A Quadratic Regulator-Based Heuristic for Rapidly Exploring State Space

by

Elena Leah Glassman

S.B., E.E. M.I.T., 2008

Submitted to the Department of Electrical Engineering and Computer Science

in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Electrical Engineering and Computer Science

at the Massachusetts Institute of Technology

February 2010

Author .... [illegible]

Department of Electrical Engineering and Computer Science
February 2, 2010

Certified by .... [illegible]

Russ Tedrake, X Consortium Associate Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by .... [illegible]

Dr. Christopher J. Terman
Chairman, Department Committee on Graduate Theses

A Quadratic Regulator-Based Heuristic for Rapidly Exploring State Space
by
Elena Leah Glassman

Submitted to the
Department of Electrical Engineering and Computer Science

February 2, 2010

In Partial Fulfillment of the Requirements for the Degree of
Master of Engineering in Electrical Engineering and Computer Science

ABSTRACT

Kinodynamic planning algorithms like Rapidly-Exploring Randomized Trees (RRTs) hold the promise of finding feasible trajectories for rich dynamical systems with complex, non-convex constraints. In practice, these algorithms perform very well on configuration space planning, but struggle to grow efficiently in systems with dynamics or differential constraints. This is due in part to the fact that the conventional proximity metric, Euclidean distance, does not take into account system dynamics and constraints when identifying which node in the existing tree is capable of producing children closest to a given point in state space. Here we argue that the RRTs' coverage of state space is maximized by using a proximity psuedometric proportional to the length, in time, of the quickest possible trajectory between two points in state space. We derive this minimum-time metric for the double integrator and show that an affine quadratic regulator (AQR) design can be used to approximate the exact minimum-time proximity pseudometric at a reasonable computational cost. We demonstrate improved exploration of the state spaces of the double integrator and simple pendulum when using this pseudometric within the RRT framework. However, for more complex nonlinear systems, experiments thus far suggest that the AQR-based proximity pseudometric and the conventional metric produce equivalent coverage of the state space, on average. This drop-off in benefit as system complexity and nonlinearity increase may be due to the linearization of system dynamics that is required to calculate the AQR-based pseudometric. Future work includes exploring methods for approximating the exact minimum-time proximity pseudometric that can reason about dynamics with higher-order terms.

Thesis Supervisor: Russ Tedrake
Title: X Consortium Associate Professor of Electrical Engineering and Computer Science

Acknowledgments

I gratefully acknowledge the contributions of my advisor, Russ Tedrake, groupmates in the Robot Locomotion Group, and members of the greater MIT community, especially David Wingate, Rick Cory, Alec Shkolnik, Ian Manchester, John Roberts, and Martin Glassman. I also appreciate the thoughtful feedback on related conference paper drafts from Seth Teller and Abraham Bachrach.

1 INTRODUCTION

Kinodynamic motion planning algorithms attempt to find feasible trajectories for a dynamical system from a start state to a goal state while respecting constraints on position, velocity, and/or acceleration. The problem is believed to be at least PSPACE-hard[8], however a number of randomized algorithms have been proposed[24, 1] which can achieve fast average-time performance for a large variety of problems[1, 2, 3, 6].

A common theme running through many path-planning algorithms is some notion of proximity in the space in which trajectories lie. In algorithms that attempt to create roadmaps, paths are found between *neighboring* nodes. In the Rapidly Exploring Random Tree (RRT) algorithm, nodes of a tree are grown toward randomly selected goals; only the node that is *closest* to the randomly selected goal is expanded [2], [3].

The proximity function that maps two points to a proximity score can be defined however the user sees fit. It does not need to meet the formal requirements of a metric, such as symmetry. It provides the user with the opportunity to incorporate his/her prior knowledge about the problem; he/she defines what makes two nodes neighbors in a roadmap, or what makes a point close enough to a goal state for a path to be considered complete, or to which nodes it is least costly to steer the system, from some specified initial state (i.e., cost-to-go).

The performance of the RRT, a particularly popular and simple randomized path-planning algorithm that is currently one of the most promising methods for planning in phase space and for solving other problems with differential constraints [14], can vary greatly as a function of the definition of proximity [9]. The basic RRT algorithm is shown in Table 1, and illustrated in Fig. 1. When used to find paths from x_{init} to a specific x_{goal}, x_{goal} is assigned to x_{rand} for some small percentage of the interations of BUILD_RRT. The definition of proximity has its effect by determining which node the NEAREST_NEIGHBOR function returns for the RRT to extend.

1.1 History

When LaValle and Kuffner first published the RRT algorithm, they explicitly tackled the problem of kinodynamic planning [2]. Their algorithm planned in state space, a space twice as large as configuration space, including both positions and velocities of all the degrees of freedom, where obstacles are not only defined by the robot colliding with itself or its environment but also laws of physical motion and the limits of finite applied forces. While they documented their randomized motion planning algorithm's successes, they acknowledged that an additional component that would address remaining barriers to more efficiently exploring state space is a perfect, quickly computable distance pseudo-metric.[1] Note that the degree of coverage of the state space can serve as a surrogate measure for the mean time to find

[1]The term *pseudometric* is used because it allows for asymmetries. If the distance between two points in state space reflects the energy necessary to drive the system from one state to the other, the distance of state B from A may not be equal to the distance of state A from B.

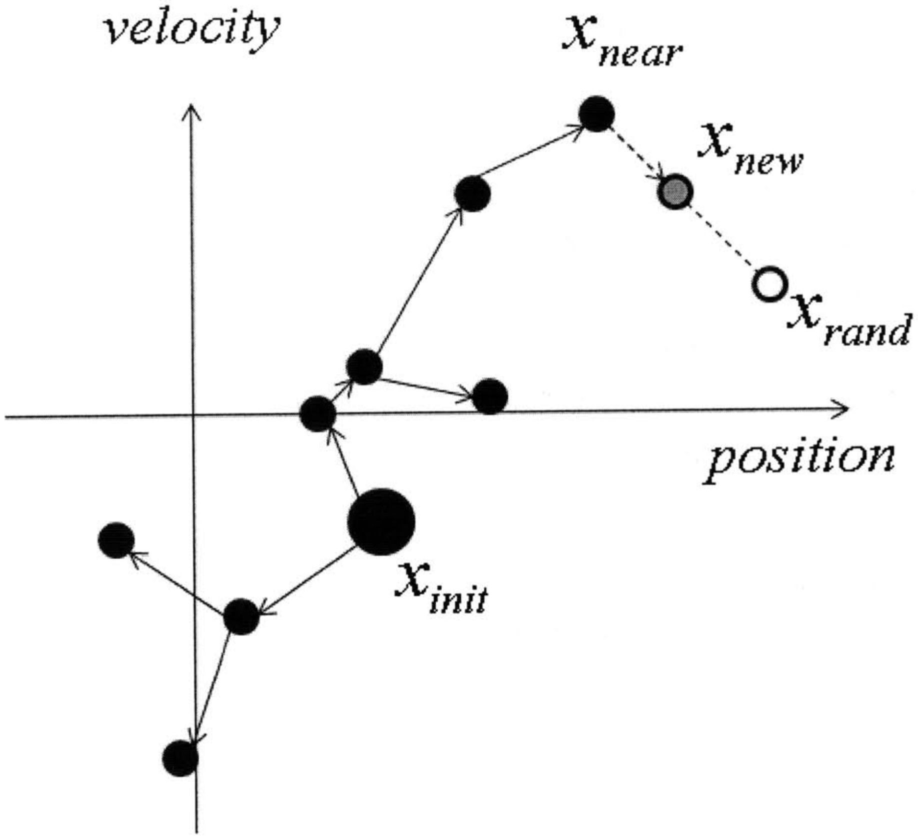


Figure 1: Illustration of one iteration of growing an RRT, adapted from [9].

```
1: **procedure** BUILD_RRT($x_{init}$)
2:     $T$.init($x_{init}$);
3:     **for** $k = 1$ to $K$ **do**
4:         $x_{rand} \leftarrow$ RANDOM_STATE();
5:         EXTEND($T, x_{rand}$)
6:     **end for**
7:     Return $T$
8: **end procedure**
9: **procedure** EXTEND($T, x$)
10:    $x_{near} \leftarrow$ NEAREST_NEIGHBOR($x, T$);
11:    **if** NEW_STATE($x, x_{near}, x_{new}, u_{new}$) **then**
12:        $T$.add_vertex($x_{new}$);
13:        $T$.add_edge($x_{near}, x_{new}, u_{new}$);
14:    **end if**
15: **end procedure**
```

Table 1: The basic algorithm for constructing RRTs, adapted from [9]

a feasible path.

An ideal pseudometric is the optimal cost-to-go function; however, computing the optimal cost-to-go is equivalent, in difficulty, to solving the original planning problem [9]. Even if an exact psuedometric does not exist, LaValle and Kuffner argue that approximations will still dramatically improve performance. Ten years after the original RRT paper, RRTs are still considered a promising method for kinodynamic planning and other problems with differential constraints [14]. Both distance pseudometric creation and RRT algorithm modification for more effective exploration of highly constrained and high-dimensional problems' state spaces are areas of active research. This thesis develops and implements the use of locally linearized systems' optimal cost-to-go functions as proximity heuristics. In the process of developing the idea, we found that LaValle and Kuffner have mentioned the possibility of using cost-to-go functions from applying optimal control to linearized systems, as part of a list of many possibilities, including Lyapunov functions, fitted spline curves, and steering methods [4].

1.2 Calculating Optimal Cost-to-Go

There is literature on how to analytically or numerically solve for the optimal path [12],[13] but applying the methods to a particular problem often requires mathematical finesse and intuition on the part of the human in the loop. A separate chapter covers methods of calculating optimal cost-to-go functions, both exact and approximate, when cost is time.

1.3 Related Terms and Function Types

If an approximate cost-to-go function is known for the goal state which (1) assigns a cost of zero at the goal, (2) assigns an infinite cost to states from which the goal is unreachable, and (3) has an associated "local operator" that selects an action at each state that results in a state with a lower cost, then the cost-to-go function is a *feasible navigation function*. This function can be greedily descended using the local operator to reach the goal state, making an RRT unnecessary [5].

Note also that if the cost-to-go function, ρ, also satisfies the principle of optimality,

$$\rho(x) = min_{u \in U(x)} \{l(x,u) + \rho(f(x,u))\} \tag{1}$$

where $l(x,u)$ is the cost of taking action u at state x, then ρ can also be referred to as an optimal cost-to-go function, also known as an optimal value function in the value iteration and dynamic programming literature.

The metric (in configuration space) induced by the cost of optimal paths between points, in the context of nonholonomic motion planning, is also called the nonholonomic, singular, Carnot-Caratheory, or sub-Riemannian metric [15].

1.4 Existing Pseudometrics

There are a varied of previously published pseudometrics. Note that while some have not been assessed in terms of state space coverage, a pseudometric that has good published performances in goal-directed path planning will also help an RRT reach the randomly selected target points that cause the tree to expand and fill the sampled state space with tree nodes.

Perhaps the most common pseudometric, which is in fact a metric, is that of Euclidean distance between two state space points as a function of their coordinates in that space:

$$distance(x, x') = \sqrt{\alpha\,(x_1 - x_1')^2 + \beta\,(x_2 - x_2')^2 + \ldots + \zeta\,(x_n - x_n')^2}, x, x' \in \Re^n \tag{2}$$

The scaling factors can be used to encode domain knowledge about the relative significance of various state components. This metric works well on holonomic systems, in terms of coverage of the sampled space, but performs far more poorly in state space, which will be demonstrated in the chapter containing experimental results. It encodes no information about the constrained relationship between position and velocity. In the phase space of a frictionless one-dimensional brick shown in Fig. 2, points A and B are equidistant from point C with respect to Euclidean distance. Yet, as observers with a priori knowledge about phase space, we know that the brick at point A is moving toward point C, while the second instance of the brick, at point B, is moving away from point C. Since the proximity function determines which branch will be extended toward C, it makes intuitive sense to define proximity so that A is in fact closer to C than B.

1.4.1 Problem-Specific Pseudometrics

With a simple energy-based pseudometric, an RRT can find a very direct path for an underactuated simple pendulum to ascend to the unstable upright position, wasting very few tree nodes on spurious paths. The pseudometric is simply the difference in energy between two states. In the two-dimensional space representing the angular position and velocity of the pendulum, there is a connected set of states which have the same energy as the goal state. If the pendulum is frictionless, then once it is driven to a state in that set, it will passively reach the goal state. For energy-conservative systems in which (1) the goal lies on a set of connected states with the same energy, (2) the dynamics along that connected set bring that system to the goal state passively, and (3) no other states outside of that connected set have

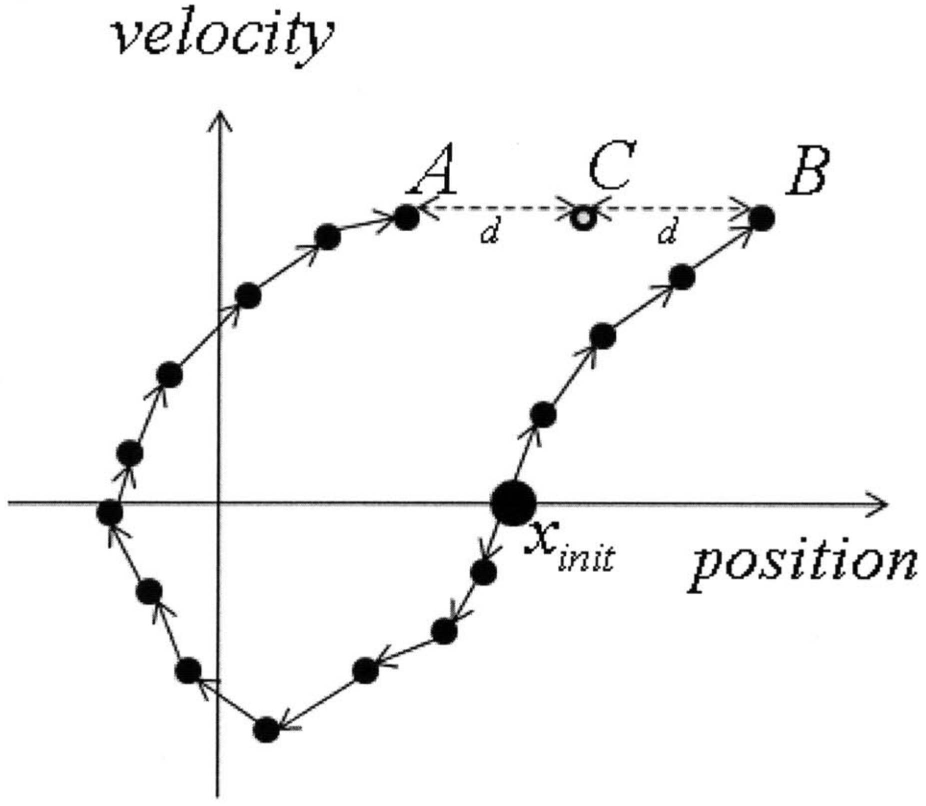


Figure 2: Nodes A and B are equidistant to point C with respect to the Euclidean distance metric. Children of A could be even closer to C with respect to Euclidean distance. However the children of node B, which has positive velocity, must be to the right of their parent, and therefore cannot get closer to C. In this case, Euclidean distance fails to distinguish between two branches, one which is clearly the right choice for extending towards C, and the other which is not.

the same energy, this can be a very effective pseudometric.

[19] applied RRTs to the control of a blimp. The approximation of the ideal metric was designed using knowledge of the blimp dynamics and the underlying Lie group structure. The blimp is modeled as a general rigid body with underactuated dynamics. If the moments of inertia are identical and no external forces are applied, then the shortest distance path between two points given by Hamilton's principle can be found analytically. This metric for the simpler case is combined, in an unspecified way, with the Euclidean distance between the two points in state space, where the relative values of the weighting factors on each state component have been determined by intuitive reasoning.

[20] used RRTs to plan dynamic trajectories for helicopters by using cost-to-go functions from the unconstrained problem to solve for combinations of trim trajectories (maneuvers/motion primitives) in an environment with obstacles. The cost-to-go calculation was made even more tractable by exploiting the (problem-specific) symmetries and relative equilibria of helicopters, along with the construction of motion primitives.

