Science Dept.
Stanford University
Stanford, CA 94305 USA
kuffner@cs.stanford.edu

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

Abstract

A simple and efficient randomized algorithm is presented for solving single-query path planning problems in high-dimensional configuration spaces. The method works by incrementally building two Rapidly-exploring Random Trees (RRTs) rooted at the start and the goal configurations. The trees each explore space around them and also advance towards each other through the use of a simple greedy heuristic. Although originally designed to plan motions for a human arm (modeled as a 7-DOF kinematic chain) for the automatic graphic animation of collision-free grasping and manipulation tasks, the algorithm has been successfully applied to a variety of path planning problems. Computed examples include generating collision-free motions for rigid objects in 2D and 3D, and collision-free manipulation motions for a 6-DOF PUMA arm in a 3D workspace. Some basic theoretical analysis is also presented.

1 Introduction

Motion planning problems arise in such diverse fields as robotics, assembly analysis, virtual prototyping, pharmaceutical drug design, manufacturing, and computer animation. Such problems involve searching the system configuration space of one or more complicated geometric bodies for a collision-free path that connects a given start and goal configuration, while satisfying constraints imposed by complicated obstacles. Although complete algorithms are known for this general class of problems [25, 6], their computational complexity limits their use to low-dimensional configuration spaces. This limitation, lower-bound hardness results [24], and strong motivation to handle practical planning problems have stimulated the development and success of many path planning methods that use randomization (e.g., [1, 3, 4, 5, 7, 10, 11, 16, 15, 17, 23, 26]). The accepted tradeoff is that the methods are incomplete, but will find a solution with any probability given sufficient running time. The key is to develop randomized methods that converge quickly in practice, yet are simple enough to yield consistent behavior and analysis.

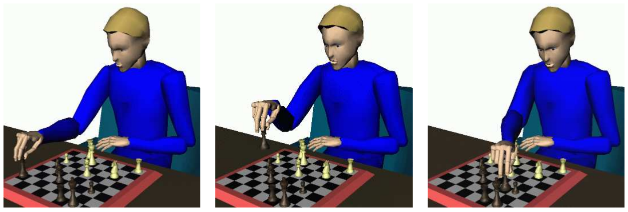

Figure 1: Path planning for a 7-DOF human arm

Randomized path planning algorithms have usually been designed for one of two contexts: *single-query* planning, and *multiple-query* planning [15]. For single-query planning, it is assumed that a single path planning problem must be solved quickly, without any preprocessing. One of the earliest and most popular methods to solve this problem was the randomized potential field approach [4]. For multiple-query planning, it is assumed that many path planning problems will be solved for the same environment. In this case, it is worthwhile to preprocess information and store it in a data structure that allows fast path planning queries. The probabilistic roadmap approach was the first to address this problem [15]. A graph is constructed in the configuration space by choosing many configurations at random, and using a local planner to connect pairs of nearby configurations.

Due to its simplicity and reliable behavior, the probabilistic roadmap approach has enjoyed considerable success in recent years, and current research is focused on analysis [3] and treatment of pathological cases [1]. Even for single-query problems where the randomized potential field planner might yield better performance, the probabilistic roadmap method has been preferred due to its reliability. The randomized potential field planner often finds fast solutions for single-query problems by encoding a greedy heuristic in the form of a potential function over the configuration space. When the planner becomes stuck in local minima, random walks are used to attempt an escape; however, it is very difficult to ensure reliable performance.

This naturally leads to the quest for a simple, reliable approach that shares many of the great properties of probabilistic roadmaps, yet is specifically designed for single-query path planning. We present a simple path planning method called RRT-Connect that combines Rapidly-exploring Random Trees (RRTs) [18] with a simple greedy heuristic that aggressively tries to connect two trees, one from the initial configuration and the other from the goal. The idea of constructing search trees from the initial and goal configurations comes from classical AI bidirectional search, and an overview of its use in previous motion planning methods appears in [12]. Recently, an interesting randomized bidirectional planner was proposed for high-DOF problems in [11]. The key to our ideas is the use of RRTs as a simple sampling scheme and data structure that reliably leads to fast and uniform exploration of the configuration space. RRT-Connect was originally developed to plan collision-free motions for 7-DOF human arms for the automatic animation of grasping and manipulation tasks for animated characters in interactive 3D virtual environments [14] (see Figure 1). However, it has also been found to be both efficient and reliable for a variety of path planning problems.

