JOINT EXPLOITATION OF LOCAL METRICS AND GEOMETRY IN SAMPLING-BASED PLANNING

Vincent Scott Pacelli

A THESIS

in

Robotics

Presented to the Faculties of the University of Pennsylvania in Partial
Fulfillment of the Requirements for the Degree of Master of Science in Engineering

2017

_____
Daniel E. Koditschek
Supervisor of Thesis

_____
Camillo J. Taylor
Program Director

Contents

Chapter 1

Introduction

Recent advances in robotics have allowed autonomous systems to be deployed in a diverse range of new settings from automobiles [1], to warehouses [2], to automated delivery systems [3]. All of these systems require motion planning algorithms capable of efficiently producing safe trajectories in high-dimensional spaces to operate safely in their dynamic environments. Motion planning is a fundamental challenge in robotics and has received a large amount of attention over the past few decades. One class of planner that has been very successful in recent years is sampling-based motion planners. These probabilistic algorithms attempt to approximate the connectivity of the underlying configuration space of the robot by developing a graph of randomly sampled configurations whose edges correspond with the connectivity of the samples in the underlying configuration space. Part of the success of these algorithms is that they are able to function efficiently even when obstacles in the robot workspace are not easily representable in terms of robot states. Their appeal also stems from their capability to handle dynamical constraints and motion uncertainty.

A key feature of sampling-based planners is their ability to incorporate information about the underlying dynamical system through the use of metrics that reflect properties of the underlying dynamical system. If made available, the algorithm will

naturally exploit the system dynamics to rapidly explore the space. Typically, the best metric for a given dynamical system is one that reflects the true distance (or other notion of "cost") that must be traversed in order to reach a given state. However, computing such a metric is often equivalent to solving the path-planning problem itself, so a tractable – and often locally applicable – approximation is used instead. Moreover, these metrics only reflect the system dynamics and not features of the workspace in which the robot operates. This thesis will show how locally informative metrics related to the robot's dynamical model can be combined with local workspace geometry to produce a sampling-based planner better captures the connectivity of the configuration space under a fixed number of samples.

1.1 Contributions

This thesis addresses how the local geometry of the workspace around a system state can be combined with local metrics describing system dynamics to improve the connectivity of the graph produced by a sampling-based planner over a fixed number of configurations. This development is achieved through generalization of the concept of the local freespace introduced in [4] to inner products other than the Euclidean inner product. This new structure allows for naturally combining the local freespace construction with a locally applicable metric.

The combination of the local freespace with two specific metrics is explored in this work. The first metric is the quadratic cost-to-go function definined by a linear-quadratic regulator, which captures the local behavior of the dynamical system. The second metric is the Mahalanobis distance for a beliefstate in a beliefspace planner. Beliefspace planners reason over distributions of states, called beliefstates, to include modeled uncertainty in the planning process. The Mahalanobis distances metric for a given beliefstate can be exploited to include notions of risk in local freespace construc-

tion. Numerical simulations are provided to demonstrate the improved connectivity of the graph generated by a sampling-based planner using these concepts.

1.2 Organization

The remainder of this thesis is organized as follows. In Chapter 2, related work in sampling-based motion planning is summarized. Chapter 3 provides a brief development of a generic sampling-based planner along with some specific steering and distance functions from literature. Chapter 4 defines the notion of the *generalized local freespaces* and demonstrates applicability to kinodynamic path planning and belief-space planning. In Chapter 5, a steering procedure that uses the generalized local freespace to exploit local information is defined. Numerical simulations demonstrating the effectiveness of this procedure in sampling-based planners are also provided. Finally, in Chapter 6, results are summarized and potential extensions are stated.

Chapter 2

Relevant Work

2.1 Sampling-Based Motion Planning

Due to their success, sampling-based motion planners and their variants have been widely studied. While many variants have been considered by researchers, they tend to fall into one of two classes: Probabilistic Road Maps (PRMs) [5] and rapidly-exploring random trees (RRTs) [6]. Both styles of planner have seen successful application to a wide number of areas including, deformable robotics [7], multi-agent UAV systems [8], spacecraft [9], and manipulators and other high-dimensional systems [10]. Both of these algorithms are *probabilistically complete*, i.e. they will find a solution trajectory with a probability of one as the number samples goes to infinity [11]. A formal descriptions of an RRT style planner is given Chapter 3 for reference throughout the paper.

The goal of a PRM algorithm is to produce a roadmap, i.e. a graph of configurations with edges representing known safe connections, that can be queried repeatedly using efficient graph-search algorithms when the robot needs to reach a different region of the workspace. This is done by first sampling some number of configurations from the configuration space and treating these as vertices. Then, a graph is built

by checking pairwise whether one vertex can be reached from another and adding an edge to the graph if so. Originally, the check for reachability was posed as a simple line-of-sight check, but many alternatives have been considered and are discussed in Section 2.2.

Since their output is intended for repeated use, these algorithms are typically not run online, and variants, for example [12], often introduce computationally expensive procedures (in [12], Monte-Carlo sampling) to produce trajectories with additional properties that can be efficiently queried online. Since a roadmap can be slow to produce, they are most effective in highly structured environments with fixed obstacles like a warehouse or factory.

In contrast, the RRT family of algorithms emphasizes building a single trajectory that navigates the robot towards a goal. Often, in highly dynamic environments, RRT-style algorithms are repeatedly run in an closed-loop online to account for moving obstacles and tracking error [1]. As such, special attention has been paid to the application of RRT planning to systems with complicated dynamics in both the extension [13–15] and sampling step [16].

The RRT works by sequentially sampling random configurations, identifying the nearest configuration currently in the tree, and checking if the robot can be moved a fixed distance toward the sample (or to the sample directly if it is close enough) without colliding with the environmental clutter. It is important to note that, the RRT algorithm is not optimal in the presence of a cost function [11]. However, there does exist a popular variant known as RRT* that does guarantee optimality by locally rewiring the graph locally around new vertices as they are added to enforce optimality [11].

2.2 Incorporating Geometric Information in Sampling-Based Planning

A fundamental challenge faced by sampling-based planners is the "narrow corridor" problem. While probabilistic completeness of the RRT and PRM algorithms guarantees that with enough samples a path will be found to the goal, common pathological environments exist that require prohibitively large numbers of samples to identify such a trajectory. One such instance is when the environment features long, narrow corridors [17, 18]. This problem is partially explained by the fact that uniform sampling strategies combined with straight-line local planning, as originally posed for both the RRT and PRM algorithms, have a distinct Voronoi bias [13]. Thus, sampling in this manner will explore large, open regions of the workspace faster than small, narrow ones. A wide body of literature has been dedicated to overcoming this challenge by incorporating geometric information into either the sampling step or steering step of the algorithm. This section will briefly review methods for doing both.

Many methods have been considered to construct a sampling distribution that reflects the workspace geometry. In [17], the freespace was "dilated" to allow penetration of a PRM partially into obstacle regions. After constructing a PRM with uniform sampling in this modified environment, vertices of the graph that lay within obstacles were removed and replaced with vertices sampled from the neighborhood of the offending vertex. In [19], uniform sampling was replaced with sampling from a mixture of Gaussian distributions centered on obstacles, therefore biasing the planner toward cluttered regions of the space. A similar method was used in [20, 21], where regions containing high quantities of samples both in and out of collision are favored during sampling. Finally, in [22], if there is no line-of-sight between two vertices, the midpoint between the vertices is also tested and added to the graph if it lies in

the freespace. This modification effectively biases sampling to be heavier in obstacle dense regions.

Many researchers have also studied more flexible ways to determine whether a connection should be made between two states than a simple line-of-sight test – thus incorporating geometry into the local planning step. For example, [23] combined PRM and RRT planning by spawning RRTs from vertices in the PRM that have few connections to other vertices. The vertices from the RRT were then added to the PRM after a stopping condition was reached. A similar approach was used in [24], except RRTs were used to connect all vertices in the PRM, resulting in fewer necessary PRM vertices. Other researchers have considered using searched-based local planners to connect configurations. In [25, 26], the configuration space is discretized and a local A* search is performed in an attempt to find a connection to a nearby node. In [27] the result of an initial any-angle search of the configuration space was used to bias the expansion of an RRT toward the goal while maintaining dynamic feasibility. Finally, in [4,28,29], a local subset of the freespace was built around each configuration out of maximum-margin separating hyperplanes between the configuration and each obstacle. This structure reflects the local geometry of the workspace, and steering was done by adding the closest point within the constructed polytope to sampled configuration to the graph.

2.3 Incorporating Dynamics in Sampling-Based Planning

Much attention has been paid to incorporating dynamical constraints into sampling-based planning algorithms. Typically, kinodynamic sampling-based planners differ from purely kinematic planners in that the local steering procedure must satisfy differential constraints and the local metric used to identify nearby tree vertices reflects

(or attempts to approximate) the true cost-to-go between states. In [13], the probabilistic completeness of a kinodynamic RRT planner was proved for finite input spaces. This algorithm was refined in [30] to be both asymptotically optimal and complete for continuous input spaces provided that the system is locally controllable. However, merely selecting a control input that drives the system closest to a new sample will not always produce a complete planning algorithm [31].

Many steering procedures have been proposed for sampling-based kinodynamic planning. For common systems, domain knowledge can often be exploited to develop useful, efficient local planners. For example, in [30], local planning strategies that guarantee completeness were introduced for continuous-time double integrator and kinematic cart dynamics. Similarly, various closed-loop control policy for local steering of non-holonomic kinematic carts have been proposed [32, 33]. Other local planning policies have been proposed for more general classes of dynamical systems using linearizations of the system dynamics around the state being extended [15, 16, 34]. This methodology assumes that designing good local control inputs for the non-linear system is challenging, but that sampling is computationally cheap and that, in the limit of infinite samples, linearizing about each state will accurately capture the non-linear system dynamics.

Similarly, various distance metrics for nonlinear systems have been studied. For non-holonomic kinematic carts [35] learns a distance metric through statistically approximating the cost-to-go between states connected via the control procedure in [32] using large quantities of samples generated offline. Meanwhile, [33] uses the Lyapunov function created by its local control strategy to estimate the cost-to-go of the kart. In [36], a similar learning approach is applied but to general nonlinear systems systems where samples are generated using an offline iterative finite-horizon optimal control policy. The approach used by [16] for create a metric for nonlinear systems is to use the cost-to-go function for an infinite-horizon linear-quadratic regulator designed for

a system linearized at one of the states in question.

Finally, sampling-based planning algorithms have been used to address problems in beliefspace planning, where noise is included as part of the dynamical model. One common approach is to apply an iterative optimization procedure to incorporate uncertainty into to a nominal path through the workspace computed via a sampling-based algorithm [37,38]. However, this iterative procedure will only produce trajectories that lie in the same homotopy class as the nominal trajectory, which may limit the robustness of the final solution. In [39], the RRT* algorithm was directly extended to directly account for uncertainty during the sampling-based planning procedure by ensuring sets approximating the uncertainty of the system remain collision free as the space is explored. Others have included uncertainty in the planning procedure by using Monte-Carlo methods to estimate the probability of successfully reaching a new state [12, 40].

Chapter 3

Background: Steering and Metrics in Sampling-Based Planning

This section outlines the structure of a sampling-based planning algorithm. A brief development of some existing local metrics and steering procedures from literature is also included.

3.1 Generic Sampling-Based Planning Algorithm Components

The following functions are the fundamental building blocks of an RRT-style sampling-based planning algorithm [13].

- `Sample`($\mathcal{X}$) draws an independent and identically distributed random sample from the closed, compact set $\mathcal{X} \subset \mathbb{R}^n$. In this thesis, the distribution will always be uniform, but other distributions such as those mentioned in Section 2.2 may be used.
- `Dist`(x_1, x_2) computes the distance between x_1 and x_2 using some metric. The Euclidean distance metric $\|x_1 - x_2\|_2$ is a standard choice.

