Algorithms for Safe Robot Navigation

by Brian Maxim Axelrod

Submitted to the
Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degree of

Master of Engineering in Electrical Engineering and Computer Science
at the
Massachusetts Institute of Technology
September 2017

Author .....
Department of Electrical Engineering and Computer Science
August 18, 2017

Certified by .....
Leslie Pack Kaelbling
Panasonic Professor of Computer Science and Engineering
Thesis Supervisor

Certified by .....
Tomás Lozano-Pérez
School of Engineering Professor in Teaching Excellence
Thesis Supervisor

Accepted by .....
Christopher J. Terman
Chairman, Department Committee on Graduate Theses

Algorithms for Safe Robot Navigation

by

Brian Maxim Axelrod

Submitted to the Department of Electrical Engineering and Computer Science
on August 18, 2017, in partial fulfillment of the
requirements for the Degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

As drones and autonomous cars become more widespread it is becoming increasingly important that robots can operate safely under realistic conditions. The noisy information fed into real systems means that robots must use estimates of the environment to plan navigation. Efficiently guaranteeing that the resulting motion plans are safe under these circumstances has proved difficult.

We build a mathematical framework for analyzing the quality of estimated geometry, rigorously developing the notion of shadows. We then examine how to use these tools guarantee that a trajectory or policy is safe with only imperfect observations of the environment. We present efficient algorithms that can prove that trajectories or policies are safe with much tighter bounds than in previous work. Notably, the complexity of the environment does not affect our method's ability to evaluate if a trajectory or policy is safe.

We also examine the implications of various mathematical formalisms of safety and arrive at a mathematical notion of safety of a long-term execution, even when conditioned on observational information.

Thesis Supervisor: Leslie Pack Kaelbling
Title: Panasonic Professor of Computer Science and Engineering

Thesis Supervisor: Tomás Lozano-Pérez
Title: School of Engineering Professor in Teaching Excellence

Acknowledgments

First I must thank Leslie Pack Kaelbling and Tomás Lozano-Pérez. Ever since I joined the Learning and Intelligent Systems group (LIS) four years ago they have dedicated countless hours to mentoring me. They have advised me in research, academics, grad school applications and life in general. They demonstrated incredible patience while listening to explanations of my work, and asked questions that defined future work. They always advocated on their students' behalf to administration and fellowships and always had our futures in mind. They welcomed us into their homes, expressed concern over our well being, and made LIS feel like a family.

I also have to thank the rest of LIS. The graduate students were always open-minded, curious and willing to talk with me about research, classes and life. They were invaluable collaborators in many projects and classes. I would especially like to thank Arial Anders, Rohan Chitnis, Caelan Garret, Patrick Barragan, Clement Gehring and Gustavo Goretkin for sharing their offices with me. Those were good times. Teresa Cataldo also deserves much credit for ensuring that we were always well fed and travel and purchases went smoothly.

I would also like to thank my fellow residents of East Campus and 4E in particular. It wasn't just a place of long nights helping each other, fun outings, awesome construction projects, winning Green building challenges and looking out for each other; it was the place that reminded me that all these geniuses around me have a human side too.

I owe a lot of my mathematical development to Michael Artin and his incredible dedication to teaching. He believed that we could become mathematicians regardless of our background and spent countless hours helping us not only fill gaps in our knowledge, but truly understand and think about the material like mathematicians.

Equally important are those who influenced me before I came to MIT. David Giandomenico, Jon Penner, Steve Cousins and Kaijen Hsiao. The opportunities they provided helped me develop intellectually and as an individual. My experiences working with them still color the way I view the world today.

Finally I would like to thank my parents for raising me in such a way that encouraged independence and challenging myself intellectually. They set an example of determination and hard work that has, and continues to, inspire me every day.

Contents

List of Figures

List of definitions

Chapter 1

Introduction

The writing of this thesis coincides with the rapid deployment of robots outside of factories. Drones are inexpensive and widely available, basic autonomy is present in mass-produced cars such as the Tesla Model S and robots are autonomously delivering room service in high-end hotels.

Deploying a robot that operates robustly in unstructured environments remains incredibly difficult. Rarely is it sufficient open a textbook, implement an algorithm and release your robot into the wild. One contributing factor is the lack of efficient, provably-correct algorithms that work on realistic models. For much of the state of the art work, we expect it to fail in certain circumstances.

Many problems in robotics are provably hard to compute efficiently. Figuring out how to move a robot between two configurations and several other variants of planning problems are PSpace and NP-hard [23, 5, 28]. Figuring out where a robot is in a map and other variants of the localization problem have also been shown to be NP-Hard [6, 33]. Common variants of decision making under uncertainty are also provably hard [17, 20]. While the dream of provably correct, efficient algorithms for robotics would make life much easier for engineers, it's not clear that it is achievable. If all problems in robotics are robustly, provably hard, it doesn't make sense to search for such algorithms.

Robotics is also very difficult from a mathematical perspective. Even reasoning about seemingly simple questions such as "what joint angles place the robot hand

in the desired location" can quickly escalate into requiring understanding lie-groups and algebraic geometry. Robotics requires reasoning about randomness in locations, transformations, and even the geometry of the environment. In these cases the sometimes the basic mathematics has not yet been developed. Probability and statistics over such objects sometimes lacks the machinery necessary to answer the questions important to robotics.

In practice, these difficulties mean that most robotics algorithms rely on heuristics and lack theoretical guarantees of correctness and efficiency. Finding such ideal algorithms is not just difficult—they might simply not exist. At the same time we often find that our algorithms often fail in practice—a result that can be extremely costly if not deadly.

1.1 Motivation

This thesis examines the safe robot navigation problem. How can we get a robot from point a to point b such that the probability of collision with a set of estimated obstacles is less than some small, fixed ϵ? As drones and autonomous cars using commodity sensors become more ubiquitous it is becoming increasingly important to understand this problem.

This leads to the following questions:

- How can we characterize the quality of our estimate of the obstacle given that we can only observe it with noisy sensors?
- What is the optimal way to handle the constant stream of information available to robots in practice? Are there issues that can cause undesired behavior in practice?
- Is it even possible to efficiently estimate the probability that a single trajectory collides in the regime where ϵ is small?
- Can we efficiently solve or approximate the safe-robot navigation problem?