2 Rapidly-Exploring Random Trees

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. A metric ρ is assumed to be defined on $\mathcal{C}$. Let $\mathcal{C}_{free}$ denote the set of configurations for which these bodies do not collide with any static obstacles. The obstacles are modeled completely in the world, and an explicit representation of $\mathcal{C}_{free}$ is not available. However, using a collision detection algorithm, a given $q \in \mathcal{C}$ can be tested to determine whether $q \in \mathcal{C}_{free}$. The single-query path planning task is to compute a continuous path from an initial configuration, q_{init}, to a goal configuration, q_{goal}, without performing any preprocessing.

The Rapidly-exploring Random Tree (RRT) was introduced in [18] 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. Related ideas have been developed in path planning research, such as Ariadne's clew algorithm [21] and expansive configuration spaces [11]. In [19] an RRT-based approach to kinodynamic and nonholonomic planning was presented that generated and connected two RRTs in a state space, which generalizes $\mathcal{C}$. In the current paper, we present an approach that is

```
BUILD_RRT(q_init)
  T.init(q_init);
  for k = 1 to K do
      q_rand ← RANDOM_CONFIG();
      EXTEND(T, q_rand);
  Return T
```

```
EXTEND(T, q)
  q_near ← NEAREST_NEIGHBOR(q, T);
  if NEW_CONFIG(q, q_near, q_new) then
      T.add_vertex(q_new);
      T.add_edge(q_near, q_new);
      if q_new = q then
          Return Reached;
      else
          Return Advanced;
  Return Trapped;
```

Figure 2: The basic RRT construction algorithm.

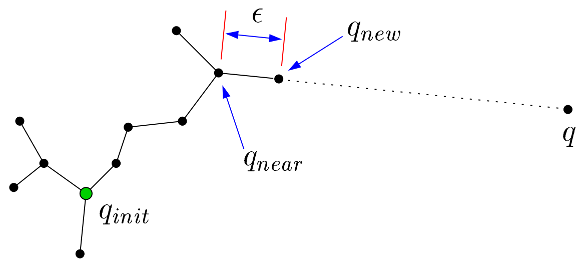

Figure 3: The EXTEND operation.

tailored to problems in which there are no differential constraints, and the problem can be expressed in $\mathcal{C}$.

The basic RRT construction algorithm is given in Figure 2. 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 configuration. The *EXTEND* function, illustrated in Figure 3, selects the nearest vertex already in the RRT to the given sample configuration, q. The function *NEW_CONFIG* makes a motion toward q with some fixed incremental distance ϵ, and tests for collision. This can be performed quickly ("almost constant time") using incremental distance computation algorithms [9, 20, 22]. Three situations can occur: *Reached*, in which q is directly added to the RRT because it already contains a vertex within ϵ of q; *Advanced*, in which a new vertex $q_{new} \neq q$ is added to the RRT; *Trapped*, in which the proposed new vertex is rejected because it does not lie in $\mathcal{C}_{free}$. The top row of Figure 4 shows an RRT 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

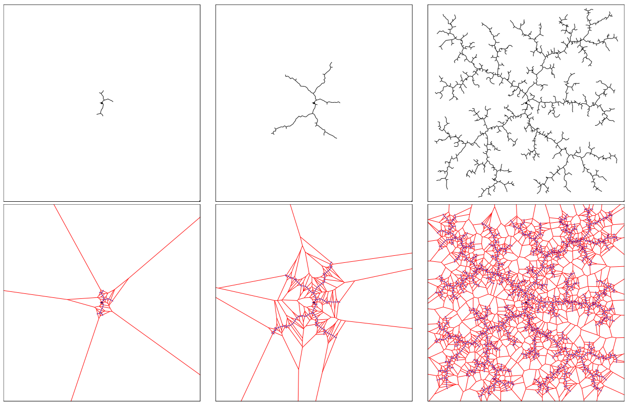

Figure 4: An RRT is biased by large Voronoi regions to rapidly explore, before uniformly covering the space.

