Optimal Sampling-Based Planning with Automatically Derived Extension Procedures for Systems with General Costs and Dynamics

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*—We introduce a method for automatically deriving the extension procedure—a critical component of the RRT* algorithm—for general systems with differentiable dynamics and twice differentiable cost functions. We first introduce a version of the RRT* algorithm that uses an LQR-based extension to plan for problems with affine dynamics and second order costs, and prove that it almost surely converges to the optimal plan. We then show how to automatically derive an extension method for problems with nonlinear dynamics and non-quadratic costs by locally approximating both functions. This allows us to automatically derive an extension method for general domains, provided their dynamics are locally differentiable and their cost function locally twice differentiable. This removes the need for domain-specific extension methods, which are a major obstacle to the general applicability of RRT*. We demonstrate our method's application in a few standard benchmark domains.**

I. Introduction

The RRT* algorithm [1] offers a practical, effective method for probabilistically complete optimal motion planning. However, like all members of the RRT family of planning algorithms [2], it builds a tree using an extension procedure which, given a pair of points, both estimates the optimal cost of moving from one to the other, and provides a method for doing so. While designing such a procedure is straightforward for kinematic systems, for more complex systems it is often difficult to design an extension procedure that accurately reflects the dynamics of the domain. Since the performance of RRT-based algorithms can be sensitive to the extension procedure used [3], [4], designing it is often the a major obstacle to their application.

Recently, several authors have proposed using methods from linear control theory to automatically derive all or part of the extension procedure, as was originally suggested by LaValle and Kuffner [2]. Glassman and Tedrake [5] derived a cost-to-go pseudo-metric based on linear-quadratic regulators (LQR), which was used to grow an RRT in domains with constrained and underactuated dynamics. Perez et al. [6] extended their work to automatically derive the extension procedure for arbitrary systems with differentiable dynamics and quadratic cost functions. Webb and van den Berg [7] proposed a method for computing trajectories between pairs of states by minimizing a cost function with a tunable trade-off between duration and control effort. ...

We propose a method for deriving the extension procedure for systems with general costs and dynamics (provided the dynamics are differentiable and the costs are twice differentiable). We begin by introducing a version of the RRT* algorithm that uses an LQR-based extension to plan for problems with affine dynamics and second order costs, and prove that it almost surely converges to the optimal plan. We then show how to automatically derive an extension procedure for more general problems using a local linear approximation to the dynamics and a local quadratic approximation to the cost function. MAYBE: WE PROVE THAT OUR METHOD IS PROBABILISTICALLY COMPLETE AND ALMOST SURELY CONVERGES TO THE OPTIMAL PLAN. THIS METHOD IS THEREFORE CAPABLE OF FINDING SUCH A PLAN FOR ANY SYSTEM ... (CONDITIONS) We demonstrate our method's application in a few standard benchmark domains.

II. Background

A. Problem Statement

Given a system with known (possibly non-linear) process dynamics, the optimal motion planning problem is to find a minimum cost feasible trajectory from an initial configuration to a goal configuration. Let *configuration space* be a compact set, $X \subseteq \mathcal{X}$, that describes all possible configurations of the system. Let $U \subseteq \mathcal{U}$ be a compact control input space. We assume that we are given the non-linear process dynamics:

$$x_{k+1} = f(x, u)$$

A *dynamically feasible trajectory* is a continuous function, $\sigma : [0, 1] \to X$, for which there exists a control function, $u : [0, 1] \to U$, such that $\forall t \in [0, 1], \dot{\sigma}(t) = f(\sigma(t), u(t))$. The set of all dynamically feasible trajectories is denoted by Σ_{traj}. Given an initial configuration, x_{init}, and a goal region X_{goal}, the motion planning problem is to find a dynamically feasible trajectory, σ, and a control, u, that starts in the initial configuration and reaches the goal region: $\sigma(0) = x_{\text{init}}$ and $\sigma(1) \in X_{\text{goal}}$. Let $c : \Sigma_{\text{traj}} \to \mathbb{R}^+$ be a *cost functional* that maps each feasible trajectory to a non-negative cost. The optimal motion planning problem is to find a solution to the motion planning problem that minimizes $c(\sigma)$.

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.

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** $(x_{\min}, \sigma_{\min})$;

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.

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);

- *Collision checking*: Given a path σ, $\texttt{CollisionFree}(\sigma)$ returns true if the path lies in the obstacle-free portion of configuration space.

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. Math Tools

Linear Quadratic Regulator (LQR) is an optimal control technique which determines the policy (mapping from state to action) for a linear dynamical system which minimizes a quadratic functional on the state and action.

