LQR-RRT* for General Kinodynamic Systems

Gustavo Goretkin, Alejandro Perez, Robert Platt Jr., and George Konidaris; TLPK?
Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology
{goretkin, aperez, rplatt, gdk}@csail.mit.edu

***Abstract*—**

I. Introduction

The RRT* algorithm [1] offers a practical and effective method for probabilistically complete optimal motion planning. Like all members of the RRT family of planning algorithms [2], it incrementally builds a tree by using a cost measure to determine "closest" vetices in the tree to a random sample and an extension method to find an optimal trajectory to the sample. While designing such procedures is straightforward for kinematic systems, it is difficult to design these for kinodynamic systems because it is non-trivial to calculate locally optimal kinodynamic trajectories even in free space. Hence, the distance measure and the extension method are the key pieces required to apply RRT* to kinodynamic systems.

Recently, several authors have applied ideas from linear control theory to estimate distance and to calculate locally optimal trajectories, as was originally suggested by LaValle and Kuffner [2]. Glassman and Tedrake applied the idea to the standard RRT formulation in order to plan in underactuated domains by using an affine version of LQR to more accurately estimate dinodynamic "distances" between a random sample and vertices in the tree [5]. Perez et al. [6] extended that idea to the RRT* algorithm by proposing that an infinite-horizon LQR controller can be used not only to estimate distances but also to calculate trajectories that extend the tree. Unfortunately, this extension method does not always yield locally optimal trajectories for the finite-time extensions (even for linear systems) because an infinite horizon LQR controller was used. Webb and van den Berg [7] proposed a similar method but use a finite-horizon optimal controller to calculate the extension trajectories. The optimal finite time-horizon itself was calculated by finding the roots of a high-order polynomial. In conunction with RRT*, this extension method was proven to converge to optimal solutions for linear systems problems with indefinite time horizons. Unfortunately, this method is complex to implement because of the root-finding requirements and can only be used with a very specific type of quadratic cost function. Moreover, it is not clear how this method should be extended to non-linear problems since the root-finding method would not apply.

In this paper, we propose a new method of applying LQR to the problem of finding optimal finite-horizon extension trajectories and costs. The key challenge is to determine the optimal time horizon of each extension of the tree. Whereas Webb and van den Berg [7] accomplish this by solving a polynomial, we propose a method based on sampling time as an additional dimension of the space in which the tree grows. Growing an RRT in time as well as state space is sometimes done in order to solve problems with dynamic obstacles environments [**?**], [**?**]. However, in our case, the purpose is to sample from the space of possible trajectory durations. Although this additional dimension can increase the asymptotic computational complexity of the algorithm, it eliminates the need to calculate an optimal extension time horizon using special-purpose tools. It is therefore much simpler to implement and enlarges the space of problem specifications that can be solved. Moreover, at least for the case of linear systems, this paper shows that in the context of RRT*, the method trivially satisfies Karaman and Frazzoli's requirements for probabilitic optimality of the resulting trajectory. Most importantly, we believe that the method can be extended directly to non-linear systems with similar optimality guarantees. We provide simulations illustrating the behavior of the algorithms for both linear and non-linear systems.

II. Background

A. Problem Statement

Given a deterministic system with known process dynamics, the optimal kinodynamic motion planning problem is to find a trajectory that begins in a given start state, terminates in a goal region, avoids obstacles, and satisfies differential dynamics constraints. Let the state space and control input space of the system be described by compact sets, $X \subseteq \mathbb{R}^n$ and $U \subseteq \mathbb{R}^m$, respectively. The system process dynamics are described by a continuously differentiable function,

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

where $x(t) \in X$ and $u(t) \in U$. Given a time horizon, $T \in \mathbb{R}_{>0}$, a dynamically feasible trajectory is a continuous function, $x : [0, T] \to X$, for which there exists a control function, $u : [0, T] \to U$, that satisifies Equation 1. The optimal kinodynamic motion planning problem is to find a dynamically feasible trajectory starts in an initial state, $x_0 \in X$, avoids obstacles, $X_{obs} \subset X$, and terminates in a goal region, $x_{goal} \subset X$ while minimizing a cost functional of Equation 2.

