LQR Funnel Graph for Kinodynamic Planning

Yukun Xia
yukunxia@g.harvard.edu

***Abstract*—LQR is a common tool for locally stabilizing nonlinear system to fixed points and a dynamically feasible trajectory. With Sum-of-Squares (SOS) Programming, a verifier based on convex optimization, the valid region of LQR can be calculated, either ellipsoids for fixed points or funnels for trajectories. In this project, I explored the algorithm of sampling and connecting funnels in the state space of the single pendulum. Regardless of all the simplifications, the funnel graph shows high coverage ratio of the state space, and its capability for future kinodynamic planning is also demonstrated.**

I. Introduction

Motion planning through searching is a classic robotics problem. The typical solutions are Rapidly-Exploring Random Trees (RRTs), and its variants. One special case of motion planning is kinodynamic planning [1], where the path should be dynamically feasible, i.e. planning in the state space. The difficulty of kinodynamic planning is the inefficiency of Euclidean distance. Two closing points in the state space can be infeasible to be connected directly. Although RRTs has been successfully applied to kinodynamic planning [2], one common drawback is the sampled trajectories are often suboptimal [3]. To incorporate dynamical information as a prior, the LQR-RRT* algorithm [4] uses the LQR cost as the heuristic for searching, and guarantees the asymptotical optimality. A more general situation is to reuse the existing paths as a probabilistic roadmap [5] for online query.

To increase the coverage of the state space, one efficient algorithm LQR-trees [6] tries to construct funnels as the building block. A state inside the funnel will move following the center trajectory, and will not exit the funnel within the valid time horizon for the center trajectory. This guarantee comes from the advancing of SOS Programming [7]., and its application in Lyapunov analysis. Although this algorithm is still complete probabilistically, it has a chance to fully cover a local region. One extension to the LQR-trees paper is to create a pre-computed funnel library [8], such that safe trajectories could be composed at runtime.

This report starts by summarizing the background knowledge, and then introduces the funnel construction with the single pendulum as example. The resultant funnels form a graph, and its capability for future kinodynamic planning is conceptually demonstrated.

II. Background

A. Fixed Point Stabilization with LQR

Consider a smooth nonlinear system

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

, with its origin a fixed point, i.e. $f(0,0) = 0$. The goal here is to stabilize the system in the vicinity of the origin. To apply LQR, we firstly linearize the system as

$$\dot{\mathrm{x}} \approx \frac{\partial f}{\partial \mathrm{x}}|_{\mathrm{x}=0,\mathrm{u}=0} \cdot \mathrm{x} + \frac{\partial f}{\partial \mathrm{u}}|_{\mathrm{x}=0,\mathrm{u}=0} \cdot \mathrm{u} = \mathrm{Ax} + \mathrm{Bu} \tag{2}$$

. The next step is to define the cost matrices Q and R, and the cost-to-go function is

$$J(\mathrm{x}) = \int_0^\infty [\mathrm{x}^T \mathrm{Q} \mathrm{x} + \mathrm{u}^T \mathrm{R} \mathrm{u}] dt \tag{3}$$

. Within the valid region for linearization, $J(\mathrm{x})$ has a quadratic form $\mathrm{x}^T\mathrm{S}\mathrm{x}$, where S is the solution to the Algebraic Riccati Equation

$$0 = \mathrm{Q} - \mathrm{SBR}^{-1}\mathrm{B}^T\mathrm{S} + \mathrm{SA} + \mathrm{A}^T\mathrm{S} \tag{4}$$

and the corresponding optimal control law is

$$\mathrm{u}^* = -\mathrm{R}^{-1}\mathrm{B}^T\mathrm{S}\mathrm{x} \tag{5}$$

B. Trajectory Stabilization and TV-LQR

Given a dynamically feasible trajectory $\mathrm{x}_0(t), \mathrm{u}_0(t)$ in the state space, we could also sequentially linearize the system into a LTI form.

$$\bar{\mathrm{x}} = \mathrm{A}(t)\bar{\mathrm{x}} + \mathrm{B}(t)\bar{\mathrm{u}} \tag{6}$$