- `NearestNeighbor`$(\mathbf{x}_{[N]}, x)$ where $\mathbf{x}_{[N]} = \{x_1, \ldots, x_N\}$ returns the closest member of the finite set $\mathcal{X} \subset \mathbb{R}^n$ to the state $x \in \mathbb{R}^n$ with respect to `Dist`. That is, `NearestNeighbor`$(\mathbf{x}_{[N]}, x) = \arg\min_{i \in \{1,\ldots,N\}} \texttt{Dist}(x_i, x)$.

- `Steer`$(x_0, x_g) = (u_0, \ldots, u_K)$ returns the control inputs $(u_0, \ldots, u_K)$ that defines a finite trajectory $(x_0, u_0), \ldots, (x_K, u_K)$ that attempts to drive the system from x_0 to x_g using a local control policy for a dynamical system $x_{k+1} = f(x_k, u_k)$. Probabilistic completeness of the overall planner depends heavily on the implementation of this function [31] and how "close" the local controller can bring the system to x_g. The control inputs $(u_0, \ldots, u_K)$ are returned.

- `CollisionFree`$(\mathbf{x}_{[K]}, \mathcal{F}) = \bigwedge_{k=2}^{K} (\{\alpha x_{k-1} + (1-\alpha)x_k \mid \alpha \in [0,1]\} \subset \mathcal{F})$ returns `true` if, for all $i = 2, \ldots, N$, the line segment joining x_{k-1} and x_k is in $\mathcal{F}$.

The metric selected for `Dist` is ideally one that describes the distance between two states subject to any existing dynamical and workspace constraints. Identifying such a metric is generally as hard as solving the path planning problem, so approximations are commonly used [16, 35, 36, 41].

3.2 Linear-Quadratic Regulator Based Metric and Steering

A linear-quadratic regulator (LQR) can be used to define both a locally applicable metric and control policy for use in a sampling-based algorithm. This section reviews some basic results from the LQR literature. The formulation of the LQR problem and its solution is derived from [42]. The section concludes with a demonstration of how the LQR can be used to define both a local steering procedure and metric.

3.2.1 LQR Formulation

For a linear time-invariant system, the LQR problem is concerned with finding a feedback control law u_k that optimally stabilizes the system over an infinite time horizon function with respect to the cost function

$$J(u_1, \ldots, u_K) = \sum_{k=1}^{K} x_k^{\mathrm{T}} Q x_k + u_k^{\mathrm{T}} R u_k \tag{3.1}$$

and subject to the system dynamics

$$x_{k+1} = Ax_k + Bu_k \qquad\qquad x_k \in \mathbb{R}^n \tag{3.2}$$

Here, $Q \in \mathbb{S}_{++}^n, R \in \mathbb{S}_{++}^m$ define penalties on the state and input error respectively. Throughout this paper, $\mathbb{S}_{++}^n$ is used as the set of positive-definite matrices.

The optimal feedback law is computed through a dynamic programming approach. For a fixed horizon K, the optimal sequence of actions $u_1, \ldots, u_K$, the Principle of Optimality states that the optimal action at each timestep can be found by recursively solving for the optimal action backwards starting with u_K. This process produces the recursive definition of the cost-to-go metric. At timestep k, the cost-to-go is given by $x_k^{\mathrm{T}} P_k x_k$ where

$$P_k = A^{\mathrm{T}} P_{k+1} A + Q - A^{\mathrm{T}} P_{k+1} B (B^{\mathrm{T}} P_{k+1} B + R)^{-1} B^{\mathrm{T}} P_{k+1} A \tag{3.3}$$

As $k \to \infty$, the equation in (3.3) converges to a steady-state if the system is controllable [42, 43], producing the discrete algebraic Riccati equation (DARE)

$$P = A^{\mathrm{T}} P A + Q - A^{\mathrm{T}} P B (B^{\mathrm{T}} P B + R)^{-1} B^{\mathrm{T}} P A \tag{3.4}$$

where $P \in \mathbb{S}_{++}^n$. DAREs are well-studied and can be solved efficiently using nu-

merical methods, e.g. [44]. For later use in algorithm descriptions, the function $\texttt{DARE}(A,\ B,\ Q,\ R)$ is defined to return the matrix $P \in \mathbb{S}^n_{++}$ that is the solution of (3.4).

Once found, P defines a quadratic cost-to-go function $V(x_k) = x_k^{\mathrm{T}} P x_k$. Moreover, P defines the infinite-time optimal stabalizing feedback control law $u_k = F x_k$ where $F = -(B^{\mathrm{T}} P B + R)^{-1} B^{\mathrm{T}} P A$.[1] Proof of stability of this controller comes from the fact that $V(x)$ can be treated as a Lyapunov function for the system

$$x_{k+1} = (A + BF)x_k \tag{3.5}$$

The function V is clearly positive-definite. The negative-definiteness of $V(x_{k+1}) - V(x_k) = x_k^{\mathrm{T}}((A+BF)^{\mathrm{T}} P (A+BF) - P)x_k$ is guaranteed by the fact that (3.4) can be factored into the form

$$(A + BF)^{\mathrm{T}} P (A + BF) - P = -(Q + F^{\mathrm{T}} R F) \tag{3.6}$$

and the positive-definiteness of $Q + F^{\mathrm{T}} R F$.

3.2.2 The Use of LQRs in Sampling-Based Planning

The LQR has two applications to sampling-based planning for time-invariant linear systems: defining a metric via the cost-to-go function $V(x)$ and steering policy [16] via the feedback law. In addition, these methods are applicable to general dynamical systems through linearization because both steering and nearest-neighbor computation are local procedures about the linearized state – especially when the statespace has been densely sampled.

The LQR distance function, `LQRDist` is given in Algorithm 1. For two states

[1] For general linear systems, it is assumed that the system is observable so that the feedback controller can be applied.

$x_1, x_2 \in \mathbb{R}^n$, the system is first linearized about x_1 and a nominal control input.[2] The DARE (3.4) is then solved for P. Then, using the following notation for the Mahalanobis distance [45]

$$\|x\|_S = \sqrt{x^{\mathrm{T}} S^{-1} x} \qquad\qquad S \in \mathbb{S}^n_{++} \tag{3.7}$$

we define the *LQR metric* to be $\|x_1 - x_2\|_{P^{-1}}$. This metric is the square root of the cost, as defined by (3.1), to travel from x_1 to x_2 according for the linear system under the LQR control policy.

Input: x_1, x_2
$A \leftarrow \frac{\partial f(x_1, u_0)}{\partial x}$, $B \leftarrow \frac{\partial f(x_1, u_0)}{\partial u}$;
$P \leftarrow \texttt{DARE}(A,\ B,\ Q,\ R)$;
return $\|x_1 - x_2\|_{P^{-1}}$

Algorithm 1: Definition of `LQRDist` (x_1, x_2)

The steering policy `LQRSteer` is given in Algorithm 2. Let $x_0, x_g \in \mathbb{R}^n$ be system states. The function will attempt to plan a trajectory from x_0 to x_g subject to the dynamical constraint $x_{k+1} = f(x_k, u_k)$. To do so, the system is linearized about x_0 and an input u_0. In [16], the zero input was used for u_0, which corresponds to not favoring any particular action.

It is assumed that, for any state in the ϵ-ball $\mathcal{B}_\epsilon(x_0)$, linearization about this state will produce the same A and B matrices and that (A, B) is controllable. The procedure will terminate if this ball is left because the linearization is no longer consistent with the nonlinear dynamics. Additionally, since the LQR policy will only cause the system to approach x_g asymptotically (i.e. $\|x_k - x_g\|_2 \to 0$ as $k \to \infty$), `LQRSteer` is terminated after $K \in \mathbb{N}$ iterations. Finally, the control inputs computed according to this feedback law are returned.

The system is linearized about x_0 and not x_g because, if x_0 is far from x_g, lin-

[2] In [16], the zero input is used, but the choice may be different if the context suggests a natural choice.

earization about x_g may not be consistent with the local behavior of the dynamical system in the neighborhood of x_0.

Input: x_0, x_g
$A \leftarrow \frac{\partial f(x_0,0)}{\partial x},\ B \leftarrow \frac{\partial f(x_0,0)}{\partial u}$;
$P \leftarrow \texttt{DARE}(A, B, Q, R)$;
$F \leftarrow -(B^{\mathrm{T}}PB + R)^{-1}B^{\mathrm{T}}PA$;
for $k = 0, \ldots, K-1$ **do**
  $u_k \leftarrow F(u_1,\ \ldots,\ u_K)$;
  $x_{k+1} \leftarrow Ax_k + Bu_k$;
  if $x_{k+1} \notin \mathcal{B}_\epsilon(x_0)$ **then**
    return $(u_0, \ldots, u_{k-1})$
  end
end
return $(u_0,\ \ldots,\ u_{K-1})$

Algorithm 2: Definition of $\texttt{LQRSteer}(x_0,\ x_g)$

3.3 A Metric from Beliefspace Planning

A second motion-planning setting that provides a natural metric is beliefspace planning – a generalization of traditional motion planning to include modeled probabilistic uncertainty. Instead of planning over the set of states, the algorithm plans over the set of distributions of states [39,46]. In general, representing arbitrary continuous state distributions is computationally intractable. However, for linear systems under Gaussian noise, i.e.

$$\begin{aligned} x_{k+1} &= Ax_k + Bu_k + w_k & w_k &\sim N(0, W) \\ y_k &= Cx_k + v_k & v_k &\sim N(0, V) \end{aligned} \tag{3.8}$$

with $x_0 \sim N(0, W_0)$, the distribution of the state will always be Gaussian with a mean and covariance given by the Kalman dynamics

$$\begin{aligned} \bar{x}_{k+1} &= A\bar{x}_k + Bu_k \\ \Sigma_{k+1|k} &= A\Sigma_k A^{\mathrm{T}} + W \\ \Sigma_{k+1} &= \Sigma_{k+1|k} - \Sigma_{k+1|k} C^{\mathrm{T}} (C\Sigma_{k+1|k} C^{\mathrm{T}} + V)^{-1} C\Sigma_{k+1|k} \end{aligned} \tag{3.9}$$

The Kalman dynamics provide an optimal filtering strategy for any quadratic error function, including mean-squared error [47]. Together, $(\bar{x}_k, \Sigma_k)$ define a Gaussian distribution that encapsulates the uncertainty about the state that is present only when measurements $y_0, \ldots, y_k$ are available. Together, we will refer to $(\bar{x}_k, \Sigma_k)$ as the belief b_k and and treat x_k as the corresponding Guassian random variable $x_k \sim N(\bar{x}_k, \Sigma_k)$ in this setting.

In this domain, $\| \cdot \|_{\Sigma_k}$, the Mahalanobis distance [45] at time k, which will be referred to as the *beliefspace metric*, forms a natural metric for use in planning. This metric weights the differences between states in directions with a higher variances less than those with lower variances. This metric is used by beliefspace planners to incorporate uncertainty with respect to both the dynamical model and obstacles into the planning process in order to bias the system to move in directions with more certainty [48, 49]. In addition, the LQR metric discussed in the previous section is still applicable in this setting because the LQR control policy is still optimal in expectation with respect to (3.1) under the noise model in (3.8) [12].

