To appear in Proc. 1999 IEEE International Conference on Robotics and Automation

Randomized Kinodynamic Planning

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

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

Abstract *This paper presents a state-space perspective on the kinodynamic planning problem, and introduces a randomized path planning technique that computes collision-free kinodynamic trajectories for high degree-of-freedom problems. By using a state space formulation, the kinodynamic planning problem is treated as a* $2n$*-dimensional nonholonomic planning problem, derived from an* n*-dimensional configuration space. The state space serves the same role as the configuration space for basic path planning; however, standard randomized path planning techniques do not directly apply to planning trajectories in the state space. We have developed a randomized planning approach that is particularly tailored to kinodynamic problems in state spaces, although it also applies to standard nonholonomic and holonomic planning problems. The basis for this approach is the construction of a tree that attempts to rapidly and uniformly explore the state space, offering benefits that are similar to those obtained by successful randomized planning methods, but applies to a much broader class of problems. Some preliminary results are discussed for an implementation that determines kinodynamic trajectories for hovercrafts and satellites in cluttered environments, resulting in state spaces of up to twelve dimensions.*

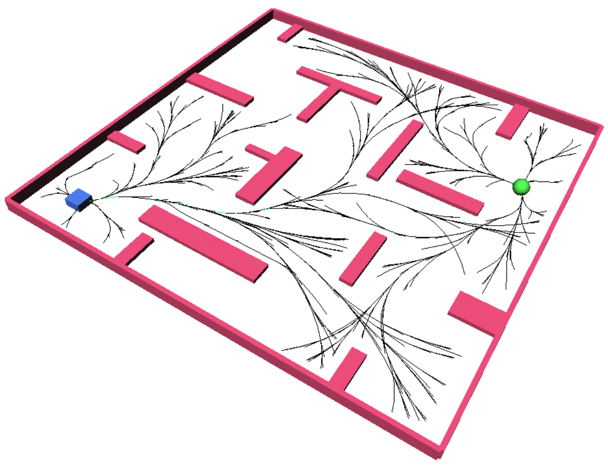

Figure 1. We consider planning problems with dynamic constraints induced by physical laws. The above image shows the state exploration trees computed for a rigid rectangular object (left). The goal location is represented by a sphere (upper right).

1 Introduction

There is a strong need for a simple, efficient planning technique that determines control inputs to drive a robot from an initial configuration and velocity to a goal configuration and velocity while obeying physically-based dynamic constraints and avoiding obstacles in the robot's environment. Although many interesting approaches exist to specific kinodynamic problems, they fall short of being able to solve many complicated, high degree-of-freedom problems. Randomized techniques have led to efficient, incomplete planners for basic path planning (holonomic and purely kinematic); however, there appears to be no equivalent technique for the broader kinodynamic planning problem (or even nonholonomic planning in the configuration space). We try to account for some of the reasons for this, and argue the need for a simple, general-purpose kinodynamic planner. We present a heuristic, randomized approach to kinodynamic planning that quickly explores the state space, and scales well for problems with high degrees-of-freedom and complicated system dynamics.

The common model in motion planning research has been to decouple the general robotics problem by solving basic path planning, and then finding a trajectory and controller that satisfies the dynamics and follows the path [3, 16, 24]. The vast majority of basic path planning algorithms consider only *kinematics*, while ignoring the system *dynamics* entirely. Motion planning that takes into account dynamic constraints as well as kinematic constraints is known as *kinodynamic planning* [10]. In this paper, we consider kinodynamic planning as a generalization of holonomic and nonholonomic planning in configuration spaces, by replacing popular configuration-space notions by their *state space* (or phase space) counterparts. A point in the state space includes both configuration parameters and velocity parameters (i.e., it is the tangent bundle of the configuration space).

It may be the case that the result of a purely kinematic planner will be *unexecutable* by the robot in the environment due to limits on the actuator forces and torques. Imprecision in control, which is always present in real-world robotic systems, may require explicitly modeling system dynamics to guarantee collision-free trajectories. Robots with significant dynamics are those in which natural physical laws, along with limits on the available controls, impose severe constraints on the allowable velocities at each configuration. Examples of such systems include helicopters, airplanes, certain-classes of wheeled vehicles, submarines, unanchored space robots, and legged robots with fewer than four legs. In general, it is preferable to look for solutions to these kinds of systems that naturally flow from the physical models, as opposed to viewing dynamics as an obstacle.