, where $\bar{\mathrm{x}} = \mathrm{x} - \mathrm{x}_0$ and $\bar{\mathrm{u}} = \mathrm{u} - \mathrm{u}_0$. Since the terminal state of the trajectory may not be a fixed point, here the time horizon is finite $t \in [0, t_f]$, and a terminal cost is defined as $\bar{\mathrm{x}}^T\mathrm{Q}_f\bar{\mathrm{x}}$. The complete form of the cost-to-go function is

$$J(\bar{\mathrm{x}}, t) = \bar{\mathrm{x}}^T\mathrm{Q}_f\bar{\mathrm{x}} + \int_t^{t_f} [\bar{\mathrm{x}}^T\mathrm{Q}\mathrm{x} + \bar{\mathrm{u}}^T\mathrm{R}\bar{\mathrm{u}}] dt \tag{7}$$

. In the vicinity of the trajectory,

$$J(\bar{\mathrm{x}}, t) = \bar{\mathrm{x}}^T\mathrm{S}(t)\bar{\mathrm{x}} \tag{8}$$

, where $\mathrm{S}(t)$ is the solution to the Differential Riccati Equation

$$-\dot{\mathrm{S}} = \mathrm{Q} - \mathrm{SBR}^{-1}\mathrm{B}^T\mathrm{S} + \mathrm{SA} + \mathrm{A}^T\mathrm{S} \tag{9}$$

, and the corresponding optimal linear controller is

$$\bar{\mathrm{u}}^* = -\mathrm{R}^{-1}\mathrm{B}^T\mathrm{S}(t)\bar{\mathrm{x}}\ . \tag{10}$$

C. Lyapunov Analysis and SOS Programming

To approximate the valid region of a LQR controller, a common routine is Lyapunov analysis.

In the local stabilization case, the Lyapunov function $V(\mathrm{x})$ is chosen to be the cost-to-go function $J(\mathrm{x})$, the existing positive definite function. According to LaSalle's Theorem, if within a Lyapunov function's sub-level set $\mathcal{G} : \{\mathrm{x}|V(\mathrm{x}) < \rho\}$, $\dot{V}(\mathrm{x})$ is negative definite, this sub-level set is guaranteed to be inside the region of attraction (RoA). For a polynomial system, the verification can be viewed as a Semi-Definite Programming (SDP) problem, requiring that

$$\mathrm{x}^T\mathrm{x}(V(\mathrm{x}) - \rho) - \lambda(\mathrm{x})\dot{V}(\mathrm{x}) \text{ is SOS} \tag{11}$$

, where $\lambda(\mathrm{x})$ is a polynomial with sufficient degree as the lagrange multiplier. This condition is known as the S-Procedure, and the maximization of ρ can be solved by commercial convex optimization solvers. Any state inside that RoA can be stabilized towards the fixed point through a time-invariant linear controller.

In the Trajectory Stabilization case, the Lyapunov function $V(\bar{\mathrm{x}}, t) = J(\bar{\mathrm{x}}, t)$ and the sub-level set

$$\mathcal{B}(\rho(t), t) = \{\bar{\mathrm{x}}|0 \leq V(\bar{\mathrm{x}}, t) \leq \rho(t)\} \tag{12}$$

vary with time. Similar to the demand of the terminal cost for a trajectory, here we define a terminal goal region

$$\mathcal{B}_f = \{\bar{\mathrm{x}}|0 \leq \bar{\mathrm{x}}^T\mathrm{Q}_f\bar{\mathrm{x}} \leq \rho_f\} \tag{13}$$

. To guarantee that $\forall\bar{\mathrm{x}}(t) \in \mathcal{B}(\rho(t), t) \Rightarrow \bar{\mathrm{x}}(t_f) \in \mathcal{B}_f$, it's required to have

$$\forall t \in [0, t_f], \forall\bar{\mathrm{x}}(t) \text{ s.t. } V(\bar{\mathrm{x}}, t) = \rho(t) \Rightarrow \dot{V}(\bar{\mathrm{x}}, t) \leq \dot{\rho}(t) \tag{14}$$

, where according to Eq. (8),

$$\begin{aligned}\dot{V}(\bar{\mathrm{x}}, t) = \dot{J}(\bar{\mathrm{x}}, t) = \bar{\mathrm{x}}^T\dot{\mathrm{S}}(t)\bar{\mathrm{x}} \\ + 2\bar{\mathrm{x}}^T\mathrm{S}(t)f(\mathrm{x}_0(t) + \bar{\mathrm{x}}, \mathrm{u}_0(t) + \bar{\mathrm{u}})\end{aligned} \tag{15}$$

, and

$$\rho(t_f) \leq \rho_f \tag{16}$$

. $\rho(t)$ describes a funnel-shaped volume along the trajectory, where any state inside the funnel will finally enter the terminal region, under the optimal TV-LQR controller.