3.4 Rapidly-Exploring Random Tree Algorithm

A generic version of the RRT algorithm [13] is now presented. Let $\mathbb{R}^n$ be the statespace of a robot and assume it is decomposed into two disjoint regions: an

obstacle set $\mathcal{O}$ and a freespace $\mathcal{F} = \mathbb{R}^n \setminus \mathcal{O}$. It is assumed that $\mathcal{F}$ is a compact set. The RRT algorithm is then given in Algorithm 3.

```
1  V ← {x_0}, E ← ∅;
2  for i = 1, ..., N do
3  |   x_rand ← Sample(F);
4  |   x_0 ← NearestNeighbor(V, x_rand);
5  |   (u_0, ..., u_{K-1}) ← Steer(x_near, x_rand) ;        /* Note:  x_k = x_near */
6  |   for k = 0, ..., K − 1 do
7  |   |   x_{k+1} ← f(x_k, u_k)
8  |   end
9  |   if CollisionFree({x_0, ..., x_K}, F) then
10 |   |   V ← V ∪ {x_new};
11 |   |   E ← {(x_0, x_K)};
12 |   end
13 end
```

Algorithm 3: Rapidly-Exploring Random Tree Algorithm

In [13], this algorithm was proved probabilistically complete when the input space is discretized. In that setting, `Steer` selects the input from a finite set of inputs that minimizes $\texttt{Dist}(x_{near}, x_{rand})$. In [30], an asymptotically optimal version of Algorithm 3 was introduced for continuous input spaces. This version of the algorithm was proved probabilistically complete if the the dynamical system is features a relaxed definition of local controllability. In addition, [30] provides examples of weak locally controllable steering procedures for linear systems and differential drive systems. In general, however, the RRT algorithm is not complete. See [31] for a counterexample.

The rest of this thesis is focused on how local metrics, such as the LQR metric and beliefspace metric, can be combined with local geometric information during the steering process. Existing steering procedures for dynamical systems, e.g. [15, 16, 32, 33, 48], often use local metrics to determine local actions, but do not consider any local geometric information. Through generalization of the concept of the local freespace introduced in [4], a local subset of $\mathcal{F}$ can be constructed that simultaneously reflects a local metric and local geometric information. This subset is then used to define a

local steering procedure for the dynamical system.

Chapter 4

Generalized Local Freespaces

This chapter generalizes the concept of the local freespace introduced in [4, 28, 29] by relaxing the choice of the Euclidean inner product to a more general class of inner products. This abstraction provides a way to organically incorporate local information about the underlying dynamical system into the local freespace by altering the metric used in its construction. A topological interpretation of this generalization in terms of an affine change of coordinates is also provided. The chapter also gives example generalized local freespaces using the LQR and beliefspace metrics.

4.1 Local Freespace Construction with Generalized Metrics

In [4, 28, 29], the concept of the local freespace as a way to incorporate the local geometry of the robot environment into the steering procedure for a kinematic robot was introduced. This steering procedure was then demonstrated empirically to be very effective at exploring narrow passages of the configuration space.

Formally, consider a setting where $\mathcal{O}_1, \ldots, \mathcal{O}_M \subset \mathbb{R}^n$ are open convex sets of states that represent obstacles. Together, they define the *freespace* $\mathcal{F} = \mathbb{R}^n \setminus (\bigcup_{i=1}^{M} \mathcal{O}_i)$,

which is assumed to be compact. Additionally, the *metric projection* of x onto some set $\mathcal{A}$ is defined to be $\Pi_{\mathcal{A}}(x) = \arg\min_{y \in \mathcal{A}} \|x - y\|_2$. The Euclidean local freespace is given in the following definition[1]

Definition 4.1.1 (Euclidean Local Freespace [4])**.** *The* ***Euclidean local freespace*** *is defined as the closed polytope*

$$\begin{aligned} \mathcal{LF}(x) &= \{\, x' \in \mathcal{F} \mid \|x' - x\| \leq \|x - s_i\|, s_i = \Pi_{\mathcal{O}_i}(x), i = 1, \ldots, M \,\} \\ &= \left\{ x' \;\middle|\; (s_i - x)^{\mathrm{T}} \left(x' - \frac{x + s_i}{2} \right) \leq 0, s_i = \Pi_{\mathcal{O}_i}(x), i = 1, \ldots, M \right\} \end{aligned} \tag{4.1}$$

For a given state x, $\mathcal{LF}(x)$ is a polytope containing x whose boundary is given by the maximum-margin separating hyperplanes between x and s_i, the projection of x onto the ith obstacle. Figure 4.1 contains an example of the local freespace defined in (4.1). When the obstacle $\mathcal{O}_i$ is a polytope, $\Pi_{\mathcal{O}_i}(x)$ is easily computed by solving a convex linearly-constrained quadratic program [50]. These may be solved numerically via efficient algorithms such as interior point [50] and active set [51] methods.

The local freespace constructed in Definition 4.1.1 can be generalized by allowing for a wider class of inner products to be used in the definition of the Euclidean local freespace. Specifically, for vectors $x, y \in \mathbb{R}^n$, the inner product is generalized from $x^{\mathrm{T}}y$ to $x^{\mathrm{T}}S^{-1}y$ for some $S \in \mathbb{S}^n_{++}$. This inner product induces the Mahalanobis distance as a norm: $\|x\|_S = \sqrt{x^{\mathrm{T}}S^{-1}x}$. For a given S, we define the metric projection using $\|\cdot\|_S$ to be $\Pi_{\mathcal{A}}(x; S) = \arg\min_{y \in \mathcal{A}} \|x - y\|_S$. Then, the definition of the Euclidean local freespace is generalized as follows:

Definition 4.1.2 (Generalized Local Freespace)**.** *For a matrix $S \in \mathbb{S}^n_{++}$, the* ***gener-***

[1]The definition given in [4] includes a constraint used to represent the robot radius. Here, the robot is assumed to be a point robot. A similar constraint is treated in Section 4.3.

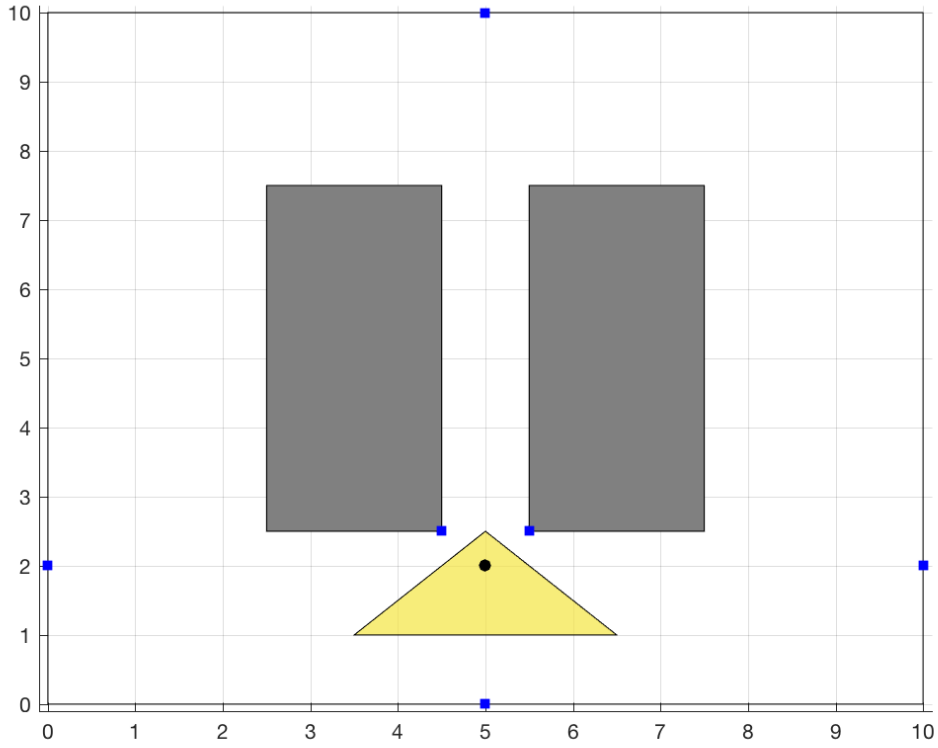


Figure 4.1: An example of the Euclidean local freespace. The robot workspace is $[0, 10] \times [0, 10]$. Obstacles are dark gray polytopes. The Euclidean local freespace is shown in gold about the state $[5\ 2]^{\mathrm{T}}$ (black circle). Blue squares designate the closest point in each obstacle set to the state under the Euclidean metric.

alized local freespace *is defined as the closed polytope*

$$\mathcal{LF}(x;S) = \{\, x' \in \mathcal{F} \mid \|x'-x\|_S \leq \|x-s_i\|_S, s_i = \Pi_{\mathcal{O}_i}(x;S), i=1,\ldots,M \,\} \tag{4.2}$$
$$= \left\{ x' \;\middle|\; (s_i-x)^{\mathrm{T}} S^{-1} \left(x' - \frac{x+s_i}{2} \right) \leq 0, s_i = \Pi_{\mathcal{O}_i}(x;S), i=1,\ldots,M \right\}$$

The metric defined by $\|\cdot\|_S$ weights differences between states in directions defined by the eigenvectors of S depending on the value of the corresponding eigenvalue. Since the inner product is taken with respect to S^{-1}, directions with smaller eigenvalues contribute more to the value of $\|\cdot\|_S$ than directions with larger eigenvalues.

Each hyperplane defining $\mathcal{LF}(x;S)$ can be interpreted as limiting motion from x in directions that produce a collision with the corresponding obstacle in the shortest distance under the metric $\|\cdot\|_S$. This observation comes from the fact that, for a given x and $\mathcal{O}_i$, s_i is the closest point in the sense of $\|\cdot\|_S$ in $\mathcal{O}_i$ to x by construction.

4.2 Properties of Generalized Local Freespaces

While the generalized local freespace is most easily presented in terms of a modification the the inner product used in the local freespace definition, analysis can be easier by interpretation $\mathcal{LF}(x;S)$ as the local freespace computed under a change of coordinate, i.e. an invertible affine transformation.

Theorem 4.2.1. *For obstacle sets $\mathcal{O}_1,\ldots,\mathcal{O}_M$, the generalized local freespace $h\left(\mathcal{LF}(x;S)\right)$ about x is equal to $\mathcal{LF}(h(x))$, where $h(x) = S^{-\frac{1}{2}}x + g$, $g \in \mathbb{R}^n$, and $\mathcal{LF}(h(x))$ is computed with respect to the obstacles $h(\mathcal{O}_1),\ldots,h(\mathcal{O}_M)$, the image of each obstacle through h.*

Proof. By taking the square root of S^{-1}, we can define a coordinate change that underwhich the Euclidean distance metric is used. First, note that because S^{-1} is

symmetric, $S^{-\frac{1}{2}}$ is as well. Then

$$\Pi_{\mathcal{A}}^{S}(x) = \arg\min_{x' \in \mathcal{A}} \|x - x'\|_S \tag{4.3}$$

$$= \arg\min_{x' \in \mathcal{A}} \sqrt{(x - x')^{\mathrm{T}} S^{-1} (x - x')} \tag{4.4}$$

$$= \arg\min_{x' \in \mathcal{A}} \sqrt{\left(S^{-\frac{1}{2}}(x - x')\right)^{\mathrm{T}} \left(S^{-\frac{1}{2}}(x - x')\right)} \tag{4.5}$$

$$= \arg\min_{x' \in \mathcal{A}} \|S^{-\frac{1}{2}} x - S^{-\frac{1}{2}} x'\|_2 \tag{4.6}$$

$$= \arg\min_{x' \in \mathcal{A}} \|S^{-\frac{1}{2}} x + g - S^{-\frac{1}{2}} x' - g\|_2 \tag{4.7}$$

$$= h^{-1}\left(\arg\min_{x' \in h(\mathcal{A})} \|h(x) - x'\|_2\right) \tag{4.8}$$

$$= h^{-1}\left(\Pi_{h(\mathcal{A})}(h(x))\right) \tag{4.9}$$

Then, since for all a, b,

$$h(x) - h(x') = S^{-\frac{1}{2}} x + g - S^{-\frac{1}{2}} x' - g \tag{4.10}$$

$$= S^{-\frac{1}{2}}(x - x') \tag{4.11}$$

and

$$\frac{h(x) + h(y)}{2} = \frac{S^{-\frac{1}{2}}(x + x')}{2} + g \tag{4.12}$$

$$= h\left(\frac{x + x'}{2}\right) \tag{4.13}$$

The following relationship between $\mathcal{LF}(x)$, $\mathcal{LF}(x; S)$, and h is established.