These concerns provide the general basis for kinodynamic planning research. Algebraic approaches solve for the trajectory exactly, though the only known solutions are for point masses with velocity and acceleration bounds in one-dimension[19] and two-dimensions[5]. Provably approximately-optimal kinodynamic trajectories are computed in [10] by performing a search of the state space by systematically applying control inputs. Other papers have extended or modified this technique [9, 8, 12, 21]. In [11], an incremental, variational approach is presented to perform state-space search. An approach to kinodynamic planning based on Hamiltonian mechanics is presented in [7]. Sensor-based motion strategies that account for robot inertia have been devised that maintain an emergency stopping path [25].

The computational complexity of kinodynamic planning problems depends upon the assumptions made for a particular instance of the problem. However, kinodynamic planning in general is believed to be at least as hard as the *generalized mover's problem*, which has been proven to be PSPACE-hard [22]. Hard bounds have also been established for time-optimal trajectories. Finding an exact time-optimal trajectory for a point mass with bounded acceleration and velocity moving amidst polyhedral obstacles in 3D has been proven to be NP-hard [10]. The need for simple, efficient algorithms for kinodynamic planning, along with the discouraging lower-bound complexity results, have motivated us to explore the development of randomized techniques for kinodynamic planning. This parallels the reasoning that led to the success of randomized planning techniques for basic path planning.

2 A State Space Formulation

We formulate the kinodynamic planning problem as path planning in an $2n$-dimensional state space that has first-order nonholonomic constraints. We would like the state space to have the same utility as a representational tool as the configuration space for a purely-kinematic problem. Let $\mathcal{C}$ denote the configuration space ($\mathcal{C}$-space) that arises from a rigid or articulated body that moves in a 2D or 3D world. Let $\mathcal{X}$ denote the state space, in which a state, $x \in \mathcal{X}$, is defined as $x = (q, \dot{q})$, for $q \in \mathcal{C}$.

Constraints When planning in $\mathcal{C}$, nonholonomic constraints often arise from the presence of one or more rolling contacts between rigid bodies, or from the set of controls that it is possible to apply to a system. When planning in $\mathcal{X}$, nonholonomic constraints also arise from conservation laws (e.g. angular momentum conservation). Using Lagrangian mechanics, the dynamics can be represented by a set of equations of the form $h_i(\ddot{q}, \dot{q}, q) = 0$. Using the state space representation, this

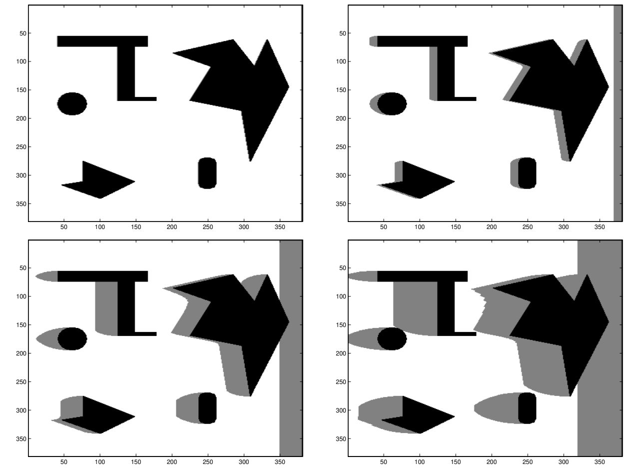

Figure 2. Slices of $\mathcal{X}$ for a point mass robot in 2D with increasingly higher initial speeds. White areas represent $\mathcal{X}_{free}$; black areas are X_{obst}; gray areas approximate $\mathcal{X}_{ric}$.

can be simply written as a set of m implicit equations of the form $G_i(x, \dot{x}) = 0$, for $i = 1, \dots, m$ and $m < 2n$. It is well known that under appropriate conditions the Implicit Function Theorem allows the constraints to be written in the form of a control system

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

in which $u \in U$, and U represents a set of allowable controls or inputs. We assume that f is a smooth function of (x, u). Equation 1 effectively yields a convenient parameterization of the allowable state transitions via the controls in U.

