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 Mechano-Informatics
University of Tokyo
Bunkyo-ku, Tokyo, Japan
kuffner@jsk.t.u-tokyo.ac.jp

Abstract

This paper presents the first randomized approach to kinodynamic planning (also known as trajectory planning, or trajectory design). The task is to determine control inputs to drive a robot from an initial configuration and velocity to a goal configuration and velocity while obeying physically-based dynamical models and avoiding obstacles in the robot's environment. We consider generic systems that express the nonlinear dynamics of a robot in terms of the robot's high-dimensional configuration space. Kinodynamic planning is treated as motion planning problem in a higher-dimensional state space that has both first-order, differential constraints and obstacle-based, global constraints. 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 trajectory planning problems in high-dimensional state spaces. The basis for this approach is the construction of Rapidly-exploring Random Trees (RRTs), which offer benefits that are similar to those obtained by successful randomized holonomic planning methods, but apply to a much broader class of problems. Theoretical analysis of the algorithm is given. Experimental results are presented for an implementation that computes trajectories for hovercrafts and satellites in cluttered environments, resulting in state spaces of up to twelve dimensions.

1 Introduction

There is a strong need for a general-purpose, 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 dynamical models and avoiding obstacles in the robot's environment. In other words, a fundamental task is to design a feasible open-loop trajectory that satisfies both global obstacle constraints and local differential constraints. We use the word kinodynamic planning, introduced in [20], to refer to such problems[1]. These solutions would be valuable in a wide variety of applications. In robotics, a nominal trajectory can be designed for systems such as mobile robots, manipulators, space robots, underwater robots, helicopters, and humanoids. This trajectory can be used to evaluate a robot design in simulation, or as a reference trajectory for designing a feedback control law. In virtual prototyping, engineers can use these trajectories to evaluate the design of many mechanical systems, possibly avoiding the time and expense of building a physical prototype. For example, in the automotive industry, the planning technique can serve as a "virtual stunt driver" that determines whether a proposed vehicle can make fast lane changes or can enter a dangerous state such as toppling sideways. In the movie and game industries, advanced animations can be constructed that automate the motions of virtual characters and devices, while providing realism that obeys physical laws. In general, such trajectories may be useful in any application area which can be described using control theoretic models, from analog circuits to economic systems.

The classic approach in robotics research has been to decouple the general robotics problem by solving basic path planning, and then find a trajectory and controller that satisfies the dynamics and tracks the path [10, 39, 64]. The vast majority of basic path planning algorithms consider only *kinematics*, while ignoring the system *dynamics* entirely. In this paper, we consider kinodynamic planning as a generalization of holonomic and nonholonomic planning in configuration spaces, by replacing popular configuration-space notions [47] by their *state space* (or phase space) counterparts. A point in the state space may include 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

[1] In nonlinear control literature, kinodynamic planning for underactuated systems is encompassed by the definition of nonholonomic planning. Using control-theoretic terminology, we characterize our work as open-loop trajectory design for nonlinear systems with drift and nonconvex state constraints. Other terms include *trajectory planning* or *trajectory design*.

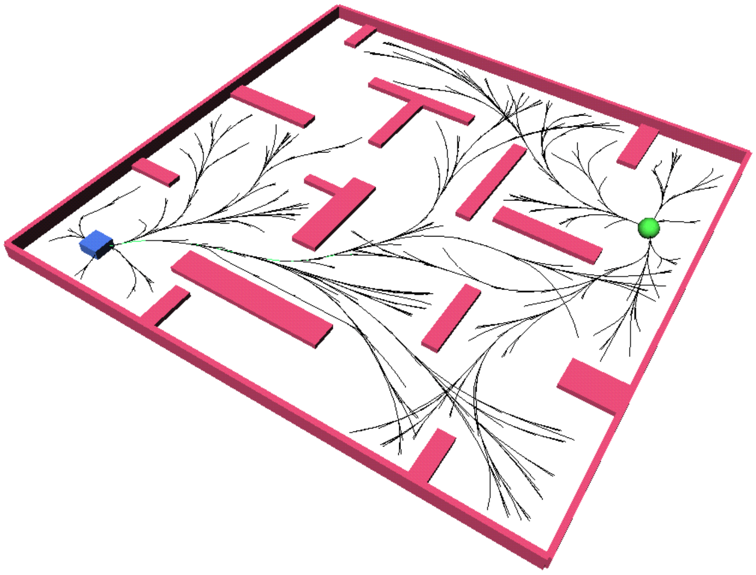

Figure 1: We consider planning problems with dynamic constraints induced by physical laws. The above image shows the state exploration trees computed for a rigid rectangular object (left). The goal location is represented by a sphere (upper right).

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. One of the earliest algorithms is presented in [61], in which minimum-time trajectories were designed by tesellating the joint space of a manipulator, and performing dynamic programming-based search that uses time-scaling ideas to reduce the search dimension. Algebraic approaches solve for the trajectory exactly, though the only known solutions are for point masses with velocity and acceleration bounds in one dimension [55] and two dimensions [14]. Provably approximately-optimal kinodynamic trajectories are computed in [20] by dynamic programming-based search on the state space by systematically applying control inputs. Other papers have extended or modified this technique [19, 18, 28, 59]. A dynamic programming-based approach to kinodynamic planning for all-terrain vehicles was presented in [16]. In [24], an incremental, variational approach is presented to perform state-space search. An approach to kinodynamic planning based on Hamiltonian mechanics is presented in [17]. An efficient approach to kinodynamic planning was developed by adopting as sensor-based philosophy that maintains an emergency stopping path which accounts for robot inertia [65].

Although several kinodynamic planning approaches exist, they are either limited to low degree-of-freedom problems or particular systems that enable simplification. Randomized techniques have led to efficient, incomplete planners for basic, holonomic path planning; however, there appears to be no equivalent technique for the broader kinodynamic planning problem (or even nonholonomic planning in the configuration space). We present a 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. Our ideas build on a large foundation of related research, which is briefly presented in Section 2.

Section 3 defines the problem and indicates some of its difficulties. Our proposed planning approach is based on the concept of Rapidly-exploring Random Trees (RRTs) [44], and is presented in Section 4. To demonstrate the utility of our approach, a series of planning experiments for hovercrafts in $\Re^2$ and spacecrafts in $\Re^3$ is presented in Section 5. Furthermore, some theoretical analysis of the planner's performance is given in Section 6. Finally, some conclusions and directions for future research are given in Section 7.

2 Other Related Research

Dynamic programming 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 [8, 9, 38, 43]. Since control theorists have traditionally preferred feedback solutions, the representation often 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 often be reduced to the construction of a tree grown from an initial state [37]. This idea has been proposed in path planning literature for nonholonomic planning [6, 48] and kinodynamic planning [16, 20]. 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.

Steering methods The steering problem has received considerable attention in recent years. The task is to compute an open-loop trajectory that brings a nonholonomic system from an initial state to a goal state, without the presence of obstacles. Given the general difficulty of this problem, most methods apply to purely kinematic models (i.e., systems without drift or momentum). For a kinematic car that has limited turning radius and moves forward only, it was shown that shortest path between any two configurations belongs to one of a family of six kinds of curves comprised of straight lines and circular arcs [21]. For a car that can move forward or backwards, optimal solutions comprised of 48 curve types have been obtained [11, 58, 69]. For more complicated kinematic models, non-optimal steering techniques have been introduced, which includes for example a car pulling trailers [54], firetrucks [13]. Techniques also exist for general system classes, such as nilpotent [35], differentially flat [53, 25], and chained form [13, 54, 67]. For systems with drift and/or obstacles, the steering problem remains a formidable challenge.

Nonholonomic planning The nonholonomic planning problem was introduced in [40], and has blossomed into a rich area of research in recent years. Rather than surveying this large body of research, we refer the reader to recent, detailed surveys [22, 41]. Most current approaches to nonholonomic planning rely on the existence of steering methods that can be used in combination with holonomic motion planning techniques. Other approaches exploit particular properties or very special systems (esp. kinematic car models). For most nonholonomic systems, it remains a great challenge to design efficient path planning methods.

Lower bounds Kinodynamic planning in general is at least as hard as the *generalized mover's problem*, which has been proven to be PSPACE-hard [60]. 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 [20]. 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 holonomic path planning.

Randomized holonomic planning It would certainly be useful if ideas can be borrowed or adapted from existing randomized path planning techniques that have been successful for basic, holonomic path planning. For the purpose of discussion, we choose two different techniques that have been successful in recent years: randomized potential fields (e.g, [7, 15]) and probabilistic roadmaps (e.g., [1, 32]). 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 probabilistic roadmap approach, a graph is constructed in the configuration space by generating random configurations and attempting to connect pairs of nearby configurations with a local planner. Once the graph has been constructed, the planning problem becomes one of searching a graph for a path between two nodes. If an efficient steering method exists for a particular system, then it is sometimes possible to extend randomized holonomic planning techniques to the case of nonholonomic planning [70, 62, 63].

