LQR-Based Heuristics for Rapidly Exploring State Space

Elena Glassman and Russ Tedrake

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 when using conventional proximity metrics like the Euclidean distance. Here we argue that linear quadratic regulator (LQR) design can be used to produce a pseudo-distance metric which captures the essential properties of proximity in state space at a reasonable computational cost. We demonstrate improved exploration of the state spaces of the double integrator, torque-limited pendulum, and acrobot when using this metric within the RRT framework.**

I. INTRODUCTION

Kinodynamic motion planning algorithms attempt to find feasible trajectories for a dynamical systems 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[6], however a number of randomized algorithms have been proposed[?], [?] which can achieve fast average-time performance for a large variety of problems[1], [2], [3], [5].

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 [12], can vary greatly as a function of the definition

E. Glassman is with the Department of Electrical Engineering and Computer Science, MIT, Cambridge, MA 02139, USA elg@mit.edu

R. Tedrake is the X Consortium Associate Professor of Electrical Engineering and Computer Science, MIT, Cambridge, MA 02139, USA russt@mit.edu

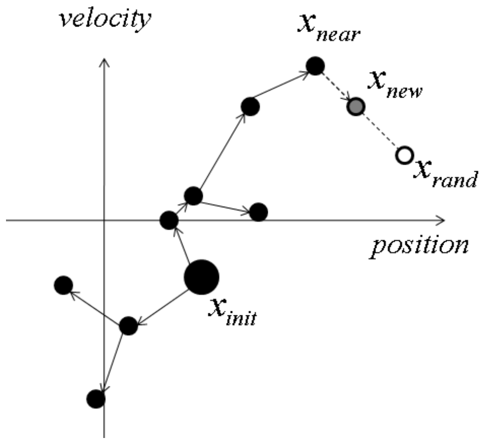


Fig. 1. Illustration of one iteration of growing an RRT, adapted from [7].

```
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 I
THE BASIC ALGORITHM FOR CONSTRUCTING RRTS, ADAPTED FROM [7]

of proximity [7]. The basic RRT algorithm is shown in Table I, 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.

For our proposed proximity function, we use a finite-horizon linear quadratic regulator (LQR) to calculate optimal cost-to-go functions of linearizations of the plant for multiple time horizons in order to locally approximate the optimal proximity measure. In Section II, we place this in the context of previously proposed metrics. We elaborate on the problem formulation in Section III and give a more detailed explanation of our metric and our method of evaluating it

Preprint submitted to 2010 IEEE International Conference on Robotics and Automation. Received September 15, 2009.

in Section IV. In Section V, we show RRT trees grown on three benchmark systems and compare outcomes when using our proximity function to the outcomes obtained when using a standard proximity function. Finally, in Section VI, we discuss the results and future work.

II. RELEVANT LITERATURE

A common proximity heuristic used within RRTs is the Euclidean distance metric. This metric works well on holonomic systems, but performs far more poorly in phase space. 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.

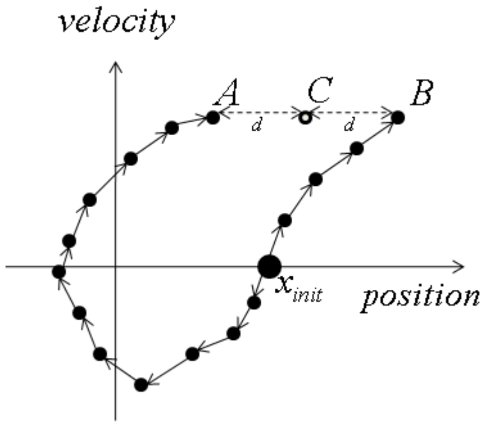


Fig. 2. Nodes A and B are equidistant to point C with respect to the Euclidean distance metric. Children of A could be even closer to C with respect to Euclidean distance. However the children of node B, which has positive velocity, must be to the right of their parent, and therefore cannot get closer 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.

When LaValle and Kuffner first published the RRT algorithm, they explicitly tackled the problem of planning in phase space [2]. While they documented their algorithm's successes, they acknowledged that the single additional component that would address remaining barriers to more efficiently exploring state space is a perfect, quickly computable proximity function.