Obstacles in $\mathcal{X}$ Assume that the world in which the robot lives contains static obstacles. There are interesting differences between finding collision-free paths in $\mathcal{C}$ versus the state space, $\mathcal{X}$. When planning in $\mathcal{C}$, it is useful to characterize the set $\mathcal{C}_{obst}$ of configurations at which the robot is in collision with an obstacle (or itself) [16]. The path planning problem involves finding a continuous path that maps into $\mathcal{C}_{free} = \mathcal{C} \setminus \mathcal{C}_{obst}$. For planning in $\mathcal{X}$, this could lead to a straightforward definition of $\mathcal{X}_{obst}$ by declaring $x \in \mathcal{X}_{obst}$ if and only if $q \in \mathcal{C}_{obst}$ for $x = (q, \dot{q})$. However, another interesting possibility exists: *the region of inevitable collision.* Let $\mathcal{X}_{ric}$ denote the set of states in which the robot is either in collision or, because of its velocity, it cannot do anything to avoid collision (there exist no controls that will prevent it). Note that $X_{obst} \subseteq X_{ric}$. Thus, it might be preferable to define $\mathcal{X}_{free} = \mathcal{X} \setminus \mathcal{X}_{ric}$, as opposed to $\mathcal{X} \setminus \mathcal{X}_{obst}$.

Figure 2 illustrates conservative approximations of $\mathcal{X}_{ric}$ for a point mass robot. The robot is assumed to have L^2-bounded acceleration, and an initial velocity pointing along the positive x axis. As expected intuitively, if the speed increases, $\mathcal{X}_{ric}$ grows.

Solution Trajectory The kinodynamic planning problem is to find a trajectory from an initial state x_{init}

to a goal state x_{goal}. A trajectory is defined as a time-parameterized continuous path $\tau : [0,T] \to \mathcal{X}_{free}$ that satisfies the nonholonomic constraints. By integrating Equation 1 from an initial state and control, we obtain a trajectory that inherently satisfies the nonholonomic constraints. Our task is then reduced to finding a control function $u : [0,T] \to U$, representing time-varying input that when applied, moves the system from x_{init} to x_{goal} while avoiding obstacles. It might also be appropriate to select a path that optimizes some criterion, such as the time to reach x_{goal}.

3 Issues in Randomized Kinodynamic Planning

One of the key differences between $\mathcal{X}$ and $\mathcal{C}$ is a factor of two in dimension. The curse of dimensionality has already contributed to the success and popularity of randomized planning methods for $\mathcal{C}$-space; therefore, it seems that there would be an even greater need to develop randomized algorithms for kinodynamic planning. One reason that might account for the lack of practical, efficient planners for problems in $\mathcal{X}$-space is that attention is usually focused on the decoupled planning problem. Another plausible reason is that kinodynamic planning is considerably harder, aside from the fact that the dimension is larger. We briefly indicate some reasons for this difficulty.

Existing Randomized Techniques It would certainly be useful if ideas can be borrowed or adapted from existing randomized path planning techniques that have been successful for planning in $\mathcal{C}$-space. For the purpose of discussion, we choose two different techniques that have been successful in recent years: randomized potential fields (e.g, [2, 6]) and randomized roadmaps (e.g., [1, 15]). In the randomized potential field approach, a heuristic function is defined on the configuration space that attempts to steer the robot toward the goal through gradient descent. If the search becomes trapped in a local minimum, random walks are used to help escape. In the randomized roadmap approach, a graph is constructed in the configuration space by generating random configurations and attempting to connect pairs of nearby configurations with a local planner. Once the graph has been constructed, the planning problem becomes one of searching a graph for a path between two nodes.

Why is Kinodynamic Planning Harder? Consider applying either of the previously mentioned randomized techniques to the problem of finding a path in $\mathcal{X}_{free}$ that also satisfies (1), instead of finding a holonomic path in $\mathcal{C}_{free}$. The potential field method appears nicely suited to the problem because a discrete-time control can repeatedly selected that reduces the potential. The primary problem is that dynamical systems usually have drift, which could easily cause the robot to overshoot the goal, leading to oscillations. Without a cleverly-constructed potential function (which actually becomes a difficult nonlinear control problem), the method cannot be expected to work well. Imagine how often the system will be pulled into X_{ric}. The problem of designing a good heuristic function becomes extremely complicated for the case of kinodynamic planning.