Drawing inspiration from previous work 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 [1, 32], 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 [41] 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. Furthermore, the probabilistic roadmap is designed for multiple queries. The underlying theme in that work is that it is worthwhile to perform substantial precomputation on a given environment, to enable numerous path planning queries to be solved efficiently.

In our approach, we are primarily interested in answering a single query efficiently, without any preprocessing of the environment. In this case, the exploration and search are combined in a single method, without substantial precomputation that is associated with a method such as the probabilistic roadmap. This idea is similar to classical AI search techniques, the Ariadne's clew algorithm for holonomic planning [50], and the related holonomic planners in [30, 72].

To directly handle differential constraints, 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. These concerns have motivated our development of RRTs [44] and the proposed planning algorithm.

Recent advances This paper is an expanded version of [45]. Since that time, several interesting developments have occurred. In [26], the ideas contained in this paper were applied to the problem of designing trajectories for a helicopter flying among polyhedral obstacles. Substantial performance benefits were obtained by using a metric based on the optimal cost-to-go for a hybrid nonlinear system that ignores obstacles. In [71], H^∞ techniques and curve fitting were applied to yield an efficient planner that builds on ideas from [45]. A randomized kinodynamic planning algorithm was proposed recently for the case of time-varying environments in [33]. A deterministic kinodynamic planning algorithm based on collocation was presented recently in [23].

3 Problem Formulation: Path Planning in the State Space

We formulate the kinodynamic planning problem as path planning in a state space that has first-order differential 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 (C-space) that arises from a rigid or articulated body that moves in a 2D or 3D world. Each configuration $q \in \mathcal{C}$ represents a transformation that is applied to a geometric model of the robot. Let $\mathcal{X}$ denote the state space, in which a state, $x \in \mathcal{X}$, is defined as $x = (q, \dot{q})$. The state could include higher-order derivatives if necessary, but such systems are not considered in this paper.

Differential constraints When planning in $\mathcal{C}$, differential or 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 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 $m < 2n$, in which n is the dimension of $\mathcal{C}$. It is well known that under appropriate conditions the Implicit Function Theorem allows the constraints to be expressed as

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

in which $u \in U$, and U represents a set of allowable controls or inputs. Equation 1 effectively yields a convenient parameterization of the allowable state transitions via the controls in U.

The proposed approach will require a numerical approximation to (1). Given the current state, $x(t)$, and inputs applied over a time interval, $\{u(t') \mid t \leq t' \leq t + \Delta t\}$, the task is to compute $x(t + \Delta t)$. This can be achieved using a variety of numerical integration techniques. For example, assuming constant input, u, over the interval $[t, t + \Delta t)$, a standard form of fourth-order Runge-Kutta integration yields

$$x' = f(x(t) + \frac{\Delta t}{2} f(x(t), u), u),$$

$$x'' = f(x(t) + \frac{\Delta t}{2} x', u),$$
$$x''' = f(x(t) + \frac{\Delta t}{2} x'', u),$$

and

$$x(t + \Delta t) \approx x(t) + \frac{\Delta t}{6}(f(x(t), u) + 2x' + 2x'' + x''').$$

For many applications, (1) might not be available. For example, motions might be generated from a complicated dynamical simulation package that considers collisions, flexible parts, vehicle dynamics, finite element analysis, etc. For these cases, our techniques can be directly applied without requiring (1). The only requirement is that an incremental simulation of the system can be generated for any current state and input.

Obstacles in the state space Assume that the environment contains static obstacles, and that a collision detection algorithm can determine efficiently whether a given configuration is in collision. It may even be assumed that an entire neighborhood around a configuration is collision free by employing distance computation algorithms [46, 51, 57]. There are interesting differences between finding collision-free paths in $\mathcal{C}$ versus in 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) [39]. 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 momentum, it cannot do anything to avoid collision. More precisely, a state lies in $\mathcal{X}_{ric}$ if there exist no inputs that can be applied over any time interval to avoid collision. Note that $\mathcal{X}_{obst} \subseteq X_{ric}$. Thus, it might be preferable to define $\mathcal{X}_{free} = \mathcal{X} \setminus \mathcal{X}_{ric}$, as opposed to $\mathcal{X} \setminus \mathcal{X}_{obst}$.

The region of inevitable collision, $\mathcal{X}_{ric}$, provides some intuition about the difficulty of kinodynamic planning over holonomic and purely-kinematic nonholonomic planning. Figure 2 illustrates conservative approximations of $\mathcal{X}_{ric}$ for a point mass robot that obeys Newtonian mechanics without gravity. 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. Ultimately, the topology of $\mathcal{X} \setminus \mathcal{X}_{ric}$ may be quite distinct from the topology $\mathcal{X} \setminus \mathcal{X}_{obst}$, which intuitively explains part of the challenge of the kinodynamic planning problem. Even though there might exist a kinematic collision-free path, a kinodynamic trajectory might not exist. For the remainder of the paper, we assume that $\mathcal{X}_{free} = \mathcal{X} \setminus \mathcal{X}_{obst}$ (one could alternatively define $\mathcal{X}_{free} = \mathcal{X} \setminus \mathcal{X}_{ric}$).

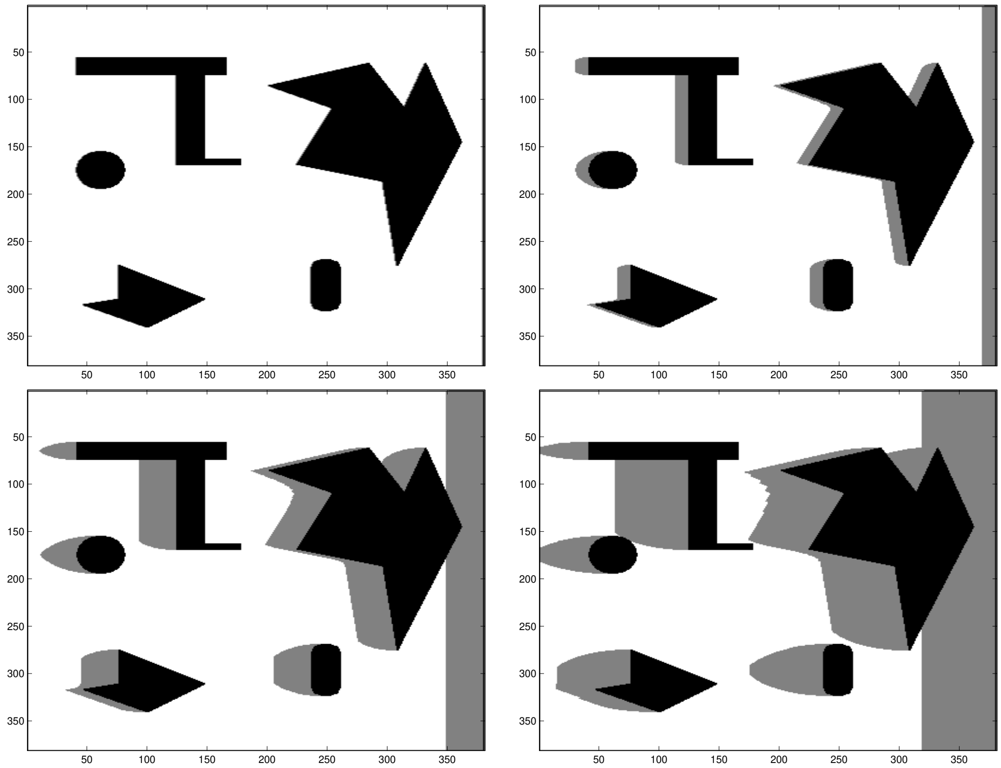

Figure 2: Slices of $\mathcal{X}$ for a point mass robot in 2D with increasingly higher initial speeds. White areas represent $\mathcal{X}_{free}$; black areas are $\mathcal{X}_{obst}$; gray areas approximate $\mathcal{X}_{ric}$.

A solution trajectory The kinodynamic planning problem is to find a trajectory from an initial state $x_{init} \in \mathcal{X}$ to a goal state $x_{goal} \in \mathcal{X}$ or goal region $\mathcal{X}_{goal} \subset \mathcal{X}$. A trajectory is defined as a time-parameterized continuous path $\tau : [0, T] \rightarrow \mathcal{X}_{free}$ that satisfies the nonholonomic constraints. Recall from (1) that the change in state is expressed in terms of an input, u. A more convenient way to formulate the problem is to find an input function $u : [0, T] \rightarrow U$ which results in a collision-free trajectory that starts at x_{init}, and ends at x_{goal} or $\mathcal{X}_{goal}$. The trajectory, $x(t)$ for $t \in [0, T]$, is determined through the integration of (1). It might also be appropriate to select a path that optimizes some criterion, such as the time to reach x_{goal}. It is assumed that optimal solutions are not required; however, here we assume that there is some loose preference for better solutions. A similar situation exists in the vast majority of holonomic planning methods.