$$\mathcal{LF}(h(x)) = \left\{ h(x') \;\middle|\; (h(s_i) - h(x))^{\mathrm{T}} \left(h(x') - \frac{h(x) + h(s_i)}{2} \right) \leq 0, h(s_i) = \Pi_{h(\mathcal{O}_i)}(h(x)) \right\} \tag{4.14}$$

$$= \left\{ h(x') \;\middle|\; S^{-\frac{1}{2}}(s_i - x)^{\mathrm{T}} S^{-\frac{1}{2}} \left(x' - \frac{x + s_i}{2} \right) \leq 0, s_i = \Pi_{\mathcal{O}_i}(x; S) \right\} \tag{4.15}$$

$$= \left\{ h(x') \;\middle|\; (s_i - x)^{\mathrm{T}} S^{-1} \left(x' - \frac{x + s_i}{2} \right) \leq 0, s_i = \Pi_{\mathcal{O}_i}(x; S) \right\} \tag{4.16}$$

$$= h\left(\mathcal{LF}(x; S)\right) \tag{4.17}$$

□

A consequence of this proof is that any property of $\mathcal{LF}(x)$ that is preserved under an affine transformation, such as convexity and the linearity of the set constraints, also applies to $\mathcal{LF}(x; S)$. We also see that if x is in the freespace, then so is any member of $\mathcal{LF}(x; S)$:

Proposition 4.2.2. *For any $x \in \mathcal{F}$, the generalized local freespace $\mathcal{LF}(x; S)$ is a nonempty subset of $\mathcal{F}$.*

Proof. Let h be the invertible affine transform $h(x) = S^{-\frac{1}{2}}x + g$. Then $\mathcal{LF}(h(x))$ computed with respect to $h(\mathcal{O}_1), \ldots, h(\mathcal{O}_N)$ lies in the freespace defined by the obstacle sets $h(\mathcal{O}_1), \ldots, h(\mathcal{O}_M)$ [4]. By (4.14), applying h^{-1} to these objects produces $\mathcal{LF}(x; S)$ defined for $\mathcal{O}_1, \ldots, \mathcal{O}_M$. Since h is bijective, no member of $\mathcal{LF}(h(x))$ will be mapped to any obstacle set $\mathcal{O}_i$, that is $\mathcal{LF}(x; S) \cap \mathcal{O}_i = \emptyset$ for $i = 1, \ldots, M$ and therefore $\mathcal{LF}(x; S)$ lies in the freespace defined with respect to $\mathcal{O}_1, \ldots, \mathcal{O}_M$. □

4.3 Generalized Local Freespaces with Ellipsoidal Constraints

Finally, we consider a definition of the generalized local freespace with ellipsoidal constraints as measured by $\|\cdot\|_S$. Specifically, if the robot occupies the set of states

$$\mathcal{E}(x;S) = \{\, y \mid \|x'-x\|_S \leq 1 \,\} \tag{4.18}$$

$$= \left\{\, x' \;\middle|\; (x'-x)^{\mathrm{T}} S^{-1}(x'-x) \leq 1 \,\right\} \tag{4.19}$$

then we would like $\mathcal{E}(x;S)$ to not intersect any obstacle for any x in the local freespace construction we define. Here, the local metric describes some kind of risk to the robot and states in $\mathcal{E}(x;S)$ contain at most a fixed amount of risk. Toward this end, we make the following definition:

Definition 4.3.1 (Contracted Local Freespace)**.** *For $S \in \mathbb{S}^n_{++}$, the **contracted local freespace** $\mathcal{LF}_C(x;S)$ is the local freespace $\mathcal{LF}(x;S)$ computed with respect to obstacle sets $\mathcal{O}_1 \oplus \mathcal{E}(0;S), \ldots, \mathcal{O}_M \oplus \mathcal{E}(0;S)$.*

The explicit form of the contracted local freespace in terms of obstacles $\mathcal{O}_1, \ldots, \mathcal{O}_M$ is given as

$$\mathcal{LF}_C(x;S) = \left\{\, x' \;\middle|\; (s_i - x)^{\mathrm{T}} S^{-1} \left(x' - \left(m_i - \frac{m_i - x}{2\|m_i - x\|_S} \right) \right) \leq 0 \,\right\} \tag{4.20}$$

where $m_i = \frac{x+s_i}{2}$, $s_i = \Pi^S_{\mathcal{O}_i}(x)$. This formulation translates each defining hyperplane of $\mathcal{LF}(x;S)$ by one half unit under the metric $\|\cdot\|_S$ toward x. This concept is adapted from [52] where it was originally proposed for disk-shaped robots. The version presented here generalizes it to use the inner product defined by S^{-1}. This abstraction still maintains the property that the robot is collision-free at any state in $\mathcal{LF}_C(x;S)$ and that $\mathcal{LF}_C(x;S)$ is nonempty for any collision-free robot:

Proposition 4.3.1. *Let $x \in \mathbb{R}^n$. Then, $\mathcal{E}(x; S) \subset \mathcal{F}$ if and only if $\mathcal{LF}_C(x; S) \neq \emptyset$.*

Proof. The forward direction follows from the definition in conjunction with Proposition 4.2.2. If $\mathcal{E}(x; S) \subset \mathcal{F}$, then it must be the case that x lies in the freespace for the obstacle set $\mathcal{O}_1 \oplus \mathcal{E}(0; S), \ldots, \mathcal{O}_M \oplus \mathcal{E}(0; S)$ and therefore $\mathcal{LF}(x; S)$ is a nonempty subset of the freespace computed with respect to that obstacle set. For the reverse direction, the fact that $\mathcal{LF}_C(x; S)$ is non-empty implies that there is a separating hyperplane between $\mathcal{E}(0; S)$ and each obstacle set $\mathcal{O}_i$ because both $\mathcal{E}(0; S)$ and $\mathcal{O}_i$ are convex. Therefore $\mathcal{E}(0; S) \subset \mathcal{F}$. □

Additionally, any ellipse centered at a member of $\mathcal{LF}_C(x; S)$ will lie in the freespace:

Proposition 4.3.2. *For any $x' \in \mathcal{LF}_C(x; S)$, $\mathcal{E}(x'; S) \subset \mathcal{F}$.*

Proof. Follows directly from the fact that $\mathcal{LF}_C(x; S)$ is $\mathcal{LF}(x; S)$ in the workspace whose obstacles are $\mathcal{O}_1 \oplus \mathcal{E}(0; S), \ldots, \mathcal{O}_M \oplus \mathcal{E}(0; S)$. □

4.4 Examples Using Metrics from Literature

In this section, the LQR metric described in Section 3.2.2 and beliefspace metric described in Section 3.3 are used to illustrate the concepts defined in this chapter.

4.4.1 Incorporating System Dynamics via the LQR Metric

To illustrate how the LQR metric modifies the local freespace construct, we will show how LQR metrics for different linearizations of a kinematic differential drive robot system impact the local freespace. The continuous-time dynamics of this system

are

$$\begin{bmatrix} \dot{x}(t) \\ \dot{y}(t) \\ \dot{\theta}(t) \end{bmatrix} = \begin{bmatrix} v(t)\cos(\theta(t)) & 0 \\ v(t)\sin(\theta(t)) & 0 \\ \omega & \end{bmatrix} \tag{4.21}$$

where $v(t)$ and $\omega(t)$ are control inputs. We consider a discrete-time approximation of this system

$$\underbrace{\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \theta_{k+1} \end{bmatrix}}_{z_{k+1}} = \underbrace{\begin{bmatrix} x_k \\ y_k \\ \theta_k \end{bmatrix} + \Delta t \begin{bmatrix} \cos(\theta) & 0 \\ \sin(\theta) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v_k \\ \omega_k \end{bmatrix}}_{f_{dd}(z_k, u_k)} \qquad u_k = \begin{bmatrix} v_k \\ \omega_k \end{bmatrix} \tag{4.22}$$

A positive value of ω rotates the vehicle counterclockwise. The variable z_k is used to represent the state to distinguish from x_k and y_k, which are the position of the robot in the workspace. At a particular state $z = [x\ y\ \theta]^{\mathrm{T}}$ and input $u = [v\ \omega]^{\mathrm{T}}$, the linearization is defined by the matrices

$$A = \frac{\partial f_{dd}(z,u)}{\partial z} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \qquad B = \frac{\partial f_{dd}(z,u)}{\partial u} = \begin{bmatrix} \Delta t\cos(\theta) & 0 \\ \Delta t\sin(\theta) & 0 \\ 0 & \Delta t \end{bmatrix} \tag{4.23}$$

However, since the original model in (4.22) is under-actuated, the linearized system in (4.23) is not controllable. Therefore, we instead consider the reduced dimension system

$$\begin{bmatrix} x_{k+1} \\ y_{k+1} \end{bmatrix} = \begin{bmatrix} x_k \\ y_k \end{bmatrix} + \begin{bmatrix} \Delta t\cos(\theta) & \delta \\ \Delta t\sin(\theta) & \delta \end{bmatrix} u_k \tag{4.24}$$

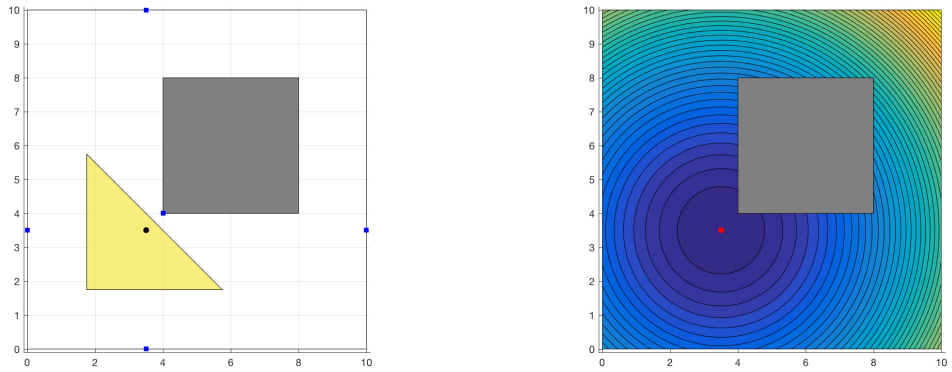
The parameter δ allows for coupling between the linear and rotational velocities and makes the system controllable. It is kept small in comparison to the terms inherited from (4.23) to limit the influence of this artificial coupling on the behavior of the system. When Δt is on the order of 1, a value of $\delta = 10^{-5}$ is used.

In each subfigure of Figure 4.2, a different local freespace is constructed. The robot is assumed to be a point robot and a collision occurs with a configuration space obstacle if $[x\ y]^{\mathrm{T}}$ lies within a workspace obstacle. In Figure 4.2a, the Euclidean local freespace is constructed. Geometrically, this local freespace definition does not favor motion in any direction around the obstacle. In Figure 4.2b and Figure 4.2c, the metric used is the LQR metric $\|\cdot\|_{P^{-1}}$, where P is the solution to the DARE (3.4) for the linearized about one of the shown state and control input with LQR penalty matrices $Q, R = I$. These local freespaces incorporate the dynamical information provided by the linearization about the respective states by favoring movement in a direction around the obstacle corresponding to the used value of θ. In Figure 4.2b, the local freespace allows more freedom along the y-axis because the chosen value of θ aligns the robot more along that axis. Similarly, the example in Figure 4.2c allows more freedom to move around the obstacle in the x-direction because the robot is aligned more along the x-axis.