1.4.2 Pseudometrics That Do Not Incorporate Problem-Specific Assumptions/Derivations

[21] uses the local first order approximation of cost-to-go, where cost is time, and the calculation is simple enough that it only requires the Euclidean distance between state space points and evaluations of given system dynamics function.

1.5 Modifications to the RRT Algorithm

There are two major themes that run through the published work on RRT algorithm modifications. These modifications are designed to make the RRT algorithm less sensitive to the quality of the pseudometric. The first theme is that of reducing repeated failed expansions of a node. For example, [22] collects collision information online and uses that to bias search. The probability of extending a given node is modulated by the *constraint violation tendency* (CVT), which is defined as the "ratio of the number of trajectories in violation in the explored region to the number of all possible constructed trajectories." The high-level reasoning behind [21]'s modification is essentially the same, but the implementation uses a *history-based weighting* instead of the CVT.

[14] categorizes child nodes into three types: those that are closer to x_{rand} than their parent, those that are further from x_{rand} than their parent but reach into previously unexplored space (*receding*), and those that are both further from x_{rand} than their parent and leading back into already explored space (*regressing*). The last category is obviously undesirable; it adds nothing to the tree. The receding type is potentially valuable, since the child node may in fact be closer to x_{rand} in terms of a more accurate pseudometric. Both receding and regressing node types are rejected by the RRT and allowed by the RRT with collision tendency ([22]), so [14] proposes RRT-Blossum, which attempts to distinguish between the two types, so that receding nodes are added while regressing nodes are ignored.

The second main theme in the work on RRT algorithm modification is adaptively biasing the sampling distribution towards regions of state space that are reachable by the tree's current set of nodes. [21] replaces the traditional uniform distribution of x_{rand} with a compactly supported Gaussian-based distribution centered around a point in region of interest. Since this kind of sample biasing can be helpful in some problems, but not all, the degree of biasing (the standard deviation of the Gaussian) is modulated by how successful the resulting node extensions are. Success, in this context, is defined as producing a child node with less distance to x_{rand} than its parent. [7] develops a sample biasing method he calls *reachability-guidance*. The Reachability-Guided RRT uses a measure of local reachability. The state space is sampled uniformly, but samples outside the estimate of the reachable set of the current tree are ignored.

1.6 Problem Formulation

Following the RRT framework, we require the following components:

1. **State Space:** A $2n$-dimensional differentiable manifold, X, that denotes the state space. A state, $x \in X$, is defined as $x = (q, \dot{q})$, for $q \in C$, where C is the n-dimensional configuration space.

2. **Metric:** A real-valued function, $\rho : X \times X \rightarrow [0, \infty)$, which specifies the cost of traveling between pairs of points in X in accordance with a specified cost function.

3. **Boundary Values:** $x_{init} \in X$ and $X_{goal} \subset X$.

4. **Constraint Satisfaction Detector:** A function, $D : X \rightarrow \{true, false\}$, which indicates when global constraints have been satisfied or violated.

5. **Inputs:** A set U of inputs containing all inputs that affect the state.

6. **Equation of Motion:** The dynamics expressed as a differential equation $\dot{x} = f(x, u)$.

7. **Incremental Simulator:** A function for generating future states of the agent given the current state, the equations of motion, a time interval, and u over that time interval.

The primary objective of this work is to develop a metric that increases the capability of the RRT to explore state space. This breaks down into two subproblems: determining the appropriate cost function for maximum state space coverage and developing an approximation of the optimal cost-to-go function that can be computed efficiently and scales well with the number of state variables. The pseudometric will be directly compared to the exact optimal cost-to-go function when the later function is known, and its impact on the ability of the RRT to explore state space will be assessed on four different dynamic systems.

2 Minimum-Time Solutions

Finding the trajectory that takes a system between given initial and final conditions in the least amount of time can be as simple as solving a matrix equation or complex enough to have no known analytical results despite the efforts of mathematicians and engineers who have attempted to find solutions over the past sixty years. This task is referred to as a minimum-time optimal control problem.

2.1 Motivation

As explained earlier, our goal is to create RRTs that have greater coverage of the state space than is presently achieved with a NEAREST_NEIGHBOR function based on the Euclidean distance metric. Efficient coverage from the RRTs comes from the Voronoi bias.

Most RRTs in the existing literature, including the basic RRT algorithm we employ in this work, expand nodes forward for a fixed Δt. In order to be efficient in the number of nodes, we would like to solve a minimum-time problem from each existing node of the tree to the subgoal. Minimum-time solutions are not known for most systems, but we review here the systems where they are known and discuss their efficiency.

2.2 Continuous Time Systems

The continuous-time double integrator with bounded input is a system that is simple enough to have an analytical minimum-time solution but complex enough to have a solution "style" that carries through to more general continuous-time systems. The dynamics and constraints of the system are as follows:

$$\ddot{x} = u, \tag{3}$$

$$B_- \leq u \leq B_+, \tag{4}$$

$$B_- < 0 < B_+ \tag{5}$$

x is u after integrating twice. The system's state evolution equation is linear, but the limit on actuation introduces a nonlinearity into the system as a whole. A physical representation of this system is a one-dimensional brick that can be pushed left or right along a line. x represents the brick's position, and u represents the magnitude and direction of the force applied to it. If u were unbounded, x could instanteously reach any desired value, and the minimum-time path is infinitely quick. (The optimal control problem would therefore not be well defined.) Also, unlimited force is unrealistic.

Even though limits on u introduce a nonlinearity into the system, the mapping of u to x is simple enough that we can still derive a closed-form expression for the minimum-time path between two arbitrary points in the state space of x, which is 2D with coordinates x and $\dot{x}$, or position and velocity in terms of the physical example of the brick. [12] presents the minimum time solution for reaching the state space origin (or any other position for which the velocity is zero, since the system's dynamics are invariant with respect to position). To compare the AQR-based pseudometric to the exact minimum-time pseudometric, the solution needed to be extended to reaching points in state space with non-zero velocities.

The major points established in [12]'s derivation for reaching the state space origin are the following:
(1) The minimum-time solution is that the input u will always be "at the rails:" either pinned to the upper or lower bound on u. (This is also referred to as a bang-bang solution.) It follows, then, that minimum-time solutions are purely composed of the bang-bang curves, the arcs through state space that the system traverses when u is at its upper or lower bound. In the case of the brick, these arcs are parabolic about the position (x) axis, and which direction, left or right, to which they open is determined by the rail to which the input is pinned. Which rail to which u is pinned also determines whether the system travels upward along the parabolic curve, which happens when u is at its upper bound, or downward, when at its lower bound. Since the dynamics are not affected by position, these two directional parabolic curves can be shifted right and left along the x axis and still be valid system trajectories.
(2) The switching curve is composed of all states from which it is possible to reach the goal in the minimum amount of time possible, but without switching which bound u is pinned to. This switching curve is shown in Figure 3. All minimum-time trajectories will converge on this switching curve because all minimum-time trajectories will be slowing down to the zero-velocity, zero-position origin point with u at one of the rails. Minimum-time trajectories can be broken down into two subtasks: getting to the switching curve in the least possible time and riding the switching curve into the origin. When the system is below the switching curve, the minimum-time route to the switching curve follows the upward-directional bang-bang curve. Likewise, when the system is above the switching curve, the minimum-time route to the switching curve follows the downward-directional bang-bang curve. Switching from the bang-bang curve that delivers the system to the switching curve to following the switching curve itself always requires pinning u to the bound it was not previously set to, and the time-optimal path has at most one switch.

Extending this solution to reaching points with non-zero velocity requires a shift in view about the problem (and possibly a corresponding change in implementation) but the method stays unchanged. The two halves of the switching curve for a point with non-zero velocity do not meet to form an inflection point at the goal. Instead, they meet at a point, as illustrated in Fig. 4. The switching curve is no longer able to be represented as a function for which each position x maps to exactly one velocity $\dot{x}$. Earlier we described the first bang-bang curve as being determined by whether the

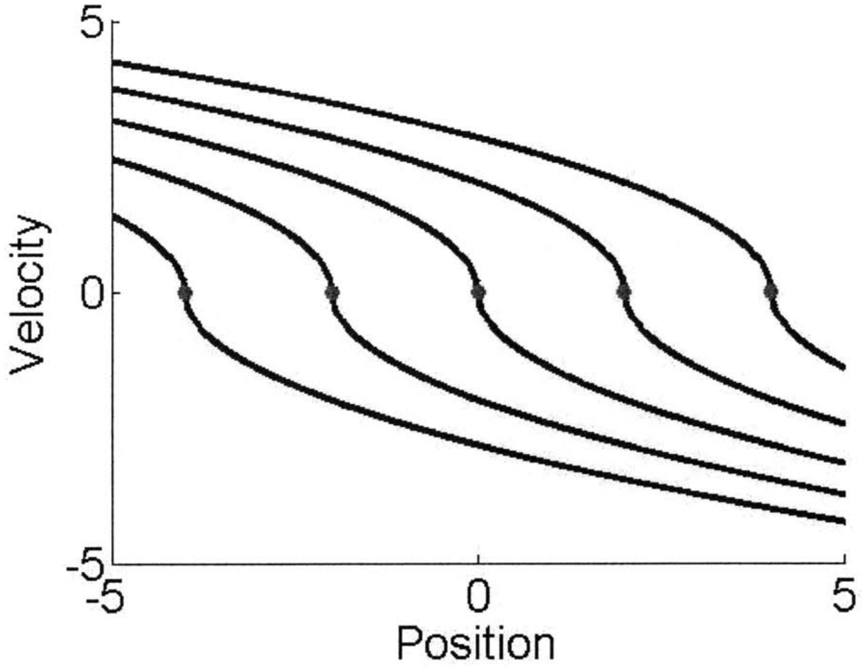


Figure 3: Switching curves for goals (shown as red points) that have zero velocity.

system starts above or below the switching curve. Equivalently, we could have described the first bang-bang curve as being determined by whether the system starts to the right or left of the switching curve, since the switching curves for goals with zero velocity are monotonically decreasing functions of position (Fig. 3). This second method (left/right rather than above/below) generalizes to determining the first bang-bang curves for goal points with non-zero velocity, whose switching curves are no longer functions of position, let alone monotonically decreasing (Fig. 4).

Now that the composition of the minimum-time trajectory between any initial and final point, with zero or non-zero velocity, has been established, it is possible to calculate the time it takes to traverse this time-optimal path. To do this, it is necessary to have mathematical descriptions of the system trajectories that sweep through state space when the input is set to one of its two bounds. When u is set to a constant,

$$\ddot{x}(t) = \gamma, \tag{6}$$

we have a closed form expression for velocity as a function of time,

$$\dot{x}(t) = \gamma t + \dot{x}_{init}, \tag{7}$$

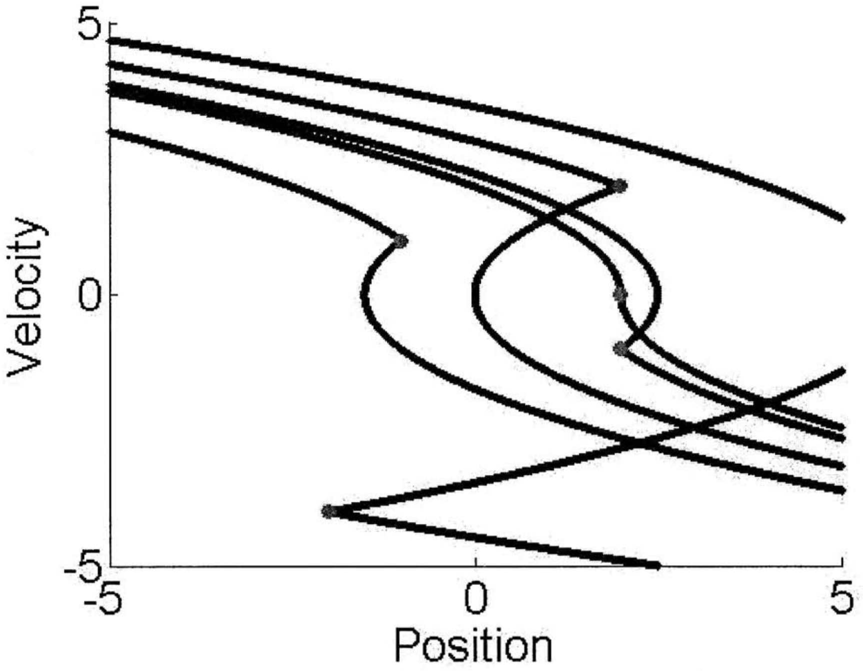


Figure 4: Switching curves for goals (shown as red points) with either zero or nonzero velocity.

and for position as a function of time, γ, and an initial position and velocity,

$$x(t) = \frac{1}{2}\gamma t^2 + \dot{x}_{init} + x_{init}. \tag{8}$$

By solving Equ. 7 for t and plugging that resulting expression in for t in Equ. 8, we get the family of all possible system trajectories for constant u:

$$\gamma\left(x - x_{init}\right) = \dot{x}_{init}\left(\dot{x} - \dot{x}_{init}\right) + \frac{1}{2}\left(\dot{x} - \dot{x}_{init}\right)^2 \tag{9}$$

which is a parabola for which (1) the curvature is scaled by the magnitude of γ, (2) the direction of opening (left or right) is determined by the sign of γ, and (3) the intersection with the position axis is determined by the state $[x_{init}, \dot{x}_{init}]$ that the curve is constrained to pass through.

If a point on the curve is the origin, Equ. 9 simplifies to:

$$\gamma x - \frac{1}{2}\left(\dot{x}\right)^2 = 0 \tag{10}$$

from which the quadratic relationship between position and velocity is easier to visualize. The dynamics are agnostic to position, so these curves can be shifted in either direction along the position axis and still be valid. Setting the left-hand side of Equ. 10 equal to some constant other than zero performs that operation. To shift the curve so that it intersects a specific state $[x_d, \dot{x}_d]$, the constant is set to the value of the left-hand side when $[x_d, \dot{x}_d]$ are plugged in for $[x, \dot{x}]$.

Above the goal point, $[x_G, \dot{x}_G]$, the switching curve will be of the downward type ($u = B_-$) and pass through $[x_G, \dot{x}_G]$:

$$B_- x - \frac{1}{2}(\dot{x})^2 = B_- x_G - \frac{1}{2}(\dot{x}_G)^2, \tag{11}$$

$$\dot{x} > \dot{x}_G \tag{12}$$

Below the goal point, the switching curve will be of the upward type ($u = B_+$) and also pass through $[x_G, \dot{x}_G]$:

$$B_+ x - \frac{1}{2}(\dot{x})^2 = B_+ x_G - \frac{1}{2}(\dot{x}_G)^2, \tag{13}$$

$$\dot{x} < \dot{x}_G \tag{14}$$

If the initial state is to the right of this switching curve, the optimal policy will be $u = B_-$ then $u = B_+$. The reverse is true if the initial state is to the left of the switching curve. We'll call these first and second assignments of u to be γ_1 and γ_2, so that the following derivation is general regardless of on which side of the switching curve the initial point falls. We can now specify the two bang-bang curves that form the time-optimal path, the first curve from the initial point, $[x_0, \dot{x}_0]$, to the intersection with the switching curve:

$$\gamma_1 x - \frac{1}{2}(\dot{x})^2 = \gamma_1 x_0 - \frac{1}{2}(\dot{x}_0)^2 \tag{15}$$

and the switching curve to the goal:

$$\gamma_2 x - \frac{1}{2}(\dot{x})^2 = \gamma_2 x_G - \frac{1}{2}(\dot{x}_G)^2 \tag{16}$$

By reorganizing the terms and setting them equal to each other, we can solve for the $\dot{x}^2$ of the intersection point. When $\gamma_1 > \gamma_2$, the positive square root is the true $\dot{x}$ of the intersection point; when $\gamma_1 < \gamma_2$, the opposite is true. Finally, the position of the intersection point can be found by substituting the $\dot{x}$ of the intersection point into either of the intersection two curves' equations. We will refer to the intersection point as $[x_C, \dot{x}_C]$.

Now that the intersection point is known, the length it takes to travel each of the two bang-bang curves can be found by substituting into Equ. 7 and solving for t:

$$t_1 = \frac{\dot{x}_C - \dot{x}_0}{\gamma_1} \tag{17}$$

$$t_2 = \frac{(\dot{x}_G - \dot{x}_C)}{\gamma_2} \tag{18}$$

The time it takes to traverse the minimum-time path is the sum of t_1 and t_2, and the policy is to set u to γ_1 for the first t_1 seconds and to γ_2 for the second t_2 seconds.