Problem 1: (Optimal kinodynamic motion planning [**?**])

Given a state space, X, obstacle region, X_{obs}, goal region, X_{goal}, control input space, U, and function f describing the system process dynamics, find a control, $u : [0, T] \to U$, for some $T \in \mathbb{R}_{>0}$ such that the corresponding dynamically feasible trajectory, $x : [0, T] \to X$, avoids the obstacle region,

reaches the goal region, and minimizes the cost functional

$$J(x,u) = \int_{t=0}^{T} g(x(t),u(t)). \tag{2}$$

B. RRT*

Optimal Rapidly-exploring random tree (RRT*) [1] is a version of the RRT algorithm [2] that has the *asymptotic optimality* property, i.e., almost-sure convergence to an optimal solution.

The major components of the algorithm are:

- *Random sampling*: The `Sample` procedure provides independent uniformly distributed random samples of states from the configuration space.
- *Near nodes*: Given a set V of vertices in the tree and a state x, the $\texttt{Near}(V,x)$ procedure provides a set of states in V that are close to x. Here, closeness is measured with respect to some metric as follows:

$$\texttt{Near}(V,x) := \left\{ x' \in V \ : \ \|x - x'\| \leq \gamma \left(\frac{\log n}{n}\right)^{1/d} \right\},$$

  where $\|\cdot\|$ is the distance using the metric, n is the number of vertices in the tree, d is the dimension of the space, and γ is a constant.
- *Steering*: Given two states x, x', the $\texttt{Steer}(x,x')$ procedure returns a path σ that connects x and x'. Most implementations connect two given states with a straight path in configuration space.
- *Collision checking*: Given a path σ, $\texttt{CollisionFree}(\sigma)$ returns true if the path lies in the obstacle-free portion of configuration space.

Algorithm 1: $\texttt{RRT}^*((V,E),N)$

1 **for** $i = 1,\ldots,N$ **do**
2 $\quad x_{\text{rand}} \leftarrow \texttt{Sample}$;
3 $\quad X_{\text{near}} \leftarrow \texttt{Near}(V, x_{\text{rand}})$;
4 $\quad (x_{\min}, \sigma_{\min}) \leftarrow \texttt{ChooseParent}(X_{\text{near}}, x_{\text{rand}})$;
5 $\quad$ **if** $\texttt{CollisionFree}(\sigma)$ **then**
6 $\quad\quad V \leftarrow V \cup \{x_{\text{rand}}\}$;
7 $\quad\quad E \leftarrow E \cup \{(x_{\min}, x_{\text{rand}})\ \}$;
8 $\quad\quad (V,E) \leftarrow \texttt{Rewire}(\ (V,E), X_{\text{near}}, x_{\text{rand}}\ )$;
9 **return** $G = (V,E)$;

Algorithm 2: $\texttt{ChooseParent}(X_{\text{near}}, x_{\text{rand}})$

1 $\text{minCost} \leftarrow \infty$; $x_{\min} \leftarrow \texttt{NULL}$; $\sigma_{\min} \leftarrow \texttt{NULL}$;
2 **for** $x_{\text{near}} \in X_{\text{near}}$ **do**
3 $\quad \sigma \leftarrow \texttt{Steer}(x_{\text{near}}, x_{\text{rand}})$;
4 $\quad$ **if** $\texttt{Cost}(x_{\text{near}}) + \texttt{Cost}(\sigma) < \text{minCost}$ **then**
5 $\quad\quad \text{minCost} \leftarrow \texttt{Cost}(x_{\text{near}}) + \texttt{Cost}(\sigma)$;
6 $\quad\quad x_{\min} \leftarrow x_{\text{near}}$; $\sigma_{\min} \leftarrow \sigma$;
7 **return** $(x_{\min}, \sigma_{\min})$;