CONNECT($\mathcal{T}$, q)
repeat
  $S \leftarrow$ EXTEND($\mathcal{T}$, q);
until not (S = *Advanced*)
Return S;

RRT_CONNECT_PLANNER(q_{init}, q_{goal})
$\mathcal{T}_a$.init(q_{init}); $\mathcal{T}_b$.init(q_{goal});
for $k = 1$ **to** K **do**
  $q_{rand} \leftarrow$ RANDOM_CONFIG();
  if not (EXTEND($\mathcal{T}_a$, q_{rand}) = *Trapped*) **then**
    if (CONNECT($\mathcal{T}_b$, q_{new}) = *Reached*) **then**
      Return PATH($\mathcal{T}_a$, $\mathcal{T}_b$);
  SWAP($\mathcal{T}_a$, $\mathcal{T}_b$);
Return *Failure*

Figure 5: The RRT-Connect algorithm.

area of its Voronoi region. This causes the RRT to be biased to rapidly explore. In addition, Section 4 shows that RRTs arrive at a uniform coverage of the space, which is also a desirable property of the probabilistic roadmap planner.

3 The RRT-Connect Path Planner

The RRT-Connect planner is designed specifically for path planning problems that involve no differential constraints. In this case, the need for incremental motions is less important. The method is based on two ideas: the Connect heuristic that attempts to move over a longer distance, and the growth of RRTs from both q_{init} and q_{goal}.

The Connect heuristic is a greedy function that can be considered as an alternative to the *EXTEND* function in Figure 2. Instead of attempting to extend an RRT by a single ϵ step, the Connect heuristic iterates the *EXTEND* step until q or an obstacle is reached, as shown in the *CONNECT* algorithm description in Figure 5. This operation serves a similar function as the artificial potential function in a randomized potential field approach. In both cases, the heuristic allows rapid convergence to a solution. However, with our method, the greedy heuristic is combined with the rapid and uniform exploration properties of RRTs, which seems to avoid the well-known pitfalls of local minima. In a sense, with the *CONNECT* heuristic, the basin of attraction continues to move around as the RRT grows, as opposed to an artificial potential field method, in which the basin of attraction remains fixed at the goal.

Figure 5 shows the *RRT_CONNECT_PLANNER* algorithm, which may be compared to the *BUILD_RRT* algorithm of Figure 2. 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 the other tree to the new vertex. Then, the roles are reversed by swapping the two trees. This causes both trees to explore $\mathcal{C}_{free}$, while trying to establish a connection between them. The growth of two RRTs was also proposed in [19] for kinodynamic planning; however, in each iteration both trees were incrementally extended toward a random configuration. The current algorithm attempts to also grow the trees towards each other, which has been found to yield much better performance.

Several variations of the above planner can also be considered. By replacing *CONNECT* by *EXTEND* in *RRT_CONNECT_PLANNER*, a simple, two-RRT planner is obtained. Adapting this planner to problems that involve differential constraints would most likely give significant performance improvement over the planner in [19]. Another variant can be obtained by replacing *EXTEND* with *CONNECT* in *RRT_CONNECT_PLANNER*. This would lead to a path planner with an even stronger greedy heuristic. 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 (each new vertex will become the nearest neighbor for the next one). This advantage motivates the choice of a greedier algorithm; however, if an efficient nearest-neighbor algorithm [2, 13] is used, as opposed to the obvious linear-time method, then it might make sense to be less greedy. Another possible variation is to make *CONNECT* add only the last vertex in the *EXTEND* iteration to the RRT, in order to reduce the number of nodes. Since these choices depend on the nearest-neighbor method and the distribution of problems, we focus on a single variation in this paper.

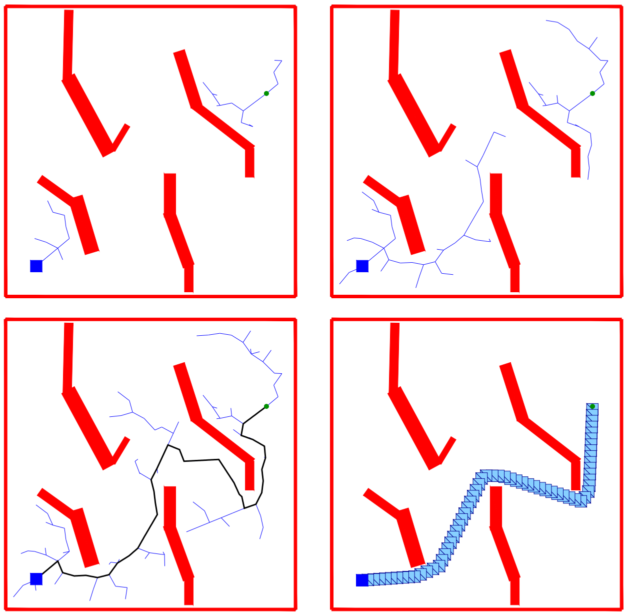

Figure 6: Growing two trees towards each other.