Why does the problem present unique challenges? The difference between $\mathcal{X}$ and $\mathcal{C}$ is usually a factor of two in dimension. The curse of dimensionality has already contributed to the success and popularity of randomized planning methods for 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 obtaining optimal solutions with guaranteed deterministic convergence. The infeasibility of this requirement for generic high-dimensional systems has led many researchers to adopt a decoupled approach in which classical motion planning is first performed, and trajectory design is optimized around a particular motion planning solution.

Another reason why randomized kinodynamic planning approaches have not appeared is that kinodynamic planning is considerably harder due to momentum. Consider adapting randomized holonomic planning 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 $\mathcal{X}_{ric}$. The problem of designing a good heuristic function becomes extremely complicated for the case of kinodynamic planning.

The probabilistic 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 [70]. One result that greatly facilitated this extension of the technique to nonholonomic planning was the existence of Reeds-Shepp curves [58] 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 [63]. 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 probabilistic 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 efficient steering.

4 A Planner Based on Rapidly-Exploring Random Trees

The unique difficulties with kinodynamic planning motivated us to design a randomized planning technique particularly suited for kinodynamic planning (it also applies to the simpler problems of nonholonomic planning in $\mathcal{C}$ and basic path planning in $\mathcal{C}$ [34]). Our intention has been to develop a method that easily "drives forward" like potential field methods or the Ariadne's clew algorithm, and also quickly and uniformly explores the space like probabilistic roadmap methods.

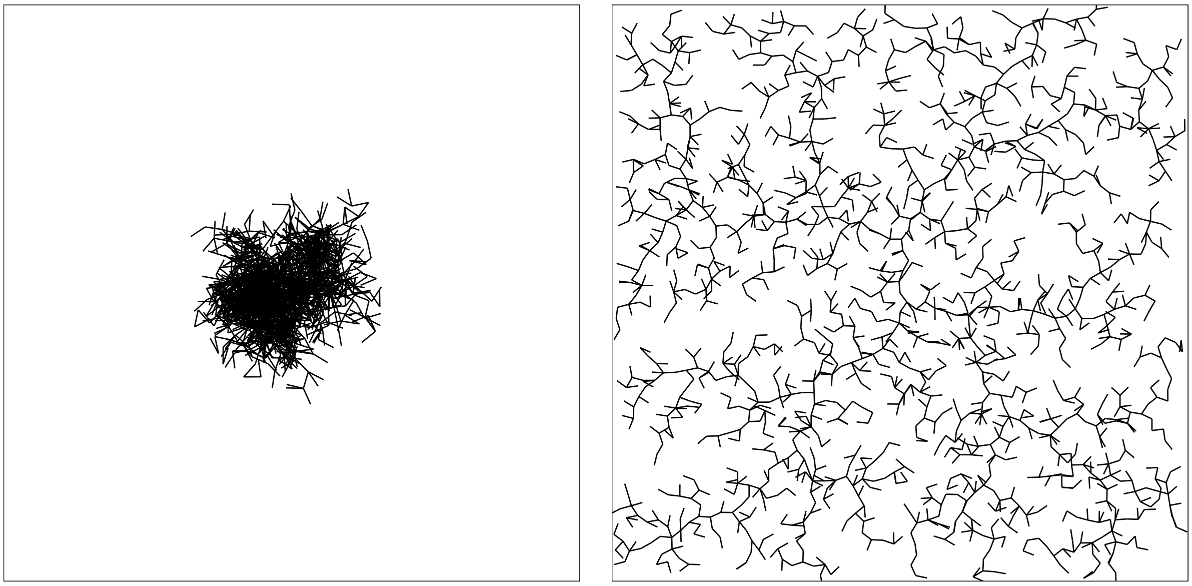

Figure 3: A Naive Random Tree vs. a Rapidly-exploring Random Tree. Each tree has 2000 vertices.

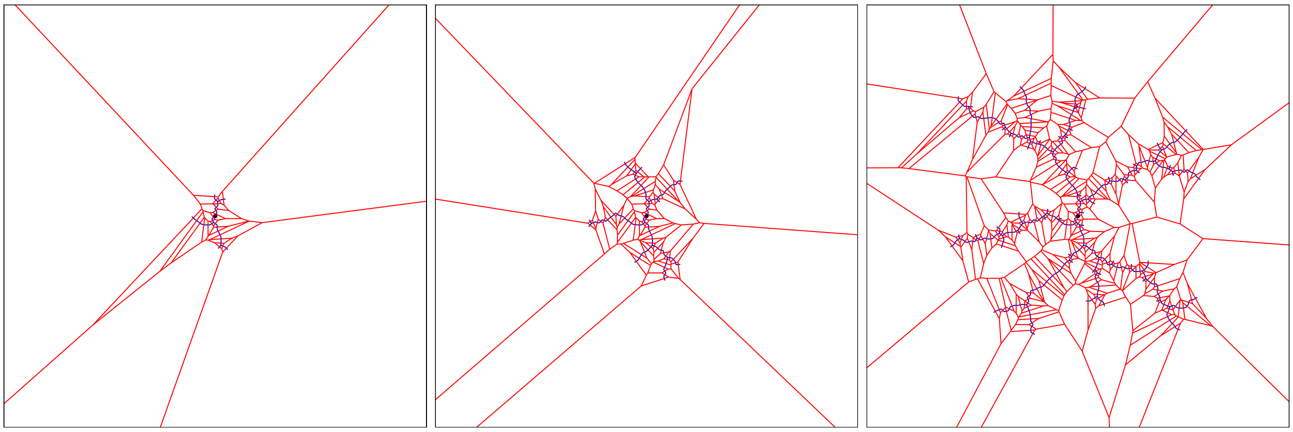

Figure 4: The RRT contains a Voronoi bias which causes rapid exploration.

```
BUILD_RRT(x_init)
T.init(x_init);
for k = 1 to K do
    x_rand ← RANDOM_STATE();
    EXTEND(T, x_rand);
Return T
```

```
EXTEND(T, x)
x_near ← NEAREST_NEIGHBOR(x, T);
if NEW_STATE(x, x_near, x_new, u_new) then
    T.add_vertex(x_new);
    T.add_edge(x_near, x_new, u_new);
    if x_new = x then
        Return Reached;
    else
        Return Advanced;
Return Trapped;
```

Figure 5: The basic RRT construction algorithm.

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 vertex 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 explores the state space. An argument for this can be made by considering the Voronoi regions of the vertices; see Figure 4. 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. It is shown in [34] that for holonomic planning, the distribution of RRT vertices converges in probability to distribution that is used for sampling, even in nonconvex spaces (regardless of the initial state).

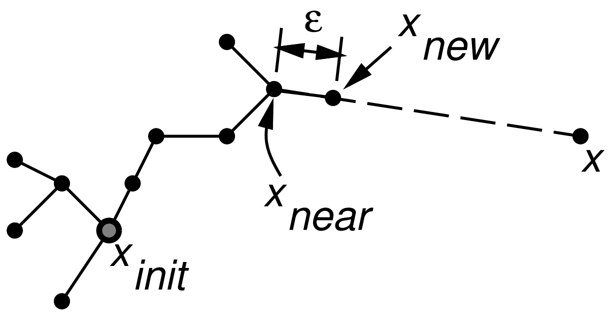

Figure 6: The EXTEND operation.

The algorithm that constructs an RRT is shown in Figure 5. A simple iteration is 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 6, 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. In general, Δt may be much larger than the time increment that is used for numerical integration. Δt can be fixed, or selected randomly at each iteration from a range of values $(0, \Delta_{max}]$. The input, u, can be chosen at random, or be selected by trying all possible inputs and choosing the one that yields a new state as close as possible to the sample, x (if U is infinite, then a discrete approximation or analytical technique can be used). 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 [27, 46, 51] 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 $\mathcal{X}_{free}$.

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 [2, 31] can be employed to help improve performance. The second complication can make it harder to wander through narrow passages, much like in the case of

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

Figure 7: A bidirectional RRT-based planner.

probabilistic roadmaps [29]. 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 is 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. Further discussion of the metric issue appears in Section 7.

A Bidirectional Planning Algorithm The basic RRT algorithm shown in Figure 5 may be used to explore the state space, but it is not designed to directly answer a path planning query. For the latter task, we borrow classical bidirectional search ideas [56] to grow two RRTs, one rooted at the initial 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 metric ρ and small $\epsilon > 0$. Our basic algorithm stops at the first solution trajectory 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).

Figure 7 shows the RRT_BIDIRECTIONAL algorithm, which may be compared to the BUILD_RRT algorithm of Figure 5. 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 the other tree to the new vertex. Then, the roles are reversed by swapping the two trees. The current algorithm is a minor variation of the algorithm presented in [45]. Previously, 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.