This thesis aims to address these questions through the development of both efficient algorithms and the mathematics necessary to understand the problem. While this thesis is primarily focused on theoretical developments, it aims to understand issues that the author encountered during more applied work in industry and academia.

1.2 Related Work

Robot navigation with uncertainty is a problem that has been studied in several different contexts. The work can be divided into several categories: methods that reason about planning under uncertainty in general fashion and are able to handle safe robot navigation; methods that reasons about uncertainty in the state of the robot; and methods that reason about uncertainty of the environment. There are also two general classes of approaches to reasoning about uncertainty. The first class of methods rely on Monte-Carlo based sampling techniques while others reason about uncertainty in a more symbolic fashion.

One general model of reasoning about uncertainty is the Partially Observable Markov Decision Process (POMDP) model—a markov decision process where the state cannot always be directly observed [13]. POMDPs can model many robotics problems with uncertainty including navigation, grasping and exploration [13, 9, 27]. Solving POMDPs is provably hard, and in some cases even undecidable [20, 17]. Though solving POMDPs is also often hard in practice, there is a large body of work aiming to solve sufficiently small, structured POMDPs [15, 26, 21, 1, 31].

In practice it is often desirable to trade generality for performance and use an algorithm tailored towards reasoning about uncertainty in robot navigation. The first class of specialized methods focuses on uncertainty in the robot's position and orientation (pose) and how well the system tracks the desired trajectory. Some methods attempt to quantify the uncertainty to identify how far the robot might stray from the ideal trajectory. They then proceed to use that information to know how far away to stay from obstacles [16, 4]. When the robot of interest has complicated, nonlinear dynamics, robust motion plans can by computed using sum-of-square funnels for

characterizing how well a robot can recover from deviations from the ideal trajectory [32, 18, 29, 30].

Other branches of work focus more on uncertainty in the shape and location of the obstacles themselves. Some works attempt to identify shadows, or regions where the obstacle could be and avoid those during planning [12, 24]. However, since models of random geometry were not widely available and understood, these works often failed to provide theoretical guarantees and sometimes make errors in their derivations resulting in trajectories that are consistently unsafe [24]. Given a generative model for obstacles, it becomes possible to develop a sampling-based method that is provably safe [25, 11]. However, these methods have a runtime that depends on $\frac{1}{\epsilon}$ where ϵ is the allowable probability of collision. In the regime where the acceptable probability of collision, ϵ, is very small, these methods can become very slow.

1.3 Contributions and Outline

This thesis builds machinery to help understand and solve the safe navigation problem. Chapter 2 builds the mathematical foundations necessary to both understand and solve the safe planning problem. It starts by addressing the question of how to define a probability of collision. It defines a distribution over shapes which we use as a model for the remainder of the work. Such obstacles are referred to as polytopes with Gaussian distributed faces (PGDF). It then discusses how to compute the intersection of probability of an arbitrary shape a random half-space drawn from any distribution. It introduces the notion of a shadow—the generalization of a confidence interval to shapes. The notion of shadows will be crucial in the development of efficient algorithms later in the work. Finally we show how to construct shadows for the distribution defined earlier in the chapter. We believe this work has applications beyond the safe navigation problem.

Chapter 3 discusses an efficient algorithm for bounding the probability of collision of a trajectory. It develops an algorithm that can be used to bound the probability of intersection of any collision checkable set with a set of PGDF obstacles. This

algorithm can be used to efficiently bound the probability that a robot trajectory will collide with a set of estimated obstacles. While the probabilities produced by the algorithm are not tight we briefly discuss tightness in the low-probability regime.

Chapter 4 discusses algorithms for the safe planning problem. It defines the safe planning problem as the search for a robot trajectory with probability of collision less than a fixed ϵ. Then we examine the approximate safe planning problem where the true probabilities of collision are replaced with the bounds computed using the algorithm in chapter 3. We examine the computational complexity of the problem. First we show a trivial, exponential time algorithm that shows the problem is in NP. We then discuss the submodularity of the cost function which we hope can be used to find a polynomial time algorithm to solve this approximate safe planning problem. We briefly discuss some hardness results of the problem with high dimensional obstacles.

Chapter 5 examines the issues with generalizing the above methods to the online setting where the robot has a constant stream of information and can change its desired trajectory. This is the "safe" analog of transitioning from motion planning to policy search. Unlike the setting where we are presented a single information set before execution, safety is not guaranteed by always following a trajectory with a low probability of collision. We present a toy example that illustrates this phenomenon. Even though the robot is always following a safe trajectory, its probability of colliding approaches 1. We suggest a different criterion for safety—guaranteeing that the lifetime probability of collision does not exceed a particular value. We then present a simple way of ensuring that such a criterion is met. We analyze when the presented criterion is necessary in addition to sufficient. In order to better understand the criterion we compare it to safety under an information oracle—an oracle which can cut off the flow of information at any point in time. This helps us develop an equivalent, more intuitive way to guarantee safety without relying on a mathematically technical formulation. Finally we show how the algorithm introduced in chapter 3 can be modified to ensure that a policy is safe.

Finally chapter 6 discusses future directions based on the presented work. We discuss other potential applications of bounding random geometry, like manipulation

under uncertainty. We also discuss the development of better models and how they can be understood using the machinery developed in chapter 2. We discuss directions for future work on safe planning that are guided by the included complexity results and known structure. Finally we take a step back and analyze the potential impact of the presented ideas to the deployment of safe, efficient and reliable robots in the real world.

Chapter 2

Shadows

In order to be able to provide safety guarantees for robot operation in domains with uncertainty about obstacles, we must develop a formal understanding of the relationship between randomness in the observations and resulting estimate. Consider the following scenario: the robot must avoid an obstacle but gets only a noisy observation of said obstacle. Given this noisy observation, it computes an estimate of the space occupied by the obstacle and avoids this region. Given the observation, however, the true space occupied by the obstacle is random and not guaranteed to be inside the estimated region. It is not sufficient for the robot to avoid the estimated region to ensure non-collision.