The randomized roadmap technique might also appear amenable to kinodynamic planning. The primary requirement is the ability to design a local planner that will connect pairs of configurations (or states in our case) that are generated at random. Indeed, this method was successfully applied to a nonholonomic planning problem in [27]. One result that greatly facilitated this extension of the technique to nonholonomic planning was the existence of Reeds-Shepp curves [20] for car-like robots. This result directly enables the connection of two configurations with the optimal-length path. For more complicated problems, such as kinematic planning for a tractor-trailer, a reasonable roadmap planner can be developed using steering results [4, 17, 23, 26]. These results enable a system to be driven from one configuration to another, and generally apply to driftless systems that are nilpotentizable (a condition on the underlying Lie algebra). In general, however, the connection problem can again be as difficult as designing a nonlinear controller. The randomized roadmap technique might require the connections of thousands of states to find a solution, and if each connection is akin to a nonlinear control problem, it seems impractical for systems that do not allow steering.

4 Rapidly-Exploring Random Trees

The observations of Section 3 motivated us to develop a randomized planning technique that nicely extends to kinodynamic planning (it also applies to the simpler problems of nonholonomic planning in $\mathcal{C}$ and basic path planning in $\mathcal{C}$). Our intention has been to develop a method that easily "drives forward" like potential field methods, and also quickly and uniformly explores the space like randomized roadmap methods. This led us to develop Rapidly-Exploring Random Trees (RRTs).

To motivate and illustrate the concepts, first consider the simple case of planning for a point robot in a two-dimensional configuration space. To prepare for the extension to kinodynamic planning, suppose that the motion of the robot is governed by a control law, $x_{k+1} = f(x_k, u_k)$, which is considered as a discrete-time approximation to (1). For this simple problem, suppose that U represents a direction in S^1 toward which the robot can be moved a fixed, small distance in time Δt. Consider Figure 3, in which the robot starts at (50,50) in an environment that ranges from $(0,0)$ to $(100,100)$, and the robot can move 2 units in one application of the discrete-time control law. The first scheme can be considered as a Naive Random Tree, which is incrementally constructed by randomly picking an existing vertex, x_k from the tree, a control $u_k \in U$ at random, and adding an edge of length 2 from x_k to $f(x_k, u_k)$. Although it appears somewhat random, this tree has a very strong bias towards places it has already explored. To overcome this bias, we propose to construct a Rapidly-Exploring Random Tree as follows. Insert the initial state as a vertex. Repeatedly select a point at random in $[0,100] \times [0,100]$, and find the nearest-neighbor, x_k, in the tree. Choose the control $u_k \in U$ that pulls the ver-

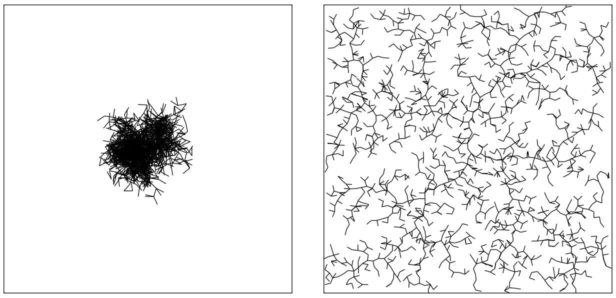

Figure 3. A Naive Random Tree vs. a Rapidly-Exploring Random Tree. Each tree has 2000 vertices.

tex toward the random point. Insert the new edge and vertex for $x_{k+1} = f(x_k, u_k)$. This technique generates a tree that rapidly and uniformly explores the state space. An argument for this can be made by considering the Voronoi regions of the vertices. Random sampling tends to extend vertices that have larger Voronoi regions, and are therefore have too much unexplored space in their vicinity. By incrementally reducing the size of larger Voronoi regions, the graph spreads in a uniform manner.

Rapidly Exploring the State Space When moving from the problem shown in Figure 3 to exploring $\mathcal{X}$ for a kinodynamic planning problem, several complications immediately occur: i) the dimension is typically much higher; ii) the tree must stay within $\mathcal{X}_{free}$; iii) drift and other dynamic constraints can yield undesired motions and biases; iv) there is no natural metric on $\mathcal{X}$ for selecting "nearest" neighbors. For the first complication, approximate nearest neighbor techniques [14] can be employed to help improve performance; it is not crucial to have the absolute closest neighbor. The second complication can make it harder to wander through narrow passageways, much like in the case of randomized roadmaps. The third complication can be partly overcome by choosing an action that brings the velocity components of x as close as possible toward the random sample. The fourth complication might lead to the selection of one metric over another for particular kinodynamic planning problems, if one would like to optimize performance. In theory, there exists a perfect metric (or pseudo-metric due to asymmetry) that could overcome all of these complications if it were easily computable. This is the optimal cost (for any criterion, such as time, energy, etc.) to get from one state to another. Unfortunately, computing the ideal metric as hard as solving the original planning problem. In general, we try to overcome these additional complications while introducing as few heuristics as possible. This enables the planner to be applied with minor adaptation to a broad class of problems.