One drawback of using a bidirectional approach is that a discontinuity in the trajectory will generally exist at the place in which the two trees connect. A number of techniques can be employed to make the trajectory continuous. Classical shooting techniques can be applied to either half of the trajectory. It might be possible to slightly perturb the starting point of the second half of the trajectory to force it to begin at the end of the first half of the trajectory. In this case, the second half of the trajectory would have to be reintegrated and tested for collision. Finally, it is possible for some systems to apply a steering method to connect the two trajectories.

If no techniques effectively remove the discontinuity, then one can use a single RRT approach to bring the system from x_{init} to a goal region $\mathcal{X}_{goal}$. Instead of sampling randomly from $\mathcal{X}$, samples can be biased toward $\mathcal{X}_{goal}$. For example, with probability p, a sample can be selected from $\mathcal{X}_{goal}$; otherwise, it is chosen random from $\mathcal{X}$. We have performed successful experiments with single-RRT planners and several different sampling techniques. It is possible to obtain reasonable performance for numerous problems; however, the bidirectional algorithm is far superior when it can be applied. In practice, we have not experienced any difficulty applying the bidirectional approach.

5 Experiments with Hovercrafts and Spacecrafts

Several kinodynamic planning experiments were conducted on challenging problems. The algorithm was implemented in C++ on an 800MHz Intel Pentium III PC with 256MB of memory running Linux. The systems considered involve both non-rotating and rotating rigid objects with velocity and acceleration bounds obeying L_2 norms.

Dynamic Model: All experiments used a dynamic model in a non-gravity environment derived from the Newtonian mechanics of a 3D rigid body[4]. For some examples, the allowable controls restrict the reachable state space to a subspace of lower dimension than the full 12 dimensions, but a general model was adopted for simplicity in testing and comparing a variety of vehicle models. We consider a rigid body $\mathcal{B}$ of mass M

and body inertia tensor I, and define the following quantities:

$$\begin{aligned}
\mathbf{p} &= [p_x \; p_y \; p_z]^T && \text{global position of the center of mass} \\
\mathbf{q} &= [q_\theta \; q_x \; q_y \; q_z]^T && \text{unit quaternion representing the rotation in } SO(3) \\
\mathbf{v} &= [v_x \; v_y \; v_z]^T && \text{linear velocity (i.e. } \mathbf{v} = \dot{\mathbf{p}}) \\
\omega &= [\omega_x \; \omega_y \; \omega_z]^T && \text{angular velocity}
\end{aligned}$$

The full state vector of $\mathcal{B}$ is given by:

$$\mathbf{x}(t) = \begin{pmatrix} \mathbf{p}(t) \\ \mathbf{q}(t) \\ \mathbf{v}(t) \\ \omega(t) \end{pmatrix}$$

The state vector consists of 13 real numbers, but the state space has 12 dimensions due to the constraint that the quaternion must be of unit norm $||\mathbf{q}||^2 = 1$. Each control $\mathbf{u} \in \mathcal{U}$ defines a force-torque pair $(\mathbf{F}, \tau)$ acting on the center of mass of $\mathcal{B}$. The equations of motion for the system can be defined as:

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t)) = \begin{pmatrix} \dot{\mathbf{p}}(t) \\ \dot{\mathbf{q}}(t) \\ \dot{\mathbf{v}}(t) \\ \dot{\omega}(t) \end{pmatrix} = \begin{pmatrix} \mathbf{v}(t) \\ \frac{1}{2}\,\hat{\omega}(t) \cdot \mathbf{q}(t) \\ \frac{\mathbf{F}}{M} \\ R(t) I^{-1} R(t)^T \tau \end{pmatrix} \tag{2}$$

where $\hat{\omega}(t) \cdot \mathbf{q}(t)$ represents the quaternion product between $[\,0 \;\; \omega_x(t) \; \omega_y(t) \; \omega_z(t)]^T$ and $\mathbf{q}(t)$. The rotation matrix $R(t)$ and its transpose $R(t)^T$ are computed by converting the quaternion $\mathbf{q}(t)$ to its equivalent matrix representation. Details on this conversion, and sample C code for a similar model is given in [4]. Useful references on the use of quaternions to represent orientation include [66] and [49].

Distance Metric: All experiments utilized a simple metric on $\mathcal{X}$ based on a weighted Euclidean distance for position, linear velocity, and angular velocity, along with a weighted metric on unit quaternions for orientation distance. The function $\rho(\mathbf{x}_1, \mathbf{x}_2)$ attempts to heuristically encode a measure the relative "closeness" between the pair of states $\mathbf{x}_1$ and $\mathbf{x}_2$ with a positive scalar function:

$$\rho(\mathbf{x}_1, \mathbf{x}_2) = w_p||\mathbf{p}_1 - \mathbf{p}_2||^2 + w_q(1 - |\mathbf{q}_1 \cdot \mathbf{q}_2|)^2 + w_v||\mathbf{v}_1 - \mathbf{v}_2||^2 + w_\omega||\omega_1 - \omega_2||^2$$

where w_p, w_p, w_p, and w_p, are weights for position, orientation, linear velocity, and angular velocity respectively. In our implementation, the weights are computed such that all component distances are normalized on the range $[0, 1]$. The quaternion scalar product $|\mathbf{q}_1 \cdot \mathbf{q}_2|$ represents the cosine of the angle formed between two unit quaternion vectors, yielding a convenient measure of closeness in orientation.

Example System	# Triangles		# Ctrl	Δt	# Trials	Computation Time (sec)		
	robot	obst.				min	max	avg
Planar Translating Body (4D)	12	228	4	0.25	100	2.39	12.15	4.59
Planar Body with Rotation (6D)	264	228	3	0.25	10	87.65	670.06	321.66
Translating 3D Body (6D)	12	300	6	0.30	50	19.32	220.78	58.12
3D Satellite (12D)	64	1921	8	0.30	10	154.26	852.02	352.51
3D Spacecraft (12D)	1289	1769	5	0.30	10	292.03	1703.94	628.07

Table 1: Performance statistics for various examples.

As indicated previously, the ideal metric is the optimal cost to go from one state to another, but its computation is as hard as the original planning problem. Additional experimentation is needed in order to evaluate the efficacy of the many different metric functions possible for different systems (See Section 7).

Applying Controls: Each example used a fixed set of controls $\mathcal{U}$. Applying no control and simply allowing the system to drift is also counted as an additional control. We used a fixed Δt, and applied each control constantly over this time interval.

Since our dynamic model does not include contacts or collisions, the equations of motion are non-stiff, and we are able to use a simple fixed-step Euler method for numerical integration. This worked well for both forwards and backwards integration using a negative time step. For systems with stiff equations, higher-order methods or implicit integration methods should be used. The magnitude of the integrator time step used for all examples was $dt = 0.01$ seconds. Note that this time step is independent of the RRT time step Δt used for applying a control, which can be much larger.

Example Systems: For each of the following, we describe the set of controls, the dimension of the reachable state space, and details of the computations performed on trial environments. A summary of the results is listed in Table 1. Standard metric units were used (i.e. lengths in meters $[m]$, forces in newtons $[N]$). The workspace bounds were $[0, 10m]$ for each axis. The linear and angular velocity bounds were $||\mathbf{v}||^2 < 2.0[m/s]$, and $||\omega||^2 < 1.5[rad/s]$ respectively.

1) Planar Translating Body (dim $\mathcal{X} = 4$) The first experiment considered a rigid rectangular object of dimensions $[0.4m, 0.2m, 0.4m]$ with a set of translational controls that restrict its motion to the xz 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 (there were no torque components to the controls).

$$\mathcal{U}_F = \left(\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix} \right)$$

Figure 8 shows snapshots during various stages of the computation. Anywhere between 400 and 2500 nodes are explored on average before a solution trajectory is found, with total computation time of approximately 5 seconds on average (see Table 1). The tolerances used for state connection were ($\epsilon_p = 0.05, \epsilon_v = 0.1$).

2) Planar Body with Rotation (dim $\mathcal{X}$ = 6) We extend the previous experiment to consider systems with rotation. First, we consider the case of a rigid spacecraft object of approximate dimensions $[0.73m, 0.13m, 0.80m]$ with thrusters enabling it to rotate in place, but only translate in the forward direction. This model was inspired by the popular video game "Asteroids". As in the previous example, the spacecraft motion is restricted to the xz plane. 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:

$$\mathcal{U}_F = \left(\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \right) \qquad \mathcal{U}_\tau = \left(\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ -0.01 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0.01 \\ 0 \end{bmatrix} \right)$$

Figure 9 shows the explored states after 13,600 nodes. The average total computation time for this example was approximately 5 minutes. The tolerances used for state connection were ($\epsilon_p = 0.075, \epsilon_q = 0.08, \epsilon_v = 0.1, \epsilon_\omega = 0.1$).