4 Analysis

Both the basic RRT and the RRT-Connect algorithms are analyzed in this section. The key result is that the distribution of the RRT vertices converges toward the sampling distribution, which is usually uniform. Furthermore, the RRT-Connect algorithm is probabilistically complete. Unfortunately, we do not have a theoretical characterization of the rate of convergence (which is observed to be very fast in practice).

Let $D_k(q)$ denote a random variable whose value is the distance of q 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).

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

Sketch of Proof: Let q be any point in $\mathcal{C}_{free}$, and let q_0 denote any initial RRT vertex. Let $B(q)$ denote a ball of radius ϵ, centered on q. Let $B'(q) = B(q) \cap \mathcal{C}_{free}$. Note that $\mu(B'(q)) > 0$, in which μ denotes the volume (or measure) of a set. Initially, $d_1(q) = \rho(q, q_0)$. At each RRT iteration, the probability that the randomly-chosen point will lie in $B'(q)$ is strictly positive. Therefore, if all RRT vertices lie outside of $B(q)$, then $E[D_k] - E[D_{k+1}] > b$ for some positive real number $b > 0$. This implies that $\lim_{k\to\infty} P[d_k(q) < \epsilon] = 1$. $\triangle$

For the statements that follow, assume that $\mathcal{C}_{free}$ is a nonconvex, open set with a single connected component.

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

Sketch of Proof: Let q_0 denote any initial RRT vertex. If q_0 and q are in the same connected component of a bounded open set, then there exists a sequence, $q_1, q_2, \ldots, q_k$, of configurations such that a sequence of balls, $\mathcal{B} = B_1(q_1), \ldots, B_k(q_k)$, can be constructed with $B_i \cap B_{i+1} \neq \emptyset$ for each $i \in \{1, \ldots, n-1\}$, $q_0 \in B_1$, and $q \in B_k$. 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 $\lim_{n\to\infty} P[d_n(q_i) < \epsilon] = 1$ for a point in $q_i \in C_i$. 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 $P[d_n(q) < \epsilon] = 1$. $\triangle$

Let 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_CONFIG* function in the *EXTEND* algorithm. Usually, X is characterized by a uniform probability density function over $\mathcal{C}_{free}$; however, we will allow X to be characterized by any smooth probability density function. Let X_k denote a vector-valued random variable that represents the distribution of the RRT vertices.

Theorem 1 *X_k converges to X in probability.*

Sketch of Proof: Consider the set $Y_k = \{q \in \mathcal{C}_{free} \mid \rho(q, v) > \epsilon \quad \forall v \in V_k\}$, in which V_k is the set of RRT vertices after iteration k. Intuitively, this represents the "uncovered" portion of $\mathcal{C}_{free}$. 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 X and 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 X is smooth, X_k converges to X in probability. $\triangle$

Corollary 1 *The RRT-Connect algorithm is probabilistically complete and vertices converge to a uniform distribution over $\mathcal{C}_{free}$.*

Sketch of Proof: The result follows by observing that Theorem 1 holds for multiple RRTs, in addition to a single RRT. Furthermore, the Connect heuristic generates all of the usual RRT vertices, plus additional vertices. These additional vertices will contribute to the covering of $\mathcal{C}_{free}$, and therefore do not adversely affect the convergence results, in which $\mu(Y_k)$ and $\mu(Z_k)$ approach zero. $\triangle$

5 Experiments

This section presents some preliminary experiments performed on a 270 MHz SGI O2 (R12000) workstation. 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. Some of these results are shown in Figure 7, in which the left column shows the RRTs, and the right column shows the corresponding solutions. Averaging over 100 trials, the (wall-clock) computation times were 0.13s, 1.52s, and 1.02s, for these three examples. The collision checking software used for all experiments was the RAPID library based on OBB-Trees developed by the University of North Carolina [8]. The performance was compared between RRT-Connect and several other RRT-based variants, including an adaptation of the algorithm presented in [19]. We determined that for most problems, the Connect heuristic improves the running time, often by a factor of three or four, especially for uncluttered environments. In very cluttered environments, the Connect heuristic only slightly increases running time in comparison to using the *EXTEND* function to construct two trees. Thus, it seems that the greedy behavior is worthwhile on average. Additional experiments may reveal other variants that further improve performance. We are currently comparing some of the variants discussed in Section 3.