A Randomized Planning Algorithm We present an algorithm that grows two RRTs, one rooted at the start state x_{init}, and the other rooted at x_{goal}. The algorithm searches for states that are "common" to both trees. Two states, x and x', are considered to be common if $\rho(x, x') < \epsilon$ some some metric ρ and small $\epsilon > 0$. Our basic algorithm stops at the first solution trajec-

GROW_RANDOM_TREES()
InsertState($\mathcal{T}_{init}$, *nil*, x_{init});
InsertState($\mathcal{T}_{goal}$, *nil*, x_{goal});
while *continuePlan* **do**
    $x_{rand} \leftarrow$ RandomState();
    $x_i \leftarrow$ GEN_STATE($\mathcal{T}_{init}$, x_{rand}, FWD);
    if $x_i \neq nil$ **and** NearbyState($\mathcal{T}_{goal}$, x_i) **then**
        record candidate solution τ connecting $\mathcal{T}_{init}$ and $\mathcal{T}_{goal}$ through x_i;
    $x_g \leftarrow$ GEN_STATE($\mathcal{T}_{goal}$, x_{rand}, $BACK$);
    if $x_g \neq nil$ **and** NearbyState($\mathcal{T}_{init}$, x_g) **then**
        record candidate solution τ connecting $\mathcal{T}_{init}$ and $\mathcal{T}_{goal}$ through x_g;

Figure 4. The algorithm incrementally grows two RRTs, from the start and the goal, until meeting at a common state.

tory found, but one could continue to grow the trees and maintain a growing collection of solution trajectories. The "best" solution found so far can be chosen according to a cost functional based on some criteria (such as execution time or energy expended).

Here we present the algorithm in detail. The pseudocode for GROW_RANDOM_TREES() is shown in Figure 4. To begin, we define two RRTs, $\mathcal{T}_{init}$ and $\mathcal{T}_{goal}$, each initialized to contain a single node representing x_{init} and x_{goal} , respectively. We pick a random state x_{rand} and generate new nodes in both trees via the function GEN_STATE(). First, the nearest neighbor of x_{rand} in $\mathcal{T}_{init}$ is selected. From this state, all possible controls are applied to the system for a fixed time interval Δt, yielding successor states derived from $(\mathbf{u}, \Delta t) - bang$motions. The successor states are generated by integrating (1) over Δt. A successor state x_{k+1} that satisfies velocity bounds, is collision-free, and minimizes $\rho(x_{k+1}, x_{rand})$ is inserted into the tree and returned. It is then tested to see if it lies within an ϵ-neighborhood of any of the states generated so far in $\mathcal{T}_{goal}$. If so, we have found a common state that joins the two trees, and we record the candidate solution trajectory τ that joins $\mathcal{T}_{init}$ and $\mathcal{T}_{goal}$ and passes through the node. If not, we invoke GEN_STATE() again on $\mathcal{T}_{goal}$ with the same random state x_{rand}. Successor states are generated exactly as before, except for one minor change. Since we are searching backwards from the goal, we integrate the state transition equation backwards in time. The returned successor state is tested to see if it lies within an ϵ-neighborhood of any of the currently explored states in $\mathcal{T}_{init}$. If so, the candidate solution trajectory is recorded. The algorithm terminates on the first successful solution found.

Our initial experiments attempted to grow a single RRT from x_{init} to connect with the goal x_{goal}. These experiments worked well for state spaces of low dimension. However, growing dual trees improves efficiency for state spaces of high dimension, at the expense of having to connect a pair of nodes between the two trees. In higher dimensions it becomes more difficult to randomly wander upon a state that is close enough to the goal state for each of the state space variables. It might also be possible to improve performance by biasing the