3) Translating 3D Body (dim $\mathcal{X}$ = 6) We consider the case of a free-floating rigid object, such as an unanchored satellite in space at a fixed orientation. The object is assumed to be equipped with thruster controls to be used for translating in a non-gravity environment. The satellite has rectangular dimensions $[0.4m, 0.2m, 0.3m]$ with three opposing pairs of thrusters along each of its principal axes forming a set of six controls spanning a 6-dimensional state space (there are no torque components to the controls):

$$\mathcal{U}_F = \left(\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix} \right)$$

The task is to thrust through a sequence of two narrow passages amidst a collection of obstacles. Figure 10 shows the RRTs generated during the planning process, and Figure 11 shows one candidate solution found after a total of 16,300 nodes were explored. The average total computation time for this case was approximately 1 minute. The tolerances used for state connection were ($\epsilon_p = 0.1, \epsilon_v = 0.1$).

4) 3D Body with Rotation (dim $\mathcal{X} = 12$) Finally, we show two results for underactuated rigid bodies in a 3D world. These examples lead to a 12D state space.

The first result is 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

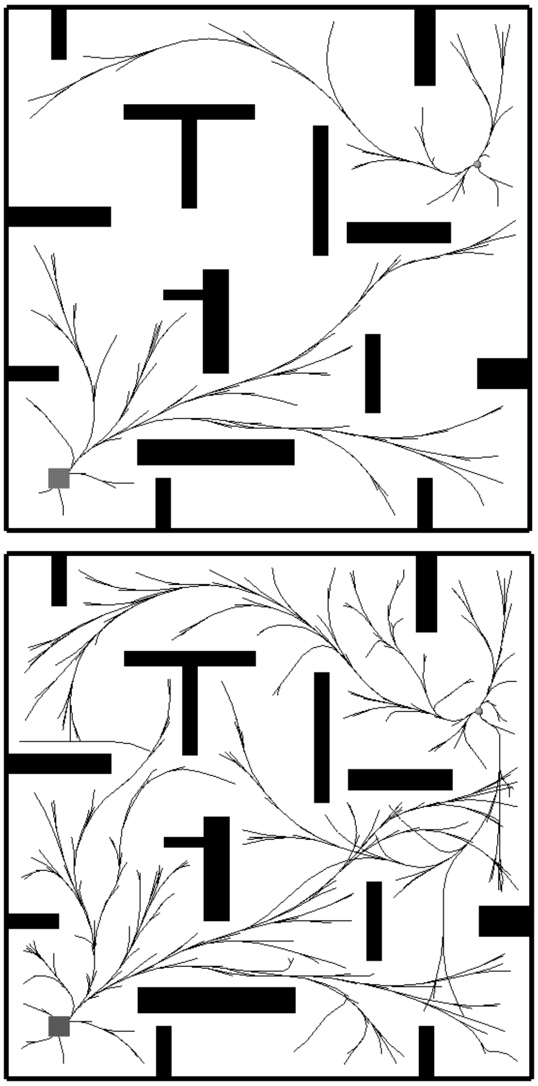

Figure 8: Various stages of state exploration during planning. The top two images show the RRTs after 500 and 1000 nodes, respectively. The bottom two images show the final trees and the computed solution trajectory after 1582 nodes.

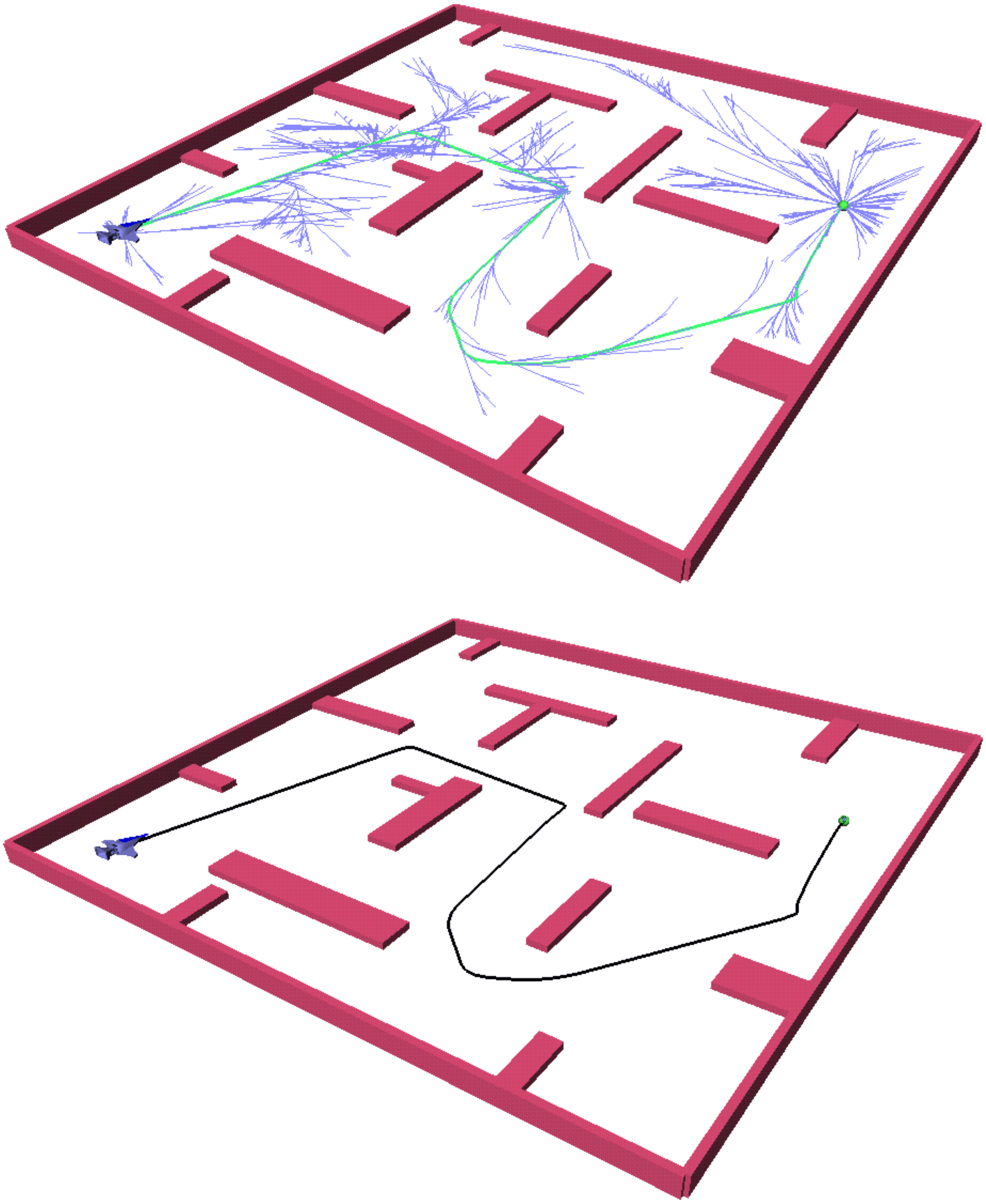

Figure 9: RRTs of 13,600 nodes and solution trajectory for the planar body with unilateral thrusters that allow it to rotate freely but translate only in the forward direction.

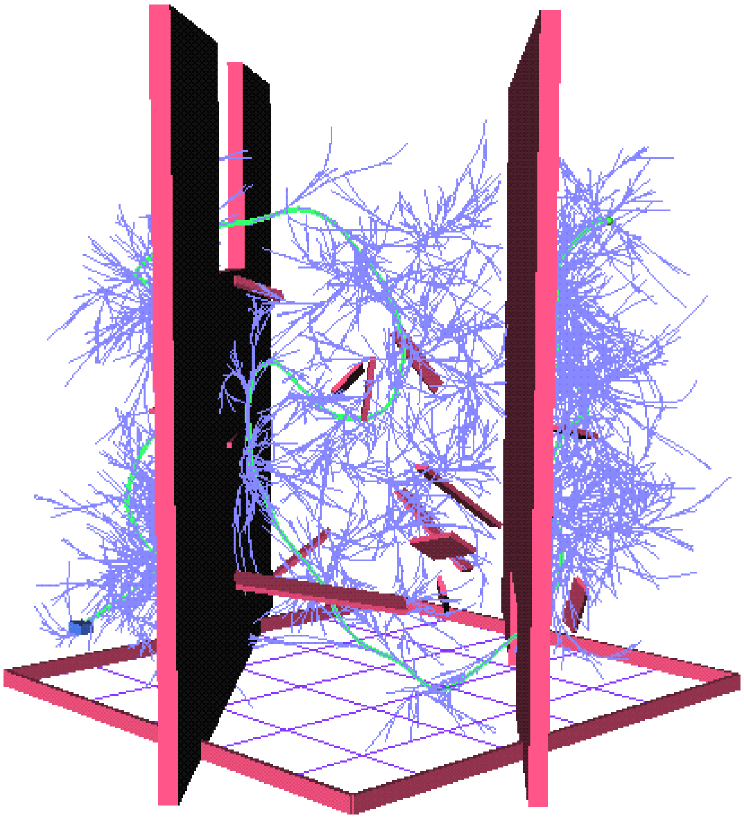

Figure 10: The RRTs computed for the task of navigating a sequence of narrow passages for the 3D translation case.