[7] articulated the possibility of using a cost-to-go function from an optimal control problem as a distance metric. should cite navigation functions here. However, computing the true cost-to-go functions is equivalent to computing a complete plan from every initial condition, as at least as costly as solving the originally specified motion planning problem. Finding general, quickly computable optimal cost-to-go functions, or their approximations, is still a continuing topic of research.

Problem-specific metrics do exist. For simple systems, like a torque-limited pendulum, an energy-based metric can be very effective. [17] used a metric based on Hamilton's principle for unforced rigid bodies. Frazzoli et al. [18] 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 symmetries and relative equilibria of helicopters, along with the construction of motion primitives. Without making problem-specific assumptions, the optimal cost-to-go, where cost was time, was approximated in [19] with a first order calculation.

[12] recently published a Ph.D. thesis on his contributions to RRT exploration of phase space, and while he acknowledges the need for a distance metric that encodes the cost-to-go more accurately than Euclidean distance, he continues to use Euclidean distance and proposes a variant of the RRT algorithm itself. The RRT variant, called RRT-Blossum, changes which nodes are eligible for expansion to reduce redundancies and relaxes constraints to allow for some regression when other nearby branches have hit dead ends. Similarly, [20] collects collision information online and uses that to bias search, rather than designing a new metric. Cheng and LaValle [9] also attempt to improve the RRT algorithm itself by proposing a way to make it less sensitive to a poor proximity heuristic, rather than generating a better proximity heuristic. While these changes improve RRT-based planning in phase space, they do not address the fundamental problems caused by using a highly inaccurate proximity function.

In [4], LaValle and Kuffner suggested considering Lyapunov functions, fitted spline curves, steering methods, and the cost-to-go functions from applying optimal control to linearizations of the systems to be steered. This paper takes that last suggestion of using locally linearized systems' optimal cost-to-go functions as proximity heuristics and develops it further.

III. 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 objective of the metric design problem is to find an approximation of the optimal cost-to-go function between any two states that can be computed efficiently and scales well with the number of state variables. Since the cost-to-go from an arbitrary point A to another arbitrary point B may not be the same as the cost of going from point B to point A, the metric design problem is technically a pseudo-metric design problem.

IV. DESIGNING THE LQR-BASED PROXIMITY HEURISTIC

The linear quadratic regulator (LQR) is an optimal controller for linear systems with quadratic cost functions of state and/or action. In the process of calculating the optimal controller, the LQR-producing function produces a closed-form expression for the exact cost-to-go when using that optimal controller, given any initial point in phase space. This cost-to-go function can be calculated by simply integrating a couple of differential matrix equations over some set period of time and since evaluating the cost-to-go function simultaneously at many points in the phase space is equivalent to a simple matrix multiplication, LQR was identified as a promising method for use within an approximate cost-to-go function for nonlinear systems.

A. Infinite-Horizon LQR

For a function that calculates the infinite-horizon LQR, the inputs are a stabilizable linear plant's dynamics

$$\dot{x} = Ax + Bu \tag{1}$$

and some matrices specifying the relative weights given to components of a quadratic cost function of state and action

$$J = \int_0^\infty \left(x^T Q x + u^T R u + 2x^T N u\right) dt \tag{2}$$

where $x \in \Re^n$ represents a point in phase space, $u \in \Re^m$ is a vector of inputs, $A, Q \in \Re^{n\times n}$, $R \in \Re^{m\times m}$, and $B, N \in \Re^{n\times m}$.

The output is a state-feedback policy, $u = -Kx$, and an *exact* cost-to-go function, $x^T S x$, where $K \in \Re^{m\times n}$ and $S \in \Re^{n\times n}$. The policy drives the plant to the origin with the minimal possible cost. The cost-to-go function evaluated at any point in phase space gives the total cost incurred for following that optimal policy for all time. S is the steady-state solution to a differential matrix equation, also called the Riccati equation, which incorporates the A and B matrices from the system dynamics and the matrices specifying the cost function.

The infinite-horizon LQR cannot, however, be used to calculate the cost-to-go to a point in phase space with a non-zero velocity, which is not a fixed point. It is physically impossible for a system to, *for all time*, get asymptotically closer and closer to being at a particular position at a particular non-zero velocity. The integrand of the cost function will not decay asymptotically and the integral will not converge as $t \to \infty$. Therefore our metric is based on the finite-horizon LQR. The differential equation for S stays the same, but rather than finding its steady-state solution, the dynamics of the time-varying $S(t)$ are simulated out to whatever finite time-horizons are selected by the user. It makes sense that the cost-to-go function for a point x, that is not a fixed point, would be time-varying. Let us consider an example where applying non-zero inputs is very costly. The cost of steering the plant from some point x' to x in t_1 seconds may be small, especially if the passive dynamics are doing most of the work and little additional input is needed. The cost of steering the plant from x' to x in t_2 seconds, on the other hand, may be very costly, if it requires heavy actuation to prevent the system from passing through x too early or too late.