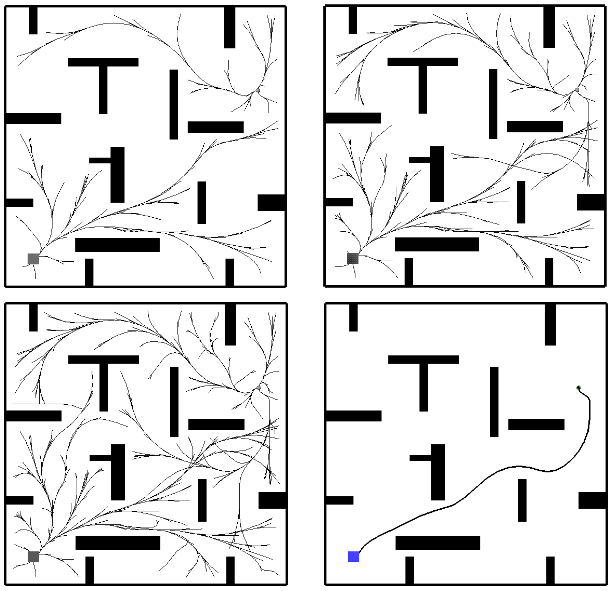

Figure 5. Various stages of state exploration during planning. The top two images show the RRTs after 500 and 1000 nodes, respectively. The bottom two images show the final trees and the computed solution trajectory after 1582 nodes.

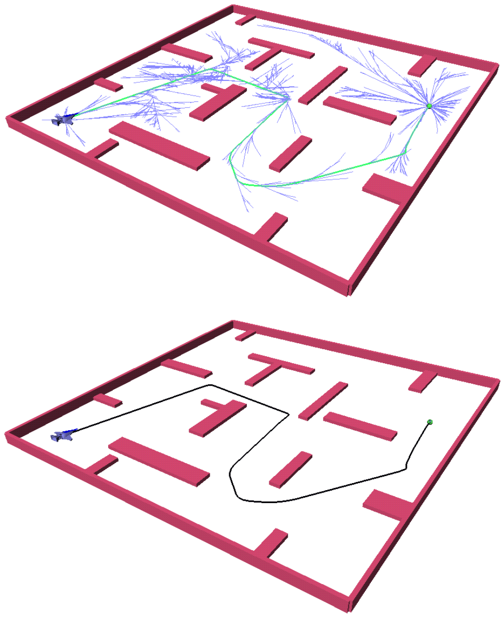

Figure 6. RRTs of 13,600 nodes and solution trajectory for the planar body with unilateral thrusters that allow it to rotate freely but translate only in the forward direction.

sampling toward goal states; this is currently under investigation as an alternative to using dual trees. We generally want to avoid defining some complicated artifical bias because we might be faced with a challenge that is similar to defining a good artificial potential function for the potential field approach to path planning (this task should be even harder for kinodyanmic planning).

5 Hovercrafts and Satellites

Basic experiments involving RRTs in the state space were conducted on simple kinodynamic systems. The algorithm was implemented in C++ on a 200MHz SGI Indigo2 with 128MB of memory. The systems considered involve both non-rotating and rotating rigid objects in 2D and 3D with velocity and acceleration bounds obeying L_2 norms. The dynamic models were derived from Newtonian mechanics of rigid bodies in non-gravity environments. All experiments utilized a simple metric on $\mathcal{X}$ 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.

Planar Translating Body (dim $\mathcal{X} = 4$) The first experiment considered a rigid object with a set of translational controls that restrict its motion to a plane. A total of 4 controls were used, consisting of a set of two pairs of opposing forces acting through the center of mass of the body. Figure 5 shows snapshots during various stages of the computation. Anywhere between 500 and 2500 nodes are explored on average before a solution trajectory is found, with total computation time ranging between 5 and 15 seconds.

Planar Body with Rotation (dim $\mathcal{X} = 6$) We extend the previous experiments to consider systems with rotation. 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 [18]. 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 6 shows the RRT after 13,600 nodes. The total computation time for this example was 4.2 minutes.

Translating 3D Body (dim $\mathcal{X} = 6$) We consider the case of a free-floating rigid object, such as an unanchored satellite in space. The object is assumed to be equipped with thruster controls to be used for translating in a non-gravity environment. The satellite has three opposing pairs of thrusters along each of its principal axes forming a set of six controls spanning a 6-dimensional state space. The task is to thrust through a sequence of two narrow passages amidst a collection of obstacles. Figure 7 shows the RRTs generated during the planning process, and Figure 8 shows the candidate solution found after a total of 16,300 nodes were explored. The total computation time for this case was 4.1 minutes.