2.2.1 More General Results

Switching functions, which provide a visual cookbook for determining the time-optimal path for bounded-input systems, potentially both nonlinear and linear, can be analytically calculated using the Pontryagin Minimum Principle, as shown in [28], when the dimension is small. For systems with three or more state variables, it becomes "generally difficult, if not impossible, to obtain an analytical expression for the switching hypersurface" [27]. The most general analytical results are described in [26], but only the form of the solution is given, without a method for calculating it.

2.3 Discrete-Time Linear Systems

The methods which give exact solutions for discrete-time (DT) systems are simpler than their counterparts for continuous-time (CT) systems. The equation that describes the system's evolution as a function of state and input is:

$$x[n+1] = Ax[n] + Bu[n] \tag{19}$$

The equation that describes the relationship between an initial condition x_0, a sequence of inputs, and the resulting system state is:

$$x[n] = A^n x[0] + C \begin{bmatrix} u[0] \\ u[1] \\ \vdots \\ u[n-1] \end{bmatrix}, C = \begin{bmatrix} B & AB & \dots & A^{n-1}B \end{bmatrix} \tag{20}$$

2.3.1 Unrestricted Inputs

The simplest method presented in this chapter is for a DT system with no restrictions on the input. Finding the time (and the associated sequence of inputs) of the minimum-time trajectory is as straightforward as solving increasingly large matrix equations.

Algorithm Given x_0 and x_f as the required initial and final states,

1. Solve

$$x_f - Ax_0 = \begin{bmatrix} B \end{bmatrix} \begin{bmatrix} u \end{bmatrix} \tag{21}$$

2. If a solution exists, return the trajectory's length in time steps (1) and the associated steering function (in this case, scalar) u

3. If no solution exists, solve

$$x_f - A^2x_0 = \begin{bmatrix} B & AB \end{bmatrix} \begin{bmatrix} u[0] \\ u[1] \end{bmatrix} \tag{22}$$

4. If a solution exists, return the trajectory's length in time steps (2) and the associated steering fuction $\vec{u}$

5. Repeat this process, attempting to solve the equation for input sequences of increasingly large duration (n), terminating when a solution is found

$$x_f - A^nx_0 = \begin{bmatrix} B & AB & \dots & A^{n-1}B \end{bmatrix} \begin{bmatrix} u[0] \\ u[1] \\ \vdots \\ u[n-1] \end{bmatrix} \tag{23}$$

Originally, I firmly believed that one only needed to attempt to solve at most N matrix equations, where N is the number of state variables, but there may actually not be any stopping criterion when it is known that no solution of any length exists. If the matrix equation has no solution when $n = N$, then the columns of the square matrix $\begin{bmatrix} B & AB & \dots & A^{N-1}B \end{bmatrix}$ do not span a space that includes the left-hand side of the equation and no additional columns (part of solving for longer steering functions) will increase that rank and expand the spanned space. However, if the vector, $x_f - A^nx_0$, on the left-hand side, which also changes as a function of n, can fall into the spanned space of $\begin{bmatrix} B & AB & \dots & A^{N-1}B \end{bmatrix}$ as n increases beyond N, then stopping at $n = N$ is not sufficient and iterations for $n > N$ are necessary to find a solution, if one exists.

2.3.2 Bounded Inputs

By imposing an additional restriction, i.e., placing bounds on each component of the input vector,

$$|u_i| \leq 1 \tag{24}$$

the problem as a whole becomes nonlinear and solveable by linear or quadratic programming instead of simply finding solutions to matrix equations. Linear and quadratic programming (LP and QP, respectively) techniques operate on matrix *inequalities* rather than equalities, but it is not difficult to set the LP or QP up in such a way that equalities, such as the relationship between current states, actions, and future states, are respected in addition to inequalities like the input bound above. If we are to consider steering functions that are n time steps long, we replace the equation for the relationship between current states, actions, and future states

$$x_f - A^n x_0 = \begin{bmatrix} B & AB & \dots & A^{n-1}B \end{bmatrix} \begin{bmatrix} u[0] \\ u[1] \\ \vdots \\ u[n-1] \end{bmatrix}, \tag{25}$$

with an equivalent set of inequalities that must be simultaneously satisfied:

$$x_f - A^n x_0 \leq \begin{bmatrix} B & AB & \dots & A^{n-1}B \end{bmatrix} \begin{bmatrix} u[0] \\ u[1] \\ \vdots \\ u[n-1] \end{bmatrix}, \tag{26}$$

$$x_f - A^n x_0 \geq \begin{bmatrix} B & AB & \dots & A^{n-1}B \end{bmatrix} \begin{bmatrix} u[0] \\ u[1] \\ \vdots \\ u[n-1] \end{bmatrix} \tag{27}$$

Multiplying the second inequality by -1 will allow for stacking the two conditions into one inequality that LP and QP can be applied to.

Since there appears to be no way to consider steering functions of different lengths without considering matrix inequalities of different sizes (the same equation cannot be used to test for the existence of steering functions of two different lengths), it is necessary, just like for DT systems with unbounded inputs, to progressively solve larger and larger LP/QP problems. Such iterations would only terminate when a solution is found, and could continue indefinitely without ever finding a solution.

Even though LP and QP optimize different cost functions of the steering vector (solution), the cost function is only used to determine which steering vector, of all n-long steering vectors, is returned. Since the cost function of interest to us in this context is purely n, it doesn't matter whether a linear and quadratic cost function is used to determine which of the smallest, n-long steering vectors is returned as the final solution.

2.3.3 Expanding Slightly beyond Simple Minimum-Time Solutions

The simple matrix-equation-solving approach can only be used to find minimum-time solutions. However, it is possible to use the LP and QP machinery to find minimal-cost (or at least locally minimized-cost) solutions for systems with bounded inputs and cost functions beyond just time, such as:

- $c^T u$
- $1 + \gamma c^T u$

within an LP and

- $u^T Q u$
- $1 + \alpha u^T Q u$
- $1 + \alpha u^T Q u + \beta c^T u$

within a QP. As previously mentioned, LP and QP only minimize these cost functions over a steering vector u of fixed length. Finding the global minimum over multiple n is not guaranteed.

If there are no bounds on the input, then an additional cost function type can be handled, which is quadratic in both state error and input, such as $1 + \alpha x^T Q x + \beta u^T R u$. The discrete form of the Riccati equation can be used to solve multiple finite-horizon linear quadratic optimal control problems.

2.4 Nonlinear Systems

Minimum-time problems can be solved for nonlinear systems using dynamic programming. [25] develops the machinery for discrete-time systems with discrete states and actions, which reduces to an algorithm that takes in the S possible states, the A possible actions, a horizon time, a dynamics function $s' = f(s, a, n)$, an instantaneous cost $g(s, a, n)$, and a terminal cost $h(s)$. A minimum-time solution would be found by using a terminal cost function $h(s)$ that is infinite everywhere but the goal and setting the instantaneous cost function to some constant value. The algorithm gives as output an optimal policy $\pi^*(s, n)$ and optimal cost-to-go function $J^*(s, n)$. The solution is exact.

However, as soon as the discrete state condition is changed to continuous state, a function approximator is needed to store the cost-to-go function, and the solution may no longer be exact.

2.5 Efficiency

These minimum-time solutions were not chosen as foundations for RRT proximity pseudometrics because of computational considerations and scalability issues. The bang-bang solutions for force-limited continuous-time systems are not efficiently computable for state spaces with a dimensionality greater than two. The solutions for discrete-time systems only solve for specific point-to-point queries; if used within a pseudometric, the solution would need to be calculated for each tree node every time a new subgoal, x_{rand}, is sampled from the state space. The solutions for nonlinear systems also do not scale well with the dimensionality of the state space. Quadratic regulators, which do not solve minimum-time problems, but which are efficient to compute and which produce solutions for all of state space simultaneously, are presented as an approximation to the minimum-time solution in the next chapter.

3 DESIGNING THE AQR-BASED PROXIMITY HEURISTIC

There is a class of optimal controllers for linear systems with quadratic cost functions of state and/or action for which the global exact closed-form cost-to-go to the specified goal state in the dynamic system's phase space can be found by numerical matrix integration. These controllers are called linear quadratic regulators (LQR), and, as mentioned in the chapter on relevant literature, LaValle identified the cost-to-go functions of optimal control problems such as these as promising candidate proximity functions for the NEAREST_NEIGHBOR function, even when the system the RRT is being built on is nonlinear. LaValle suggested this class, and this thesis work develops the idea and explains the significant design choices made to compose the final proposed proximity function.

Consider a smoothly differentiable, possibly nonlinear system:

$$\dot{x} = f(x, u), x \in \Re^n, \quad u \in \Re^m \tag{28}$$

and x_{rand}, a random sample in the phase space produced by the RRT algorithm that is building a tree on this system. For notational simplicity, we will define a new coordinate system centered about x_{rand}:

$$\bar{x} = x - x_{rand}. \tag{29}$$

The RRT algorithm must select a node of the tree to grow towards x_{rand}, so, in order to get a cost-to-go to x_{rand} from each RRT node, we will use a controller that attempts to drive the system to x_{rand}.

Since, at this time, there is no efficient systematic design approach available for finding optimal controllers for general nonlinear systems [12], we linearize the general, possibly nonlinear dynamic system, define a quadratic cost function on state $x(t)$ and action $u(t)$, and solve for the optimal controller of the new system. The cost-to-go function associated with that optimal controller will be most accurate around the linearization point. We chose x_{rand} as the linearization point so that the global optimal cost-to-go could be found once, using the dynamics from linearizing about the sample goal x_{rand}, rather than finding the global optimal cost-to-go to x_{rand} using the dynamics from linearizing about RRT node i and then repeating that process for each node in the tree.

The linearized system dynamics are defined by the zero and first-order terms of the Taylor expansion of the original system's dynamics. We have to specify both the state x_{rand} and the input u being applied at that state to find the linearized dynamics. However, in practice, u is always set to a zero vector. However, for generality, we will refer to this specified u as u_f.

3.1 The Trouble with Non-Fixed (Unstabilizable) Points

Since fixed points lie on subspaces of the full state space when the dynamic system is not degenerate, the probability of randomly sampling a fixed point as the next sample goal, (x_{rand}, u_f), when the full state space is being sampled uniformly, is zero. When (x_{rand}, u_f) (the linearization point) is not a fixed point, the zero-order term of the Taylor expansion of $\dot{x}$ will not be a zero vector. In this case, while the system is referred to as linearized, the product of the linearization process will in fact be affine. The derivation that follows mirrors that of an open-loop LQR, but has been generalized to affine systems, and for that reason, we refer to the resulting controller as an affine quadratic regulator (AQR).

Under the control of an infinite-horizon LQR designed to drive the linearized system to a fixed point, the system will converge on that fixed point asymptotically.

An infinite-horizon affine version of LQR cannot exist for non-fixed points, which are unstabilizable and impossible to approach asymptotically. If the system actually reaches the non-fixed point, the dynamic constraint that defines a non-fixed point (a combination of state space location and input for which $\dot{x}$ is non-zero) forces the system to move away from the non-fixed point it just reached. The assumed smoothness of the dynamics as a function of state and action allows us to extend this property to the neighborhood around the non-fixed point and conclude that the non-fixed point cannot be approached asymptotically. The AQR derivation below is an example of a finite-horizon affine version of LQR.

3.2 Derivation of the Affine Quadratic Regulator

The following series of equalities and approximations shows the derivation of the linearized (affine) system from the derivative of the state in the new coordinate frame. Note that the constant offset that places the origin of the new coordinate frame at the coordiates of x_{rand} in the old frame has no effect on the change in state as a function of states and inputs.

$$\begin{aligned}\dot{\bar{x}} &= \frac{d}{dt}\left(x(t) - x_{rand}\right) = \dot{x}(t) && (30)\\ &\approx f(x_{rand}, u_f) + \frac{\delta f}{\delta x}\left(x(t) - x_{rand}\right) + \frac{\delta f}{\delta u}\left(u - u_f\right) && (31)\\ &= c + A\bar{x} + B\bar{u}, \quad A \in \Re^{nxn}, \quad B \in \Re^{nxm} && (32)\end{aligned}$$

We define a quadratic cost on $\bar{u}$, plus a constant. Since the constant is inside the integrand, it adds the total length of a path to the cost of that path, serving as a penalty on paths of longer durations. The relative cost of quick paths and paths which require little input to follow is determined by the positive definite R.

$$J(\bar{x}, t_0, t_f) = \int_{t_0}^{t_f} \left[1 + \frac{1}{2}\bar{u}^T(t)R\bar{u}(t)\right] dt, R = R^T > 0, \tag{33}$$

$$s.t.\ \bar{x}(t_f) = \vec{0} \tag{34}$$

$$\bar{x}(t_0) = \bar{x}_0 \tag{35}$$

$$\dot{\bar{x}} = A\bar{x} + B\bar{u} + c. \tag{36}$$

While this cost function, J, is parameterized by both t_0 and t_f, the linearized dynamics are autonomous, so the cost function can be re-parameterized with respect to time, without loss of generality, by T, which is $t_f - t_0$. That total trajectory length T can have a significant impact on the cost-to-go from a state to x_{rand}. Consider a situation where, just by the passive linearized dynamics alone, the linearized system will arrive at x_{rand} from some state x_i in t_i seconds. Depending on how B maps u to a change in state, it can take a lot of additional control effort to speed or slow the system's arrival at x_{rand}.

The optimal control solution for this constrained system can be found using Pontryagin's minimum principle, which is a necessary condition for optimality. To apply this principle, we must define the Hamiltonian; it is a linear combination of the integrand of the cost function J and the expression for the change in state $\dot{\bar{x}}$ as a function of state and action [12].[2] The $\lambda(t)$ is the factor that makes the Hamiltonian a linear combination of terms from the objective function and the constraints imposed by the dynamics, and it is referred to as the Lagrange multiplier.

$$H(t) = 1 + \frac{1}{2}\bar{u}^T(t)R\bar{u}(t) + \lambda(t)\left[A\bar{x} + B\bar{u} + c\right] \tag{37}$$

The Hamiltonian function should be at a minimum or stationary point with respect to changes in the control function $\bar{u}(t)$ so we also constrain the derivative of $H(t)$ with respect to $\bar{u}$ to be zero [12]. If we define L as the integrand of J, we can more concisely show the specific constraint for our linear quadratic problem.

$$0 = \frac{\delta H}{\delta \bar{u}} \tag{38}$$

$$= \frac{\delta L}{\delta \bar{u}} + \frac{\delta f^T}{\delta \bar{u}}\lambda(t) \tag{39}$$

$$= R\bar{u} + B^T\lambda(t) \tag{40}$$

[2]The change in state, $\dot{\bar{x}}$, as a function of state and action is not time-varying in this case, but the definition of the Hamiltonian is general enough to handle such a problem specification [25].

This "stationarity condition" and the positive definiteness of R allow us to solve for the optimal control function in terms of the Lagrange multiplier:

$$\bar{u}^* = -R^{-1}B^T\lambda(t) \tag{41}$$

We can solve for $\lambda(t)$. λ has its own dynamics, as shown in [12]'s Pontryagin's minimum priniciple-based derivation of the continuous nonlinear optimal controller, and those dynamics are simple enough for linear quadratic problems that if we know the final value of λ, we know λ for all time.

The time-varying Lagrange multiplier's dynamics will satisfy

$$-\dot{\lambda} = \frac{\delta H}{\delta x}, \quad 0 \leq t \leq T \tag{42}$$

Since this is the linear quadratic problem, we can be more specific:

$$-\dot{\lambda} = A^T\lambda(t), \quad 0 \leq t \leq T \tag{43}$$

so the closed-form solution for λ in terms of its final value is

$$\lambda(t) = e^{A^T(T-t)}\lambda(T) \tag{44}$$

However, we do not know the final value of λ. It's much easier to define boundary conditions in terms of the system's state, x. In fact, it makes intuitive sense that, since we are interested in the cost incurred by actually reaching x_{rand} (with the linearized system), not just minimizing some quadratic function of our distance from x_{rand} at the end of the trajectory, we should impose a strict final boundary value condition on x:

$$x(T) = x_{rand} \tag{45}$$

In the coordinate system we prefer:

$$\bar{x}(T) = 0 \tag{46}$$

Of course, this boundary value for $\bar{x}$ will only help us find the boundary value for λ if we know the relationship between $\bar{x}$ and λ. By substituting the optimal control solution $\bar{u}^*$, which is in terms of λ, for $\bar{u}$ in the equation governing the dynamics of $\bar{x}$, we get that necessary relation:

$$\dot{\bar{x}}(t) = A\bar{x}(t) - BR^{-1}B^T e^{A^T(T-t)}\lambda(T) + c \tag{47}$$

The relationship between the state and the Lagrange multiplier (Equ. 47) and the dynamics of the Lagrange multiplier (Equ. 42) together form what is referred to as the *Hamiltonian system*. With initial and final boundary values for the state, which are the states at the beginning and end of the trajectory, x_0 and x_{rand}, we can solve the Hamiltonian system for $\lambda(t)$.