B. Finite-Horizon LQR with a Hard Constraint on the Final State and a Free Final Time

We chose to place an additional constraint on the LQR used within our proposed proximity heuristic: a constraint requiring the policy to drive the system to the desired final state by the end of the specified finite time horizon. This is in contrast to a soft constraint, where there is only a penalty for not being at the desired final state at the end of the specified time horizon. The extra requirement also allows us to the remove Q from the cost function. As result, how far a path strays away (in terms of squared Euclidean distance $x^T Q x$) from the desired final state does not affect the path's overall cost, as long as the path does ultimately reach the desired final state at the final time.

Since the finite-horizon LQR solution is recursive (the solution for the t_1 time horizon is dependent on the solution for the t_1^- time horizon), solving for some maximum time horizon, t_{max}, also produces the solutions for all time horizons less than t_{max}. We can therefore treat the time horizon as a free variable to optimize over for each pair of nodes whose proximity we wish to calculate.

C. The LQR-Based Proximity Heuristic

Let us consider a controllable, smoothly differentiable, nonlinear system:

$$\dot{x} = f(x, u) \tag{3}$$

and x_{rand}, a random sample in the phase space produced by the RRT algorithm building a tree on this nonlinear system. Define a new coordinate system, where x_{rand} (which is also serving as the goal for this particular example iteration of the RRT algorithm) in the old coordinate system coorresponds to the origin in the new coordinate system:

$$\bar{x} = x - x_{rand}. \tag{4}$$

Preprint submitted to 2010 IEEE International Conference on Robotics and Automation. Received September 15, 2009.

Now linearize the system around x_{rand}:

$$\dot{\bar{x}} = \frac{d}{dt}\left(x(t) - x_{rand}\right) = \dot{x}(t) \tag{5}$$

$$\approx f(x_{rand}, u_f) + \frac{\delta f}{\delta x}\left(x(t) - x_{rand}\right) + \frac{\delta f}{\delta u}\left(u - u_f\right) \tag{6}$$

$$= A\bar{x} + B\bar{u} + c \tag{7}$$

Assume u_f, the input vector at t_f, the end of the finite horizon, is zero. Define a quadratic cost on $\bar{u}$, plus an added constant, which serves as a penalty on paths of longer durations:

$$J(\bar{x}, t_0, t_f) = \int_{t_0}^{t_f} \left[1 + \frac{1}{2}\bar{u}^T(t)R\bar{u}(t)\right] dt, R = R^T > 0. \tag{8}$$

To complete the specification of a finite time horizon LQR problem with a hard constraint on the final state, add the requirements that

$$\bar{x}(t_f) = \vec{0} \tag{9}$$

$$\bar{x}(t_0) = x_0 \tag{10}$$

Using Pontryagin's minimum principle, a necessary condition for optimality, it can be derived that the inverse of $S(t)$, referred to here as $P(t)$, is governed by the following differential matrix equation:

$$\dot{P}(t) = AP(t) + P(t)A^T + BR^{-1}B^T, P(t_f) = 0 \tag{11}$$

The resulting cost-to-go is

$$J^*(\bar{x}, t_f) = t_f + \frac{1}{2}d^T(\bar{x}, t_f)P^{-1}(t_f)d(\bar{x}, t_f) \tag{12}$$

where

$$d(t) = e^{At_f}x + \int_0^{t_f} e^{A(t_f - \tau)}cd\tau \tag{13}$$

. The LQR-based proximity heuristic's value for the distance from x to x_{rand} is:

$$\texttt{LQR-based\_proximity}(x \rightarrow x_{rand}) = min_{t_f} J^*(\bar{x}, t_f) \tag{14}$$

This is shown in Table II.