Algorithm 3: $\texttt{Rewire}(\ (V,E), X_{\text{near}}, x_{\text{rand}}\ )$

1 **for** $x_{\text{near}} \in X_{\text{near}}$ **do**
2 $\quad \sigma \leftarrow \texttt{Steer}(x_{\text{rand}}, x_{\text{near}})$;
3 $\quad$ **if** $\texttt{Cost}(x_{\text{rand}}) + \texttt{Cost}(\sigma) < \texttt{Cost}(x_{\text{near}})$ **then**
4 $\quad\quad$ **if** $\texttt{CollisionFree}(\sigma)$ **then**
5 $\quad\quad\quad x_{\text{parent}} \leftarrow \texttt{Parent}(x_{\text{near}})$;
6 $\quad\quad\quad E \leftarrow E \setminus \{x_{\text{parent}}, x_{\text{near}}\}$;
7 $\quad\quad\quad E \leftarrow E \cup \{x_{\text{rand}}, x_{\text{near}}\}$;
8 **return** (V,E);

The algorithm proceeds as follows. Firstly, a state is sampled, denoted as x_{rand}, from the configuration space (Line 2), and near neighbors are evaluated using the `Near` function (Line 3).[1] Next, RRT* calls `ChooseParent` (Algorithm 2) to find a candidate for a parent node to x_{rand} (Line 4). `ChooseParent` searches through the nodes in the set X_{near} of near nodes and returns the node, denoted as $x_{\min}$, that reaches x_{rand} with minimum cost along with the path $\sigma_{\min}$ connecting $x_{\min}$ to x_{rand}. After that node is found, the algorithm checks $\sigma_{\min}$ for collision (Line 5). If the path is obstacle-free, the algorithm adds x_{rand} to the tree and connects $x_{\min}$ to x_{rand}, and attempts to "rewire" the nodes in X_{near} using the `Rewire` procedure (Algorithm 3). The `Rewire` procedure attempts to connect x_{rand} with each node in the set X_{near} of near nodes. If the path that connects x_{rand} with a near node x_{near} reaches x_{near} with cost less than that of its current parent, then the x_{near} is "rewired" to x_{rand} by connecting x_{rand} and x_{near}.

C. Affine LQR

Linear quadratic regulation (LQR) is an optimal control technique that efficiently calculates an optimal policy for linear systems with quadratic cost functions. This paper uses finite-horizon LQR that solves the following problem: given deterministic linear process dynamics,

$$\dot{x}(t) = Ax(t) + Bu(t)), \tag{3}$$

find a state and control trajectory, x and u, that minimizes the cost functional,

$$\begin{aligned} J_{LQR}(x,u) =& x(T)^T Q_F x(T) \\ &+ \int_{t=0}^{t=T} x(t)^T Q x(t) + u(t)^T R u(t), \end{aligned} \tag{4}$$

where we have initial and final value constraints on the state trajectory, $x(0) = x_0$ and $x(T) = x_F$. The linear quadratic regulator solves for the optimal solution to the above problem in two steps. First, it calculates the optimal value function by integrating the differential Riccati equation backward in time starting at the final time T with $P(T) = Q_F$ and integrating toward 0:

$$-\dot{P}(t) = A^T P(t) + P(t)A - P(t)BR^{-1}B^T P(t) + Q.$$

[1] Although it is not explicitly noted here, we assume that the algorithm proceeds to the next iteration if the `Near` function returns an empty set.

Second, LQR calculates the optimal action to take at time $t \leq T$ using:

$$u(t) = -R^{-1}B^T Px.$$

The standard LQR formulation requires process dynamics to be linear and the goal state to be at the origin (*i.e.* the state cost function to be measured with respect to the origin). However, our RRT application requires goals in locations different than the point about which the system is linearized. Also, we allow for affine process dynamics. Specifically, suppose that the process dynamics are now

$$\dot{x}(t) = Ax(t) + Bu + c, \tag{5}$$

and instead of minimizing Equation 4, suppose that our goal is now to minimize

$$J_{aff}(x,u) = (x(T) - x^*)^T Q_F (x(T) - x^*) + \int_{t=0}^{t=T} (x(t) - x^*)^T Q(x(t) - x^*) + u(t)^T Ru(t). \tag{6}$$