Since the relationship between the state and Lagrange multiplier is actually between the *change* in state and the Lagrange multiplier, we must integrate Equ. 47 before evaluating the relationship at T so that we can impose our state final boundary value constraint. Integrating Equ. 47 produces

$$\bar{x}(t) = e^{A(t-t_0)}\bar{x}_0 - \int_{t_0}^{t} e^{A(t-\tau)}BR^{-1}B^T\lambda(\tau) + e^{A(t-\tau)}c d\tau \tag{48}$$

Now we can evaluate $\bar{x}$ at T and set it equal to our final boundary value on state, 0:

$$\bar{x}(T) = e^{AT}\bar{x}_0 - \int_0^T e^{A(T-\tau)}BR^{-1}B^T\lambda(\tau) + e^{A(T-\tau)}c d\tau = 0 \tag{49}$$

If we substitute in the expression for λ in terms of its final value (shown earlier in Equ. 44), we can solve for the final value of λ in terms of the dynamics of the system, the cost on action, and the starting and ending states of the trajectory. Recall that with this final value of λ we know $\lambda(t)$, $\bar{u}^*(t)$, and $\bar{x}(t)$ for the entire optimal trajectory.

$$\bar{x}(T) = e^{A(T)}\bar{x}_0 - \int_0^T e^{A(T-\tau)}BR^{-1}B^T e^{A^T(T-\tau)}\lambda(T) + e^{A(T-\tau)}c d\tau = x_{rand} \tag{50}$$

The integral amounts to a symmetric matrix, referred to as the *continuous reachability gramian*, and the final value of the Lagrange multiplier multiplied together plus a constant. If we define

$$P(t) \equiv \int_0^t e^{A(t-\tau)}BR^{-1}B^T e^{A^T(t-\tau)} \tag{51}$$

then

$$0 = e^{AT}\bar{x}_0 - P(T)\lambda(T) + \int_0^T e^{A(T-\tau)}c d\tau \tag{52}$$

$$\lambda(T) = P^{-1}(T)\left[e^{AT}\bar{x}_0 + \int_0^T e^{A(T-\tau)}c d\tau\right] \tag{53}$$

The symmetry of P allows us to invert it. For notational simplicity, we'll define the expression above in brackets as d:

$$\lambda(T) = P^{-1}(T)d(\bar{x}_0, T) \tag{54}$$

While Equ. 51 describes one way of calculating P, [12] asserts that there is a preferable method. My understanding of this alternate method was enhanced by recognizing the connection between linear quadratic regulators which have final boundary value constraints on state and linear quadratic regulators developed from cost functions whose only state error terms are from the state at the final time T. The relationship is easiest to see when the cost functions and final state constraints, when applicable, are shown side by side.

The linear quadratic regulator which has been discussed thus far has the following cost function and final state boundary value constraint:

$$J(\bar{x}_0, T) = \int_0^T \left[1 + \frac{1}{2}\bar{u}^T(t)R\bar{u}(t)\right] dt, \quad R = R^T > 0, \tag{55}$$

$$s.t.\ \bar{x}(T) = \vec{0}. \tag{56}$$

The cost function of a closely related linear quadratic regulator is the same as above, but with the additional quadratic term for final state error and with a final state constraint:

$$J(\bar{x}_0, T) = \bar{x}_f^T Q_f \bar{x}_f + \int_0^T \left[1 + \frac{1}{2}\bar{u}^T(t)R\bar{u}(t)\right] dt, \quad R = R^T > 0, \quad Q_f = Q_f^T > 0 \tag{57}$$

The optimal cost-to-go functions associated with the solutions to both control problems have the following form:

$$J^*(\bar{x}_0, T) = T + h(\bar{x}_0, T)^T S(T) h(\bar{x}_0, T), \quad h(\bar{x}_0, T) \in \Re^n, \quad S(T) \in \Re^{nxn} \tag{58}$$

When Q_f has an infinite determinant, the optimal cost-to-go functions' $S(t)$ are the same. The latter controller's infinite Q_f assigns an infinite cost to all trajectories that do not end at the final desired state, so all the optimal trajectories will also satisfy the final state constraint of the former controller.

This $S(t)$ is the solution to

$$-\dot{S} = -SBR^{-1}B^TS + SA + A^TS, \quad S(T) = Q_f \tag{59}$$

which is a matrix Riccati equation and bilinear in S. When Q_f does not have an infinite determinant, $S(t)$ for all $t < T$ can be found by integrating the dynamics of S backwards in time from $S(T)$ at T. However, when the determinant of Q_f is ∞ and integrating backwards from infinity is impractical, we instead solve for S^{-1} by integrating the dynamics of S^{-1} backwards in time from the corresponding final boundary value, $S^{-1}(T)$, which is 0. Using the matrix relation $\frac{\delta S^{-1}}{\delta t} = -S^{-1}\frac{\delta S}{\delta t}S^{-1}$ [25], the dynamics of S^{-1} is

$$\dot{S^{-1}}(t) = AS^{-1}(t) + S^{-1}(t)A^T - BR^{-1}B^T \tag{60}$$

This is a Lyapunov equation, and linear in S^{-1}. [3] The solution S^{-1} is equivalent to P^{-1}, and the method described above for calculating it is used in the code developed for this thesis, instead of the original definition of P^{-1}.

Now $\lambda(t)$ has not just been defined in terms of known (given) quantities; we have identified the preferred method of its calculation, and since $\bar{u}^*(t)$ and $\bar{x}(t)$ are defined in terms of $\lambda(t)$, they are also known in terms of originally given quantities. However, the quantity upon which the AQR-based pseudometric will be based is the cost-to-go between x_0 and x_f. By plugging $\bar{u}^*$ into the cost function, we get the following:

$$\begin{aligned}
J^*(\bar{x}, T) &= \int_0^T \left[1 + \frac{1}{2}\bar{u}^{*T}(t)R\bar{u}^*(t)\right] dt && (61)\\
&= T + \frac{1}{2}\int_0^T \left[-d^T(\bar{x}_0, T)P^{-1}(T)e^{A(T-t)}BR^{-1}\right] R \left[-R^{-1}B^Te^{A^T(T-t)}P^{-1}(T)d(\bar{x}_0, T)\right] && (62)\\
&= T + \frac{1}{2}\int_0^T d^T(\bar{x}_0, T)P^{-1}(T)\left[e^{A(T-t)}BR^{-1}B^Te^{A^T(T-t)}\right] P^{-1}(T)d(\bar{x}_0, T) && (63)\\
&= T + \frac{1}{2}d^T(\bar{x}_0, T)P^{-1}(T)d(\bar{x}_0, T) && (64)
\end{aligned}$$

Finally, since the cost of traveling along the optimal trajectory from $\bar{x}_0$ to the origin of our state coordinate system in T units of time is highly dependent on T, we search for the horizon time with the least cost

$$T^* = \text{argmin}_T J^*(\bar{x}_0, T), \quad 0 \le T \le T_{max} \tag{65}$$

$J^*(\bar{x}_0, T^*)$ is the distance when traveling from x_0 to x_{rand} according to the AQR-based pseudometric.

[3] This can be solved explicity, rather than with numerical integration [25].

3.3 Computational Efficiency

The user must choose some maximum T by considering the possibility of the lowest $J^*(\bar{x}, T)$ for a given $\bar{x}$ occurring at a longer T and the additional computation time of finding $J^*(\bar{x}, T)$ for longer T. The additional computation of considering longer T is made more efficient by observing that both P and d can be solved recursively by integrating backwards from the final conditions. As explained earlier, P^{-1} is integrated backwards in time from 0, and can be integrated backwards indefinitely to find the cost for any T. d is the sum of $e^{AT}\bar{x}$ and an integral over the entire length of the time horizon T; therefore d for a given $\bar{x}_0$ and T can be expressed as

$$d(\bar{x}_0, T) = r(T) + e^{AT}\bar{x}_0 \tag{66}$$

As $J^*(\bar{x}_0, T)$ is computed for longer and longer T, r can be updated from the r of the previous (shorter) T using the differential equation whose solution is the integral term in the original definition of d:

$$\dot{r}(t) = Ar(t) + c, r(0) = 0 \tag{67}$$

As P^{-1} integrated from its final conditions and a time horizon of 0 backwards to find P^{-1} for longer and longer T, r can simultaneously be integrated to find $r(T)$ for longer and longer T.

And while there is a distinct T^* for each $\bar{x}_0$, P^{-1} and r are valid across all $\bar{x}_0$. The computation of $J^*(\bar{x}_0, T^*)$ is done by first calculating P^{-1} and r via integration from 0 to T_{max}. Then, for each chosen increment of T, we produce a $d(\bar{x}^i, T)$ from $r(T)$ for each node, where $\bar{x}^i$ is the i^{th} node in the RRT of p nodes. A new matrix, $D(T)$, is formed; its p columns are the p node's $d(\bar{x}^i, T)$. $J^*(\bar{x}, T)$ can then be computed for all nodes simultaneously by computing $T\,[1\ 1\ \ldots\ 1] + D(T)P^{-1}(T)D(T)$. Finding the T^* for each node is trivial and efficient.

To further increase computational efficiency, we rely on the fact that the subroutine that returns the closest node with respect to this AQR-based pseudometric does not need to evaluate all the nodes in the RRT to determine its output with certainty. Since $J(\bar{x}, T)$ is a quadratic form in terms of d plus an offset, it can be broken down into a sum of squares

$$J(\bar{x}, T) = T + \frac{1}{2}d^T(\bar{x}_0, T)P^{-1}(T)d(\bar{x}_0, T) \tag{68}$$

$$= T + \frac{1}{2}\left[\lambda_1 c_1^2 + \lambda_2 c_2^2 + \ldots + \lambda_n c_n^2\right] \tag{69}$$

where c_i is the projection of $d(\bar{x}_0, T)$ onto the orthonormal axis associated with the eigenvalue λ_i. Assume λ_L is the largest eigenvalue. If $\lambda_L c_L^2$ for some node i is greater than the entire sum of squares, $\left[\lambda_1 c_1^2 + \lambda_2 c_2^2 + \ldots + \lambda_n c_n^2\right]$ for another node, j, in the tree, then the total J for node i at T is also guaranteed to be greather than the total J for node j at T.

Since the matrix multiplication involved in calculating $D(T)P^{-1}(T)D(T)$ can become a computational bottleneck when trees have many nodes (producing very wide $D(T)$), this fact about the relative costs of nodes can be used to identify which nodes' total costs cannot be the least of all the nodes of the tree *before* their d becomes another column in $D(T)$. For each T, we want to identify the node which will enable us to eliminate the greatest number of nodes from consideration, but without a lot of computation. If we limit ourselves to computing only one $\lambda_i c_i^2$ per node before deciding which node's total $J(T)$ will serve as a threshold for eliminating other nodes, then it makes sense to calculate $\lambda_L c_L^2$ at T for all p nodes, choose the node with the smallest $\lambda_L c_L^2$ at T, calculate its total $J(T)$ and call it C, and not add any nodes' d vector to $D(T)$ whose $\lambda_L c_L^2$ at T is greater than C. The total $J(T)$ of nodes whose d does become a column of $D(T)$ will be calculated. Note that whether or not a node's total $J(T)$ is calculated for some time horizon T_1 does not affect whether or not its total $J(T)$ is calculated for a different time horizon T_2. Overall, this reduces the time to find the node in the tree with the least distance to x_{rand} ($J^*(T^*)$) with respect to the AQR-based psuedometric.

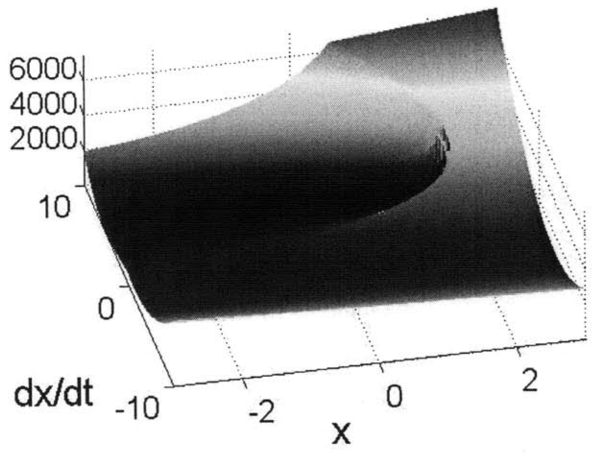
3.4 The Shape of the AQR and AQR-based Pseudometric Cost-to-Go Functions

When the linearization point is a fixed point, the cost-to-go, as a function of state, for that linearized system under control of the AQR will be quadratic bowl.[4] The same shape of the cost-to-go function results from the infinite horizon LQR solution.

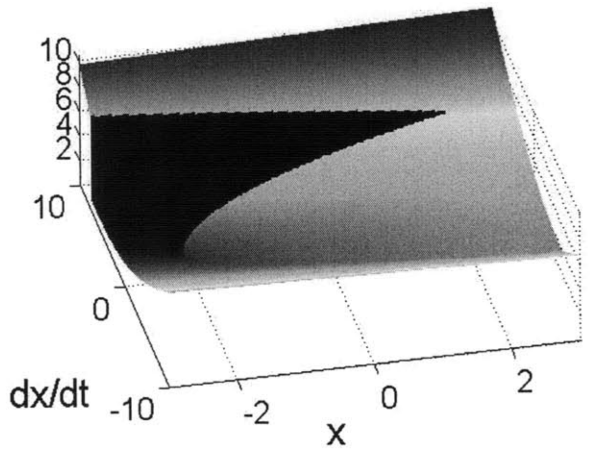
For the case of a linearization point (x_{rand}, u_f) that is not a fixed point, the c term of the linearized dynamics shifts the location of the center of the AQR's quadratic cost-to-go function bowl, as can be seen by examining Equ. 69. The minimum-cost state, which is the center of the quadratic bowl of cost, is the state in which the passive dynamics of the linearized system will bring the system back to x_{rand} in exactly T units of time. As T is changed, the principal axes, eigenvalues, center location, and magnitude offset of $J(\bar{x}, T)$ change.

Finding $J^*(\bar{x}, T^*)$ for each $\bar{x}$ enables the AQR-based pseudometric to approximate the discontinuities in cost-to-go to a nonfixed point that exists for exact minimum-time pseudometrics, even though the AQR-based pseudometric is composed entirely of a state by state minimization over T of magnitude-offset quadratic bowls.

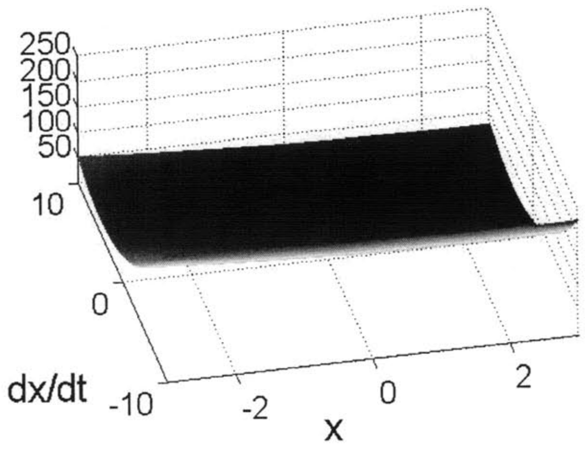
[4]AQR and finite-horizon LQR are equivalent due to the linearization point being a fixed point.



(a) AQR-Based Proximity



(b) Exact Min-Time



(c) Euclidean Distance

Figure 5: Maps of proximity to a sample point, [2,5].