controls that allow it to translate along the primary axis of the cylinder. This model has a 12-dimensional state space.

$$
\begin{aligned}
\mathcal{U}_F &= \left(\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix} \right) \\
\mathcal{U}_\tau &= \left(\begin{bmatrix} 0.01 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} -0.01 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0.01 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ -0.01 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0.01 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ -0.01 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \right)
\end{aligned}
$$

The task of the satellite, modeled as a rigid cylindrical object of radius $0.2m$ and height $0.6m$, is to perform a collision-free docking maneuver into the cargo bay of the space shuttle model amidst a cloud of obstacles. Figure 12 illustrates a typical example of the trajectories explored during the planning process, and Figure 13 shows a candidate solution found after 23,800 states were explored. The tolerances used for state connection were $(\epsilon_p = 0.15, \epsilon_q = 0.1, \epsilon_v = 0.3, \epsilon_\omega = 0.5)$. The average total computation time was approximately 6 minutes.

The second result, given in Figure 14, shows a fictitious, underactuated spacecraft of approximate di-

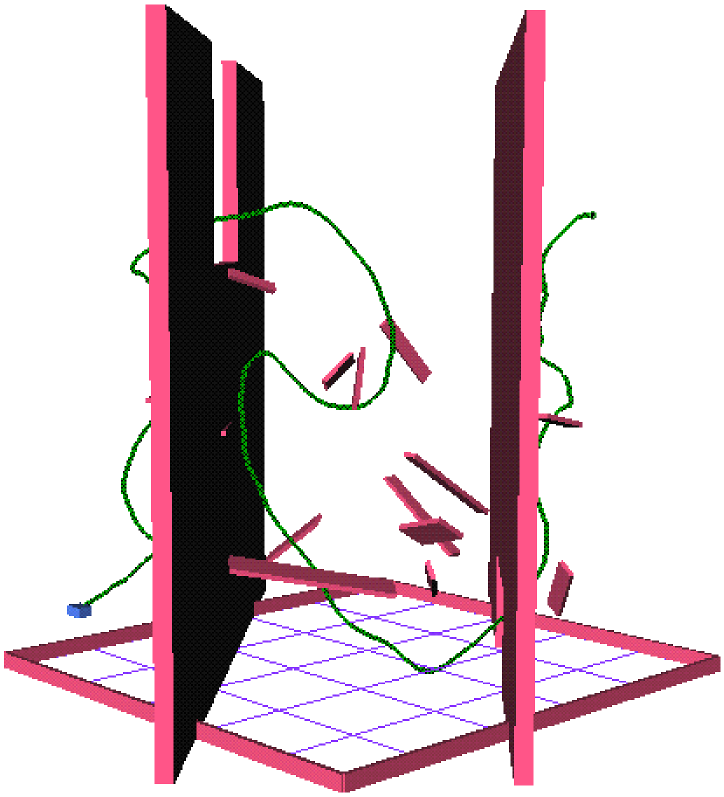

Figure 11: Solution trajectory for navigating through a sequence of narrow passages for the 3D translation case. The initial state is at the lower left; the goal is at the upper right.

mensions $[0.65m, 0.17m, 0.77m]$ that must maneuver through two narrow gates and enter a hangar that has a small entrance. There are five inputs, each of which applies a thrust impulse. The possible motions are: 1) forward, 2) up, 3) down, 4) clockwise roll, 5) counterclockwise roll:

$$\begin{aligned}
\mathcal{U}_F &= \left(\begin{bmatrix} 0 \\ 0 \\ 0.5 \end{bmatrix}, \begin{bmatrix} 0 \\ 0.25 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ -0.25 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \right) \\
\mathcal{U}_\tau &= \left(\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0.01 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ -0.01 \end{bmatrix} \right)
\end{aligned}$$

Planning is performed directly in the 12-dimensional state space. The tolerances used for state connection were ($\epsilon_p = 0.05, \epsilon_q = 0.02, \epsilon_v = 0.3, \epsilon_\omega = 0.5$). The average total computation time was approximately 11 minutes.

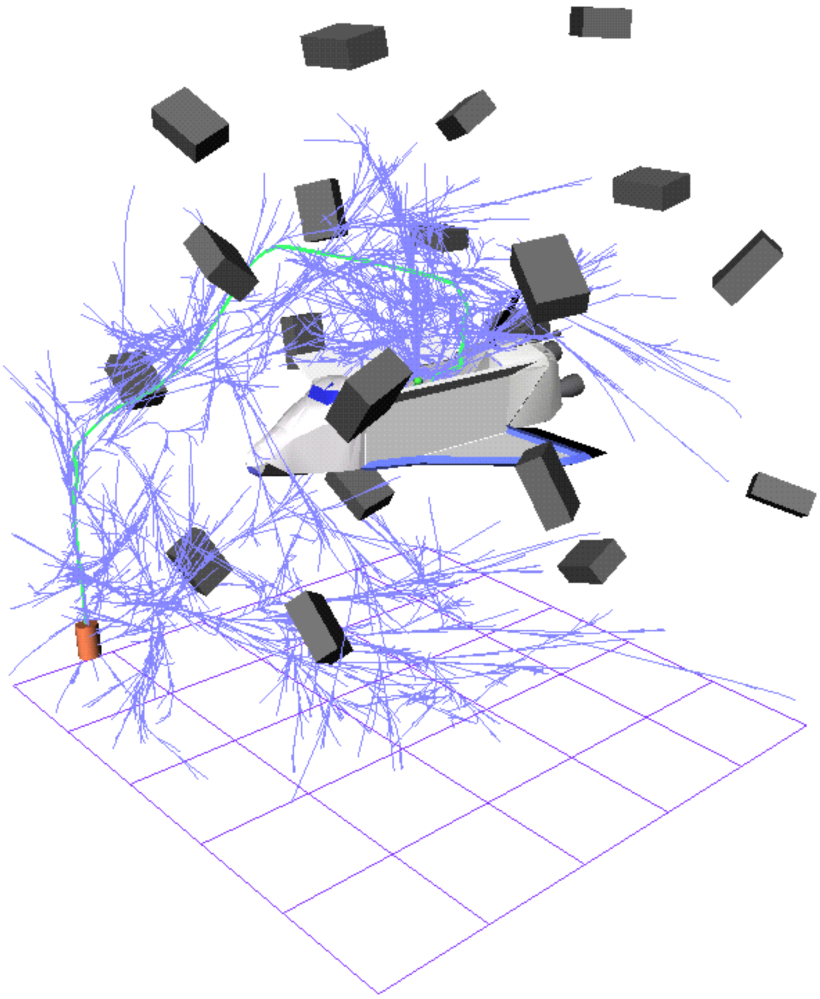

Figure 12: The RRTs constructed during planning for the fully-orientable satellite model with limited translation. A total of 23,800 states were explored before a successful candidate solution trajectory was found.

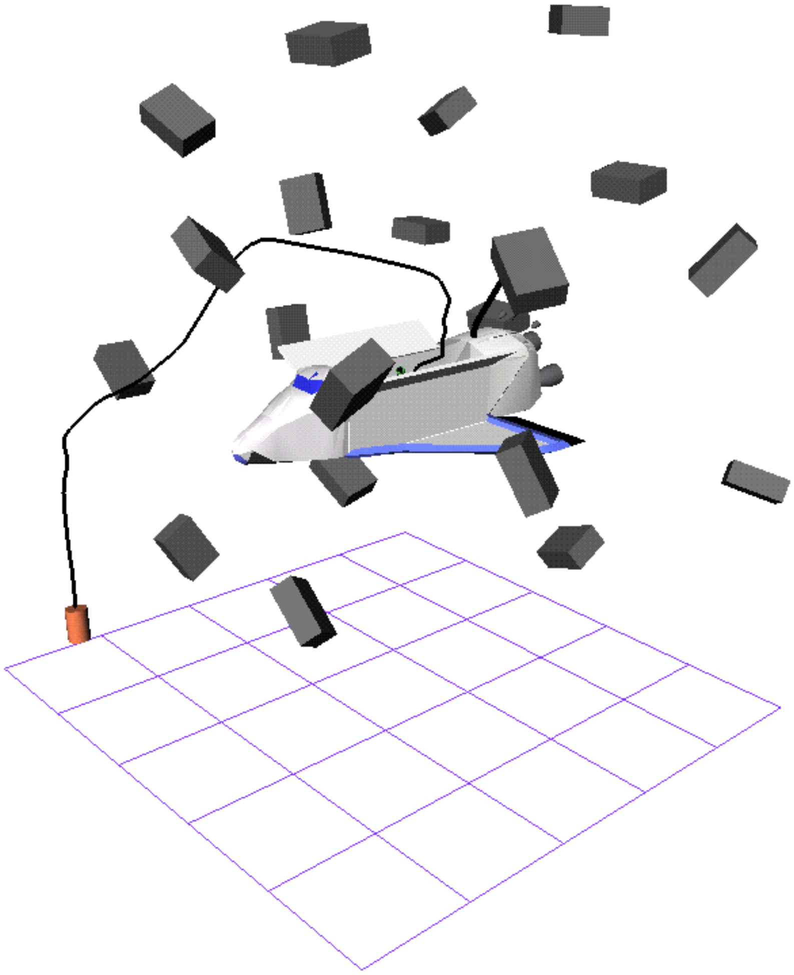

Figure 13: The docking maneuver computed for the fully-orientable satellite model. The satellite's initial state is in the lower left corner, and the goal state is in the interior of the cargo bay of the shuttle.