In order to provide theoretical guarantees about a robot's operation in such a domain we must develop mathematics regarding the *random shapes* that come from estimating geometry with noisy observations. The ultimate aim of this chapter is to develop *shadows*, a geometric equivalent of confidence intervals for uncertain obstacles. The concept of shadows will prove useful in the development of provably correct algorithms for safe robot navigation problem later in the thesis.

Understanding the mathematics behind shadows turns out to be rather involved. Section 2.1 begins by formally defining the notion of an ϵ-shadow. A significant portion of the section is spent motivating our definition of shadows and seeing how it avoids issues with previous methods of bounding random geometry. We then provide an example of a distribution from which these random shapes can be drawn. We

introduce polytopes with Gaussian distributed faces (PGDF) which will be used to develop examples and algorithms throughout this thesis.

It turns out that shadows can have surprising and non-intuitive behavior. In order to build an intuition for shadows we provide some theorems that characterize half-space shadows of any distributions. First we show how to find collision probabilities between a random half-space and an any set. This helps introduce some of the basic geometric notions that will be used to understand and construct shadows. We then develop a theorem that helps us understand the existence and uniqueness of an ϵ-shadow for any distribution and a fixed ϵ. The answers here are can be surprising and suggest that random geometry can behave quite differently than random vectors.

Section 2.2 addresses the algorithmic question of how to compute shadows for PGDF obstacles. It provides several algorithms for computing shadow **lower bounds** for PGDF obstacles. Unlike the shadows of section 2.1, these bounds are often loose: there are situations in which the computed $\epsilon-$shadow is also a $\epsilon'-$shadow for $\epsilon' < \epsilon$. We compare the tightness of these different shadow bounds in different scenarios. We also present a general method of building ϵ-shadows for polytopes given only a mechanism for generating shadows of half-spaces, and we comment on the tightness of the bounds generated with the method.

2.1 Mathematics of Shadows

In this section we develop the mathematical machinery necessary to build and characterize shadows. Computational and algorithmic developments will be presented in a later section.

2.1.1 Definitions

In this section we define the notion of a shadow as well as give an example distribution of obstacles for which we can compute shadows for concrete obstacle estimates.

In order to be able to distinguish the space that the robot operates in from the space of half-space parameters we introduce the following notation: bold, nonscript

capital letters refer to subsets of $\mathbb{R}^n$ corresponding to the robot's workspace. Our robots usually operate in $\boldsymbol{X} = \mathbb{R}^n$. $\boldsymbol{X}$ refers to this vector space throughout the document. Capital script letters refer to subsets of the dual of the robot's workspace' i.e., sets of half-space parameters over the robot's workspace. Half-space obstacles can be represented as points in $\mathcal{X} = \mathbb{R}^n$. $\mathcal{X}$ refers to this space of half-space parameters throughout the document. We also note that the goal of the chapter is to bound *random obstacles*. A random obstacle is defined as a random convex polytope which the robot is not allowed to collide with. This analysis can be made applicable

Shadows

Definition 1 ($\epsilon-$shadow)**.** *A set $\boldsymbol{S} \subseteq \boldsymbol{X}$ is an $\epsilon-$shadow of a random obstacle $\boldsymbol{O}$ if $P(\boldsymbol{O} \subseteq \boldsymbol{S}) \geq 1 - \epsilon$.*

It is important to note that an $\epsilon-$shadow may be quite different than the set of points with probability more than ϵ of being inside the random obstacle as used by Sadigh and Kapoor [24].

Guaranteeing non-intersection of the robot trajectory with the obstacle given only the set of points of low probability of colliding, requires either understanding the correlation between the event that nearby points collide or using a union bound over all points in the trajectory.

Since robot trajectories necessarily pass through an infinite number of points, a union-bound alone cannot be used to guarantee safety. While in this case, a guarantee can be found using smoothness of the Gaussian PDF and constructing simplices that enclose the trajectory, doing so introduces additional restrictions on the resulting trajectories. In practice this usually causes the the computed probability bound to be greater than 1, rendering the theoretical guarantees of the algorithm useless.

An example of the difference between a shadow and the set of points likely in the obstacle can be see in figure 2-1. The shadow is a subset of the likely points, but can used to verify the trajectory of the safety. The set of likely points is not sufficient information to verify the safety of the trajectory.

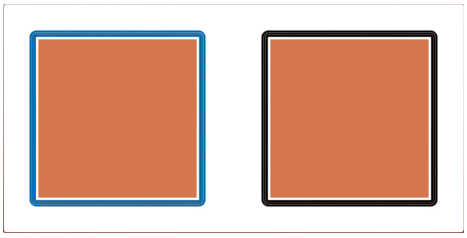


Figure 2-1: A fair coin is flipped. If the coin is heads the square is in the left position, otherwise in the right. The orange region identifies the points with probability at least 0.5 of being in the obstacle. Either outline represents a valid 0.5−shadow. The shadow is sufficient to show that a trajectory that avoids the shadow has probability less than 0.5 of colliding with the obstacle. Merely knowing that the orange region is the set of points with at least 0.5 probability of being in the obstacle is not sufficient to provide any guarantees about the safety of a trajectory. We also note that shadows need not be unique—two 0.5-shadows are shown.

Polytopes with Gaussian Distributed Faces

Computing shadows requires having some sort of characterization of the uncertainty with respect to obstacles in the environment. We formalize this notion as assumptions about the distribution from which obstacles are drawn.

We use homogeneous coordinates throughout for simplicity. We recall that a polytope $\boldsymbol{P} \subseteq \boldsymbol{X}$ is the intersection of half-spaces:

$$\boldsymbol{P} = \bigcap_i \alpha_i^T x \leq 0.$$

We consider the case where the parameters $\alpha_{1\ldots k}$ are drawn from Gaussian distributions with known parameters. From here on, a polytope with parameters drawn from Gaussian distributions is referred to as a polytope with Gaussian distributed faces (PGDF). Note that for any $\lambda \geq 0$, $\lambda\alpha$ and α correspond to the same half-space. We will show that this model can be derived under reasonable conditions and discuss the fundamental limitations and drawbacks of the model.