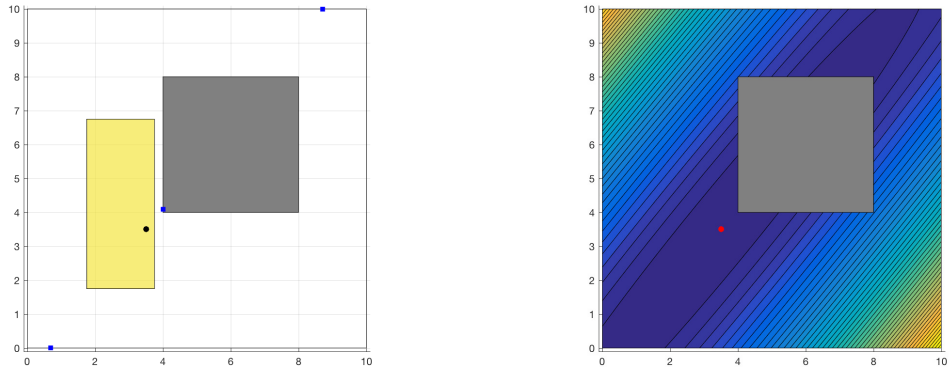
Additionally, in each example, the contour of the metric used to compute each local freespace is shown. Different linearizations produce different values of P, which, in turn, induce different metrics with ellipsoidal contours. In Figure 4.2b, the alignment of the contours is rotated clockwise from the alignment in Figure 4.2c, which is what causes the different local freespace structures.

4.4.2 Incorporating Uncertainty via the Beliefspace Metric

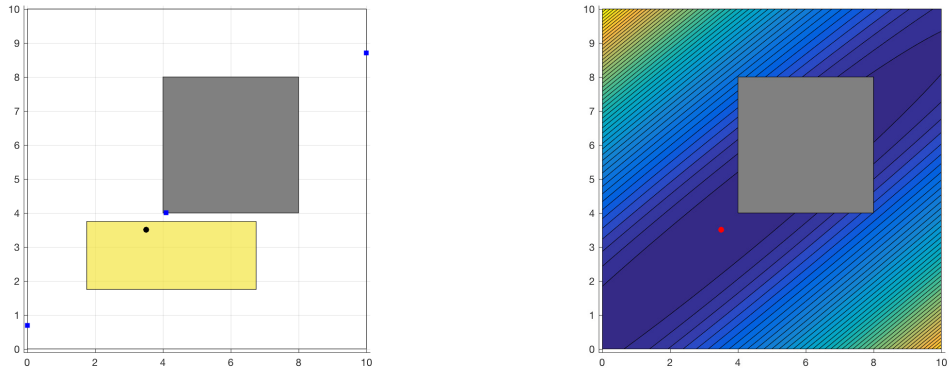
The beliefspace metric in conjunction with the contracted local freespace provides a natural way to combine uncertainty with the local geometry. A common requirement



(a)



(b)



(c)

Figure 4.2: Three local freespaces of a differential drive robot under different metrics. The robot workspace is $[0, 10] \times [0, 10]$. The local freespace is shown as the gold polytope, while an obstacle is shown as a dark gray polytope. In (a), the Euclidean metric is used, while (b) and (c) use LQR metrics for the system in (4.22) linearized about the respective states $[1\ 1\ \frac{\pi}{4} + 0.01]^{\mathrm{T}}$, $[1\ 1\ \frac{\pi}{4} - 0.01]^{\mathrm{T}}$, same control input $[0.01\ 0]^{\mathrm{T}}$, and $\Delta t = 1$. The robot location in the workspace is black circle while the closest point on each obstacle under the metric is shown as a blue square. In addition, the contour of each metric is shown projected onto the workspace (robot position is the red circle).

of a beliefspace planner [37, 39, 46] is to ensure that, for each beliefstate $b_k = (x_k, \Sigma_k)$ along a solution trajectory

$$\Pr\{x_k \in \mathcal{F}\} = \sum_{i=1}^{N} \Pr\{x_k \notin \mathcal{O}_i\} \geq \delta \qquad\qquad x_k \sim N(\bar{x}_k, \Sigma_k) \tag{4.25}$$

That is, no collision occurs with at least some probability δ. However, computing $\Pr\{x_k \in \mathcal{O}_i\}$ is challenging to do numerically due to having to integrate a Gaussian distribution over each obstacle. A standard approximate approach is to simply ensure that the confidence region defined by b_k lies entirely within $\mathcal{F}$ [39, 53, 54]. The *confidence region* [55] of a beliefstate b_k is an ellipsoid

$$\mathcal{E}_\delta(b_k) = \left\{ x \;\middle|\; (x - \bar{x}_k)^{\mathrm{T}} (\gamma \Sigma_k)^{-1} (x - \bar{x}_k) \leq 1 \right\} \qquad\qquad \gamma = \chi_n^{-1}(\delta) \tag{4.26}$$

where $\chi_n(\cdot)$ is the cumulative distribution function (CDF) of chi-squared distribution with n degrees of freedom [55]. The confidence region is constructed such that $\Pr\{x_k \in \mathcal{E}_\delta(b_k)\} = \delta$. If $\mathcal{E}_\delta(b_k) \cap \left(\bigcup_{i=1}^{M} \mathcal{O}_i \right) = \emptyset$, then (4.25) is satisfied.

The quadratic form in the definition of the confidence region of b_k suggests choosing $\| \cdot \|_{\gamma\Sigma}$ as a metric for local freespace construction. This metric is a scaled version of the beliefspace metric from Section 3.3. This choice of metric guarantees by construction that any point lying within $\mathcal{LF}_C(\bar{x}; \gamma\Sigma_k)$ will satisfy (4.25), as summarized in the following proposition

Proposition 4.4.1. *For the beliefstate $b = (\bar{x}, \Sigma)$ and $\gamma \in \mathbb{R}_+$, let $\bar{x} \in \mathcal{F}$ and $\mathcal{LF}_C(\bar{x}, \gamma\Sigma)$ be non-empty. Then, for any belief $(\bar{y}, \Sigma)$ with $\bar{y} \in \mathcal{LF}_C(\bar{x}, \gamma\Sigma)$, it is the case that $\Pr\{y \in \mathcal{F}\} = \sum_{i=1}^{N} \Pr\{y \notin \mathcal{O}_i\} \geq \delta$ for $\gamma = \chi_n^{-1}(\delta)$ and $y \sim N(0, \Sigma)$.*

Proof. Proposition 4.3.2 implies that $\mathcal{E}(\bar{y}; \gamma\Sigma) \subset \mathcal{F}$ for any $\bar{y} \in \mathcal{LF}_C(\bar{x}; \gamma\Sigma)$. Since $\Pr\{y \in \mathcal{E}(\bar{y}, \gamma\Sigma)\} = \delta$, it is such that $\Pr\{y \in \mathcal{F}\} \geq \delta$ for $y \sim N(\bar{y}, \Sigma)$. □

Some example contracted local freespaces are presented in Figure 4.4 using $\delta = 0.5$.

As a point of comparison, a local freespace is constructed using the Euclidean metric in Figure 4.3. Each subfigure of Figure 4.4 uses a different covariance Σ, and the boundary of the confidence region about the state (black dot) is drawn as a black ellipsoid. Each subfigure also shows (on the right), the workspace and local freespace under the transformation h from Section 4.2 for that example's Σ. In Figure 4.4a, the confidence region is aligned along the x-axis and can easily fit through the gap between the obstacles. Note that, under h, this confidence region becomes spherical. The local freespace reflects this by funneling local motion in the positive x-direction through the gap. In contrast, in Figure 4.4b, the confidence region cannot maneuver through the gap as easily due to its origin along the y-axis. The local freespace adapts by allowing for more motion in the y-axis as a result. Finally, in Figure 4.4c, the confidence region has a diagonal orientation. This example best demonstrates how the contracted local freespace moves different faces of the local freespace depending on their orientation with respect to the state uncertainty. Directions with more certainty are shrunk more than those with less uncertainty.

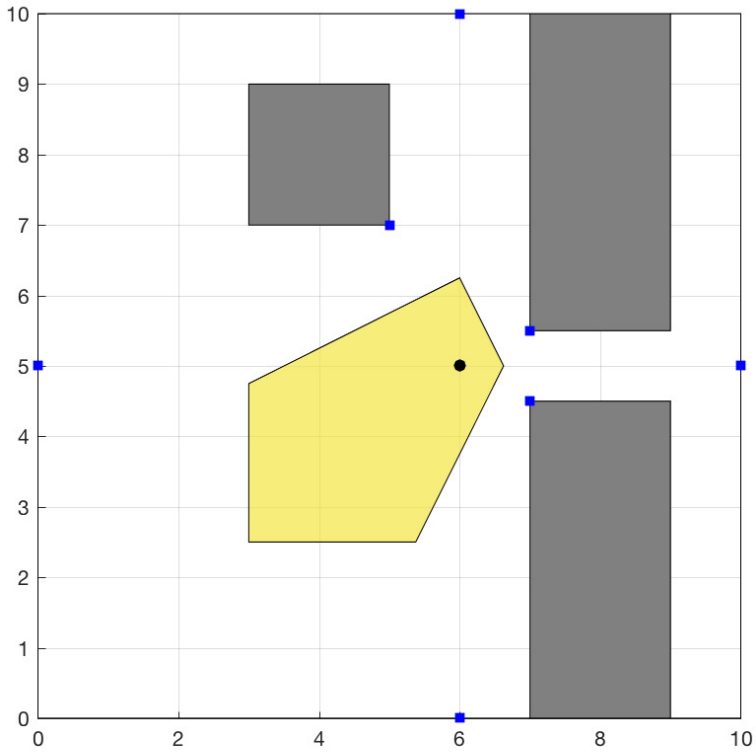
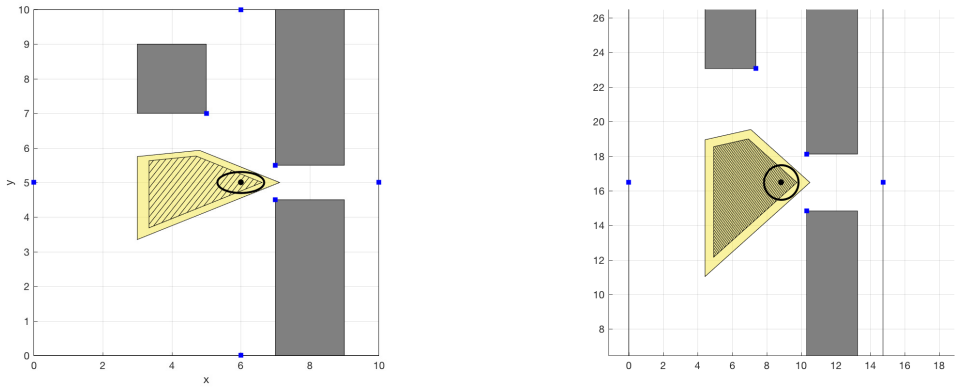
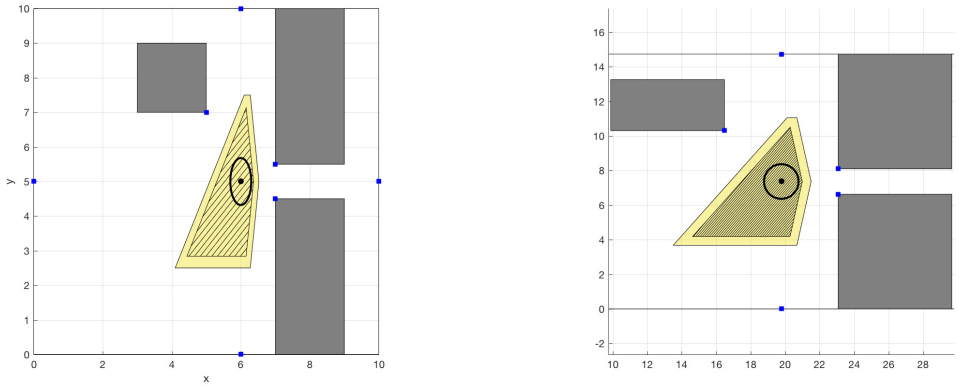