3.5 Justifications for the Parameter Choices of the AQR-based Pseudometric

The AQR-based pseudometric has a hard final constraint on state, equivalent to a Q_f with an infinite determinant, and no state error term $\bar{x}^T Q \bar{x}$ within the cost J's integral, equivalent to a Q with a determinant of zero in a closed-loop LQR. The quickest paths are not necessarily also the closest in terms of the weighted squared Euclidean distance that the incremental state error term $\bar{x}^T Q \bar{x}$ accumulates; this is an argument for not including that incremental state error term. However, if the term is included, the RRT will be biased toward expanding nodes whose optimal trajectories spend greater amounts of time near the sample goal x_{rand} in terms of the weighted squared Euclidean distance, which might subsequently bias the RRT toward expanding nodes which themselves are closer to x_{rand} in terms of the weighted squared Euclidean distance. This injects the influence of the Euclidean distance metric that we earlier rejected, but also biases the RRT toward expanding nodes for which the linearized dynamics (and therefore the cost-to-go), which are only locally accurate, are more accurate. Ultimately, Q was chosen to be a zero matrix; the arguments for and against this choice are both reasonable, so the tie-breaker was that eliminating the incremental state error term reduces the number of parameters to tune.

Since the exact minimum-time proximity to x_{rand} can have discontinuities, such as cliffs, that run through x_{rand} and assign states of equal final state error $\bar{x}^T Q_f \bar{x}$ to significantly different distances to x_{rand}, it makes sense not to again minimize some weighted squared Euclidean distance when it is possible to enforce that at least the linearized system arrives not near but at x_{rand}.

R remains as the sole tunable parameter, and based on personal experience, the accuracy of the minimum-time pseuodmetric approximation is increased when the magnitude of the elements of a diagonalized R are proportional to the severity of the bounds set on the input. LQR and AQR cannot take explicit input bounds into account; penalizing high-magnitude actions is the closest related possible option.

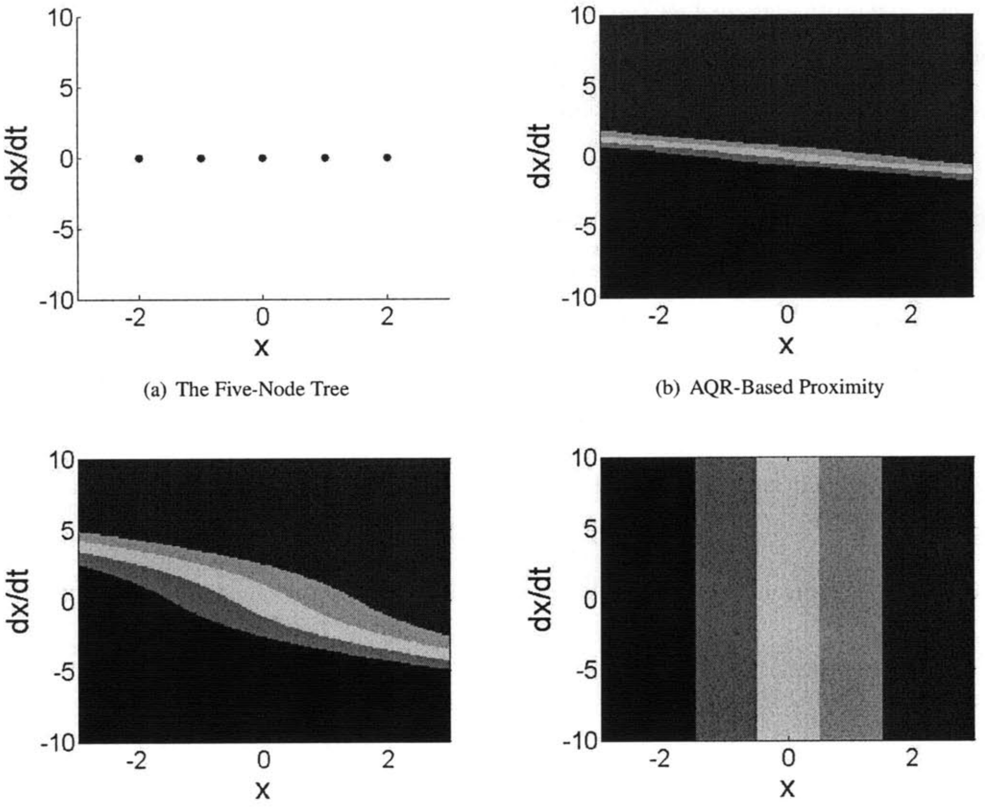
4 INTERPRETING VORONOI DIAGRAMS AND PROXIMITY MAPS

Every new sampled point (x_{rand}) is mapped back to the nearest node to it in the RRT (x_{near}) so that x_{near} can be expanded towards x_{rand}. Since the sampled space is sampled uniformly with respect to the Euclidean distance, the probability of expanding node i in the RRT is proportional to the Euclidean area of the Voronoi region of node i, where the Voronoi region contains all points closer to node i than any other RRT node. This is referred to as the Voronoi bias, the bias RRTs have towards exploring places not yet visited. Regions on the frontier of the tree and regions where little exploration has occurred contain the fewest nodes; the farther apart the nodes are in a region, the larger their Voronoi regions and the more likely they are to be nearest of all RRT nodes to x_{rand} and expanded.

While the samples of space (x_{rand}) are spread uniformly through the sampled space with respect to the Euclidean distance, the NEAREST_NEIGHBOR function is free to use any measure of distance. Each measure of distance defines its own set of Voronoi regions. By determining the Voronoi regions, the distance metric also determines what regions of the sampled space are least explored, and which children of an expanded x_{near} have brought the RRT closer to that less explored space.

The classic demonstration of RRTs' effectiveness for exploring spaces comes from running the RRT algorithm on a holonomic system and using a NEAREST_NEIGHBOR function that returns the node with the least Euclidean distance. LaValle's early RRT papers show this classic example, with the Voronoi regions overlaid. This thesis focuses on defining and implementing a NEAREST_NEIGHBOR function that, when used with the standard RRT algorithm shown in Table 1, produces trees on nonholonomic systems that most faithfully reproduce the coverage of trees produced by RRTs with Euclidean-distance-based NEAREST_NEIGHBOR functions running on holonomic systems. Half of this chapter looks at this goal from the perspective of Voronoi biases imposed by various definitions of the NEAREST_NEIGHBOR function; the other half looks at this goal from the the perspective of gradients of the various (pseudo) distance metrics the NEAREST_NEIGHBOR function can use.

In Fig. 6(d), it is possible to see that, regardless of the velocity of a state with the same position as node i in the RRT, which in this toy RRT are all stationary, the Euclidean distance metric identifies node i as the closest of the RRT's nodes. Expressed in another way, when the NEAREST_NEIGHBOR function relies on Euclidean distance, the RRT algorithm will try to expand a stationary node towards any state with the same position, regardless of its velocity. Such a path from a stationary to high-velocity state at the same position requires moving far enough away from that position so that it can move back toward the end position while building up to the end velocity. This corresponds to a spiral in the phase space, which can be visualized directly overtop of the Voronoi diagrams. The more restrictive the bounds on the force that can be applied to the brick, the wider the narrowest feasible trajectory (which happens to be



(a) The Five-Node Tree

(b) AQR-Based Proximity

(c) Exact Min-Time

(d) Euclidean Distance

Figure 6: Voronoi Diagrams for a 5-node tree on the force-limited brick system.

generatable by a bang-bang minimum-time controller) will be in phase space.

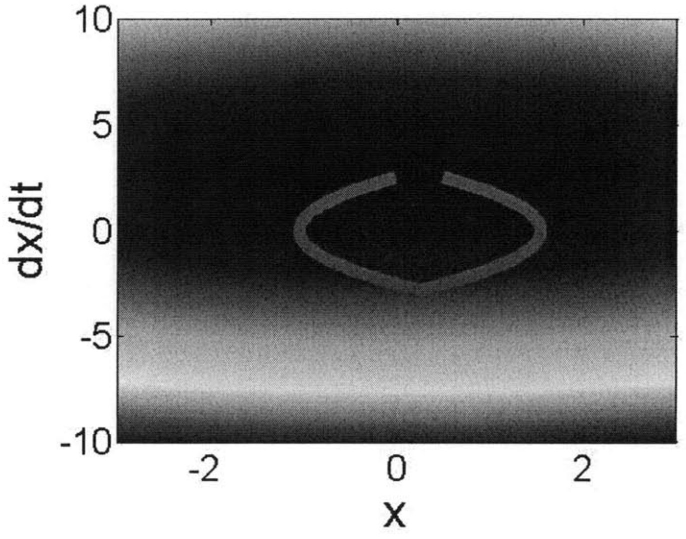
Note that points along any feasible path between the stationary and high-velocity end states (x_{rand}) may be farther in Euclidean distance from the end state than the stationary starting node, as shown in Fig. 7. In contrast, by definition, if the cost is trajectory length in time, the exact optimal cost-to-go function evaluated along the minimum-time path decreases monotonically. We can confirm this visually in Fig. 8.

While variants of the RRT algorithm exist, the version used in this thesis (Table 1) does not add that whole path to the tree at once; the feasible paths are unknown. Instead, the child(ren) of the nearest node in the RRT (x_{near}) are a small incremental step away from their parent, and the child that is farthest down the gradient of the same (psuedo) distance metric used by NEAREST_NEIGHBOR, i.e., closer to x_{rand}, is added to the tree.

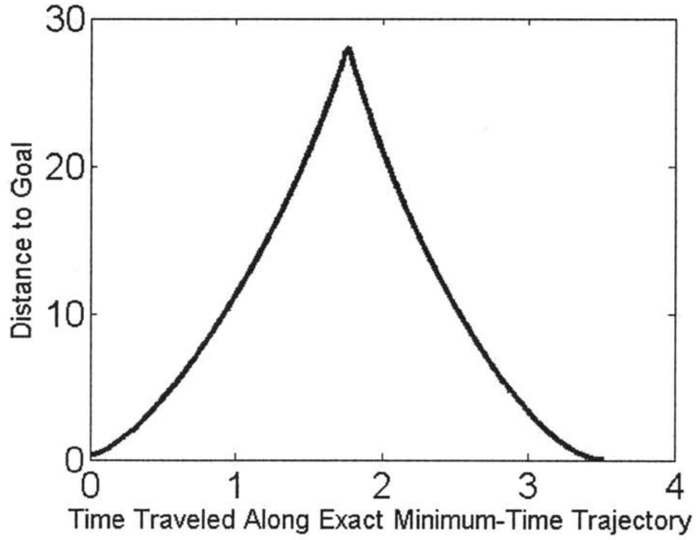
When the (pseudo) distance metric evaluated along the feasible paths from x_{near} to x_{rand} is not monotonically decreasing, this simple method for adding nodes will be inefficient at best and ineffectual at worst. (This is presumably why so many papers and theses have proposed algorithmic modifications to the RRT.) Fig. 9 shows how the AQR-based proximity function approximates the exact solution for this system's dynamics and constraints, so that its gradient along the path more closely resembles that of the monotonically decreasing exact solution than that of the Euclidean distance. The sharp increase in cost at the end of the trajectory, when the system is extremely close to the goal, is caused by poorly conditioned matrices at very short finite horizons of the AQR-based proximity calculation. This is not expected to have any significant effect on RRT growth.

In the Voronoi diagrams, we cannot see *how close* states are to each other, or whether or not the proximity to an end state would decrease monotonically along a feasible path, but we can see which node of an RRT will be considered closest to, and therfore expanded toward, any x_{rand} and how well that corresponds to what node of the RRT from which it is actually least costly to reach x_{rand}.

This can be made more concrete by examining Fig. 6(c). It shows, by the color of each point, exactly which node in the RRT is closest, with respect to the minimum-time cost. By definition, the optimal minimum-time path will not pass out of the Voronoi region it starts in, and will end at the RRT node associated with the region. The Voronoi region of the center stationary RRT node includes the stationary states whose positions are closer to it than to the two stationary RRT nodes that flank it to the left and right. It also includes the states with non-zero velocities from which it is possible, with enough force exerted against the direction of motion, to reach zero velocity at the center stationary RRT nodes' position. The same is true about the Voronoi regions for the stationary RRT nodes to either side of the center RRT node, which are also flanked on the left and right by stationary RRT nodes. The Voronoi regions of the left- and right-most RRT nodes are much larger. Those nodes are, as indicated by the Voronoi bias imposed by the exact cost-to-go pseudometric, on the frontier of the tree, since the probability of one of these nodes being x_{near} for the

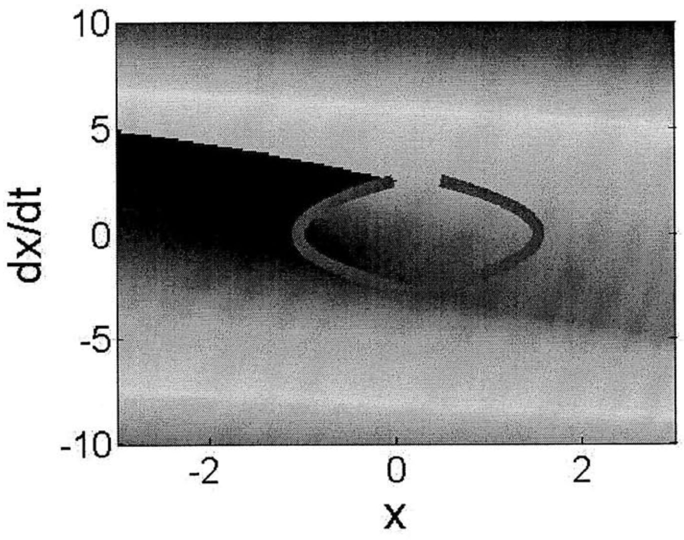


(a) The Trajectory in Phase Space Superimposed on the Map of Euclidean Distance to x_{rand}

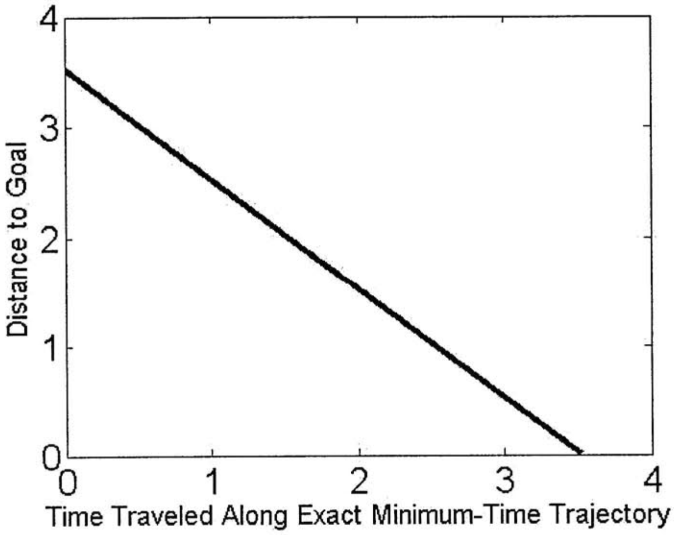


(b) Euclidean Distance Along the Trajectory

Figure 7: Visualization of the non-monotonically decreasing Euclidean distance to an x_{rand} along a feasible (in this case, time-optimal) path on the force-limited brick system.