This model is relevant because it can be derived from segmented point-cloud data. If we are given a point cloud representation of the obstacle with points grouped by face and the points are observed with Gaussian noise then the true faces can be estimated with a linear regression. If this regression is performed in a Bayesian setting, the

posterior distribution over the face parameters is a Gaussian distribution, suggesting that assuming α is Gaussian distributed is reasonable [2, 22].

The model, however, does discard information often available to robotic systems. Robot sensors such as Kinects and LIDARs often provide point clouds in conjunction with free space information. In addition to identifying the first time each "ray" hits an obstacle in the environment, these sensors also provide the information that the space in between the sensors and the point is likely collision free.

Later in this chapter we show how to efficiently compute $\epsilon-$shadows for the PGDF model in practice.

2.1.2 A characterization of half-space Shadows

Before we discuss how to compute shadows efficiently in practice, we develop a mathematical theory of shadows to help get intuition for their behavior. For random half-spaces of any distribution we will answer the following questions: When does there exist an ϵ-shadow? Are shadows unique?

We connect the notion of a shadow with the classic notions of dual and polar cones in convex analysis. The analysis in this section is done with complete generality and does not assume that obstacles are PGDF. The only restriction, which we take to make analysis cleaner, is that sets with an empty interior under the standard topology have zero measure. More intuitively, we assume the probability that we select parameters from a zero volume set is zero.

Exact Probabilities of Intersection

First we compute the exact probability that a random half-space $\alpha^T x \leq 0$, parametrized by α, will intersect an arbitrary set $\boldsymbol{A} \subseteq \boldsymbol{X}$. We relate this value to the measure of a dual cone, a notion from convex geometry. More formally, we examine how to find:

$$P(\exists x \in \boldsymbol{A} \text{ such that } \alpha^T x \leq 0).$$

Theorem 1. *$P(\exists\, x \in \boldsymbol{A} : \alpha^T x \leq 0) = 1 - \mu(\mathcal{C}^\star)$ where $\mathcal{C}^\star$ is the dual cone of $\boldsymbol{A}$.*

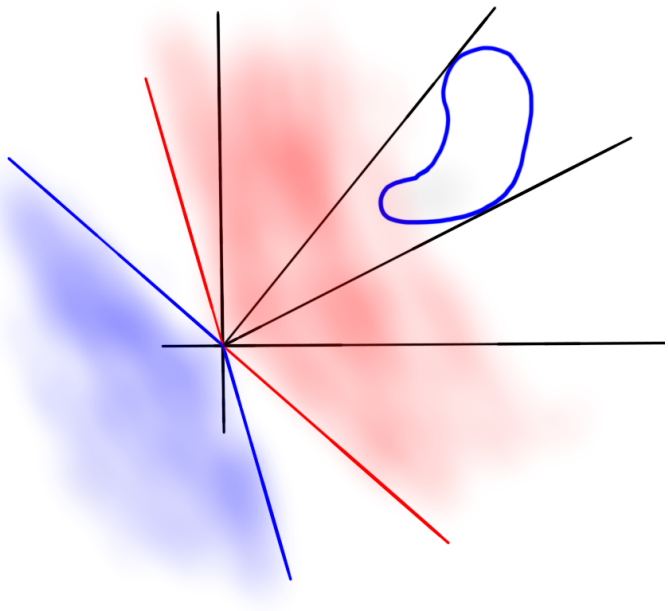


Figure 2-2: The red cone is the dual cone of blob outlined in blue. The blue cone is the polar cone of the same region. Note how the polar cone is a reflection of the dual cone. The red and blue cones are in the dual space while the blob outlined in blue is not.

Proof. Recall that the dual cone $\mathcal{C}^\star$ of $\boldsymbol{A}$ is the set $\{\alpha \mid \alpha^T x \geq 0,\ \forall x \in \boldsymbol{A}\}$.

First we examine the set of parameters α such that there exists an $x \in \boldsymbol{A}$ such that $\alpha^T x \leq 0$. The complement of this set is the set $\{\alpha \mid \alpha^T x > 0\ \forall x \in \boldsymbol{A}\}$. Thus

$$P(\exists\ \boldsymbol{A} \in X : \alpha^T x \leq 0) = 1 - P(\alpha^T x > 0,\ \forall x \in \boldsymbol{A}).$$

Up to the boundary, which is measure zero, $\{\alpha \mid \alpha^T x > 0,\ \forall x \in \boldsymbol{A}\}$ is exactly the dual cone of $\boldsymbol{A}$ up to the zero-measure boundary.

$$\begin{aligned} P(\exists\ x \in \boldsymbol{A} : \alpha^T x \leq 0) &= 1 - P(\alpha^T x > 0,\ \forall x \in \boldsymbol{A}) \\ &= 1 - \mu(\mathcal{C}^*). \end{aligned}$$

□

Shadows and Polar Cones

A related notion in convex geometry, the polar cone, allows us to relate the construction of shadows with a distribution in the halfspace parameter space. Please see appendix A for the related definitions. A correspondence theorem defines a bijection between a set we understand (polar cones) and a set we are trying to characterize (shadows). This will allow use to use our knowledge of the existence and uniqueness of polar cones to understand the existence and uniqueness of shadows.

The proof of the correspondence theorem (theorem 3) will highlight a problem with our current definitions. Since a $0.25-$shadow is also a $0.5-$shadow, it will prove difficult to construct a one-to-one map between all shadows and another object which we understand. Thus we consider only maximal shadows.

Definition 2 (Maximal ϵ-shadow)**.** *An $\epsilon-$shadow $\boldsymbol{A} \subseteq \boldsymbol{X}$ of a random shape O is maximal if $P(O \subseteq \boldsymbol{A}) = 1 - \epsilon$.*

This leads to our first, and most general, method of constructing shadows.

Theorem 2. *For every set $\mathcal{Y} \subseteq \mathcal{X}$ of measure $1 - \epsilon$, the polar cone $\mathcal{C}^0$ of $\mathcal{Y}$ is a $\epsilon-$shadow for the random shape defined by the measure.*

Proof. Recall that a point x is in collision with the halfspace defined by α if $\alpha^T x \leq 0$.