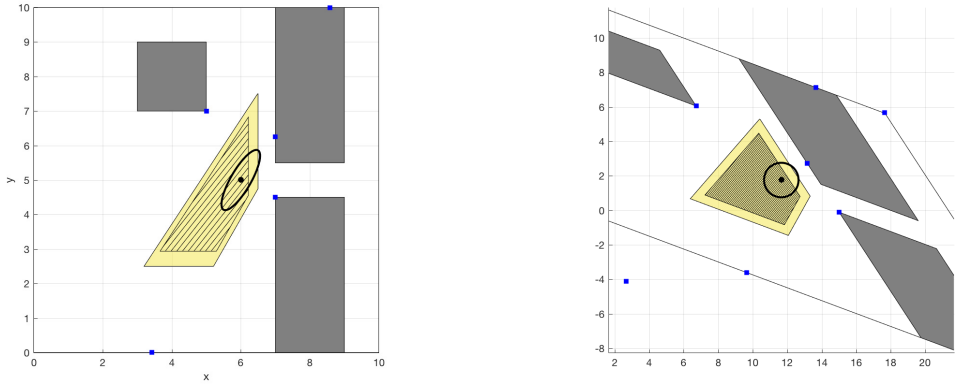
Figure 4.3: An example local freespace constructed about $[6\ 5]^{\mathrm{T}}$ (black circle). Blue squares represent the closest point on each obstacle used to construct the local freespace.



(a)



(b)



(c)

Figure 4.4: Three examples of local freespaces computed about the same point (black circle) using belief metrics defined by different covariances. The hatched region is the contracted freespace corresponding to the local freespace shown in the gold region. The confidence region corresponding to $\delta = 0.5$ is shown in each figure as an ellipse. The figures on the left is the original workspace, where the figures on the left are the workspace under the transformation h. The blue squares are the nearest point in each obstacle set under the norm used to define each local freespace.

Chapter 5

Adaptive Steering Using Generalized Local Freespaces

This chapter describes how the generalized local freespace can be used to define a local steering algorithm for general nonlinear systems. The steering algorithm assumes that sampling is a cheap part of the sampling-based planning algorithm and therefore sufficient samples will be drawn to make the dynamical linearizations used a good local approximation. Efficacy of the steering procedure is given in terms of numerical simulations at the end of the chapter.

5.1 Generalized Local Freespace Steering

The generalized local freespace concepts introduced in Chapter 4 provide local convex decompositions of the environment that merges information about the underlying dynamical process, such as local behavior and uncertainty, with the local geometry. Since the generalized local freespace is convex and exploits local information, a sensible steering policy is given by a constrained LQR program.

The new steering procedure, `GLFSteer`, definition is given in Algorithm 4. The algorithm designs a local trajectory over a finite horizon $K \in \mathbb{N}$ that drives the state

x_0 toward another x_{near}. To do so, the system is linearized about x_{near} and a nominal control input u_0. Then, the following convex linearly constrained quadratic program is solved:

$$\begin{aligned} \underset{u_0,\ldots,u_{K-1}}{\text{minimize}} \quad & \sum_{k=0}^{K-1} (x_k - x_{near})^{\mathrm{T}} Q (x_k - x_{near}) + u_k^{\mathrm{T}} R u_k \\ \text{subject to} \quad & x_{k+1} = A x_k + B u_k \\ & x_k \in \mathcal{LF}(x_0, S_k) \end{aligned} \tag{5.1}$$

where S_0 is provided to the algorithm and $S_{k+1} = T(S_k)$ for some function $T : \mathbb{S}^n_{++} \to \mathbb{S}^n_{++}$. This optimization can be solved efficiently numerically as it is a convex linearly constrained quadratic program [50, 51]. The generalized local freespace constraint simultaneously incorporates the local metric and geometric structure into the steering procedure. If obstacles are very far away from x_0, then `GLFSteer` will behave similarly to `LQRSteer`.[1] When obstacles are near x_0, the local freespace constraint limits the motion of the robot in a way that simultaneously reflects the local metric and geometry. We allow the metric to evolve over the finite horizon in some way that is known to the algorithm modeled by notated by the transform T. (e.g. projecting the covariance of the system forward to use an accurate beliefspace metric at each time step). In order to preserve the convexity of the program, all local freespace constraints are computed with respect to x_0.

Input: $x_0,\ x_g$
1 $A \leftarrow \frac{\partial f(x_0,u_0)}{\partial x},\ B \leftarrow \frac{\partial f(x_0,u_0)}{\partial u}$;
2 $(u_1, \ldots, u_{K-1}) \leftarrow$ `CLQR` $(A,\ B,\ Q,\ R,\ S_0, T, K)$; `/* Solve (5.1) */`
3 **return** $(u_1, \ldots, u_{K-1})$

Algorithm 4: Definition of `GLFSteer` $(x_{near},\ x_{rand})$

[1] The path will not be exactly the same because `LQRSteer` solves an infinite horizon version of the problem.

5.2 Numerical Demonstrations in Sampling-Based Planners

This section demonstrates the results of some numerical simulations of the RRT algorithm (Algorithm 3) using the steering procedure `GLFSteer` (Algorithm 4) with different metrics. A comparison trial using `LQRSteer` (Algorithm 2) will be also be shown. In each experiment, both trials will use the same `LQRDist` function and only the steering function will differ. All examples use a horizon $K = 4$.

First, a simple kinematic setting is considered, namely

$$x_{k+1} = x_k + u_k \tag{5.2}$$

where $x_k, u_k \in \mathbb{R}^2$. This model is very common in sampling-based motion planning due to the simple local control laws needed to steer the system [4, 39]. The robot is modeled as a point robot in a workspace whose boundary is $[0, 10] \times [0, 10]$. In Figure 5.1, the LQR cost function is defined with matrices $Q = \text{diag}(2, 1)$ and $R = I$. The steering procedures under consideration are different from the straight steering procedure considered in [4] because the LQR cost in conjunction with a horizon $K > 1$ will not always produce a straight connection between vertices. Both simulations in this example were run with $N = 1000$ iterations and the trees were grown from an initial state $x_0 = [1\ 1]^{\mathrm{T}}$. Since `LQRSteer` does not incorporate any geometric information during planning, many of these trajectories lie in collision with obstacles and are not added to the graph of the tree. Meanwhile, trajectories produced by `GLFSteer`. The final number of vertices in the tree is $|\mathcal{V}| = 470$. In contrast, `GLFSteer` adapts to the environment through use of the local freespace and is capable of navigating the narrow channel between the obstacles. Since (5.1) is always feasible for this system due to the lack of constraints on u_k and the first-order nature of the dynamics, a

vertex is added to the tree at each iteration. Therefore, the final number of vertices in the RRT tree is $|\mathcal{V}| = 1001$ (including the initial state) using `GLFSteer`.

The results of a similar experiment with the same dynamical system are shown in Figure 5.2. The LQR weights Q, R remain unchanged. Again, `LQRSteer` performs poorly, resulting in few connections made during planning. After $N = 500$ iterations, the number of vertices in the tree of the RRT planner is $|\mathcal{V}| = 213$. Meanwhile, the `GLFSteer` steering procedure is, again, more successful at navigating the environment and leaves the RRT with $|\mathcal{V}| = 1001$ vertices. In both examples, the local freespaces used during planning were constructed using the LQR metric.

Unlike the steering procedure in [4], `GLFSteer` is not limited to the dynamics of the form in (5.2). Consider a second-order linear model

$$x_{k+1} = Ax_k + Bu_k \qquad A = \begin{bmatrix} I & I \\ 0 & I \end{bmatrix} \qquad B = \begin{bmatrix} 0 \\ I \end{bmatrix} \tag{5.3}$$

which is another common form of linear model used in motion planning [15, 46]. In Figure 5.3 provides the results of an experiment using this model. Each state can be decomposed into a positional and velocity component – i.e. for $x_k = [q_k \ \dot{q}_k]^{\mathrm{T}}$, q_k is the position of the robot and $\dot{q}_k$ is its velocity. In the example, $q_k, \dot{q}_k \in \mathbb{R}^2$. Collisions with workspace obstacles are only checked with respect to q_k. When sampling states, $\dot{q}_k$ is drawn from $[-0.1, 0.1] \times [-0.1, 0.1]$ uniformly.

For a second-order system, (5.1) is not strictly feasible because an input u_k will impact position q_{k+1} but has no influence over q_k. Therefore, a large enough $\dot{q}_k$ will ensure that q_{k+1} is outside of the local freespace regardless of u_k. If `GLFSteer` is not able to produce a trajectory, the planner treats it as a collision and no action is taken that cycle. Despite the lack of feasibility guarantee, the RRT still contains more vertices at the end of this experiment using `GLFSteer` than the `LQRSteer`. After $N = 500$ iterations, the RRT features $|\mathcal{V}| = 455$ vertices using `GLFSteer` while

`LQRSteer` leaves the RRT with $|\mathcal{V}| = 122$ vertices. Again, the coverage of the tree, as shown in Figure 5.3, is much greater using `GLFSteer`. The local freespaces were computed using the LQR metric.

Next, the kinematic differential drive system given in (4.22) is considered. The results of the trial are summarized in Figure 5.5. In order to design local LQR control policies for steering and distance computation for this under-actuated system, the reduced dimension linearization given in (4.24) is used. In this scenario, the LQR costs matrices are were $Q, R = I$ and the model parameters are $\Delta t = 0.2, \delta = 10^{-5}$. Both planners were run for $N = 500$ iterations. Due to the fact that the steering policy was designed for a reduced dimension approximation of the system dynamics, the planner is limited in how it can explore the space. Nevertheless, the `GLFSteer` procedure was capable to explore the cluttered region at the center of the workspace more successfully than `LQRSteer`. Using `LQRSteer`, only $|\mathcal{V}| = 414$ vertices remained in the tree while $|\mathcal{V}| = 500$ vertices were added using `GLFSteer`.