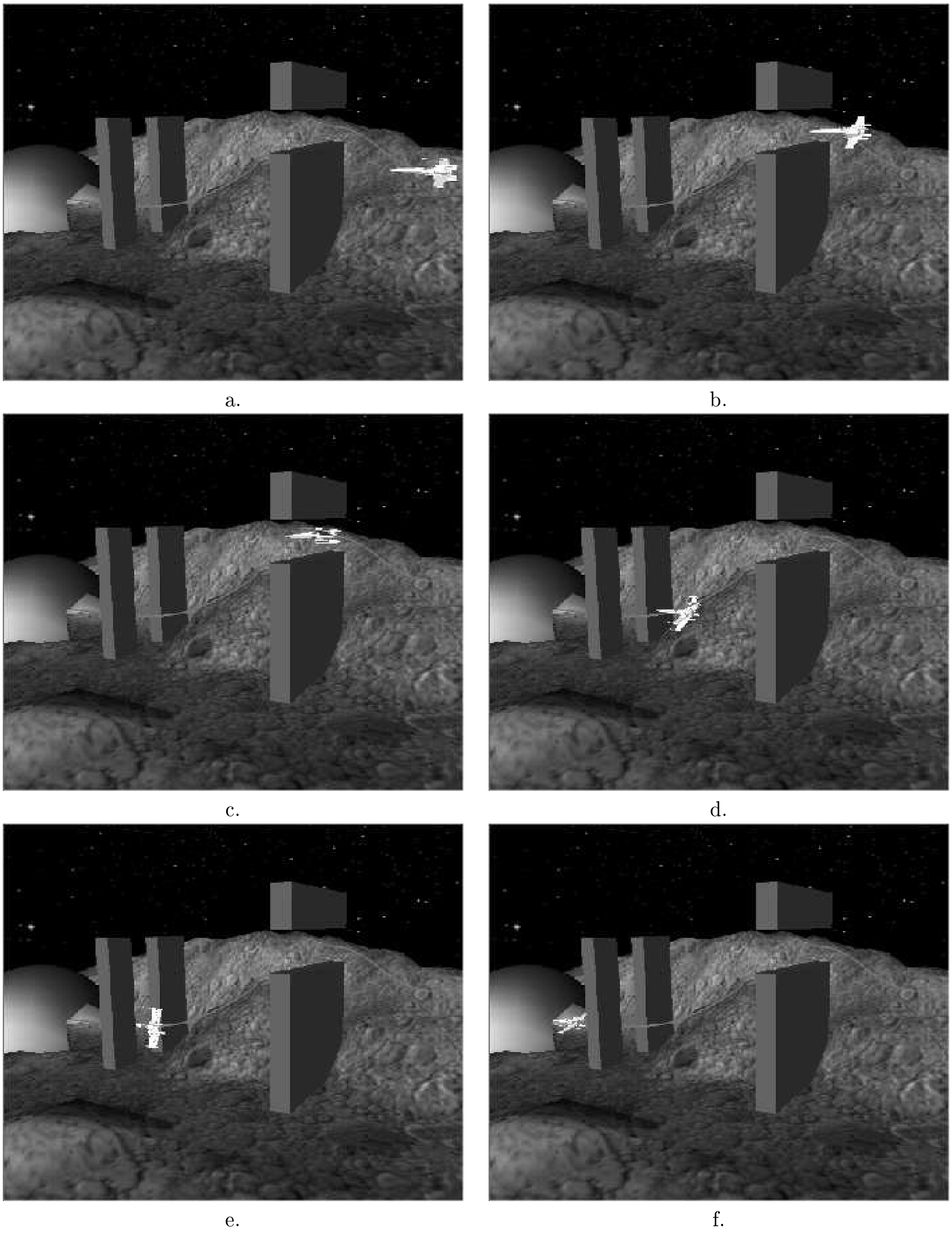

Figure 14: An underactuated spacecraft that performs complicated maneuvers. The state space has twelve dimensions, and there are five inputs.

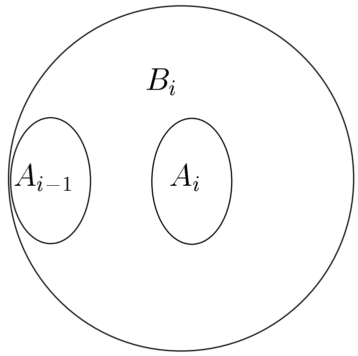

Figure 15: The attraction sequence has two properties: 1) in terms of the metric, ρ, any point in A_{i-1} is closer to any point in A_i than any point outside of B_i; 2) any point within B_i can be attracted to A_i.

6 Analysis

In this section, the theoretical behavior of the planning method is characterized. Theorems 1 and 2 express the rate of converge of the planner, and Theorem 3 establishes that the planner is probabilistically complete (i.e., the probability that a solution is found tends to one as the number of iterations tends to infinity). These results represent a significant first step towards gaining a complete understanding of behavior of the planning algorithm; however, the convergence rate is unfortunately expressed in terms of parameters that cannot be easily measured. It remains an open problem to characterize the convergence rate in terms of simple parameters that can be computed for a particular problem. This general difficulty even exists in the analysis of randomized path planners for the holonomic path planning problem [30, 36].

For simplicity, assume that the planner consists of a single RRT. The bidirectional planner is only slightly better in terms of our analysis, and a single RRT is easier to analyze. Furthermore, assume that the step size is large enough so that the planner always attempts to connect x_{near} to x_{rand}.

Let $\mathcal{A} = \{A_0, A_1, \ldots, A_k\}$ be a sequence of subsets of $\mathcal{X}$, referred to as an *attraction sequence*. Let $A_0 = \{x_{init}\}$. The remaining sets must be chosen with the following rules. For each A_i in $\mathcal{A}$, there exists a *basin*, $B_i \subseteq \mathcal{X}$ such that the following hold:

1. For all $x \in A_{i-1}$, $y \in A_i$, and $z \in \mathcal{X} \setminus B_i$, the metric, ρ, yields $\rho(x, y) < \rho(x, z)$.

2. For all $x \in B_i$, there exists an m such that the sequence of inputs $\{u_1, u_2, \ldots, u_m\}$ selected by the EXTEND algorithm will bring the state into $A_i \subseteq B_i$.

Finally, it is assumed that $A_k = \mathcal{X}_{goal}$.

Each basin B_i can intuitively be considered as both a safety zone that ensures an element of B_i will be selected by the nearest neighbor query, and a potential well that attracts the state into A_i. An attraction sequence should be chosen with each A_i as large as possible and with k as small as possible. If the space contains narrow corridors, then the attraction sequence will be longer and each A_i will be smaller. Our analysis indicates that the planning performance will significantly degrade in this case, which is consistent with analysis results obtained for randomized holonomic planners [29]. Note that for kinodynamic planning, the choice of metric, ρ, can also greatly affect the attraction sequence, and ultimately the performance of the algorithm.

Let p be defined as

$$p = \min_i \{\mu(A_i)/\mu(\mathcal{X}_{free})\},$$

which corresponds to a lower bound on the probability that a random state will lie in a particular A_i.

The following theorem characterized the expected number of iterations.

Theorem 1 *If a connection sequence of length k exists, then the expected number of iterations required to connect x_{init} to X_{goal} is no more than k/p.*

Proof: If an RRT vertex lies in A_{i-1}, and a random sample, x, falls in A_i, then the RRT will be connected to x. This is true because using the first property in the definition of a basin, it follows that one of the vertices in B_i must be selected for extension. Using the second property of the basin, inputs will be chosen that ultimately generate a vertex in A_i.