LQR is used to solve the following problem:
Find $\langle u_0, \cdots u_{T-1} \rangle$ such that

$$x_{k+1} = Ax_k + Bu_k$$

minimizing

$$J(x, u) = \sum_{k=0}^{T-1} \left(x_k^T Q x_k + u_k^T R u_k \right) + x_T^T Q_T x_T$$

given A, B, Q, R, x_0, x_T, and T.

We can enforce that LQR policies obey the final-value constraint on x_T by placing very high cost (practically infinite) on the Q_T fed to the finite-horizon LQR solver. The textbook LQR problem statement requires that the quadratic bowl $x^T Q x$ be centered at $x = 0$. A shift of coordinates can change the center of all bowls, which is not what we want.

We can enrich the class of LQR problems to have arbitrary second-order penalty (along with first-order dynamics, which will be essential later) by considering the follwing transformation:

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

$$J(x, u) = \sum_{k=0}^{T-1} \hat{x}_k^T \hat{Q} \hat{x}_k + \hat{u}_k^T R \hat{u}_k + \hat{x}_T^T \hat{Q}_T \hat{x}_T$$

[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.

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

Here we augment the state vector with an additional dimension whose value is constrained by the dynamics to remain constant. As long as this additional dimension is unity in x_0 , then it will be unity along all trajectories.

In addition to allowing first-order dynamics (increasing the class of systems from linear to affine), this transformation also allows an arbitrary second-order function on x and u. For example, say we want the following bowl:

$$(x - x_0)^T Q (x - x_0) = x^T Q x + -2x_0^T Q x + x_0^T Q x_0$$

Then choose $\hat{Q} = \begin{pmatrix} Q & -Qx_0 \\ -x_0^T Q & x_0^T Q x_0 \end{pmatrix}$.

III. Constrained Linear Systems with Second-Order Cost Functions

Necessary since in general, not possible or practical for steer function to be exact.

Logically equivalent to common formulation of RRT*, but easier to plug in the primitives.

Algorithm detailing the different structure here

For this class of systems, the LQR method is the standard way to find optimal solution. LQR, though, cannot handle constaints on the state nor on actuation.

We can use LQR in the distance metric calculation.

A. Augmenting State with Time

Augmenting time in the RRT* state space allows us to set explicit time goals and allows us to apply the LQR heuristic. Without this, it's not clear what the cost between two states should be. Penalizing time may not be the metric we want to optimize for. Choosing the lowest cost over all possible time horizons does not guarantee completeness. These are crucial points.

Many scenarios include time in the state anyway. For example: time-varying dynamics or obstacles and constraints.

Let $\langle x, k \rangle \in \mathbb{R}^n \times \mathbb{R}_{0+}$ be the space of the RRT state.

1) Primitives: $Steer(s_1, s_2)$

$T = k(s_2) - k(s_1)$

if $T \leq 0$, return NullAction

follow LQR gain matrices around $x(s_1)$ with goal $x(s_2)$, time horizon T

$Distance(s_1, s_2)$

use LQR cost-to-go

2) Optimality Proof Sketch: LQR solves a relaxed version of the problem – no obstacle and no actuation constraints. This is directly analogous to a Euclidean distance metric being a relaxed version of the shortest path in the kinematic case.

IV. Application to General Dynamical Systems

Consider a discrete-time dynamical system in the form

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

with additive cost function

$$J(\mathbf{u}, \mathbf{x}) = \sum_{k=0}^{T} g(x_k, u_k) \tag{2}$$

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 sequence $\mathbf{u} = \{u_0, \cdots, u_T\}$ which induces a trajectory $\mathbf{x} = \{x_1, \cdots, x_T\}$ satisfying the dynamics (1) such that C is minimized according to (2).

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

$$C(x, x') = \min_{\mathbf{u}} J(\mathbf{u}, \mathbf{x})$$
$$\text{subject to (1), } x_0 = x,\ x_T = x'$$

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

$$x_{k+1} \approx Ax_k + Bu_k + c \tag{3}$$

$$J(\mathbf{u}, \mathbf{x}) \approx \sum_{k=0}^{T} x_k^T Q x_k + u_k^T R u_k + 2q^T x_k + 2r^T u_k + d \tag{4}$$

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 3 and 4 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 3 and 4 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:

V. Related Work

vdB, Glassman, Perez

VI. Results

A. Linear Domain

spaceship no orientation

B. Non-linear Domain

spaceship orientation

C. Results

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

VII. Discussion and Conclusion and Future Work

Code available. Spatial Data structure future

VIII. Acknowledgments

IX. 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.