Finally, a beliefspace example is considered in Figure 5.5. The setting is similar to the first example except with additive Gaussian process and observation noise, i.e.

$$\begin{aligned} x_{k+1} &= x_k + u_k + w_k & w_k &\sim N(0, W) \\ y_k &= x_k + v_k & v_k &\sim N(0, V) \end{aligned} \tag{5.4}$$

where $W = \text{diag}(0.01, 0.02)$ and

$$V = \begin{bmatrix} 0.0098 & 0.0153 \\ 0.0153 & 0.0447 \end{bmatrix} \tag{5.5}$$

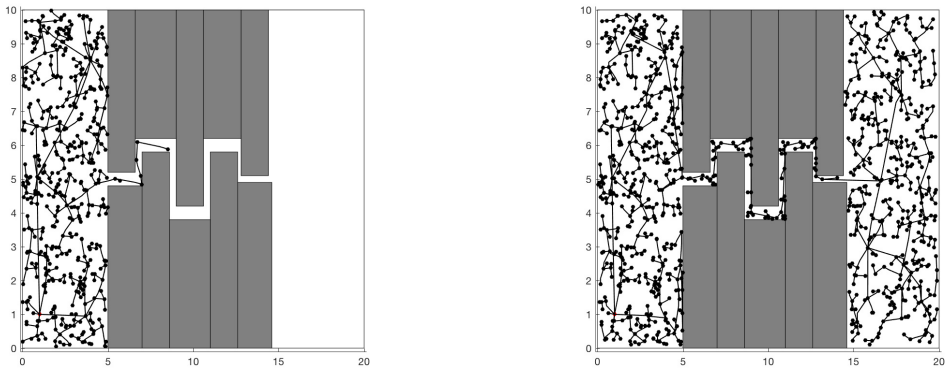
The initial belief was $b_0 = \left([3\ 8]^{\mathrm{T}}, \text{diag}(0.001, 0.001)\right)$ and $\delta = 0.95$ was the enforced probability of safety. The vertices in the RRT are beliefs as opposed to merely dynamical states, and covariances are propagated between vertices by applying the Kalman

update (3.9) along the trajectory. For the beliefspace setting, `CollisionFree` is replaced with the function

$$\texttt{BeliefFree}(\bar{\mathbf{x}}_{[K]}, \mathcal{F}) = \bigwedge_{k=2}^{K} \left(\mathcal{E}_\delta(\{\alpha \bar{x}_{k-1} + (1-\alpha)\bar{x}_k, \Sigma_{k|k-1}) \mid \alpha \in [0,1]\} \subset \mathcal{F} \right) \quad (5.6)$$

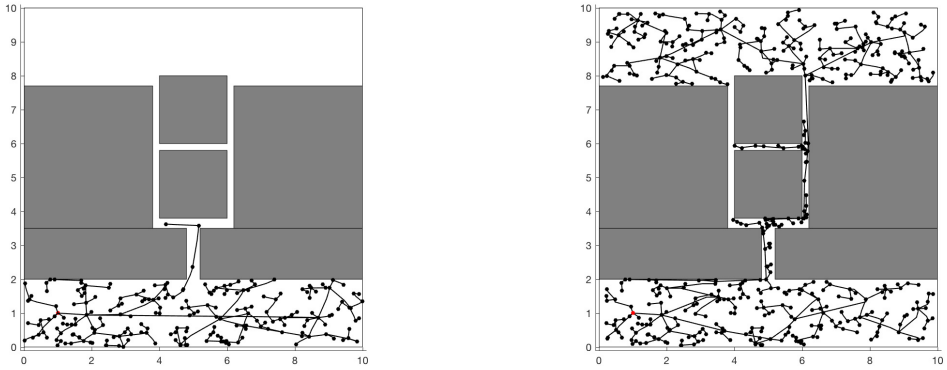
which checks if the confidence ellipse corresponding to the Kalman process update $\Sigma_{k|k-1}$ is collision-free along each line segment joining consecutive pairs of trajectory states. The process covariance matrix $\Sigma_{k|k-1}$ is used as opposed to Σ_k or Σ_{k-1} because it encodes the uncertainty about the system during the state transition before z_k is observed. The metric used for `GLFSteer` is the beliefspace metric using the process covariance for the beliefstate. Specifically, given an initial belief $b_0 = (\bar{x}_0, \Sigma_0)$, each state $\bar{x}_k$ along the trajectory planned by `GLFSteer` is required to lie within $\mathcal{LF}_C(\bar{x}_0, \Sigma_{k|k-1})$. This constraint ensures that any satisfying trajectory will be collision free according to `BeliefFree`.

The results of the simulation are summarized in Figure 5.5. Over the aloted $N = 1000$ iterations, `LQRSteer` was only able to progress into two out of the four chambers in the environment and produced a tree with $|\mathcal{V}| = 231$ vertices. In comparison, the RRT reaches all the chambers when using `GLFSteer` and contains $|\mathcal{V}| = 1001$ vertices.



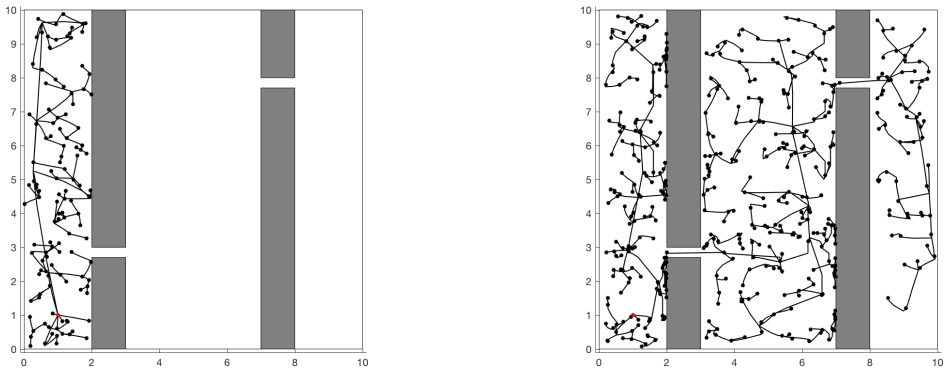
(a) `LQRSteer` **(b)** `GLFSteer`

Figure 5.1: First simulation environment using the model in(5.2). Obstacles are shown in gray. Circles are graph vertices in the RRT. The red circle (lower left) is the initial state.



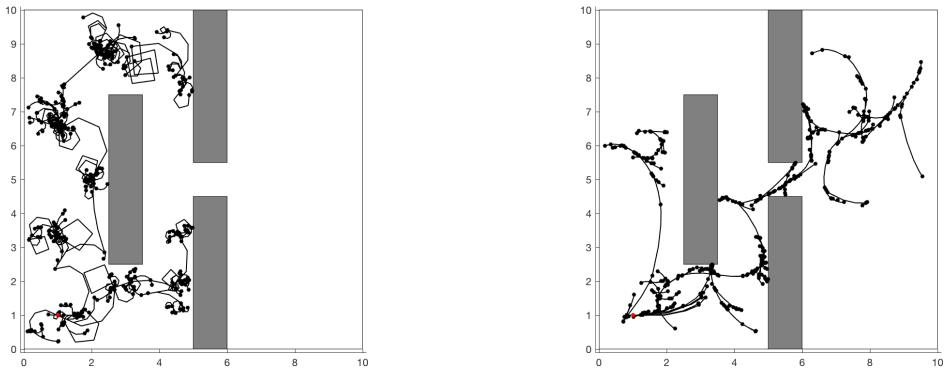
(a) `LQRSteer` **(b)** `GLFSteer`

Figure 5.2: Second simulation environment again using the model in(5.2).



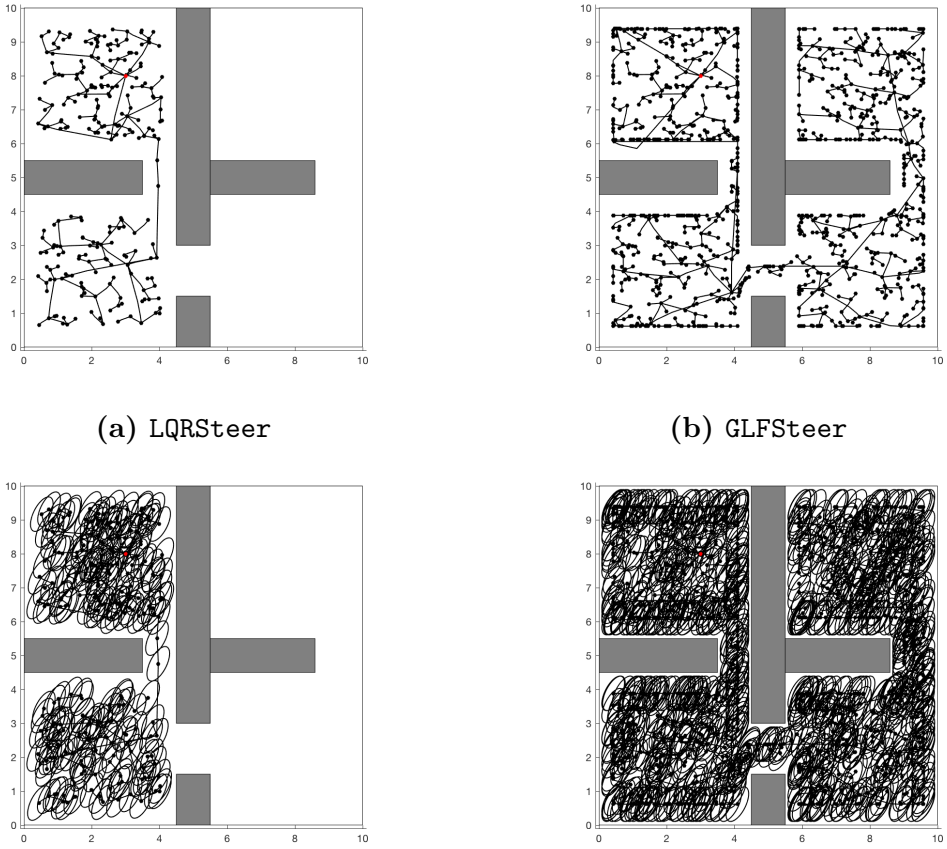
(a) `LQRSteer` **(b)** `GLFSteer`

Figure 5.3: Third simulation environment using the linear dynamical model in (5.3)



(a) `LQRSteer` **(b)** `GLFSteer`

Figure 5.4: Forth simulation environment using the kinematic kart model in (4.22)



(a) `LQRSteer` **(b)** `GLFSteer`

(c) `LQRSteer` **(d)** `GLFSteer`

Figure 5.5: Fifth simulation environment depcting the beliefspace system given in (5.4). The top row contains the RRT output while the bottom row also contains the confidence region of each vertex.

Chapter 6

Conclusion

This thesis proposed the generalized local freespace, a novel structure that attempts to combine information about workspace geometry and the behavior of an underlying dynamical system. To do so, it abstracts the choice of inner product from the Euclidean one used in the local freespace structure from [4, 28, 29]. The choice of inner product corresponds to the specification of a new metric used during for constructing the local freespace. It was also demonstrated that the generalized local freespace is equivalent to computing the Euclidean local freespace in the workspace under an affine transformation. Finally, the idea of the contracted local freespace was proposed as a way to incorporate risk, as measured by the selected metric, into the local freespace. Two motivating metrics from sampling-based planning literature were provided as motivating examples: the LQR cost-to-go function and the Mahalanobis distance. The LQR cost-to-go captures local information about the dynamical system, while the Mahalanobis distance provides a measure of risk for linear-Gaussian systems at a certain timestep.

The generalized local freespace was then used to design a steering function for a sampling-based planner. The steering procedure solves a finite-horizon, constrained linear-quadratic regulation problem whose state constraints stem from a generalized

local freespace about the initial state. The efficacy of this local steering procedure was demonstrated for a number of dynamical settings, including a second-order linear system, a kinematic differential drive system, and a linear-Gaussian beliefspace system. In each case, the generalized local freespace steering procedure increased the connectivity of the graph produced by the sampling-based planner compared to an example LQR-based steering procedure from the literature.

6.1 Future Directions

To conclude, we present some possible extensions and applications of the generalized local freespace for future work.

First, a critical assumption in the construction of the generalized local freespace is that the configuration space obstacles are in a form that it is easy to compute the metric projection of a configuration onto. In this work, all configuration space obstacles were convex polytopes that were available to the algorithm in closed form, thus allowing for computation of the metric projection via an efficient convex program. However, sampling-based planners are often deployed in settings where such representations are not readily available, such as manipulators [10, 56]. In these environments, it is often efficient to test when the robot is in collision, despite representing explicit obstacles sets being intractable. How the generalized local freespace can be adapted to such systems is one possible next step.