In each iteration, the probability that the random sample lies in A_i is at least p; hence, if A_{i-1} contains an RRT vertex, then A_i will contain a vertex with probability at least p. In the worst-case, the iterations can be considered as Bernoulli trials in which p is the probability of a successful outcome. A path planning problem is solved after k successful outcomes are obtained because each success extends the progress of the RRT from A_{i-1} to A_i.

Let $C_1, C_2, \ldots, C_n$ be i.i.d. random variables whose common distribution is the Bernoulli distribution with parameter p. The random variable $C = C_1 + C_2 + \cdots + C_n$ denotes the number of successes after n iterations. Since each C_i has the Bernoulli distribution, C will have a binomial distribution,

$$\binom{n}{k} h^k (1-h)^{n-k},$$

in which k is the number of successes. The expectation of the binomial distribution is n/p, which also represents an upper bound on the expected probability of successfully finding a path. $\triangle$

The following theorem establishes that the probability of failure decreases exponentially with the number of iterations.

Theorem 2 *If an attraction sequence of length k exists, for a constant $\delta \in (0, 1]$, the probability that the RRT fails to finds a path after n iterations is at most $e^{\frac{-1}{2}(np-2k)}$.*

Proof: The random variable C from the proof of Theorem 1 has a binomial distribution, which enables the application of a Chernoff-type bound on its tail probabilities. A theorem from [52] is directly applied to establish the theorem. If C is binomially distributed, $\delta \in (0, 1]$, and $\mu = E[C]$, then $P[C \leq (1-\delta)\mu] < exp(\mu\delta^2/2)$, in which $\delta = 1 - k/(np)$. The expression in the exponent can be simplified to $-\frac{1}{2}np + k - \frac{k^2}{2np}$. Note that $e^{\frac{-k^2}{2np}} \leq 1$. This implies that $exp(\mu\delta^2/2) \leq e^{\frac{-1}{2}(np-2k)}$. $\triangle$

We now consider probabilistic completeness. 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 $\mathcal{X}_{free}$.

The following establishes the probabilistic completeness of the nonholonomic planner.

Theorem 3 *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.*

Proof: 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 $\mu(Vor(x_i))/\mu(\mathcal{X}_{free})$ of being selected. There exists a second positive real number, c_2 (which depends on c_1), such that the probability 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. $\triangle$

7 Conclusions

We have presented the first randomized approach to kinodynamic planning [45]. We believe that this approach and other randomized kinodynamic planning techniques will prove useful in a wide array of applications that includes robotics, virtual prototyping, and computer graphics. Recently, our planner has been applied to automating the flight of helicopters in complicated 3D simulations that contain obstacles

[26]. 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 problems of up to 12 degrees-of-freedom. 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 $\mathcal{C}$). Some analysis of the planning algorithm has also been given; however, it remains an open problem to obtain convergence results expressed in terms of parameters that can be computed for a given example.

Several issues and topics for future research are mentioned below.

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 5 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 approach various random samples, 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 pseudo-metric 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 [41]), the value function [68], and the cost-to-go function [3, 42]. 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. For a particular system, it may be possible to derive ρ from several alternatives, including a Lyapunov function, a steering method, a fitted spline curve, or an optimal control law for a locally-linearized system. In [26], the cost-to-go function from a hybrid optimal controller was used as the metric in an RRT to generate efficient plans for a nonlinear model of a helicopter.

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., [2, 31]). 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. Our initial implementation and experimentation with efficient nearest neighbor techniques indicate dramatic performance improvements (typically an order of magnitude or two in computation time). Three additional concerns must be addressed to incorporate efficient nearest neighbor techniques into the algorithm: 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 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, [12, 73]). Due to randomization, it is obvious that the generated trajectories are not optimal, even within their path (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 path 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 path 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 [27, 46, 51, 57]. For pairs of convex polyhedral bodies, the methods proposed in [46, 51] 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. It might be 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.

Acknowledgments

Steve LaValle thanks Bruce Donald for his advice on the kinodynamic planning problem. 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. This work has also benefitted greatly from discussions with Nancy Amato, Jim Bernard, Francesco Bullo, Peng Cheng, Brian George, David Hsu, Yan-Bin Jia, Lydia Kavraki, Jean-Claude Latombe, Jean-Paul Laumond, Kevin Lynch, Ahmad Masoud, and Jeff Yakey.

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

[4] D. Baraff. Rigid body simulation I - unconstrained rigid body dynamics. In *SIGGRAPH '97 Course Notes on Physically-based Modeling*, pages D1–D30. ACM SIGGRAPH, 1997.

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

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

[9] D. P. Bertsekas. Convergence in discretization procedures in dynamic programming. *IEEE Trans. Autom. Control*, 20(3):415–419, June 1975.

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

[11] J.-D. Boissonnat, A. Cérézo, and J. Leblond. Shortest paths of bounded curvature in the plane. *J. Intelligent and Robotic Systems*, 11:5–20, 1994.

[12] A. E. Bryson and Y.-C. Ho. *Applied Optimal Control.* Hemisphere Publishing Corp., New York, NY, 1975.

[13] 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.

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

[15] 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.

[16] M. Cherif. Kinodynamic motion planning for all-terrain wheeled vehicles. In *IEEE Int. Conf. Robot. & Autom.*, 1999.

[17] 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.

[18] 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.

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

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

[21] L. E. Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics*, 79:497–516, 1957.

[22] I. Duleba. *Algorithms of Motion Planning for Nonholonomic Robots.* Oficyna Wydawnicza Politechniki Wroclawskiej, Wroclaw, Poland, 1998.

[23] N. Faiz and S. K. Agrawal. Trajectory planning of robots with dynamics and inequalities. In *IEEE Int. Conf. Robot. & Autom.*, pages 3977–3983, 2000.

[24] 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.

[25] M. Fliess, J. Levine, P. Martin, and P. Rouchon. Flatness and defect of nonlinear systems. *International Journal of Control*, 61(6):1327–1361, 1993.

[26] E. Frazzoli, M. A. Dahleh, and E. Feron. Robust hybrid control for autonomous vehicles motion planning. Technical Report LIDS-P-2468, Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, 1999.

[27] 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.

[28] 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.

[29] D. Hsu, L. E. Kavraki, J.-C. Latombe, R. Motwani, and S. Sorkin. On finding narrow passages with probabilistic roadmap planners. In et al. P. Agarwal, editor, *Robotics: The Algorithmic Perspective*, pages 141–154. A.K. Peters, Wellesley, MA, 1998.

[30] D. Hsu, J.-C. Latombe, and R. Motwani. Path planning in expansive configuration spaces. *Int. J. Comput. Geom. & Appl.*, 4:495–512, 1999.

[31] 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.

[32] 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.

[33] R. Kindel, D. Hsu, J.-C. Latombe, and S. Rock. Kinodynamic motion planning amidst moving obstacles. In *IEEE Int. Conf. Robot. & Autom.*, 2000.

[34] 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.

[35] G. Laffierriere and H. J. Sussman. Motion planning for controllable systems without drift. In *IEEE Int. Conf. Robot. & Autom.*, 1991.

[36] F. Lamiraux and J.-P. Laumond. On the expected complexity of random path planning. In *IEEE Int. Conf. Robot. & Autom.*, pages 3306–3311, 1996.

[37] R. E. Larson. A survey of dynamic programming computational procedures. *IEEE Trans. Autom. Control*, 12(6):767–774, December 1967.

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

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

[40] 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.

[41] 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.

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

[43] 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.

[44] 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.

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

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

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

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

[49] R.A. Mayo. Relative quaternion state transition relation. *Journal of Guidance and Control*, 2:44–48, 1979.

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

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

[52] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[53] R. M. Murray, M. Rathinam, and W. M. Sluis. Differential flatness of mechanical control systems. In *Proc. ASME International Congress and Exposition*, 1995.

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

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

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

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

[58] J. A. Reeds and L. A. Shepp. Optimal paths for a car that goes both forwards and backwards. *Pacific J. Math.*, 145(2):367–393, 1990.

[59] 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.

[60] 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.

[61] G. Sahar and J. M. Hollerbach. Planning minimum-time trajectories for robot arms. *Int. J. Robot. Res.*, 5(3):97–140, 1986.

[62] 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.

[63] 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.

[64] 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.

[65] 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.

[66] K. Shoemake. Animating rotation with quaternion curves. In *Computer Graphics : Proc. of SIGGRAPH '85*, pages 245–254. ACM SIGGRAPH, 1985.

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

[68] 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.

[69] H. Sussman and G. Tang. Shortest paths for the Reeds-Shepp car: A worked out example of the use of geometric techniques in nonlinear optimal control. Technical Report SYNCON 91-10, Dept. of Mathematics, Rutgers University, 1991.

[70] 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.

[71] G. J. Toussaint, T. Başar, and F. Bullo. Motion planning for nonlinear underactuated vehicles using hinfinity techniques. Coordinated Science Lab, University of Illinois, September 2000.

[72] 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.

[73] 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.