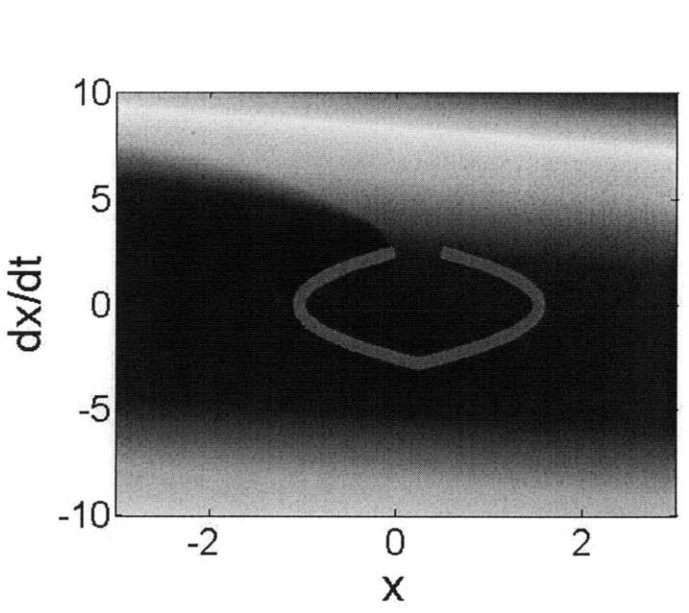


(a) The Trajectory in Phase Space Superimposed on the Map of Exact Time-to-Go to x_{rand}

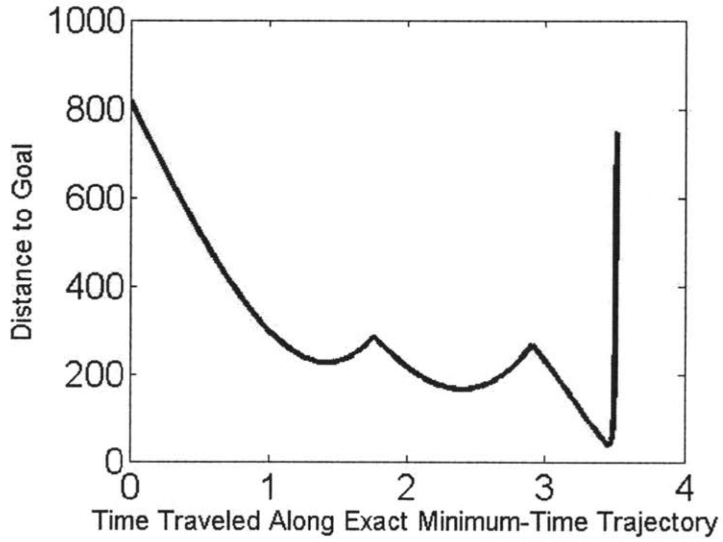


(b) Exact Time-to-Go Along the Trajectory

Figure 8: Visualization of the monotonically decreasing exact time-to-go to an x_{rand} along the time-optimal path on the force-limited brick system.



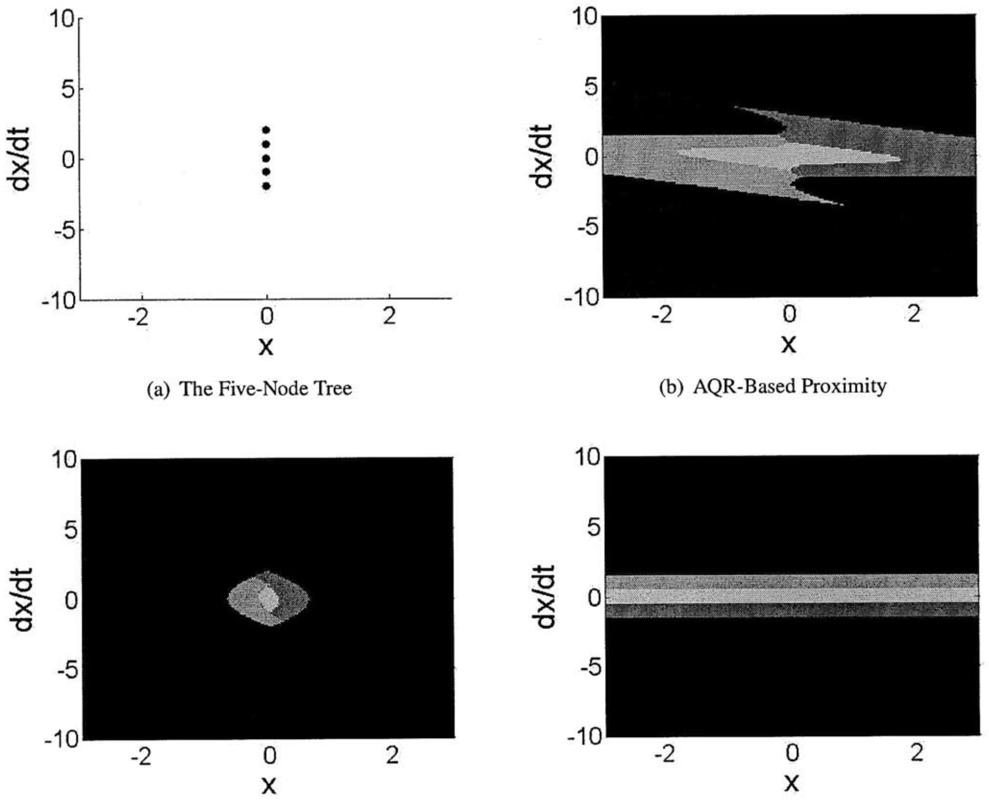
(a) The Trajectory in Phase Space Superimposed on the Map of AQR-based Proximity to x_{rand}



(b) AQR-based Proximity Along the Trajectory

Figure 9: Visualization of the AQR-based Proximity to an x_{rand} along the time-optimal path on the force-limited brick system. The sharp increase in cost at the end of the trajectory, when the system is extremely close to the goal, is caused by poorly conditioned matrices at very short finite horizons of the AQR-based proximity calculation. This is not expected to have any significant effect on RRT growth.

next x_{rand} is, as explained earlier, proportional to the size of their (large) Voronoi regions. The quickest way to reach those outer edges of the sampled space is from the left- and right-most RRT nodes, so the exact cost-to-go distance pseudometric appears to fulfill the earlier stated goal of reproducing the tendency of Euclidean-distance-guided RRTs running on holonomic systems to explore places not yet visited. As can be seen in Figs. 6(b) and 6(d), the Voronoi regions built from the AQR-based proximity function approximate those regions built from the exact minimum-time cost-to-go pseudometric far better than those built from the Euclidean distance. Figs. 10 and 11 show that this is true for a variety of (toy) sets of tree nodes.



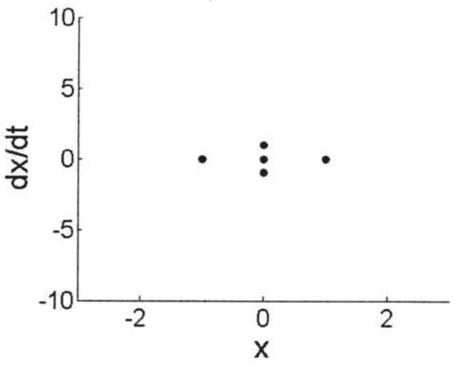
(a) The Five-Node Tree

(b) AQR-Based Proximity

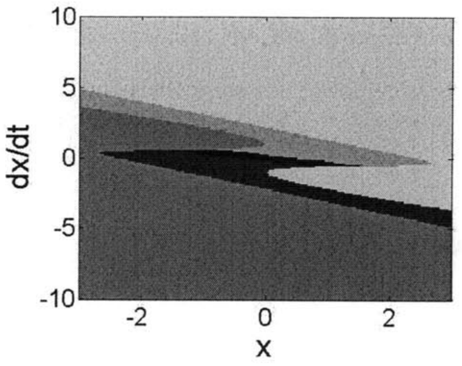
(c) Exact Minimum-Time

(d) Euclidean Distance

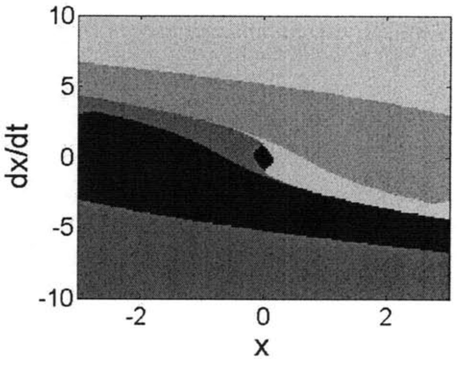
Figure 10: Voronoi Diagrams for a 5-node tree on the force-limited brick system.



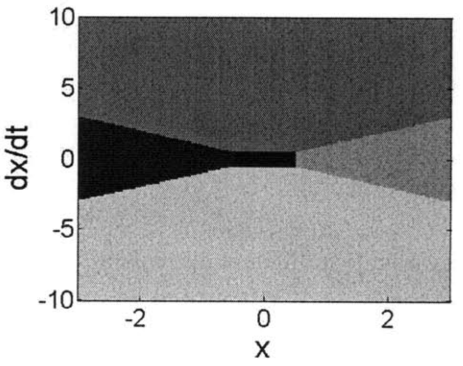
(a) The Five-Node Tree



(b) AQR-Based Proximity



(c) Exact Minimum-Time



(d) Euclidean Distance

Figure 11: Voronoi Diagrams for a 5-node tree on the force-limited brick system.

5 EXPERIMENTS

Since we developed our pseudometric with a specific, measurable performance index in mind, i.e., state space coverage, the design of experiments was straightforward. Note that [21] also used this technique to quantitatively compare planning algorithms' effectiveness.

For each dynamic system, we build two RRTs. The parameters of the system and the RRT-building algorithm are held constant, except for the distance pseudometric, so that any resulting differences in state space coverage between the trees can be attributed solely to the choice of psuedometric. The two trees do not receive identical random x_{rand} to grow towards, and due to this inherent randomness, the reported results are the averages of repeated trials. While RRTs can be biased to grow towards a particular point, these RRTs have no goal-bias, since we are interested in creating trees whose branches reach out into all regions of the sampled state space.

5.1 Measuring Coverage

The RRT algorithm attempts to grow the tree towards random samples (x_{rand}) which are taken from some finite-volume subset of the infinitely large phase space. This sampled space was divided into bins. The percentage of populated bins is our measure of coverage. For two-dimensional state spaces, a 10x10 grid of identically sized bins was used. For four-dimensional state spaces, a 6x6x6x6 grid of identically sized bins was used. (That amounts to 1296 bins in total to cover the space.)

5.2 The Dynamic Systems

RRTs were grown on four different dynamic systems: the brick, pendulum, cartpole, and acrobot. All four systems have bounds on the force/torque that can be applied. The brick moves along a single dimension without friction; it is equivalent to a double integrator. The pendulum is a point-mass on a massless rod attached to an actuated pivot point, and is also undamped. Trees are grown from a root node where the pendulum is at its stable equilibrium. The cartpole is the same classic system that control textbooks address. It is a pendulum attached to a brick, with no actuation at the pivot point of the pendulum. The entire system is driven by forces applied to the brick. Trees were also grown from a root node where the pendulum is at its stable equilibrium. The acrobot is a double (two-link) pendulum with actuation only at the joint between the two links, and trees were grown from a root node where the system is at its stable equilibrium. The green "X" indicates the location of the tree root (x_{init}), and the axes of the plot are set such that only the sampled region of state space (the region from which x_{rand} is uniformly, randomly drawn) is visible.

5.3 Determining Parameter Settings

The RRT algorithm's, dynamic systems' and the psuedometrics' settings may all have a significant effect on the results. For example, it is possible that large sections of the sampled state space will not be reachable without first traveling through unsampled regions of space, which the RRT is unlikely to do, given that, by definition, no x_{rand} sample goals will be in those regions. All 2π radians of each joint angle were sampled, and the sample intervals for linear displacements and both linear and angular velocities were arbitrarily set.

In the Euclidean distance metric, the squared differences between two states along each axis are equally weighted. Theoretically, scaling factors could be introduced and optimized based on empirical data, but this is not customary in the literature.

The AQR-based psuedometric has two parameters: the maximum considered horizon for the finite-time AQR optimal control problem and R, the penalty factor for applying force/torque in the AQR cost function. The pseudometric-defined distance between x_{rand} and nodes in the existing tree is calculated once per node added to the tree, and the maximum horizon can be set large enough to make the repeated computations of the psuedometric impractical. However, setting the maximum horizon to a very short length of time will inflate the cost of states that are, for example, (1) intuitively quite close to the sample goal, x_{rand}, because the system dynamics will take the system at that state straight to it and (2) only able to reach the sample goal with a trajectory longer than the maximum considered horizon. For systems with nonlinear dynamics, a shorter maximum horizon may actually lead to improved RRT coverage, since considering longer trajectories may bias the system toward expanding nodes that are farther away from the linearization point, x_{rand}, and the further the nodes are from the linearization point, the less accurate the pseudometric's estimate of true cost (distance) will be. For these experiments, the maximum considered finite horizon length was arbitrarily set to 5 seconds. However, R, the penalty factor for applying force/torque, was varied in order to find the value which produced the greatest coverage.

The dynamic systems' parameters are the masses and lengths of the various virtual components and the force/torque bounds. The masses and lengths were left unchanged. The force/torque bounds were set to be relatively unconstraining. For example, the bounds for the pendulum were set such that there was enough available torque to push the pendulum from its stable equilibrium (hanging downward) to its unstable equilibrium (inverted) directly, without needing to swing it back and forth first while pumping energy into it. The AQR-based pseudometric does not reason about force/torque bounds, so it was hypothesized that if most or all of state space could be reached without forces/torques exceeding the bounds, the pseduometric would be a better approximation of the true minimum-time pseudometric.

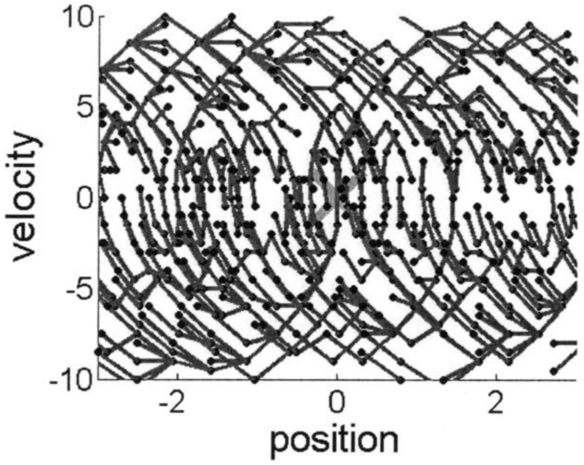
The next step was to build RRTs with various R values, differing by orders of magnitude (1, 0.1, 0.01...), in order to find an R at which coverage was at a local maximum. This was done for each system independently, so that the R setting could be system-specific. I then checked whether or not the trajectories for the linearized system, created by the AQR-based pseudometric but not used in the RRT in these experiments, violated the (possibly nonlinear) system's force/torque bounds. This was done to further understand the relationship between the AQR-based pseudometric's parameters and RRT coverage, and had no influence on the R used in final tests, which was the R that maximized coverage in the previous step. Finally, RRTs, with system-specifc R values, were grown on each system, multiple times with each distance pseudometric, so that the average coverage of the RRTs could be compared as a function of pseudometric.

Sometimes it was clear from watching the growing RRTs that coverage was not being measured at an appropriate point during tree growth. For example, for the simple pendulum example in the results that follow, the state space can be almost completely covered by a 1000-node tree, regardless of the metric used. How well the RRT reaches out and explores state space while using a particular distance psuedometric is, at least for that system, better assessed by measuring the coverage of the RRTs earlier in the growth process, when they have fewer nodes. Likewise, for the more dynamically complicated, high-dimensional systems like the acrobot, the difference in coverage between RRTs using different distance psuedometrics may not be apparent until there are many more nodes in the trees so that the RRTs that, given their distance psuedometric, *are* able to steer themselves to new unexplored regions of state space have the opportunity to do so.

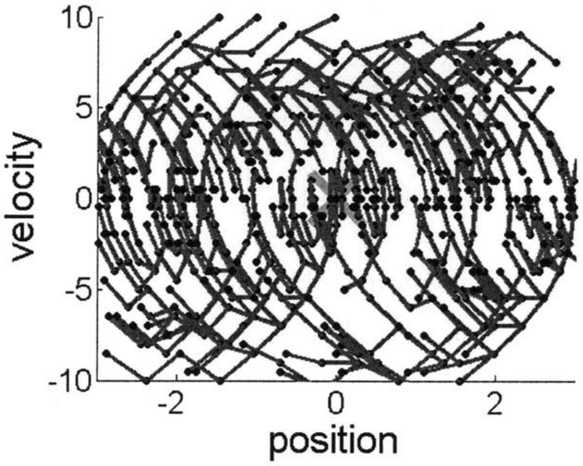
5.4 Brick

The brick system is unique among the four dynamic systems considered, and for two reasons. First, ignoring the bounds on the force that can be applied, its dynamics are linear. There is no need for linearization in order to apply the AQR-based pseudometric, which eliminates one source of error. Second, the true minimum-time trajectory between any two points is known; the corresponding minimum-time pseudometric returns the length, in time, of the node that can reach a given state in the least amount of time. The AQR-based pseudometric is intended to approximate the true minimum-time pseudometric. We have already compared maps of the distances they assign to a mesh of points around a given goal. In this chapter, we can see how well the AQR-based pseudometric approximates the true minimum-time pseudometric in terms of state space coverage.