A similar direction is how other metrics may be incorporated into the generalized local freespace. Often, locally informative functions, such as robustness to a logical specification [57], are not available in closed-form and must be approximated through (often efficient) simulation. Determining how a local representation of such a metric can be incorporated into local freespace construction is another potential direction.

Finally, a different vein of inquiry is how local freespaces can be used to define local

control policies for use online once planning is complete. A local freespace provides a useful underapproximation of the safe region of the statespace surrounding the trajectory. In addition, the local freespace constructed around subsequent trajectory states overlap. Once can envision employing a combination of barrier and Lyapunov functions [58] to funnel sets of states from the neighborhood of the origin toward the goal by sequentially employing local controllers constructed for each local freespace.

Bibliography

[1] Y. Kuwata, G. A. Fiore, J. Teo, E. Frazzoli, and J. P. How, "Motion planning for urban driving using rrt," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on.* IEEE, 2008, pp. 1681–1686.

[2] R. D'Andrea, "Guest editorial: A revolution in the warehouse: A retrospective on kiva systems and the grand challenges ahead," *IEEE Transactions on Automation Science and Engineering*, vol. 9, no. 4, pp. 638–639, 2012.

[3] K. Dorling, J. Heinrichs, G. G. Messier, and S. Magierowski, "Vehicle routing problems for drone delivery," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 47, no. 1, pp. 70–85, 2017.

[4] O. Arslan, V. Pacelli, and D. E. Koditschek, "Sensory steering for sampling-based motion planning," in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on.* IEEE, In Press.

[5] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. on Robot. Autom.*, vol. 12, no. 4, pp. 566–580, 1996.

[6] S. M. Lavalle, "Rapidly-exploring random trees: A new tool for path planning," Iowa State University, Tech. Rep., 1998.

[7] O. B. Bayazit, J.-M. Lien, and N. M. Amato, "Probabilistic roadmap motion planning for deformable objects," in *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, vol. 2. IEEE, 2002, pp. 2126–2133.

[8] M. Kothari, I. Postlethwaite, and D.-W. Gu, "Multi-uav path planning in obstacle rich environments using rapidly-exploring random trees," in *Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on.* IEEE, 2009, pp. 3069–3074.

[9] P. Cheng, Z. Shen, and S. La Valle, "Rrt-based trajectory design for autonomous automobiles and spacecraft," *Archives of control sciences*, vol. 11, no. 3/4, pp. 167–194, 2001.

[10] M. V. Weghe, D. Ferguson, and S. S. Srinivasa, "Randomized path planning for redundant manipulators without inverse kinematics," in *Humanoid Robots, 2007 7th IEEE-RAS International Conference on.* IEEE, 2007, pp. 477–482.

[11] S. Karaman and E. Frazzoli, "Incremental sampling-based algorithms for optimal motion planning," *Robotics Science and Systems VI*, vol. 104, 2010.

[12] A.-A. Agha-Mohammadi, S. Chakravorty, and N. M. Amato, "Firm: Sampling-based feedback motion-planning under motion uncertainty and imperfect measurements," *The International Journal of Robotics Research*, vol. 33, no. 2, pp. 268–304, 2014.

[13] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *Int. J. Robot. Res.*, vol. 20, no. 5, pp. 378–400, 2001.

[14] R. Tedrake, I. R. Manchester, M. Tobenkin, and J. W. Roberts, "Lqr-trees: Feedback motion planning via sums-of-squares verification," *The International Journal of Robotics Research*, vol. 29, no. 8, pp. 1038–1052, 2010.

[15] D. J. Webb and J. van den Berg, "Kinodynamic rrt*: Asymptotically optimal motion planning for robots with linear dynamics," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 5054–5061.

[16] 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 *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 2537–2542.

[17] D. Hsu, L. E. Kavraki, J.-C. Latombe, R. Motwani, and S. Sorkin, "On finding narrow passages with probabilistic roadmap planners," in *The Int. Workshop on the Algorithmic Foundations of Robotics*, 1998.

[18] S. R. Lindemann and S. M. LaValle, "Incrementally reducing dispersion by increasing Voronoi bias in RRTs," in *IEEE International Conference on Robotics and Automation*, vol. 4, 2004, pp. 3251–3257.

[19] V. Boor, M. H. Overmars, and A. F. van der Stappen, "The gaussian sampling strategy for probabilistic roadmap planners," in *IEEE Int. Conf. Robot. Autom.*, vol. 2, 1999, pp. 1018–1023.

[20] S. Rodriguez, S. Thomas, R. Pearce, and N. M. Amato, "Resampl: A region-sensitive adaptive motion planner," in *Algorithmic Foundation of Robotics VII*. Springer, 2008, pp. 285–300.

[21] S. W. H. Wong and M. Jenkin, "Exploiting collision information in probabilistic roadmap planning," in *IEEE International Conference on Mechatronics*, 2009, pp. 1–5.

[22] D. Hsu, T. Jiang, J. Reif, and Z. Sun, "The bridge test for sampling narrow passages with probabilistic roadmap planners," in *IEEE Int. Conf. Robot. Autom.*, vol. 3, 2003, pp. 4420–4426.

[23] K. Shi, J. Denny, and N. M. Amato, "Spark PRM: Using RRTs within PRMs to efficiently explore narrow passages," in *IEEE International Conference on Robotics and Automation*, 2014, pp. 4659–4666.

[24] K. E. Bekris, B. Y. Chen, A. M. Ladd, E. Plaku, and L. E. Kavraki, "Multiple query probabilistic roadmap planning using single query planning primitives," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 1, 2003, pp. 656–661.

[25] N. M. Amato, O. B. Bayazit, L. K. Dale, C. Jones, and D. Vallejo, "OBPRM: An obstacle-based PRM for 3D workspaces," in *Workshop on the Algorithmic Foundations of Robotics*, 1998, pp. 155–168.

[26] P. Isto, "Constructing probabilistic roadmaps with powerful local planning and path optimization," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3, 2002, pp. 2323–2328.

[27] L. Palmieri, S. Koenig, and K. O. Arras, "Rrt-based nonholonomic motion planning using any-angle path biasing," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 2016, pp. 2775–2781.

[28] O. Arslan and D. E. Koditschek, "Exact robot navigation using power diagrams," in *Robotics and Automation, 2016 IEEE International Conference on*, 2016, pp. 1–8.

[29] ——, "Sensor-based reactive navigation in unknown convex sphere worlds," in *The 12th International Workshop on the Algorithmic Foundations of Robotics*, 2016.

[30] S. Karaman and E. Frazzoli, "Optimal kinodynamic motion planning using incremental sampling-based methods," in *Decision and Control (CDC), 2010 49th IEEE Conference on*. IEEE, 2010, pp. 7681–7687.

[31] T. Kunz and M. Stilman, "Kinodynamic rrts with fixed time step and best-input extension are not probabilistically complete," in *Algorithmic foundations of robotics XI*. Springer, 2015, pp. 233–244.

[32] L. Palmieri and K. O. Arras, "A novel rrt extend function for efficient and smooth mobile robot motion planning," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*. IEEE, 2014, pp. 205–211.

[33] J. J. Park and B. Kuipers, "Feedback motion planning via non-holonomic rrt* for mobile robots," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 4035–4040.

[34] E. Glassman and R. Tedrake, "A quadratic regulator-based heuristic for rapidly exploring state space," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 5021–5028.

[35] L. Palmieri and K. O. Arras, "Distance metric learning for rrt-based motion planning with constant-time inference," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 637–643.

[36] M. Bharatheesha, W. Caarls, W. J. Wolfslag, and M. Wisse, "Distance metric approximation for state-space rrts using supervised learning," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*. IEEE, 2014, pp. 252–257.

[37] J. Van Den Berg, S. Patil, and R. Alterovitz, "Motion planning under uncertainty using iterative local optimization in belief space," *The International Journal of Robotics Research*, vol. 31, no. 11, pp. 1263–1278, 2012.

[38] V. Indelman, L. Carlone, and F. Dellaert, "Planning in the continuous domain: A generalized belief space approach for autonomous navigation in unknown environments," *The International Journal of Robotics Research*, vol. 34, no. 7, pp. 849–882, 2015.

[39] A. Bry and N. Roy, "Rapidly-exploring random belief trees for motion planning under uncertainty," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 723–730.

[40] N. A. Melchior and R. Simmons, "Particle rrt for path planning with uncertainty," in *Robotics and Automation, 2007 IEEE International Conference on*. IEEE, 2007, pp. 1617–1624.

[41] N. M. Amato, O. B. Bayazit, L. K. Dale, C. Jones, and D. Vallejo, "Choosing good distance metrics and local planners for probabilistic roadmap methods," in *IEEE International Conference on Robotics and Automation*, vol. 1, 1998, pp. 630–637.

[42] F. Borrelli, A. Bemporad, and M. Morari, *Predictive control for linear and hybrid systems*. Cambridge University Press, 2017.

[43] F. L. Lewis, D. Vrabie, and V. L. Syrmos, *Optimal control*. John Wiley & Sons, 2012.

[44] T. Pappas, A. Laub, and N. Sandell, "On the numerical solution of the discrete-time algebraic riccati equation," *IEEE Transactions on Automatic Control*, vol. 25, no. 4, pp. 631–641, 1980.

[45] S. Xiang, F. Nie, and C. Zhang, "Learning a mahalanobis distance metric for data clustering and classification," *Pattern Recognition*, vol. 41, no. 12, pp. 3600–3612, 2008.

[46] M. P. Vitus and C. J. Tomlin, "Closed-loop belief space planning for linear, gaussian systems," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on.* IEEE, 2011, pp. 2152–2159.

[47] M. S. Grewal and A. P. Andrews, "Kalman filtering: Theory and practice using matlab," 2001.

[48] P. Missiuro and N. Roy, "Adapting probabilistic roadmaps to handle uncertain maps," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*

[49] R. Platt, L. Kaelbling, T. Lozano-Perez, and R. Tedrake, "Simultaneous localization and grasping using belief space planning," in *Robotics and Automation (ICRA), IEEE International Conference on, Workshop on Manipulation Under Uncertainty*, 2011.

[50] S. Boyd and L. Vandenberghe, *Convex Optimization.* Cambridge University Press, 2004.

[51] J. Nocedal and S. J. Wright, *Sequential quadratic programming.* Springer, 2006.

[52] O. Arslan and D. Koditscheck, "Sensor-based reactive navigation in unknown convex sphere worlds," in *submitted to) the 12th International Workshop on the Algoritmic Foundations of Robotics (WAFR)*, 2016.

[53] L. Blackmore and M. Ono, "Convex chance constrained predictive control without sampling," in *Proceedings of the AIAA Guidance, Navigation and Control Conference*, 2009, pp. 7–21.

[54] M. Ono and B. C. Williams, "Iterative risk allocation: A new approach to robust model predictive control with a joint chance constraint," in *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on.* IEEE, 2008, pp. 3427–3432.

[55] J. Berger, "A robust generalized bayes estimator and confidence region for a multivariate normal mean," *The Annals of Statistics*, pp. 716–761, 1980.

[56] J. Cortés and T. Siméon, "Sampling-based motion planning under kinematic loop-closure constraints," in *Algorithmic Foundations of Robotics VI*. Springer, 2004, pp. 75–90.

[57] G. E. Fainekos and G. J. Pappas, "Robustness of temporal logic specifications," in *FATES/RV*. Springer, 2006, pp. 178–192.

[58] A. D. Ames, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs with application to adaptive cruise control," in *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*. IEEE, 2014, pp. 6271–6278.