Consider a set $\mathcal{Y}$ of measure $1 - \epsilon$. Since sets with empty interiors have zero measure, we can assume without loss of generality that $\mathcal{Y}$ is open.

First we identify the set of points $\boldsymbol{A} = \{x \mid \alpha^T x > 0,\ \forall \alpha \in \mathcal{Y}\} \subseteq \boldsymbol{X}$, that is the set of points not inside any halfspace defined by a point in $\mathcal{Y}$. Up to the boundary, which is a set of measure zero, this is exactly the polar cone, $\boldsymbol{C}^0$ of $\mathcal{Y}$.

With probability $1 - \epsilon$, a draw of α will be in $\mathcal{Y}$ and thus not correspond to a halfspace that intersects $\boldsymbol{C}^0$. This implies that $\boldsymbol{C}^0$ is an $\epsilon-$shadow. □

Theorems 1 and 2 can be combined to give a correspondence theorem that will allow us to better understand shadows.

Theorem 3. *There is a one-to-one correspondence between convex cones in parameter space of measures $1 - \epsilon$ and maximal ϵ-shadows.*

Proof. Our proof will show that applying the construction in theorems 1 and 2 and reversed yields the identity.

First we start with a convex cone in the space of half-space parameters, $\mathcal{Y}$, of measure $1 - \epsilon$. Theorem 2 tells us that the polar cone of $\mathcal{Y}$ is an $\epsilon-$shadow. The construction in theorem 1 tells us that the probability of the shadow not containing the random halfspace is the measure of the negative of it's dual cone.

Since $\mathcal{Y}$ is a convex cone, the negative of the dual cone of the polar cone the original set $\mathcal{Y}$ itself.

The same procedure works for the reverse direction. □

Theorem 3 gives us guidelines about how to construct shadows. It shows that the sets used to compute the probabilities of shadows should be convex cones if we wish for our shadows to be tight.

It also gives insight to when to when $\epsilon-$shadows are not unique. Any set of measure $1 - \epsilon$ with a distinct polar cone can be used to create a distinct shadow.

The non-uniqueness gives insight into why not all shadows are equivalent when bounding the probability of intersection. One $\epsilon-$shadow may be sufficient to certify non-collision, but another might not. This suggests that we will have to search through a continuous set of shadows for the one that gives the strongest bound.

Finally it helps us answer the question of whether nontrivial shadows always exist. In $\mathcal{X} = \mathbb{R}^n$, if the distribution over parameters is such that any halfspace through the origin has measure greater than ϵ then the only shadow is the entire space (the trivial shadow). This comes from the fact that the minimal convex cone with sufficient measure then becomes the entire space. For example, consider constructing an $\epsilon = 0.25$-shadow with $\alpha \sim \mathcal{N}(0, I)$. The distribution is symmetric and all halfspaces through the origin have measure 0.5. Thus we cannot construct any non-trivial $0.25-$shadows for this distribution.

This suggests a procedure by which we can find the maximal ϵ such that no ϵ-shadow smaller than the full space cannot exist.

Theorem 4. *Let*

$$\epsilon^\star = \inf_{\alpha \in \mathcal{X}} \mu(\alpha^T x \leq 0)$$

*Then for all $\epsilon' < \epsilon$, there do not exist **any** $\epsilon'-$shadows.*

Since any convex cone that strictly contains a halfspace through the origin must be the entire vector space, and no halfspace has measure more than ϵ, for any $\delta < \epsilon$ there cannot exist an $\delta-$shadow. In other words there is no shadow that contains the set with probability more than $1-\epsilon$. We note that for PGDF obstacles there always exists an $\epsilon \in (0, 1)$ such that a non-trivial shadow does not exist. This fact is surprising, does not match our intuition and suggests that care must be taken in ensuring that planning methods are truly safe. Future work also includes understanding whether other important models of random geometry also undergo this phase transition.

2.2 Computing Shadows for PGDFs

In the previous section we constructed shadows for halfspace obstacles in full generality. Unfortunately, even the integrals required to determine the probability that a shadow contains an obstacle can be difficult to compute. In this section we will suggest three different ways to compute shadow lower bounds. Instead of trying to compute a shadow that is tight with respect to its probability, we will construct shadows for which it is easy to compute a lower bound on the probability that a shadow contains a random shape. At first we will restrict our attention to halfspaces from a PGDF faces. In section 2.2.3 we will use the developments in the previous section to construct a shadow for a PGDF with an arbitrary number of faces. Sections 2.2.1 and 2.2.1 present bounds that are more effective when the parameters of the halfspace are relatively well known (when the mean of the distribution is far from the origin with respect to the norm induced by the precision matrix) and section 2.2.2 presents a bound that works better when there is relatively little information about the parameters.

2.2.1 Gaussian Elliptical Shadows

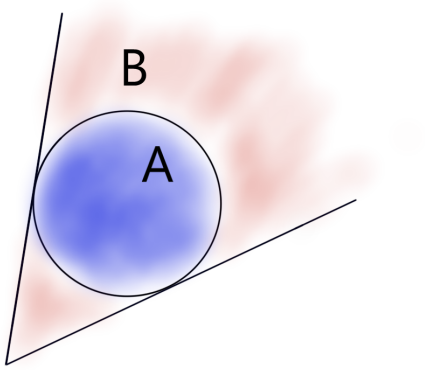


Figure 2-3: Set B corresponds to parameters of half-spaces that will be contained within our shadow. We lower bound the measure of set B using the measure of set A, an ellipsoid of measure $1 - \epsilon$.

In this section we construct a shadow for a given distribution of random shapes as follows. We identify a sufficiently large scaled covariance ellipse around the mean parameter vector such that its measure is $1 - \epsilon$. We then take the polar cone of this ellipse and use it as our shadow. A pictorial outline of the derivation is presented in figure 2-3.

We begin by identifying required size of the covariance ellipse.

Lemma 1. *Let $\alpha \sim \mathcal{N}(\mu, \Sigma)$, ϕ be cdf of the Chi-Squared distribution with n degrees of freedom and:*

$$\mathcal{X} = \{\beta \mid (\beta - \mu)^T \Sigma^{-1} (\beta - \mu) \leq \phi^{-1}(1 - \epsilon))\}.$$

Then $P(\alpha \in X) = 1 - \epsilon$.