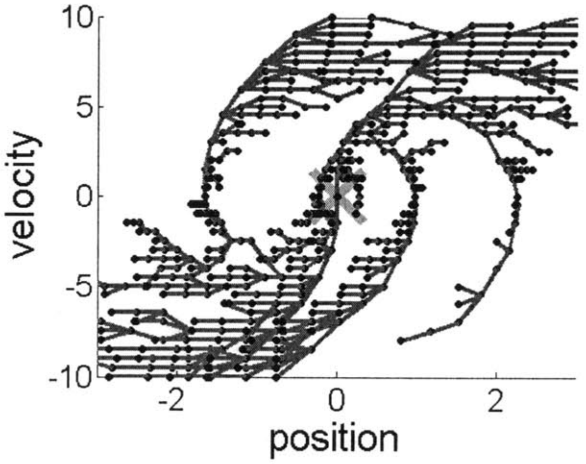
The axes of the figures in Figure 12 are set such that only sampled space from which x_{rand} is randomly uniformly drawn is visible, and it is perhaps most readily apparent that the exact minimum-time pseudometric leads to a very uniform coverage of the entire sampled space. We can also see that coverage of the RRT using the AQR-based approximation of the minimum-time pseudometric reaches almost as much of the sampled space. Finally, we can see



(a) Using the Exact Minimum-Time Pseudometric



(b) Using AQR-Based Proximity



(c) Using Euclidean Distance

Figure 12: Examples of 1000-node RRTs grown on an undamped brick (double integrator) with bounds on the forces applied to the system.

that the Euclidean-based RRT is not able to explore the upper left and lower right quadrants of state space as well as the RRTs using the other pseudometrics. This can be explained by the fact that many of the x_{rand} in the upper left and lower right quadrants were closest in Euclidean distance to the branches representing states where the system is constrained by its dynamics to continue moving away, not toward, that x_{rand}.

5.5 Pendulum

In Figure 14, it is clear that the RRT with the AQR-based pseudometric is able to reach a greater percentage of the state space than the RRT with the Euclidean metric. This comparison holds true over repeated trials, as shown in Figure 15.

5.6 Cartpole

In Figure 16, there is no clear difference between the coverage of the RRTs using the two different distance pseudometrics. Figure 17 confirms this lack of differentiation.

Tripling the size of the trees does not change the relative performance of the RRTs using these two pseudometrics.

5.7 Acrobot

In Figure 18, there appears to be some advantage to using the AQR-based distance psuedometric, in terms of coverage. However, it is not possible to know this from the figure because information is lost when the four-dimensional space is projected onto two two-dimensional graphs. Figure 19 shows that on average, there is no significant difference in coverage.

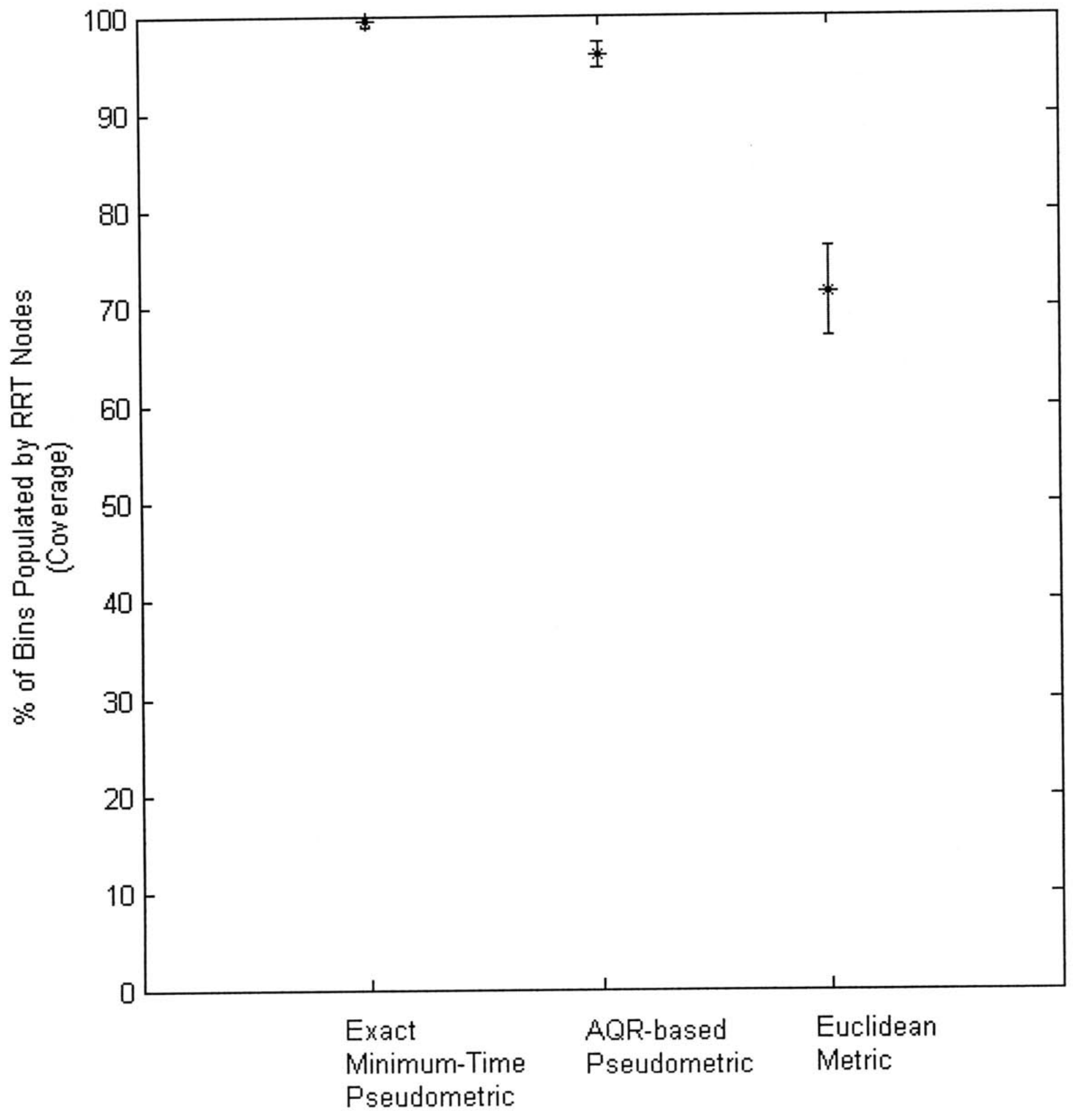
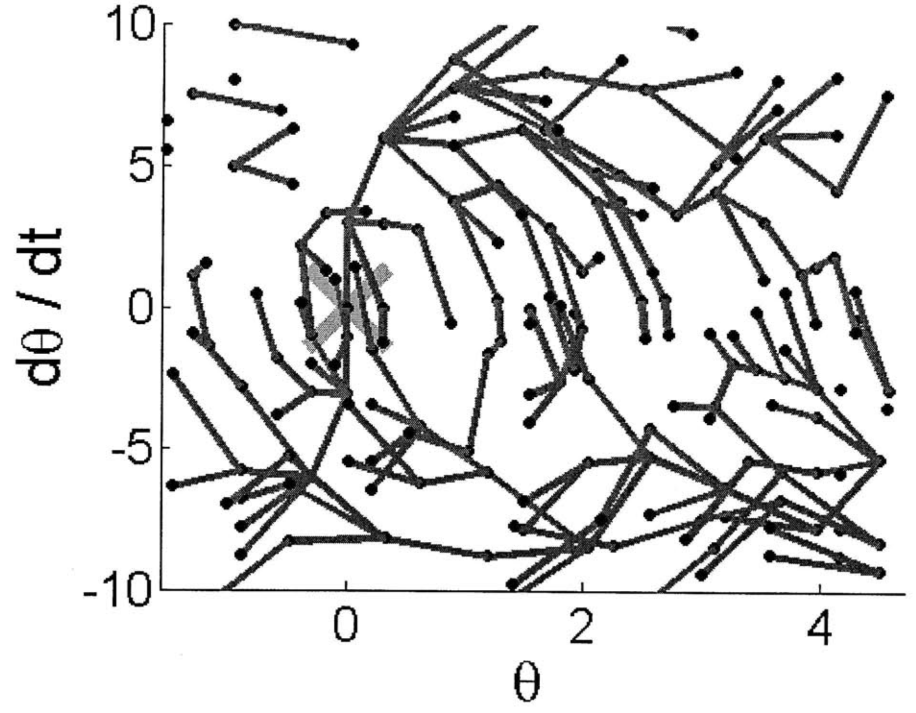
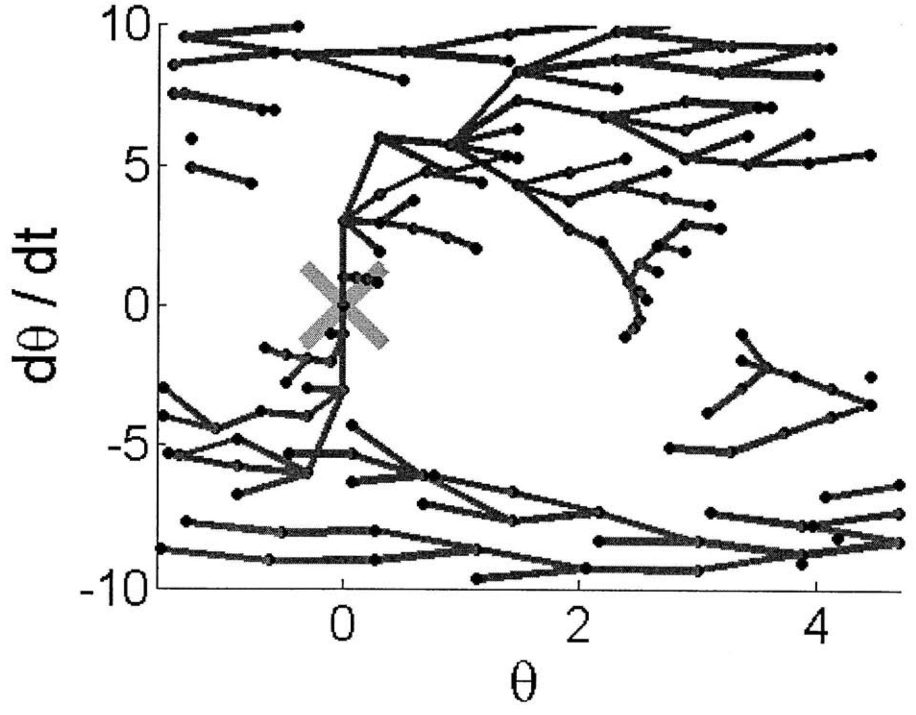


Figure 13: Comparative coverage of 1000-node RRTs' exploration of the brick's state space. Fifty RRTs were grown for each distance pseudometric, and the mean and standard deviation of those RRTs' coverage is shown.



(a) Using AQR-Based Proximity



(b) Using Euclidean Distance

Figure 14: 200-node RRTs grown on a torque-limited pendulum.

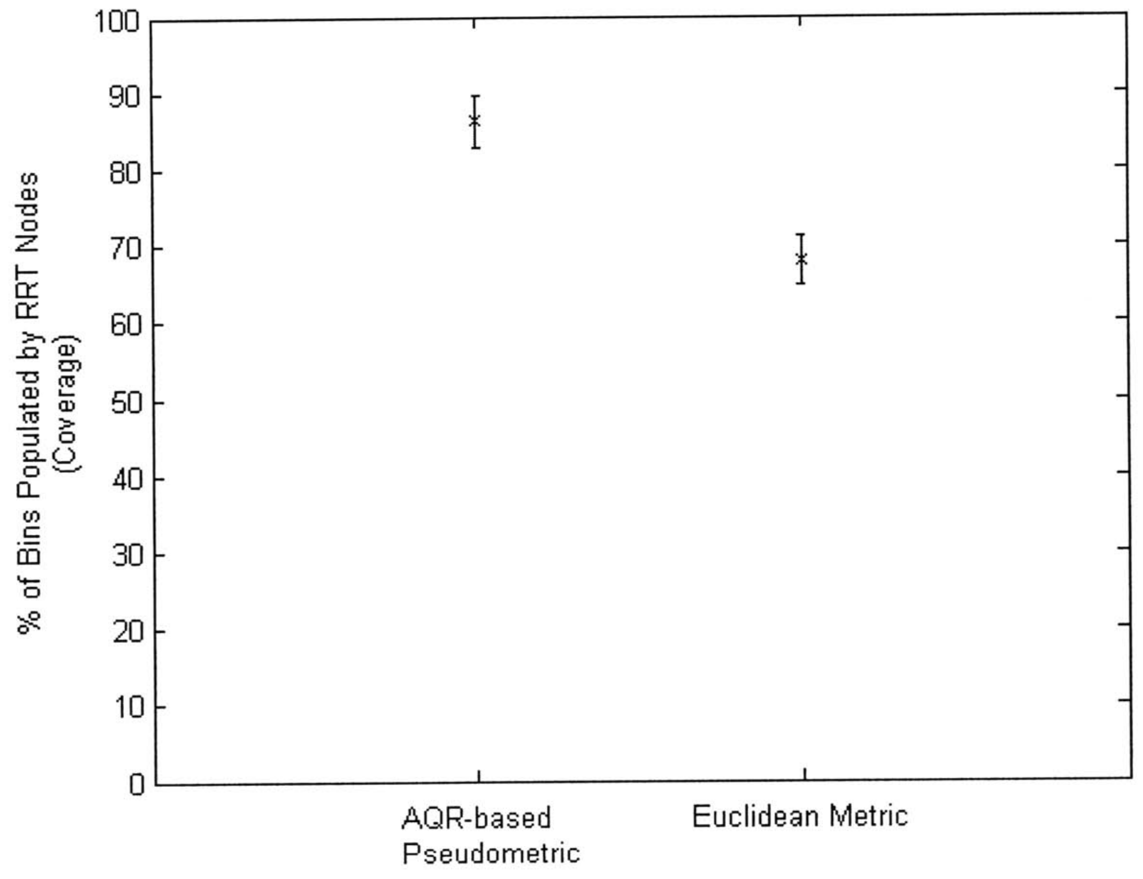
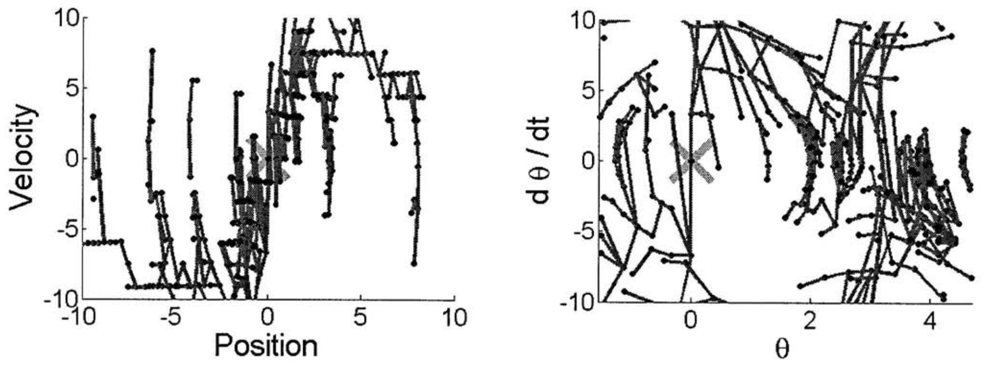
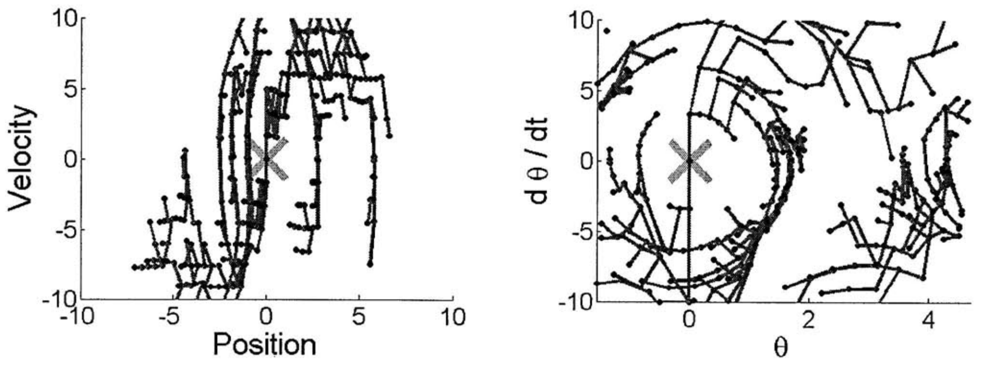


Figure 15: Comparative coverage of 200-node RRTs' exploration of the pendulum's state space. Fifty RRTs were grown for each distance pseudometric, and the mean and standard deviation of those RRTs' coverage is shown.