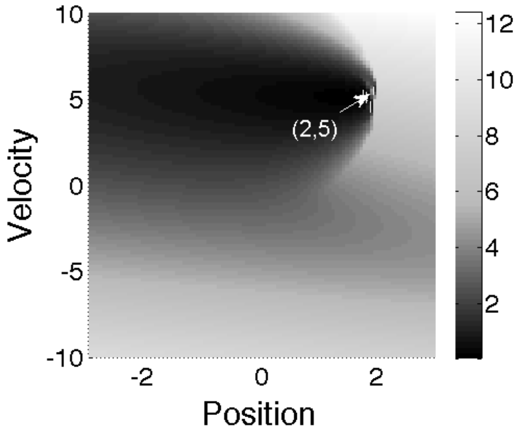
In order to handle state variables that require wrapping, such as angles, one must not only minimize over t_f but also over different unwrapped points in phase space that wrap back to the same point.

In Figure 3, we have visualized the proximity to the state $(2, 5)$, given the dynamics of a undamped brick, which is equivalent to a double integrator. It is clear that the LQR-based proximity function agrees with our intuition about the relative benefit of extending branch A or B in the original, motivating example illustrated in Figure 2.

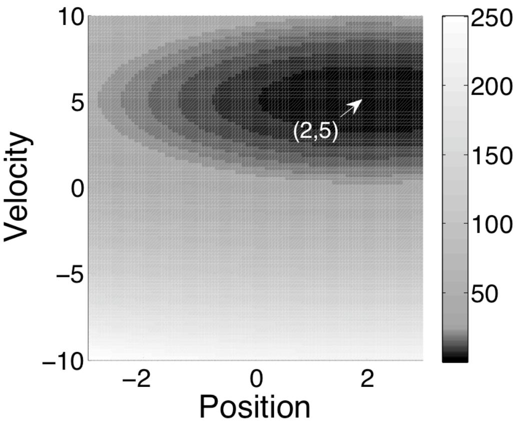
```
1:  procedure LQR-BASED_PROXIMITY($x$,$x'$,plant_dynamics)
2:      $A, B, c \leftarrow$ Linearize plant_dynamics at $x'$.
3:      $S(t) \leftarrow$ COST_TO_GO(linearized_plant_dynamics)
4:      $d(t) \leftarrow e^{At_f}x + \int_0^{t_f} e^{A(t_f-\tau)}cd\tau$
5:      RETURN the minimum value of $t + \frac{1}{2}d^T S(t)d$ over all $t$ from
        $t = 0$ to $t_f$. ($t + \frac{1}{2}d^T S(t)d$ is the time-dependent cost-to-go from $x$
        to $x'$ for the linearized system.)
6:  end procedure
7:  procedure COST_TO_GO(linearized_plant_dynamics)
8:      Substitute matrices $A$, $B$, $c$, and $R$ into Equ. 11.
9:      Calculate $P(t)$ by integrating the Equ. 11 backwards in time from
        $t = t_f$ to $t = 0$ and saving the intermediate results for $P$. ($t_f$ is the
        maximum considered trajectory duration.)
10:     $S(t) \leftarrow$ the inverse of the matrix $P(t)$ at each $t$.
11:     RETURN $S(t)$
12: end procedure
```

TABLE II

PSUEDOCODE FOR THE LQR-BASED PROXIMITY HEURISTIC, WHICH COMPUTES THE APPROXIMATE COST-TO-GO FROM x TO x'



(a) LQR-Based Proximity



(b) Euclidean Distance

Fig. 3. Maps of the proximity to the state $(2, 5)$, given the dynamics of a undamped brick (equivalent to a double integrator).

V. EXPERIMENTS

A. Evaluation

We wish to assess the performance of metrics for planning paths to any arbitrary goal region in state space. A good metric should enable the goal-directed RRT algorithm to find a feasible path from the initial state to the goal region quickly and/or without creating an unreasonably large number of tree nodes. We run an RRT on the demonstration problem without specifying a goal region and compute a measure of how much phase space the resulting tree reaches. It serves as a surrogate measure for the mean time to find a feasible path. [19] also used this technique to quantitatively compare planning algorithms' effectiveness.

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.

B. Qualitative Coverage Results

We have grown RRTs on three classic dynamical systems, without bias toward a goal state. The only differences between the trees is the algorithm defining the distance between points in state space. R was arbitrarily set to 1, and the maximum considered finite horizon length was 5 seconds. The green "X" indicates the location of the tree root (x_{init}).