III. Graph Graph and Planning

In this section, I'll introduce the algorithms for constructing graphs with every edge representing a dynamically feasible trajectory, and also examine whether these graphs can transfer kinodynamic planning to a graph searching problem. All the examples are based on the single pendulum dynamics $ml^2\ddot{\theta} + b\dot{\theta} + mgl\sin\theta = \tau$, with $m = 1$, $l = 0.5$, $b = 0.1$, $g = 9.8$.

A. Direct Connection

To better clarify the significance of funnels in graph construction, I firstly checked the effectiveness of constructing graph by trying to directly connect random points in the state space. To ensure the connection of edges, one simple method is to pre-sample enough random points, and exhaustively try the connecting each pair with trajectory optimization. To reduce the computational complexity, only neighbor pairs, defined by a Delaunay Triangulation, are considered. The implicit heuristic here is that the feasible trajectories passing a point are likely to be closed to a neighbor point.

Figure 1 shows the resultant graph. Obviously, the graph has isolated segments, and two blank regions exist at the first and third quadrant, where the failure probability of trajectory optimization is pretty high. Besides, there are two fundamental problems with the direct connection method. The first is that no matter how many points been sampled, the coverage ratio of the state space is always 0. Therefore, when a starting point and ending point is chosen, local replanning is required, and the success is not guaranteed. Secondly, the edges in graph often have much longer trajectories under the hood than the represented blue lines. Only considering one single connection between point pairs wastes the majority of dynamical information.

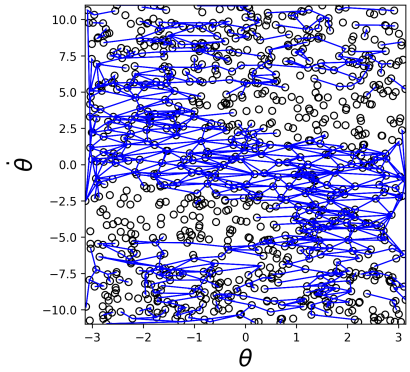


Fig. 1: Graph from directly connecting neighbor pairs of 960 random points. Black hollow circles: sampled points; Blue lines: feasible connections.

B. Random Funnel Sampling

To overcome the drawbacks of the direct connection method, one improvement here is that the point pairs will be randomly sampled in the state space. The starting points are uniformly sampled within the box $\theta \in [-\pi, \pi]$ and $\dot{\theta} \in [-3.5\pi, 3.5\pi]$, and the ending points are at the distance of π from the corresponding starting points. The result is shown in Figure 2, and the trajectories look almost uniform at the sampling region, except for the four corners and the vicinity of the origin. The trajectory distribution can be further fine-tuned through, for instance, changing the maximum time interval of the trajectory optimization, the sampling region for starting points, and the distance between the point pairs. Additionally,

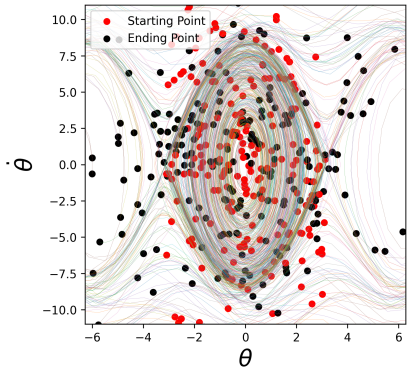


Fig. 2: 200 random trajectories

every trajectory will be explicitly represented as a funnel, as described in the LQR tree paper [6]. The first step is to backwardly integrate the matrix S, with Eq. (9. Although the ending point may not be a fixed point, I used the S from LQR as Q_f, so that $\mathrm{S}(t_f) = \mathrm{Q}_f$. The integrator used here is Scipy's *solve_ivp* function with *RK45* mode [1]. Uptill here, $\mathrm{S}(t)$, $\mathrm{x}_0(t)$, $\mathrm{u}_0(t)$ are calculated and easy to be discretized. The next step is to solve $\rho(t)$. In the origin lqr tree paper [6], $\rho(t)$ is piecewise linear at each time interval

$$\rho(t) = \begin{cases} \beta_{1k}t + \beta_{0k} & t \in [t_k, t_{k+1}) \\ \rho_f & t = t_f \end{cases} \tag{17}$$