Proof. First we note that α is equal in distribution to $\Sigma\alpha' + \mu$ with $\alpha' \sim \mathcal{N}(0, I)$. $\alpha'^T \alpha'$ is then a Chi-Squared random variable with n degrees of freedom. Thus $P(\alpha'^T \alpha' \leq \phi^{-1}(1 - \epsilon)) = 1 - \epsilon$. If we let $Z = \{\alpha \mid \alpha^T \alpha \leq \phi^{-1/2}(1 - \epsilon)\}$ then $P(\alpha' \in Z) = 1 - \epsilon$. Let Y be the image of Z under the map we used to generate a random variable

identical in distribution to α. Then $Y = \{\beta \mid (\beta - \mu)^T \Sigma^{-1} (\beta - \mu) \leq \phi^{-1}(1 - \epsilon)\}$ and the measure of Y under the distribution over α must be the same as the measure of Z under the distribution over α'. Thus $P(\alpha \in Y) = 1 - \epsilon$. □

Now, given this ellipse of sufficient measure, we can compute its polar cone and resulting $\epsilon-$shadow. We note that if the ellipse given in lemma 1 contains the origin in its interior, the resulting polar cone will be empty. We compute the polar cone by first computing the minimal cone $\mathcal{C}$ which contains the ellipse, and then computing the polar cone of $\mathcal{C}$.

For the remainder of the section we assume that the space has been rotated and scaled such that $\mu = (0..., 0, 1)$.

Theorem 5. *For nondegenerate PGDF halfspaces, there exists Σ' such that*

$$\boldsymbol{X} = \{< x_1 ....x_{n-1}, z > |\ x^T \Sigma x \geq z\}$$

is an $\epsilon-$shadow.

Our proof is constructive and tells us how to compute Σ.

Proof. Let $\mathcal{X} = \{\beta \mid (\beta - \mu)^T \Sigma (\beta - \mu) \leq r^2\}$ be the ellipse identified in lemma 1.

We expand the equation of the above surface for convenience:

$$\begin{aligned} r^2 &= (\beta - \mu)^T \Sigma (\beta - \mu) \\ &= \beta^T \Sigma \beta - 2\beta^T \Sigma \mu + \mu^T \Sigma \mu. \end{aligned}$$

Now we compute the equation for the normals to the surface at point x

$$2\Sigma x - 2\Sigma \mu.$$

Then we identify the set where the normal vectors are orthogonal to the vector to the

point x:

$$\begin{aligned} x^T(\Sigma x - \Sigma\mu) &= 0 \\ x^T\Sigma x - x^T\Sigma\mu &= 0 \\ x^T\Sigma x &= x^T\Sigma\mu. \end{aligned}$$

Plugging this into the equation of the original ellipse gives us the equation for the plane that contains the set where the ellipse is tangent to the minimal containing cone:

$$\begin{aligned} \beta^T\Sigma\mu - 2\beta^T\Sigma\mu + \mu^T\Sigma\mu &= r^2 \\ -\beta^T\Sigma\mu + \mu^T\Sigma\mu &= r^2 \\ \mu^T\Sigma\beta &= \mu^T\Sigma\mu - r^2. \end{aligned}$$

This is a linear equation in β which we can use this to solve for β_n in terms of the remaining indices of β. Substituting it plugged into the original equation yields the equation of a new ellipse. Let Σ^E, x_0, r' be the parameters of this new ellipse in the form $(\beta^T_{1\ldots n-1} - x_0)^T\Sigma^E(\beta_{1\ldots n-1} - x_0) \leq r'^2$.

Now we can directly identify the equation of the cone as:

$$\beta^T \begin{pmatrix} \Sigma^E & 0 \\ 0 & -r^2 \end{pmatrix} \beta \leq 0;\ \alpha_n \geq 0.$$

Alternatively we can describe the set as $\{\beta \mid \sqrt{\beta^T_{1..n-1}\Sigma'\beta_{1..n-1}} \leq \beta_n\}$, where $\Sigma' = \frac{\Sigma^E}{r^2}$.

This identifies our cone as a standard-norm cone with the norm induced by Σ', and makes finding the dual cone a standard problem. The dual cone is just $||\beta_{1\ldots n}||_\star \leq \beta_n$ where $||\cdot||_\star$ denotes the dual norm [see 3, example 2.25, page 52]. The dual norm is the natural one induced by Σ'^{-1}. Thus the dual cone is

$$\beta^T \begin{pmatrix} \Sigma'^{-1} & 0 \\ 0 & 1 \end{pmatrix} \beta \leq 0;\ \beta_n \geq 0.$$

Finally since $C^0 = -C^\star$ we can write down the equation of the polar cone which defines the shadow.

$$\beta^T \begin{pmatrix} \Sigma'^{-1} & 0 \\ 0 & 1 \end{pmatrix} \beta \leq 0;\ \beta_n \leq 0$$

□

When we dehomogenize the coordinate system we get a conic section as a shadow. We refer to Handlin's Conic Sections Beyond $\mathbb{R}^2$ for a classification of different sections and a discussion of conic sections in high dimensions [8], but we note that this step implies that the procedure does not always produce non-degenerate shadows.

Double Ellipse Refinement

Figure 2-3 clearly illustrates that the shadows computed using the above method are not tight. We note that the resulting probabilities can be improved as follows: instead of using a single ellipse to lower bound the measure of set B we will use two half-ellipses as in the figure below. The first half-ellipse is computed using the same computation as above, and the second is the maximal-radius half-ellipse with the same center as the original ellipse. This is illustrated in figure 2-4

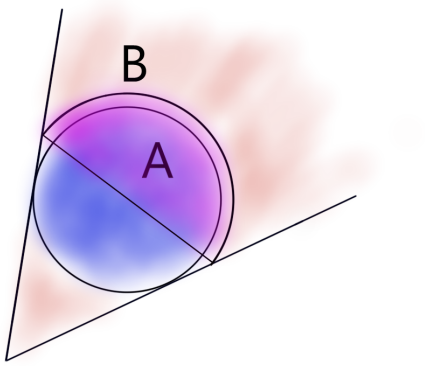


Figure 2-4: Using two ellipses allows us to compute a tighter bound

2.2.2 Conic Shadows

We note that the shadow bounds derived in section 2.2.1 fail to provide a non-trivial shadow when the covariance is too "small" relative to the distance of the mean from the origin. As this becomes the case another class of shadow lower bound approaches being tight. We suggest how such a bound can be computed below. We can compute the measure of a cone going through the mean of the distribution by transforming our space to one where the Gaussian distribution is Isotropic and then computing the portion of the full space taken up by cone. This process is illustrated in figure 2-5.