The difficulty is that the cost function is no longer a quadratic form. It turns out that this problem can easily be solved by reformulating the problem using a change of coordinates in state space and an augmentation of the state vector. Redefine state to be $\bar{x}(t) = x(t) - x^*$, and augment the new state vector with an additional coordinate that will always be equal to one. The new process dynamics are:

$$\begin{pmatrix} \dot{\bar{x}}(t) \\ 1 \end{pmatrix} = \begin{pmatrix} A & c + Ax^* \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \bar{x}(t) \\ 1 \end{pmatrix} + \begin{pmatrix} B \\ 0 \end{pmatrix} u.$$

It may be verified that these process dynamics are the same as those in Equation 3 except that they are now expressed in terms of the new variable. We can now express the offset cost functional in Equation 7 as:

$$J(\bar{x},u) = \begin{pmatrix} \bar{x}(T) \\ 1 \end{pmatrix} \bar{Q}_F \begin{pmatrix} \bar{x}(T) \\ 1 \end{pmatrix} + \int_{t=0}^{t=T} \begin{pmatrix} \bar{x}(t) \\ 1 \end{pmatrix} \bar{Q} \begin{pmatrix} \bar{x}(t) \\ 1 \end{pmatrix} + u(t)^T Ru(t), \tag{7}$$

where $\bar{Q}_F = \begin{pmatrix} Q_F & 0 \\ 0 & 1 \end{pmatrix}$, and $\bar{Q} = \begin{pmatrix} Q & 0 \\ 0 & 1 \end{pmatrix}$ and we solve the affine version of the problem by applying LQR to the augmented system in the standard way.

III. LQR-RRT* FOR AFFINE SYSTEMS

The key idea of this paper is to solve kinodynamic RRT* problems using LQR by sampling in time as well as in state space. A tree is constructed where each vertex has an associated state *and* time. Each vertex is also associated with the one-step cost associated with reaching that vertex from its parent.

A. Affine version of the kinodynamic planning problem

Consider the version of Problem 1 where the system process dynamics (Equation 1) are linear or affine and where the cost functional (Equation 2) is quadratic. Notice that except for the presence of obstacles, this would be a linear problem solvable using LQR. Specifically, we require the process dynamics to be affine (of the form in Equation 5) and the cost function to be of the form,

$$J(x,u) = \int_{t=0}^{T} (x(t) - x^*)^T Q(x(t) - x^*) + u(t)^T Ru(t). \tag{8}$$

The obstacle region, X_{obs}, may be of any shape. We consider two instances of the problem: a) where the time horizon, T, is fixed and given as input, and b) where the time horizon is free to vary in order to minimize cost (*i.e.* T is an optimization variable).

B. Tree extension

First, suppose that we are given a final time constraint, T, at which the trajectory must end. We sample points from state-time, $(x_{rand}, t_{rand}) \sim X \times T$. This happens on each iteration of the FOR loop in Algorithm 1. Given this sample, we calculate the affine LQR solution by integrating the differential Riccati equation for the affine dynamics backward in time to 0 starting from t_{rand} with the quadratic cost function centered at the sample point, $x^* = x_{rand}$ (see Section II-C). The LQR cost function used in the Riccati integration has Q and R equal to their values in the global quadratic cost function. Q_F is set to a multiple of the identity matrix that is sufficiently large to cause LQR to find trajectories that terminate very close to the random sample [2]. The result of this integration will be written as a function, $P : [0,T] \rightarrow S_+^{n+1}$ (the result of the Riccati integration is always a positive semi-definite matrix).