1) Brick Example: The first example system is an undamped brick that moves along a single dimension, and to which forces of limited magnitude can be applied. It is equivalent to a double integrator.

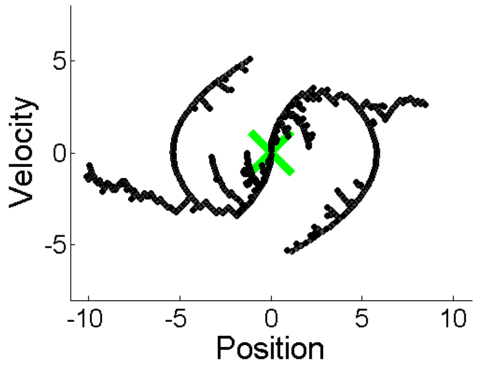
In Figure 4, the 500 nodes of the RRT built using Euclidean distance has only two prominent paths extending out from the tree's root: one in which the brick is pushed to the left with the maximum allowable force and one in which the brick is pushed to the right with the maximum allowable force. In contrast, the 500 nodes of the RRT built using the LQR-based heuristic has four prominent paths, each extending into a different quadrant of phase space surrounding the initial state of stillness at the origin. The two additional prominent paths represent being pushed away from the origin and then reapproaching the origin.

2) Pendulum Example: The second example is a torque-limited simple pendulum.

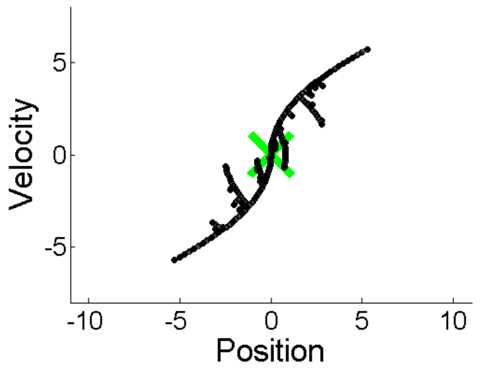
In Figure 5, the 500-node RRT built using Euclidean distance grows paths that make the pendulum swing faster and faster around its pivot point. As a result, a good deal of the phase space is reached. The 500-node RRT built using the LQR-based heuristic, however, doesn't just sweep quickly through a good percentage of the sampled space, it hits *many different* points in each bin.

3) Acrobot Example: The third and final example is the torque-limited acrobot, which is a double pendulum with actuation only at the joint between the two links.

In Figure 6, it is clear that more of the sampled phase space is reached by branches of the 500-node RRT using the LQR-based heuristic than by the 500-node RRT using the Euclidean distance measure.



(a) Using LQR-Based Proximity



(b) Using Euclidean Distance

Fig. 4. 500-node RRTs grown on an undamped brick (double integrator).

C. Quantitative Coverage Results

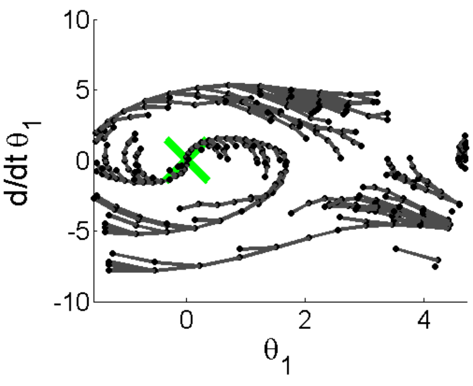
In the final figure, Figure 7, the coverage metric is applied to the preceeding graphs and shown in bar graphs so that the coverage of the trees can be quantitatively compared.

Interestingly, for trees of 200 nodes, the coverage of the ED-based trees was just as good or better than that of the LQR-based trees. However, when the size of the trees was increased to 500 nodes, the LQR-based trees' coverage overtook that of the ED-based trees.

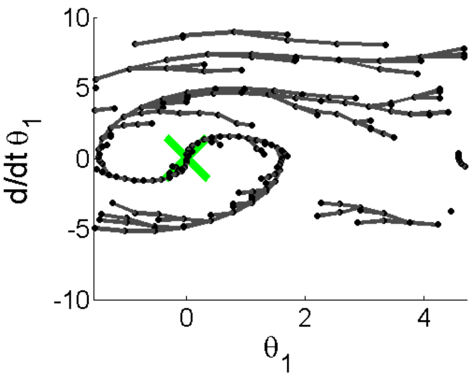
VI. DISCUSSION

When limited to some given maximum number of nodes in the tree, the LQR-based proximity heuristic clearly enhances the RRT's ability to grow branches into all the possible places that a goal region might be in state space, when compared to the typical Euclidean distance metric. This is important because some distance-function-based planning algorithms, like LQR-Trees [21], benefit immensely from covering the greatest amount of phase space with the fewest possible nodes.