3D Body with Rotation (dim $\mathcal{X} = 12$) Finally, we 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

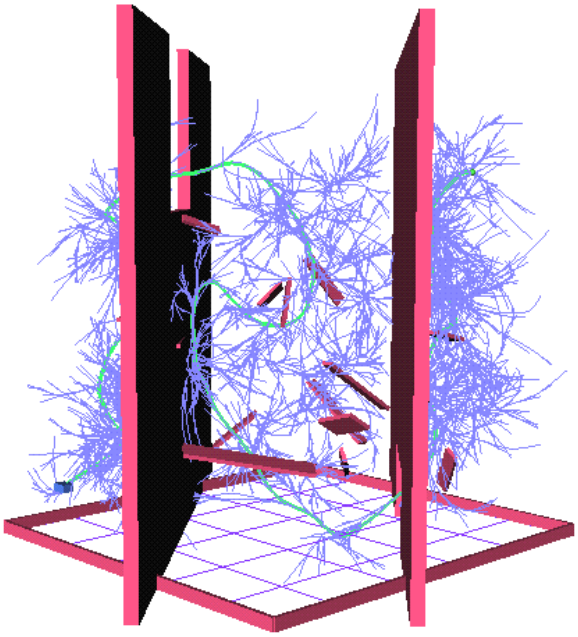

Figure 7. The RRTs computed for the task of navigating a sequence of narrow passages for the 3D translation case.

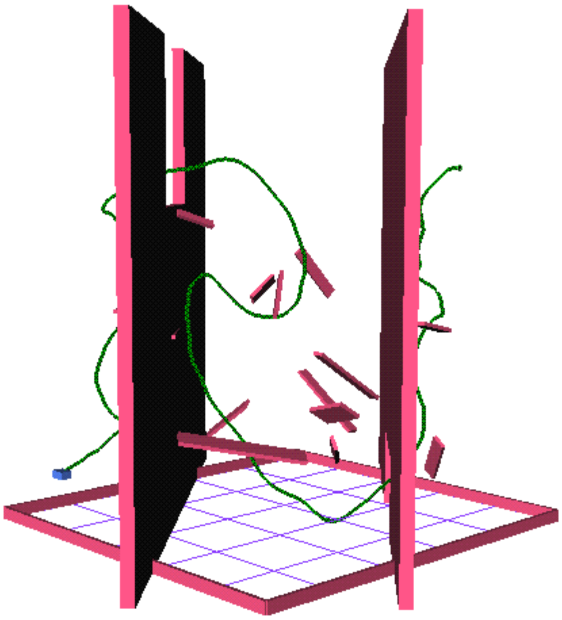

Figure 8. Solution trajectory for navigating through a sequence of narrow passages for the 3D translation case. The initial state is at the lower left; the goal is at the upper right.

cloud of obstacles. Figure 9 illustrates all trajectories explored during the planning process, and Figure 10 shows the candidate solution found after 23,800 states were explored. The total computation time was 8.4 minutes.

6 Discussion

We believe randomized kinodynamic planning techniques will prove useful in a wide array of applications that includes robotics, virtual prototyping, and computer graphics. We presented a state-space perspective on the kinodynamic planning problem that is modeled after the configuration-space perspective on basic path planning. We then presented an efficient, randomized planning technique that is particularly suited to the difficulties that arise in kinodynamic planning. We implemented this technique and generated experiments for hovercraft problems of up to 12 degrees-of-freedom. We still consider this work to be in a preliminary stage, and we are experimenting with different RRT-based variants of the algorithm. Many more experiments will have to be performed, on a wide array of dynamical systems, to assess the full generality and adaptability of the approach. The planning technique appears to generate good paths; however, we make make no claims that the paths are optimal or near optimal (this assumption is common for path planning algorithms in $\mathcal{C}$). One idea for further investigation might be to construct RRTs to find initial trajectories, and then employ a variational technique to optimize the trajectories (see, for example, [28]). We are currently exploring the use of RRTs for other problems, such as kinodynamic planning for automobiles).

Acknowledgments Steve LaValle thanks Bruce Donald for his advice on the kinodynamic planning problem. He also thanks Brian George for his helpful suggestions. Steve LaValle is supported in part by an NSF CAREER award. James Kuffner thanks Nobuaki Akatsu and Bill Behrman for providing him with sound technical advice during the initial stages of this research. James Kuffner is supported in part by an NSF Graduate Fellowship in Engineering.

References

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

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

[3] J. Bobrow, S. Dubowsky, and J. Gibson. Time-optimal control of robotic manipulators. *Int. Journal of Robotics Research*, 4(3), 1985.