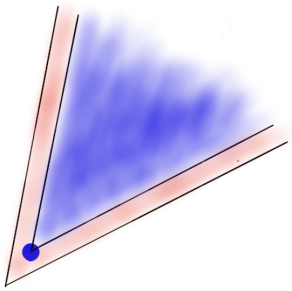


Figure 2-5: Using a cone through the mean to compute a shadow lower bound.

2.2.3 From Halfspaces to Polytopes

The previous sections focused on half-spaces because duality is very intimately related to the construction of shadows. This relationship is not as powerful for general polytopes, and makes computing the exact probabilities corresponding to shadows difficult without a large enumeration.

Recall that a polytope is defined as the intersection of half-spaces $\bigcap_i \alpha_i^T x \leq 0$. Constructing tight shadows for polytopes in practice depends critically on understand the correlation between $\alpha_{1\ldots n}$. In this section we make no assumptions regarding the correlation between the α's. In practice, we may have strong prior shape information but a poor estimate of the location of the object. In this case our shadows would not

be maximal. In other words, there may exist a strictly smaller set that is also a valid ϵ-shadow. We demonstrate an example where our shadows are tight, and provide some analysis suggesting cases the shadows we construct are close to tight. Exploring how knowledge of correlation may be used to produce tighter shadows for polytopes remains an interesting direction for future work.

We can construct an ϵ-shadow from shadows for individual faces and a union bound.

Theorem 6. *Given $\lambda_i \in \mathbb{R}^+$, $\sum_i \lambda_i = 1$ and Y_i an $\lambda_i\epsilon$ shadow of the half-space defined by α_i, $\bigcap_i Y_i$ is an $\epsilon-$shadow of $\bigcap_i \alpha_i^T x \geq 0$.*

Proof. We note for the object to not be contained within the shadow, at least one face must fall outside of its corresponding shadow Y_i. We will bound the probability that any face falls outside its shadow.

The probability that the face corresponding to α_i falls outside of its shadow is $\lambda_i\epsilon$. Applying a union bound gives us that the probability that any face falls outside its shadow is upper bounded by $\sum \lambda_i\epsilon = \epsilon$. □

We note that one natural choice of λ_i is $\frac{1}{m}$ where m is the number of faces. We note that this may, however, be suboptimal if some faces are much less likely to cause problems for robot safety than others.

Chapter 3

Algorithm for Bounding Probability of Collision

In this section we focus on bounding the probability that a given robot trajectory collides with a set of obstacles drawn from the posterior distribution of a Bayesian estimation process. One of the most important consequences of the developments of chapter 2 is a fast algorithm for bounding the probability that a fixed set intersects a PGDF obstacle. We develop a generic algorithm that can bound the probability of intersection between a set of PGDF obstacles with any collision-checkable set.

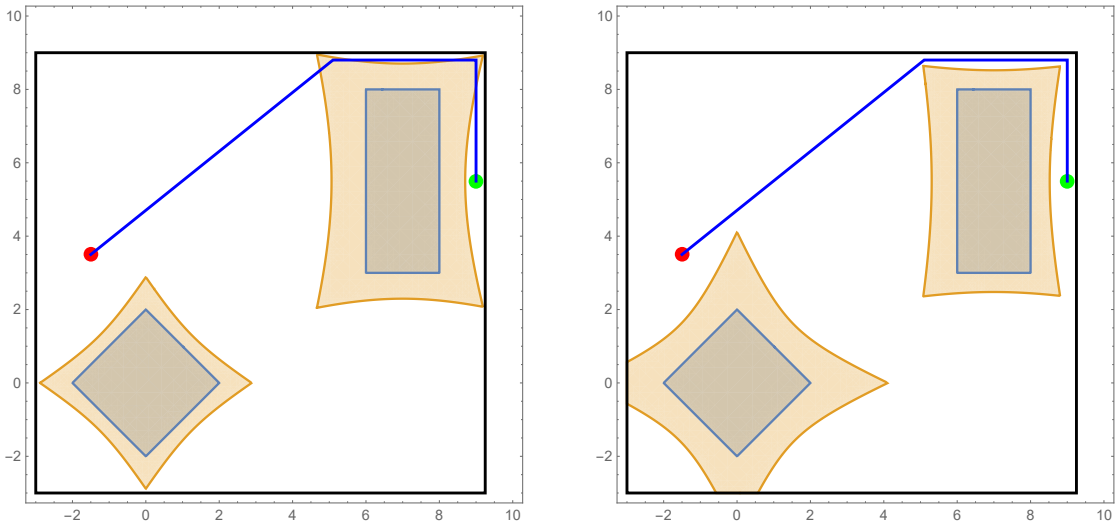
3.1 Motivation

One way to estimate the probability of collision is to use a Monte Carlo technique based on sampling the location of obstacles and collision checking the trajectory. Take for example the following naïve algorithm. Sample n sets of obstacle geometries and let p be the portion of these geometries for which there is a collision. This algorithm is very simple and does not introduce any slackness ($E[p]$ is the exact probability of collision). However, if you want to guarantee the probability of collision is less than ϵ, n must be at least $O\left(\frac{1}{\epsilon}\right)$ samples for this method to be useful. In essence the algorithm has to see at least a couple of failures in expectation to be useful. This a fundamental limitation of sampling based algorithms that do not sample in a

transformed space.