However, some applications are more dependent on time, not the number of nodes in the tree. Methods for speeding up the calculation of the LQR-based proximity heuristic will soon be implemented, so that meaningful comparisons of coverage can be made as a function of time, not just nodes. Furthermore, scaling factors within the Euclidean and



(a) Using LQR-Based Proximity



(b) Using Euclidean Distance

Fig. 5. 500-node RRTs grown on a torque-limited pendulum.

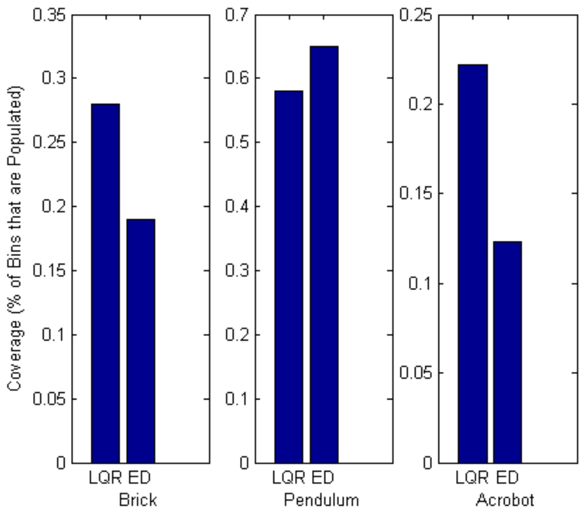
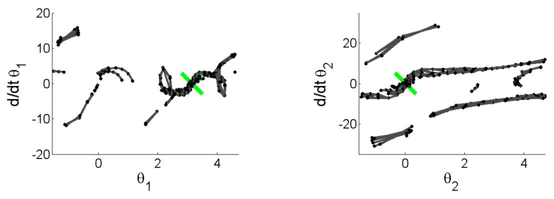
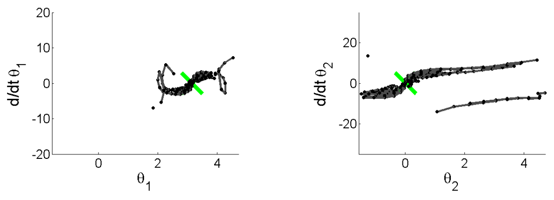


Fig. 7. Comparative Coverage of 500-node RRTs' Exploration of the State Space



(a) Using LQR-Based Proximity



(c) Using Euclidean Distance

Fig. 6. 500-node RRTs grown on the Acrobot

LQR-based proximity measures will be optimized in future testing, and overall coverage scores will be the average of the coverage scores of RRTs grown from each of many randomly selected initial states, rather than just one, as was done here. We have shown the promise of kinodynamic planning with the LQR-based proximity heuristic within the context of three simple two- and four-dimensional systems, and we hope it will signficantly reduce the difficulty of planning sophisticated, dynamic movements for high-dimensional systems, like humanoid and industrial robots.

ACKNOWLEDGMENTS

The authors gratefully acknowledge the contributions of David Wingate, Rick Cory, Alec Shkolnik, Ian Manchester, John Roberts, and Martin Glassman. We also appreciate the thoughtful feedback of Seth Teller and Abraham Bachrach.

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] 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.
[6] J. H. Reif. Complexity of the movers problem and generalizations. In *Proceedings IEEE Symposium on Foundations of Computer Science*, page 421427, 1979.
[7] Steven M. Lavalle. From dynamic programming to rrts: Algorithmic design of feasible trajectories. In *Control Problems in Robotics*. Springer-Verlag, 2002.
[8] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control.* Athena Scientific, 2nd edition, 2000.

[9] 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.

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

[11] 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.

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

[13] 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.

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

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

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

[17] 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.

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

[19] 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.

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

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

Preprint submitted to 2010 IEEE International Conference on Robotics and Automation. Received September 15, 2009.