(a) Using AQR-Based Proximity



(c) Using Euclidean Distance

Figure 16: 500-node RRTs grown on a force-limited cartpole.

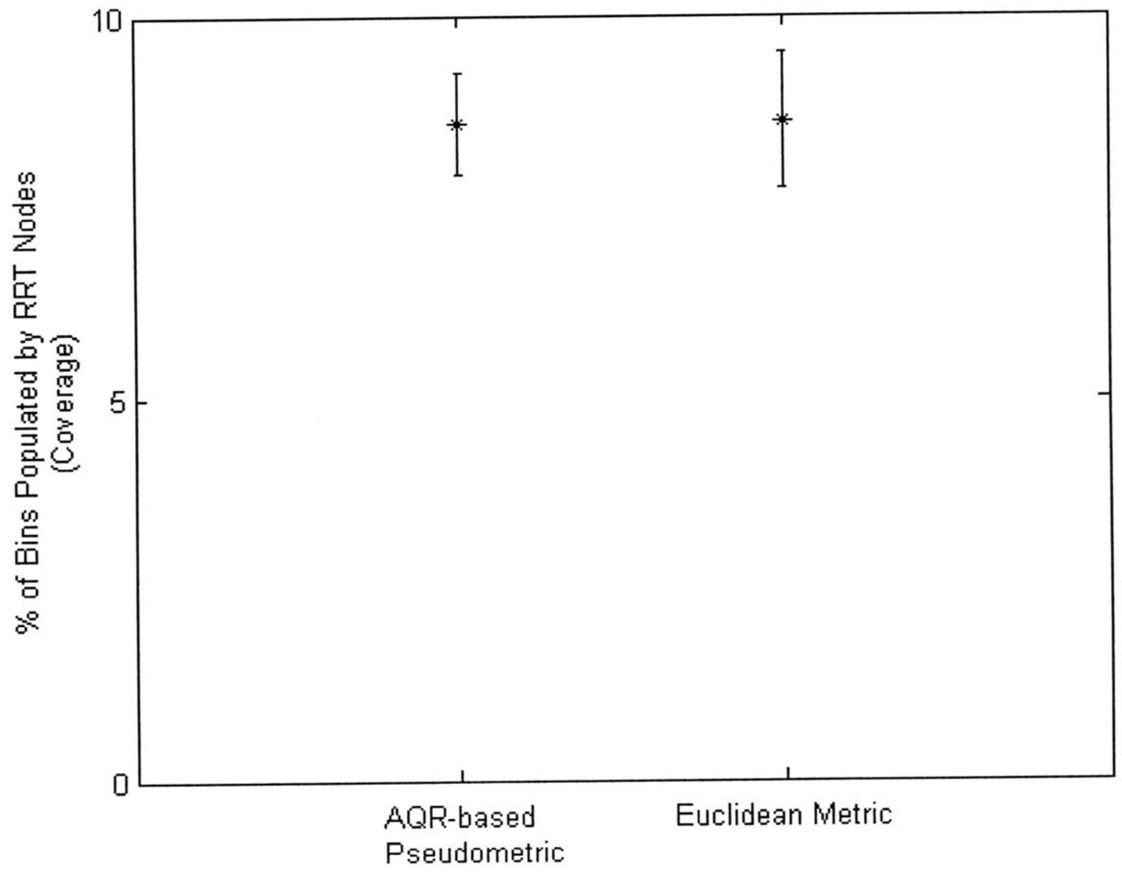
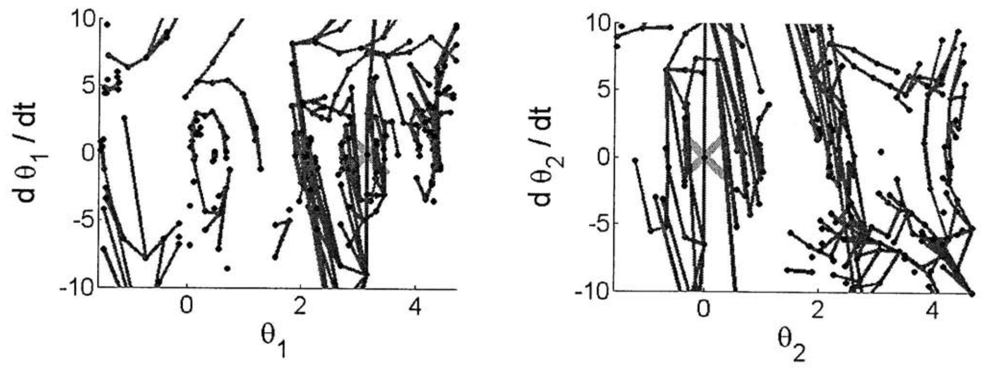
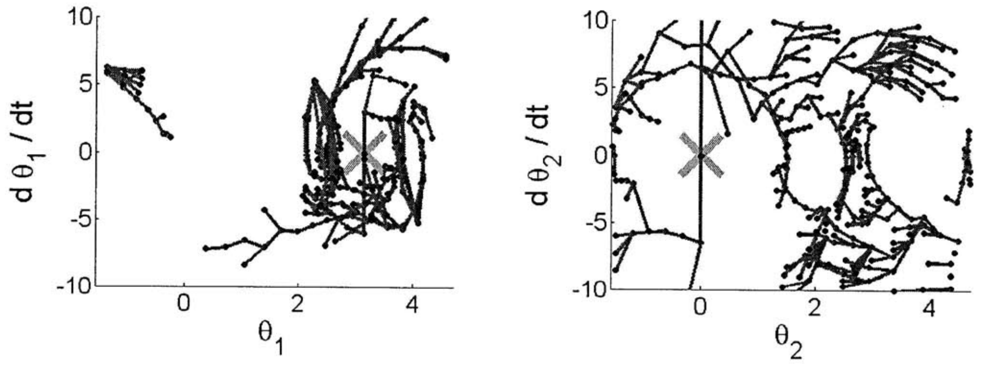


Figure 17: Comparative coverage of 500-node RRTs' exploration of the cartpole's state space. Ten RRTs were grown for each distance pseudometric, and the mean and standard deviation of those RRTs' coverage is shown.



(a) Using AQR-Based Proximity



(c) Using Euclidean Distance

Figure 18: 500-node RRTs grown on a torque-limited acrobot.

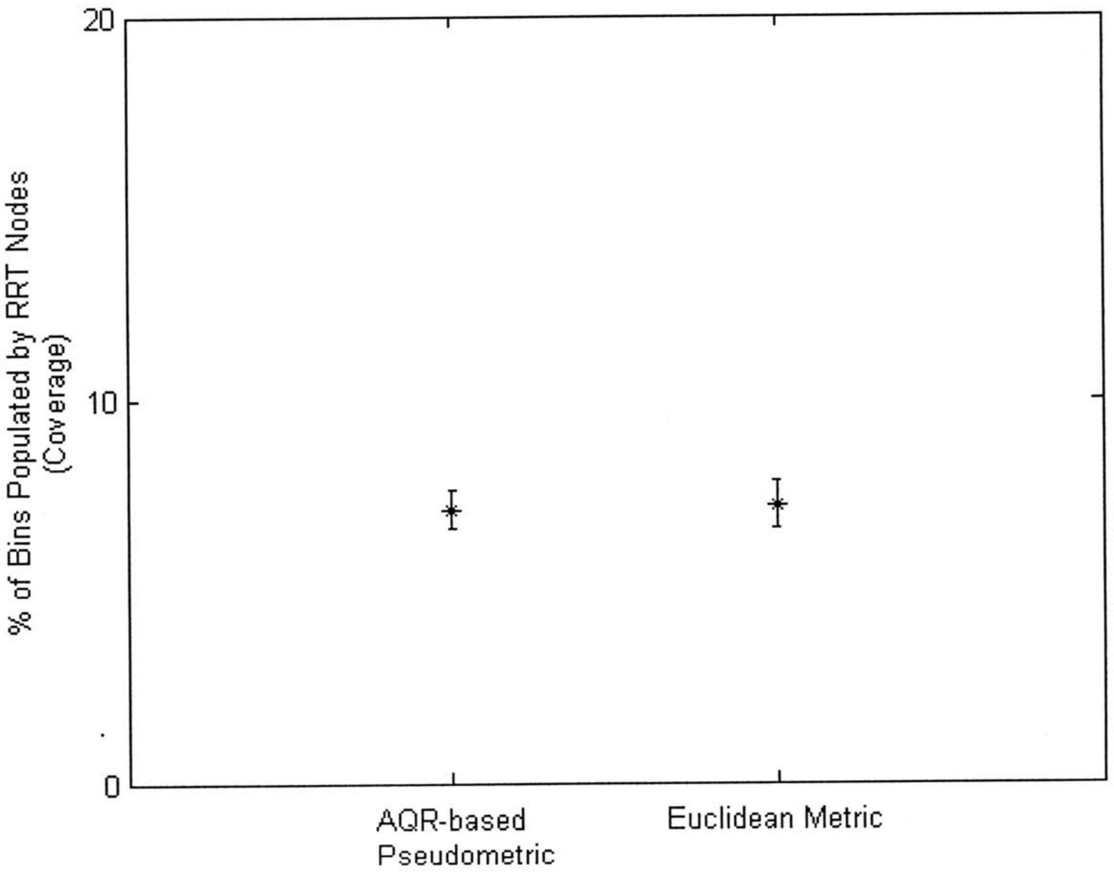


Figure 19: Comparative coverage of 500-node RRTs' exploration of the acrobot's state space. Ten RRTs were grown for each distance pseudometric, and the mean and standard deviation of those RRTs' coverage is shown.

6 DISCUSSION

6.1 Trends Across Systems

It is impossible to make sweeping statements about the value of the AQR-based pseudometric based on just four dynamic systems, but trends can be discussed, and perhaps verified in future work. There appears to be a negative correlation between the complexity/nonlinearity of a system's dynamics and the benefit of using the AQR-based distance pseudometric over the Euclidean distance. This makes intuitive sense since, for systems with more complex, nonlinear dynamics, the accuracy of the cost-to-go (distance) estimates of the AQR-based pseudometric will degrade faster as a function of distance to the linearization point.

There are several other factors that may have also made coverage results on the higher-dimensional, more complex, nonlinear systems less differentiable. Perhaps it is necessary to sample a larger region of state space because regions of the currently sampled state space are unreachable without first branching out into currently unsampled state space. It may also be necessary to optimize the maximum considered finite horizon time with respect to coverage.

A more subtle trend, which may or may not be a coincidence, is that the coverage of any given RRT grown using the AQR-based pseudometric may be less variable than when using the Euclidean metric. This is suggested by the fact that the standard deviation of coverage across repeated trials was less, for three out of the four dynamic systems, when using the AQR-based pseudometric compared to the Euclidean metric. (For the fourth dynamic system, the standard deviations were about the same.)

6.2 Future Work

Since we observe a drop-off in benefit as system complexity and nonlinearity increase, future work could include exploring methods for approximating the exact minimum-time proximity pseudometric which can reason about dynamics with higher-order terms.

The results presented in this thesis are focused solely on quantifying the impact of the AQR-based pseudometric on RRTs' coverage of state space. However, there may be other advantages to using this pseudometric as well. AQR's cost function allows for the user to bias the RRT towards solutions which require low input energy. Growing an RRT with this bias may not lead to the greatest coverage, but may in of itself be of interest to the research community.

References

[1] S. LaValle. Rapidly-exploring random trees: A new tool for path planning. Technical Report 98-11, Iowa State University, Dept. of Computer Science, 1998.

[2] LaValle, S.M., Kuffner, J.J., and Jr. Randomized kinodynamic planning. *Proc. of the IEEE Int. Conf. on Robotics and Automation*, 1:473-479, 1999.

[3] S. LaValle and J. Kuffner. Rapidly-exploring random trees: Progress and prospects. In *Proceedings of the Workshop on the Algorithmic Foundations of Robotics*, 2000.

[4] Steven M. LaValle, James J. Kuffner, and Jr. Randomized kinodynamic planning. *The International Journal of Robotics Research*, 20(5):378-400, 2001.

[5] Steven M. LaValle. Planning Algorithms. Cambridge University Press, 2006.

[6] Yershova, A., Jaillet, L., Simeon, T., LaValle, and S.M. Dynamic-domain rrts: Efficient exploration by controlling the sampling domain. *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 3856-3861, April 2005.

[7] Alexander Shkolnik, Matthew Walter, and Russ Tedrake. Reachability-guided sampling for planning under differential constraints. In Proceedings of the International Conference on Robotics and Automation (ICRA), pages 2859-2865. IEEE/RAS, 2009.

[8] J. H. Reif. Complexity of the movers problem and generalizations. In *Proceedings IEEE Symposium on Foundations of Computer Science*, pages 421427, 1979.

[9] Steven M. Lavalle. From dynamic programming to rrts: Algorithmic design of feasible trajectories. In *Control Problems in Robotics*. Springer-Verlag, 2002.

[10] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control.* Athena Scientific, 2nd edition, 2000.

[11] Peng Cheng, LaValle, and S.M. Reducing metric sensitivity in randomized trajectory design. *Intelligent Robots and Systems, 2001. Proceedings. 2001 IEEE/RSJ International Conference on*, 1:43-48 vol.1, 2001.

[12] Frank L. Lewis. *Applied Optimal Control and Estimation.* Digital Signal Processing Series. Prentice Hall and Texas Instruments, 1992.

[13] John Canny, Ashutosh Rege, and John Reif. An exact algorithm for kinodynamic planning in the plane. In *SCG '90: Proceedings of the sixth annual symposium on Computational geometry*, pages 271-280. ACM, 1990.

[14] Maciej Kalisiak. *Toward More Efficient Motion Panning with Differential Constraints.* PhD thesis, University of Toronto, 2008.

[15] 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 153. Springer-Verlag, Berlin, 1998.

[16] S. Sundar and Z. Shiller. Optimal obstacle avoidance based on the Hamilton-Jacobi-Bellman equation. *IEEE Trans. Robot. & Autom.*, 13(2):305310, April 1997.

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

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

[19] Jongwoo Kim, Jim Keller, and R. Vijay Kumar. Design and verification of controllers for airships. In *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)*, volume 1, pages 54-60, 2003.

[20] Emilio Frazzoli. *Robust Hybrid Control for Autonomous Vehicle Motion Planning.* PhD thesis, Massachusetts Institute of Technology, June 2001.

[21] Jongwoo Kim, Joel M. Esposito, and Vijay Kumar. An rrt-based algorithm for testing and validating multi-robot controllers. In *Robotics: Science and Systems I.* Robotics: Science and Systems, June 2005.

[22] Peng Cheng. *Sampling-based motion planning with differential constraints.* PhD thesis, 2005. Adviser-Steven M. Lavalle.

[23] Russ Tedrake. LQR-Trees: Feedback motion planning on sparse randomized trees. In *Proceedings of Robotics: Science and Systems (RSS)*, page 8, 2009.

[24] N.M. Amato and Y. Wu. A randomized roadmap method for path and manipulation planning. In Proceedings of the IEEE International Conference on Robotics and Automation, volume 1, pages 113 - 120. IEEE, 1996.

[25] Russ Tedrake. Underactuated Robotics: Learning, Planning, and Control for Efficient and Agile Machines: Course Notes for MIT 6.832. Working draft edition, 2009.

[26] J. P. LaSalle. Time optimal control systems. Proceedings of the National Academy of Sciences of the United States of America, 45(4):573-577, 1959.

[27] Donald E. Kirk. *Optimal control theory: an introduction*. Dover Publications, 2004.

[28] Dimitri P. Bertsekas. *Dynamic Programming & Optimal Control*, volume I and II. Athena Scientific, 3rd edition, May 1 2005.