A variety of more challenging experiments were performed. Figure 8 depicts a computed solution for a 3D model of a grand piano (over 4500 triangles) 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. The average computation time was 12.5 seconds (100 trials). 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-Connect 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 9. This set

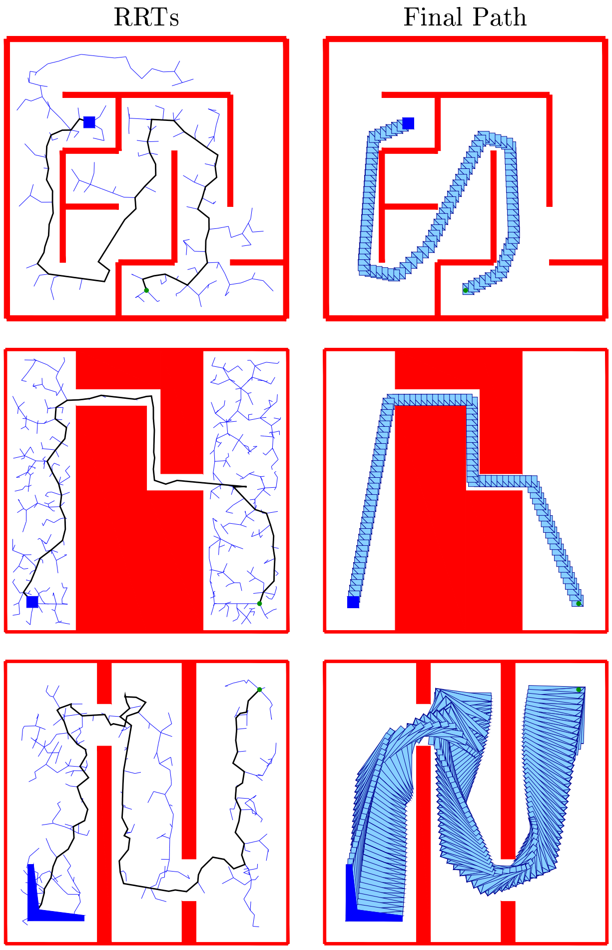

Figure 7: Several basic results.

of three queries were each solved in under 4 seconds on average.

As mentioned previously, the Connect heuristic works most effectively when one can expect relatively open spaces for the majority of the planning queries. The Connect heuristic was originally developed with this kind of problem in mind [14]. Figure 1 shows a human character playing chess. Each of the motions necessary to reach, grasp, and reposition a game piece on the virtual chessboard were generated using the RRT-Connect planner in under 2 seconds on average. The human arm is modeled as a 7-DOF kinematic chain, and the entire scene contains over 8,000 triangle primitives. The speed of the planner allows for the user to interact with the character in real-time, and engage in a game of “interactive virtual chess.” The planner can also handle more complicated queries with narrow passages in $\mathcal{C}_{free}$, 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