. To gaurantee Eq. (14), an SOS program is set to be

$$\begin{aligned} &\text{find } h_1(\bar{\mathrm{x}}), h_2(\bar{\mathrm{x}}), h_3(\bar{\mathrm{x}}) \\ &\text{s.t. } \dot{J} - \dot{\rho}_k(t) + h_1(\rho_k(t) - J) \\ &\quad + h_2(t - t_k) + h_3(t_{k+1} - t) \le 0 \end{aligned} \tag{18}$$

, where h_1, h_2 and h_3 are polynomial Lagrange multiplier with sufficient degree. To optimize ρ for larger volume of the funnel, ρ at each time step needs to be as large as possible

$$\begin{aligned} &\text{maximize}_{\beta_{\cdot k}} \ \rho_k(t_k) = \beta_{1k}t_k + \beta_{0k} \\ &\text{s.t. } \rho_k(t+1) \le \rho(t_{k+1}) \\ &\qquad \text{SOS Program (18)} \end{aligned} \tag{19}$$

. However, it's quite complicated to fully optimize $\rho(t)$, especially when there's a trade-off between locally optimizing the ρ_k to increase the volume of the current funnel section or sacrificing it for larger later funnel sections. The simplifications I made was

- S is piecewise constants
- $\rho(t)$ stays constant through the trajectory

. Now the optimization problem becomes

$$\begin{aligned} &\text{maximize } \rho_k \\ &\text{s.t. } J(\bar{\mathrm{x}}, t_k) = \rho_k \Rightarrow \dot{J}(\bar{\mathrm{x}}, t_k) \le 0 \\ &\text{for } k = 0, 1, \dots N-1 \end{aligned} \tag{20}$$

[1]One caveat here is that small numerical error might lead S to be not positive definite, but for the single pendulum, that failure never appeared during this project.

, and

$$\rho_{\text{common}} = \max\{\rho_0, \rho_1, ..., \rho_f\} \tag{21}$$

. The verification of the optimization program (20) is just a sequential of separate SOS programming as (11). The major drawback of this simplified funnel implementation is that, as t decrease from t_f to t_0, ρ_k also decreases, and at some point, ρ_k becomes a number close to 0. The trick here I used was to check if $\rho_k > 0.5$, and if not, the iteration for calculating ρ_k will be terminated. Besides, as I tried to use a self-defined Q_f, eg. an identity matrix or a diagonal matrix, ρ_f will always be closed to 0, but from t_{f-1}, situation will be similar to the LQR Q_f cases. I observed that as t decrease from t_f, $\mathrm{S}(t)$ will quickly get close to (but not necessarily converge to) S corresponding to the LQR solution of $(\mathrm{x}_0, \mathrm{u}_0)$, independent on the choice of S_f in a wide range. It's reasonable because as trajectories are long enough, the influence of the terminal cost will be very small. These analysis shows that the early termination might be related to the change of S, but the proof and explanation need further investigation.

Combining the simplified optimization problem (20) and the sampled trajectories in Figure (2), I got a group of funnels shown below. The direction of the funnels are consistent with the vector field in the state space of the single pendulum.

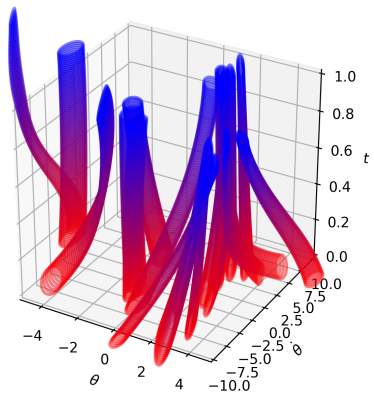


Fig. 3: Funnels constructed from optimization program (20) and Eq. (21). The vertical axis is the inverse of nominal $t \in [0, 1]$. Funnel number is limited here for better visualization. R = 2.

There are four tricks for optimizing the funnel construction process. The first one is to force the new ending points outside of the existing terminal funnels, for better exploration. Secondly, once a trajectory's SOS programming is early terminated, the ending point of the next trajectory will be the terminal state, and the staring points is the same. This trick accelerates the speed of construction $\sim$6 times, and the ratio of these deterministic mode trajectories occupies $\sim 25\%$. This vast acceleration further implies the relationship between the rapid change of S and the early termination. Additionally, more connections between funnels is also more favorable for the planning in the next section. The third trick is to discard those funnels with too short time steps, so that more informative

funnels will be conserved. At last, I decreased R from 20 to 2, because the torque limit is not considered yet, and that reduces the difficulty of trajectory optimization and the failure ratio of funnel construction.