[4] 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.

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

[6] D. Challou, D. Boley, M. Gini, and V. Kumar. A parallel formulation of informed randomized search for robot motion planning problems. In *IEEE Int. Conf. Robot. & Autom.*, pages 709–714, 1995.

[7] 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.

[8] 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.

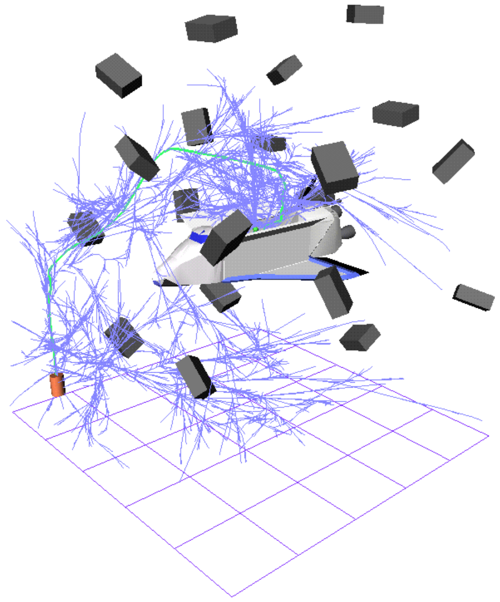

Figure 9. The RRTs constructed during planning for the fully-orientable satellite model with limited translation. A total of 23,800 states were explored before a successful candidate solution trajectory was found.

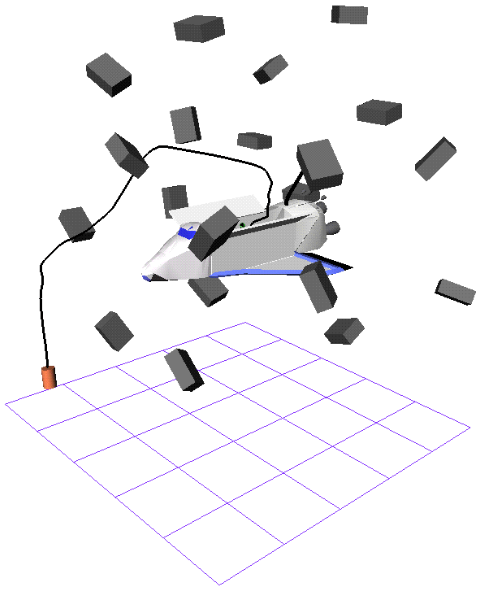

Figure 10. The docking maneuver computed for the fully-orientable satellite model. The satellite's initial state is in the lower left corner, and the goal state is in the interior of the cargo bay of the shuttle.

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

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

[11] 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.

[12] 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.

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

[14] 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.

[15] L. Kavraki, P. Švestka, 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.

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

[17] A. D. Lewis and R. M. Murray. Configuration controllability of simple mechanical control systems. *SIAM Journal on Control and Optimization*, 35(3):766–790, 1997.

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

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

[20] J.A. Reeds and R.A. Schepp. Optimal paths for a car that goes both forward and backward. *Pacific Journal of Mathematics*, 145(2), 1990.

[21] 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.

[22] J. H. Reif. Complexity of the mover's problem and generalizations. In *Proc. 20th IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 421–427, 1979.

[23] S. Sekhavat, F. Lamiraux, J.-P. Laumond, G. Bauzil, and A. Ferrand. Motion planning and control for hilare pulling a trailer: experimental issues. In *Proc. of IEEE Int. Conf. on Robotics and Automation*, April 1997.

[24] Z. Shiller and S. Dubowsky. On computing time-optimal motions of robotic manipulators in the presence of obstacles. *IEEE Trans. on Robotics and Automation*, 7(7), December 1991.

[25] A. Shkel and V. Lumelsky. The jogger's problem : Control of dynamics in real-time motion planning. *Automatica, A Journal of the Int. Federation of Automatic Control*, 33(7):1219–1233, July 1997.

[26] H. K. Struemper. *Motion Control for Nonholonomic Systems on Matrix Lie Groups*. PhD thesis, University of Maryland, College Park, MD, 1997.

[27] 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.

[28] M. Zefran, J. Desai, and V. Kumar. Continuous motion plans for robotic systems with changing dynamic behavior. In *Proc. of IEEE Int. Conf. on Robotics and Automation*, 1996.