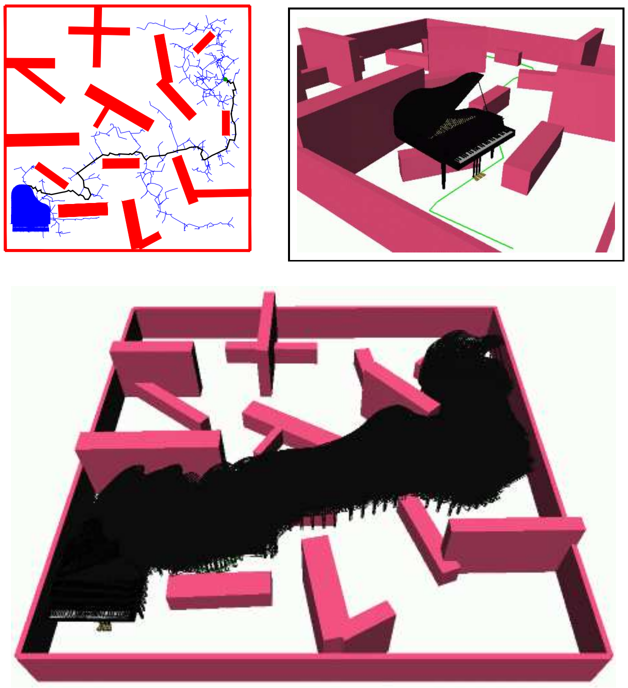

Figure 8: Moving a Piano

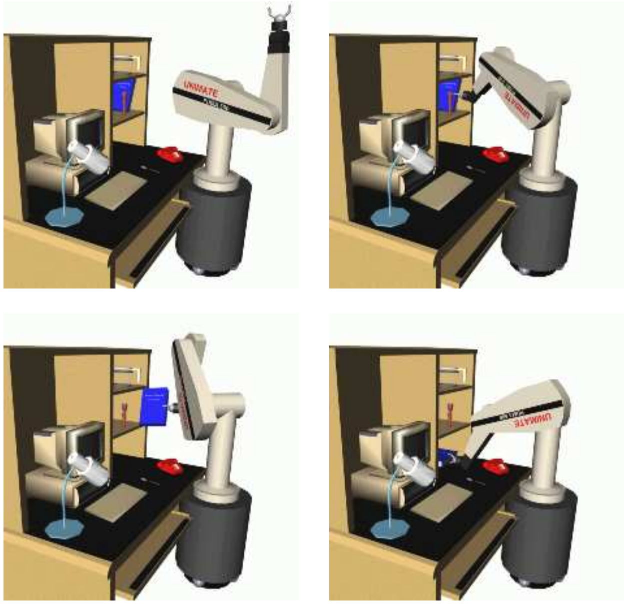

Figure 9: Path planning for a 6-DOF Puma robot.

of queries takes an average of 17 seconds. The scene contains over 13,000 triangles. Since we used a non-incremental 3D collision checking algorithm, performance could potentially be improved significantly by using an alternate algorithm (for example [9, 20, 22]).

6 Conclusions

A randomized approach to single-query path planning is proposed that yields good experimental performance over a wide variety of examples. The technique is based on Rapidly-exploring Random Trees (RRTs) and the Connect heuristic. Some of the key practical advantages of the planning method include: 1) it does not require parameter tuning; 2) preprocessing is not required, yet interactive performance can be obtained for many difficult problems; 3) simple and consistent behavior was observed through repeated experiments; 4) a reasonable balance has been struck between greedy searching (as in a potential field planner) and uniform exploration (as in a probabilistic roadmap planner); 5) the method is well-suited for incremental distance computation algorithms and fast nearest-neighbor algorithms. The practical performance observed so far is encouraging; however, an extensive study that involves many benchmarking examples would be useful, and is currently under investigation. Undoubtedly, pathological cases exist for RRT-Connect, and more experimental work is needed to determine conditions under which RRT-Connect will yield very poor performance. We have shown theoretically that the planner is probabilistically complete and the vertices tend to a uniform distribution over $\mathcal{C}_{free}$. Theoretical analysis of the convergence rate remains, which is one topic under current investigation.

Although the RRT-Connect has proven to be very successful in our experiments, we are aware of several intertwined factors that could improve performance even further. Experimental evaluation of these issues form the basis of our future research: 1) the length of each RRT step can be optimized by computing the radius of a collision-free ball in $\mathcal{C}_{free}$ using the result of the distance computation algorithm; 2) the *CONNECT* heuristic can be used entirely in the RRT-Connect planner, as opposed to a combination of *CONNECT* and *EXTEND*; 3) vertices that are discovered during each incremental step within *CONNECT* do not need to be added to the RRT to increase efficiency; 4) approximate nearest neighbor methods can be used to reduce computation time for n vertices from $O(n)$ to near-logarithmic time; 5) incremental collision detection algorithms can be used.