C. Clustering of funnels

One extra interesting phenomenon is the clustering of funnels. If R is high, eg. equal to 20, the funnels tend to grow around certain centers. From Figure (4), it clear that funnels forms three clusters around three fixed points, i.e. $(-\pi, 0)$, $(0, 0)$ and $(\pi, 0)$. Besides, the funnels around the top fixed point have have cross sections with much more narrow ellipse when $t_{\text{inverse}} = 1$. Note that the narrow degree can be described by the eigenvalues of S. Combining the clustering and the early termination, one guess here is that different clusters represents different skills, and a hierarchical representation can be realized by firstly using machine learning algorithms to divide the cases, by the eigenvalues and eigenvectors of S from LQR. That I think is related to the embedding/encoding of funnels.

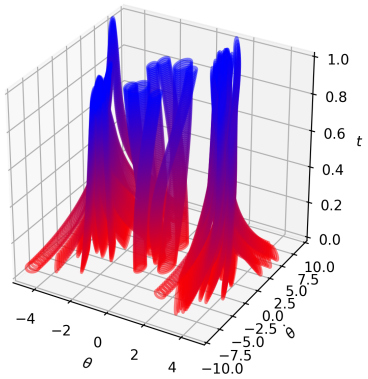
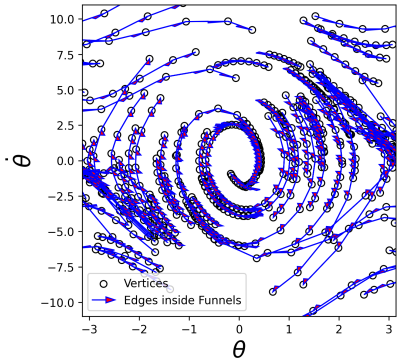


Fig. 4: 40 funnels constructed with R = 20

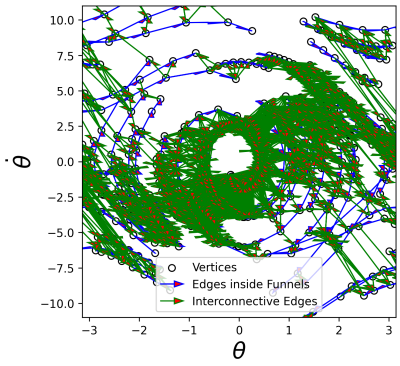
D. Planning with Funnels

Back to the main line of this project, with the funnels, we could examine their capability for future planning.

However, there're several algorithmic and geometric difficulties. Since the starting points of the funnels here are not controllable, it's hard to simply connect tails and heads of funnels to construct a graph. Besides, there could be many interconnection at the middle of trajectories. Another difficulty is how to online shrink the value ρ of a funnel to make its final ellipse fully contained in the other funnel. I haven't had time to dive into that geometry problem. The third one comes from the fast judgement of point containment in all other ellipses, with their ρ changing during the time of planning. That means the graph will dynamically change as the planning continues and new planning tasks show up. Exhaustively checking this at every iteration can be quite time consuming, if no efficient and parallel algorithm is used. Lastly, the trajectories inside funnels is continuous, and thus have infinite internal points.



(a)



(b)

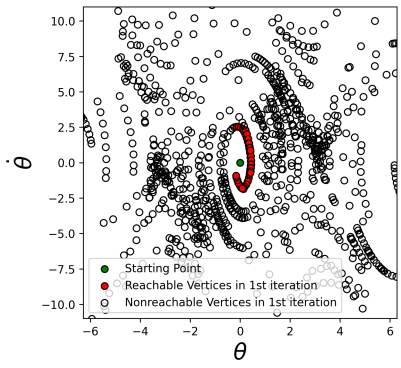
Fig. 5: Graphs constructed after simplifications. Every trajectory is discretized into 8 points.(a) Graph without interconnection between trajectories. Blue arrows are the edges of the graph. (b) Graph with interconnections between trajectories. Green arrows are the interconnections.

For the purpose of demonstration, I simplified the problem to a less rigorous reachability check problem. Each funnel or trajectory is discretized into several pieces, and if one point is inside the RoA of the other points, then the connection is made, as if the movement is feasible and stable. Comparing Figure (5) with Figure (1), the edges here are more compliant with the pendulum vector field. Interestingly, the number of interconnection edges are 10 times more than the edges inside each funnel.