RRT* begins with the standard RRT tree extension. We calculate the cost of traveling to the sampled state by the sampled time from each of the vertices in the tree (this is implemented in the **Near** function of Algorithm **??**). This can be accomplished by evaluating the LQR cost-to-go function for each vertex, $v_i = (x_i, t_i) \in V$, in the tree:

$$c_1(v_i, x_{rand}) = (x_{rand} - x_i)^T P(t_i)(x_{rand} - x_i).$$

We define the cost to any vertex with a vertex time larger than the sample time, $t_i \geq t_{rand}$, to be infinite. Without loss of generality, we assume the time associated with the root vertex to be 0 (non-zero start times can be handled by shifting all times forward). As a result, there is always at least one vertex with a time smaller than the sample time. Once the vertex "closest" to the sample is found, we generate a trajectory between (x_i, t_i) and (x_{rand}, t_{rand}) using LQR. We check whether this trajectory intersections the obstacle region or not. If it does intersect, then we discard the trajectory and re-sample.

[2] In principle, it is possible to integrate an inverse form of the differential Riccati equation that allows setting Q_F to infinity (*i.e.* $Q_F^{-1} = 0$). However, we have found this approach to be numerically troublesome and recommend setting Q_F to a large value that is determined to work well in practice.

C. Tree re-wiring

In the context of a kinodynamic problem, once the tree has been extended, RRT* perform two re-wiring steps [**?**]. In the first, we identify the set of vertices for which

$$c_1(v_i, x_{rand}) \leq \gamma \left(\frac{\log n}{n} \right)^{-\frac{1}{d}}. \tag{9}$$

For each vertex in this set, we calculate the total cost to the root by tracing the tree back to the root and summing the costs. After ranking these vertices in order of cost, we start at the top of this list and attempt to connect the vertex to the random sample. If the resulting trajectory intersects an obstacle or if LQR does not terminate sufficiently close to the sample point, that connection is discarded and the algorithm moves down the list to the vertex associated with the next smallest cost. Once a successful trajectory is found, the tree is correspondingly re-wired.

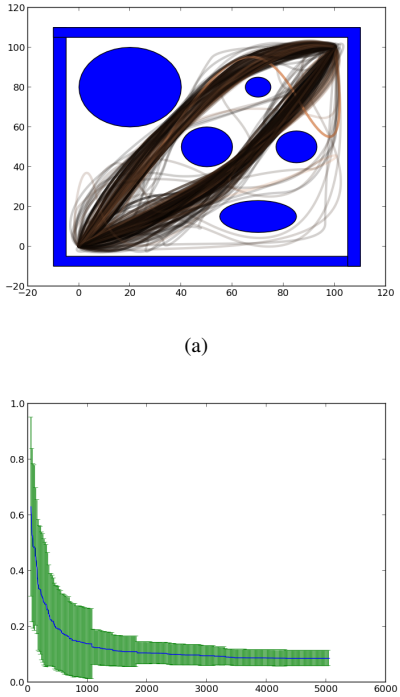
Similarly, RRT* searches for vertices that can be reached *from* the sample and re-wires as necessary. Costs are calculated in the same way as above. This time, it is necessary to integrate the differential Riccati equation backward from each vertex in the tree. Instead of requiring this integration to occur for the entire tree on each re-wire step, we store the backward Riccati integration for each vertex in the tree when it is originally added and access the appropriate P function as necessary. Only vertices within the cost threshold of Equation 9 are evaluated.

D. Optimality of LQR-RRT* for affine systems

Recognizing the optimality of the proposed approach for affine systems is straightforward. The optimality guarantee for RRT* requires two things. First, the estimate of cost used during re-wiring as well as the trajectory generation itself must be optimal in the absence of obstacles. Second, the domain must satisfy a weakened local controllability criterion (Assumption 3 in [**?**]) and an ϵ-collision-free approximate trjectories criterion (Assumption 4 in [**?**]). Since we are growing a tree in state-time, these requirements must be met in the augmented space. With regard to the optimal cost and re-wiring requirements, note that we automatically have this optimality guarantee because we are using LQR, a method known to find optimal trajectories for affine systems over finite time horizons. Furthermore, a reading of the weakened local controllability criterion and the ϵ-collision-free approximate trjectories criterion reveals that the addition of time as a dimension does not invalidate these criteria if they already hold for the un-augmented system.