Acknowledgments

This work has benefitted greatly from discussions with Nancy Amato, David Hsu, Lydia Kavraki, Karl MacDorman, and Jean-Claude Latombe. Kuffner has been supported in part by a National Science Foundation Graduate Fellowship in Engineering, and MURI grant DAAH04-96-1-007. LaValle has been

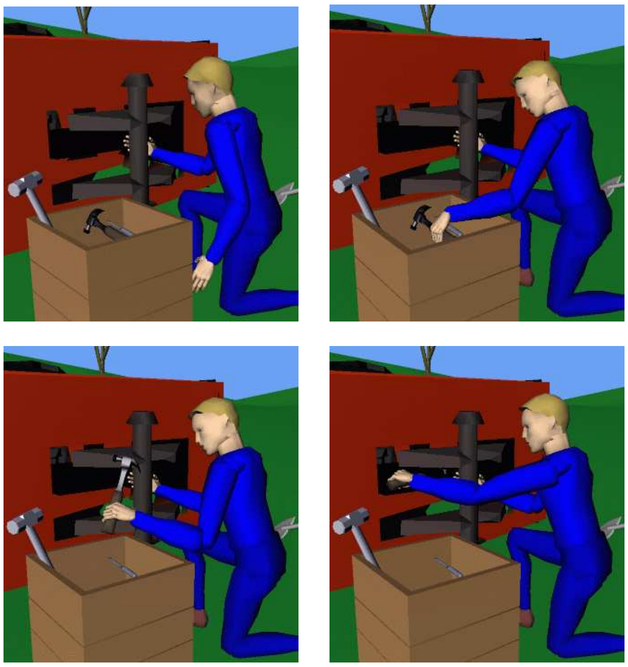

Figure 10: A path planning problem that involves finding and using a hammer in a virtual world.

supported in part by NSF CAREER Award IRI-9875304 (LaValle).

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] 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.

[3] J. Barraquand, L.E. Kavraki, J.C. Latombe, T.Y. Li, R. Motwani, and P. Raghavan. A random sampling scheme for path planning. *Int. J. Robot. Res.*, 16(6):759–774, 1997.

[4] 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.

[5] V. Boor, M. Overmars, and A.F. van der Stappen. The Gaussian sampling strategy for probabilistic roadmap planners. In *Proc. of IEEE Int. Conf. Robotics and Automation*, Detroit, MI, 1999.

[6] J. F. Canny. *The Complexity of Robot Motion Planning.* MIT Press, Cambridge, MA, 1988.

[7] D. Chalou and M. Gini. Parallel robot motion planning. In *Proc. of IEEE Int. Conf. Robotics and Automation*, pages 24–51, Atlanta, GA, 1993.

[8] S. Gottschalk, M. C. Lin, and D. Manocha. Obbtree: A hierarchical structure for rapid interference detection. In *SIGGRAPH '96 Proc.*, 1996.

[9] 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.

[10] T. Horsch, F. Schwarz, and H. Tolle. Motion planning for many degrees of freedom : Random reflections at c-space obstacles. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA'94)*, pages 3318–3323, San Diego, CA, April 1994.

[11] D. Hsu, J.-C. Latombe, and R. Motwani. Path planning in expansive configuration spaces. In *Int. J. of Computational Geometry and Applications*, 1997.

[12] Y. K. Hwang and N. Ahuja. A potential field approach to path planning. *IEEE Trans. Robot. & Autom.*, 8(1):23–32, February 1992.

[13] 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.

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

[15] 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.

[16] L.E. Kavraki and J.-C. Latombe. Randomized preprocessing of configuration space for fast path planning. Technical report, Dept. of Computer Science, Stanford University, September 1993.

[17] Y. Koga, K. Kondo, J. Kuffner, and J.-C. Latombe. Planning motions with intentions. In *Proc. SIGGRAPH '94*, pages 395–408, 1994.

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

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

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

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

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

[23] M. Overmars. A random approach to motion planning. Technical report, Dept. Computer Science, Utrect University, The Netherlands, October 1992.

[24] J. H. Reif. Complexity of the mover's problem and generalizations. In *Proc. of IEEE Symp. on Foundat. of Comp. Sci.*, pages 421–427, 1979.

[25] J. T. Schwartz and M. Sharir. On the piano movers' problem: Coordinating the motion of several independent bodies. *Int. J. Robot. Res.*, 2(3):97–140, 1983.

[26] P. Svestka. A probabilistic approach to motion planning for car-like robots. Technical report, Dept. Computer Science, Utrect Univ., April 1993.