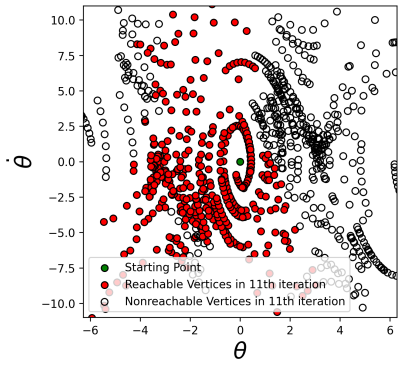
The last task of the project is to use the graph to swing up the pendulum, a trajectory that doesn't exist in our funnel set. Still, for simplicity, edges here are viewed as guaranteed movements. The reachable vertices from the origin in the graph are updated in each iteration, until no new vertices is added. Shown in Figure (6), at the first iteration, only the vertices very closed to the origin are reachable. After 21 iterations, most of the vertices are reachable, include the ones in the vicinity of the top fixed point.

IV. Conclusion

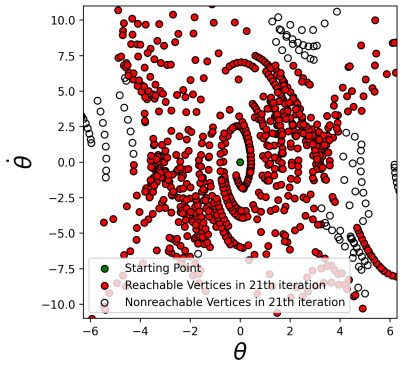
In this project, I started with directly connecting points in the state space with trajectory optimization. Its problems can be largely solved by replacing trajectories with funnels.



(a)



(b)



(c)

Fig. 6: Iterative reachability check for the origin. (a) 1st iteration, (b) 11th iteration, (c) 21th iteration

For the funnel graph construction, several simplification and tricks was chosen to reduce the complexity, but still, the funnels are rigorously provided by the SOS Programming. Lastly, to verify the capability of the graph for kinodynamic planning, I checked the reachability of the origin conceptually. The result shows that from origin, most region for point sampling is reachable. Therefore, the funnel graph planning is generalizable.

In the future, there're several possible followups. To controllably construct better graphs, it's necessary to fully optimize the $\rho(t)$ along the full trajectory. Except the optimization ideas in the LQR tree paper [6], another paper [9] provides even larger funnels by iteratively optimizing V, ρ and $\bar{u}$. Besides, as shown in Figure (5), the graph is far from uniform, partially because the duplication check was only done for the terminal state. To reduce the duplication of funnels, some efficient sampling algorithm for the state space should be incorporated. Thirdly, at the end of Section (III-B), the potential combination of machine learning clustering and the LQR might also provides interesting results. In the planning section, the reachability proof should be more rigorous, potentially accelerated by parallel computing. It will be more convincing if a full trajectory can be planned for swinging up the pendulum at runtime. With all these work done, more challenging tasks can be torque-limit pendulum, acrobot, cart-pole or even high dimensional cases.

REFERENCES

[1] Bruce Donald, Patrick Xavier, John Canny, and John Reif. Kinodynamic motion planning. *Journal of the ACM (JACM)*, 40(5):1048–1066, 1993.

[2] Steven M LaValle and James J Kuffner Jr. Randomized kinodynamic planning. *The international journal of robotics research*, 20(5):378–400, 2001.

[3] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7):846–894, 2011.

[4] Alejandro Perez, Robert Platt, George Konidaris, Leslie Kaelbling, and Tomas Lozano-Perez. Lqr-rrt*: Optimal sampling-based motion planning with automatically derived extension heuristics. In *2012 IEEE International Conference on Robotics and Automation*, pages 2537–2542. IEEE, 2012.

[5] Lydia E Kavraki, Petr Svestka, J-C Latombe, and Mark H Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation*, 12(4):566–580, 1996.

[6] Russ Tedrake, Ian R Manchester, Mark Tobenkin, and John W Roberts. Lqr-trees: Feedback motion planning via sums-of-squares verification. *The International Journal of Robotics Research*, 29(8):1038–1052, 2010.

[7] Pablo A Parrilo. *Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization*. PhD thesis, California Institute of Technology, 2000.

[8] Anirudha Majumdar and Russ Tedrake. Funnel libraries for real-time robust feedback motion planning. *The International Journal of Robotics Research*, 36(8):947–982, 2017.

[9] Anirudha Majumdar, Amir Ali Ahmadi, and Russ Tedrake. Control design along trajectories with sums of squares programming. In *2013 IEEE International Conference on Robotics and Automation*, pages 4054–4061. IEEE, 2013.