E. Experiments

We evaluated our approach to affine systems control using a two-dimensional double integrator. Similar evaluations of RRT* performance for the double integrator have appeared in [**?**] and [**?**]. Figure 1 illustrates the working of the algorithm for an example domain. Starting at $(0, 0)$ with zero velocity at time $t = 0$, the system must reach the goal at $(8, 0)$ at time $t = 10s$. The cost function (Equation 8) has $Q = 0$ and



(a)

(b)

Fig. 2. (a) Best-cost trajectories found during 50 separate runs of the algorithm. Darkness of the line is inversely proportional to trajectory cost. (b) Average and standard deviations of the costs of the best trajectories as a function of algorithm iteration number.

$R = I$. After 600 iterations of the RRT* algorithm, there are 249 vertices in the tree and the current best-cost path has been improved eight times. Initially, the algorithms finds a trajectory (Figure 1(a)) with cost 3515. The subsequent eight best-cost trajectories have costs 2495, 2495, 215, 53, 47, 27, 22, and 14.

We also evaluated average performance over multiple runs of the algorithm (Figure 2). Figure 2(a) shows the set of best paths during 50 different runs of the algorithm for the double integrator domain shown. Notice that the different runs of the algorithm tend to identify two different homotopies that have nearly equal costs. Figure 2(b) shows the mean and standard deviations of the best-cost paths as a function of algorithm iteration.

IV. Application to General Dynamical Systems

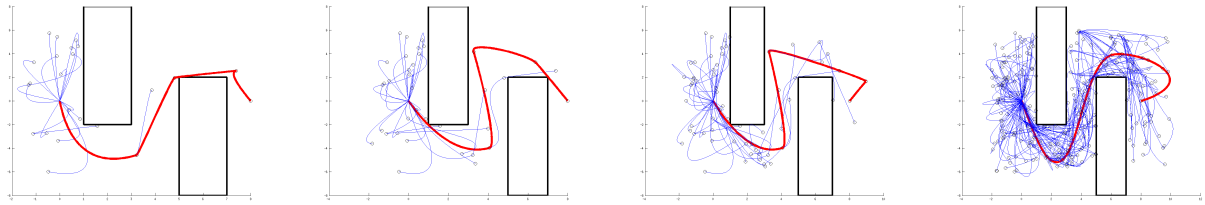
Consider a discrete-time dynamical system in the form

$$x_{k+1} = f(x_k, u_k) \tag{10}$$

with additive cost function

$$J(\mathbf{u}, \mathbf{x}) = \sum_{k=0}^{T} g(x_k, u_k) \tag{11}$$

and starting point x_0. The state vector, x, is n-dimensional and the control vector, u, is m-dimensional. We aim to find



(a) (b) (c) (d)

Fig. 1. Example of using LQR-RRT* to find an optimal solution to the kinodynamic motion planning for the double integrator in the presence of obstacles. The four images show the tree at different points during growing. The blue lines show the tree. The red path denotes the current best path to the goal. The cost of the best path in each of the four frames is, respectively, 3515, 2495, 53, and 14.

a sequence $\mathbf{u} = \{u_0, \cdots, u_T\}$ which induces a trajectory $\mathbf{x} = \{x_1, \cdots, x_T\}$ satisfying the dynamics (10) such that C is minimized according to (11).

The real cost of moving from point x' to x'' is

$$C(x, x') = \min_{\mathbf{u}} J(\mathbf{u}, \mathbf{x}) \quad \text{subject to (10), } x_0 = x,\ x_T = x'$$

Note that the minimization happens over control sequences $\mathbf{u}$ of a fixed time lengths, according to

We approximate $C(x', x'')$ by taking a first-order approximation of the dynamics and a second-order approximation of the cost and applying LQR control. In general, the approximated dynamics and cost are of the following form

$$x_{k+1} \approx Ax_k + Bu_k + c \tag{12}$$

$$J(\mathbf{u}, \mathbf{x}) \approx \sum_{k=0}^{T} x_k^T Q x_k + u_k^T R u_k + 2q^T x_k + 2r^T u_k + d \tag{13}$$

A and Q are $n \times n$, B is $m \times n$, R is $n \times n$. c and q are $n \times 1$, r is $m \times 1$ and d is a scalar.

$$A = \left.\frac{\partial f}{\partial x}\right|_{x^*, u^*}$$
$$B = \left.\frac{\partial f}{\partial u}\right|_{x^*, u^*}$$
$$c = -Ax^* - Bu^* + f(x^*, u^*)$$

x^*, u^* is the point about which the linearization is performed. Typically u^* is taken to be $\mathbf{0}$ and $x^* = x'$

Equations 12 and 13 are the truncated Taylor expansions of f and g. The dynamics f must be once-differentiable and addition cost g must be twice-differentiable.

A. Reduction to the Previous Problem

It is possible to transform the problem specified with 12 and 13 into LQR form (where there is only an A, B, Q, R matrix) using the following:

$$\underbrace{\begin{bmatrix} x_{k+1} \\ 1 \end{bmatrix}}_{\hat{x}_{k+1}} = \underbrace{\begin{pmatrix} A & c - R^{-1}r \\ 0 & 1 \end{pmatrix}}_{\hat{A}} \underbrace{\begin{bmatrix} x_k \\ 1 \end{bmatrix}}_{\hat{x}_k} + \underbrace{\begin{pmatrix} B \\ 0 \end{pmatrix}}_{\hat{B}} \hat{u}_k$$

$$C(\mathbf{u}, \mathbf{x}) = \sum_{k=0}^{T} \hat{x}_k^T \hat{Q} \hat{x}_k + \hat{u}_k^T R \hat{u}_k$$

with $\hat{u}_k = u_k + R^{-1}r$ and $\hat{Q} = \begin{pmatrix} Q & q \\ q^T & d \end{pmatrix}$.

The $\hat{A}$, $\hat{B}$, $\hat{Q}$, and R matrices specify a linear dynamical system with quadratic costs to which an optimal solution can be found with LQR.

B. Nuances and Subtleties

1) Non-exact steering: rewiring and propagating dynamics

2) Uncontrollable Dynamics: The linearized system may be uncontrollable – the A and B matrices are such that it's not possible to control all the modes of the system. This is the case, for example, for a cart with two inverted pendulums of the same length linearized about the upward-pointing fixed point. The control input to the system affects both linearized pendulums in the same way, so it's not possible to independently stabilize them. For the infinite-horizon LQR control problem, there is no solution. For the finite-horizon problem, there is a solution, though it might not be possible to go to any arbitrary location. If the system linearized at x' cannot reach x'', then $C(x', x'')$ needs to be defined in another way.

is Therefore using the LQR cost metric cannot approximate the cost

3) Indefinite Cost:

4) Actuation Constraints: The LQR framework does not permit actuation constraints.

5) Asymmetric Cost:

C. Non-linear Domain

spaceship orientation

D. Results

Quick mention of performance (or not). Picture of tree, cost over iteration

V. Discussion and Conclusion and Future Work

Code available. Spatial Data structure future

VI. Acknowledgments

VII. References

References

[1] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, June 2011.

[2] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, May 2001.

[3] S. M. Lavalle, "From dynamic programming to RRTs: Algorithmic design of feasible trajectories," in *Control Problems in Robotics*. Springer-Verlag, 2002.

[4] P. Cheng and S. M. Lavalle, "Reducing metric sensitivity in randomized trajectory design," in *In IEEE International Conference on Intelligent Robots and Systems*, 2001, pp. 43–48.

[5] E. Glassman and R. Tedrake, "A quadratic regulator-based heuristic for rapidly exploring state space," in *Proceedings of the IEEE International Conference on Robotics and Automation*, May 2010.

[6] A. Perez, R. Platt, G. Konidaris, L. Kaelbling, and T. Lozano-Perez, "LQR-RRT*: Optimal sampling-based motion planning with automatically derived extension heuristics," in *Proceedings of the IEEE International Conference on Robotics and Automation*, May 2012, pp. 2537–2542.

[7] D. J. Webb and J. V. D. Berg, "Kinodynamic rrt*: Optimal motion planning for systems with linear differential constraints," arXiv:1205.5088v1, submitted.