In the context of robot safety, where we want robot collisions to be extremely rare events (ex. $p = 10^{-7}$), this implies that Monte Carlo methods tend to perform poorly. While variance reduction techniques have been shown to improve performance, it is not sufficient in the regime where collisions happen rarely [10].

In this chapter we show how the $O\left(\frac{1}{\epsilon}\right)$ dependence can be avoided by searching for shadows instead of sampling. We will search for a certificate, or proof, that the probability is less than ϵ. This will enable our method to have a complexity $O\left(\log \frac{1}{\epsilon}\right)$, linear in the number of required digits of precision. Searching for a shadow also results in tighter bounds than previous work without sacrificing correctness. Another benefit of searching for shadows is that it avoids the slackness that would be introduced by taking shadows of the same probability and using a union bound to compute the probability bound. The difference is highlighted in figure 3-1.



(a) An even allocation of shadows. Notice the intersection of the shadow and trajectory in the top right.

(b) A carefully chosen allocation of shadows that can prove the trajectory is safe. Notice how the bottom left obstacle has a much larger shadow.

Figure 3-1: An optimal pair of shadows can show that the trajectory is safe, while shadows of equal probability cannot. The sum of the probabilities corresponding to the two shadows in figures a) and b) are equal.

This contrasts with many previous works that used uniformly sized shadows such as [12] and [24]. This implies that in the worst case a method using uniformly sized

shadows can only find a plan if an $\frac{\epsilon}{n}$ safe plan exists. Not only can this lead to unnecessarily conservative behavior, it also has a dependence on the environment that does not necessarily affect the true probabilities of collision in a way that matches our intuition. For safety what matters is the number of obstacles "close" (measured by a combination of distance and uncertainty) to the trajectory, not just the number of obstacles present in the environment. Searching for shadows instead of choosing their size uniformly allows us to close the quality gap between geometric/symbolic algorithms and Monte-Carlo algorithms by a factor of n, without sacrificing correctness.

An interesting byproduct of our algorithm is a safety certificate. Our method finds a set of shadows that show the trajectory in question is safe with probability at least $1-\epsilon$. These shadows provide easy to verify, interpretable, proof that the trajectory is safe. These certificates can be used to understand which obstacles are likely to collide with the robot and which are not. Formally the problem we address is follows: given a set X and PGDF obstacles drawn from distributions with parameters μ_{ij}, Σ_{ij}, find the minimal ϵ such that the probability that an obstacle intersects X is less than ϵ. We explain our algorithm in two parts. First, in section 3.2 we present an algorithm that bounds the probability that a single obstacle intersects the fixed set. This allows us to focus on the search of a single shadow and understand the restrictions of our algorithm. In section 3.3 we present serial and parallel versions of the algorithm for many obstacles and discuss computational complexity in further detail.

3.2 The Single Obstacle Case

In the case of a single obstacle, we can frame finding a probability bound as the search for the maximal shadow. In other words we wish to find a solution to the following optimization problem where X is the set of states visited by the robot trajectory (the

swept volume):

$$\begin{aligned} &\underset{\epsilon\in(0,1)}{\text{minimize}} \quad \epsilon \\ &\text{subject to} \quad \text{shadow}(\epsilon) \cap X = \varnothing \end{aligned} \tag{3.1}$$

It is not immediately clear that this optimization is convex, or otherwise solvable in polynomial time. In order to make the optimization easily solvable we restrict ourselves to a class of easily computable shadow lower bounds obtained in chapter 2. In particular, we restrict ourselves to shadows coming from the single and double ellipse bounds for PGDF obstacles presented in sections 2.2.1 and 2.2.1 combined with theorem 6. When this mapping is non-degenerate, it is a one-to-one from probabilities to shadows. Let $S(\epsilon)$ be one of these two maps. We restrict ourselves to the following optimization:

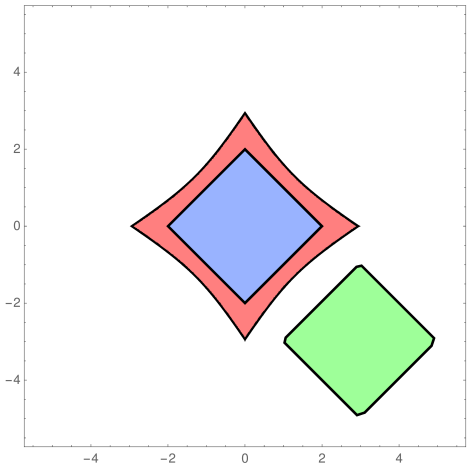
$$\begin{aligned} &\underset{\epsilon\in(0,1)}{\text{minimize}} \quad \epsilon \\ &\text{subject to} \quad \mathrm{S}(\epsilon) \cap X = \varnothing \end{aligned} \tag{3.2}$$

An important property of the map is that it is monotone. If $\epsilon_1 \geq \epsilon_2$ then $S(\epsilon_1) \subset S(\epsilon_2)$. This implies that the set of probabilities that map to shadows that do not intersect X is an interval subset of $(0, 1)$. This means that we can find the minimal ϵ for this class of shadows with a simple bisection search on the interval $(0, 1)$. The pseudocode for this search is presented in algorithm 1 and a figure illustrating the execution is shown in figure 3-2.

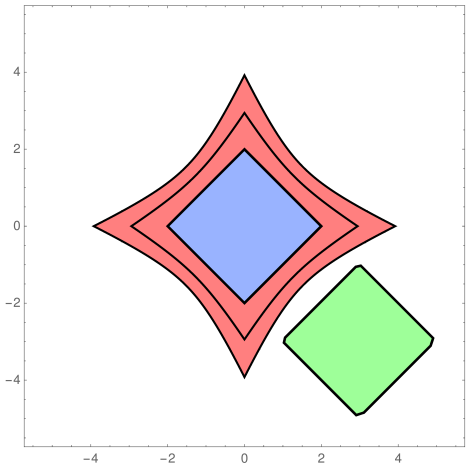
To achieve precision ϵ, algorithm 1 requires $O\left(\log \frac{1}{\epsilon}\right)$ iterations due to the bisection search. In every iteration there is a single call to a collision checker.

3.3 Generalization to Multiple Obstacles

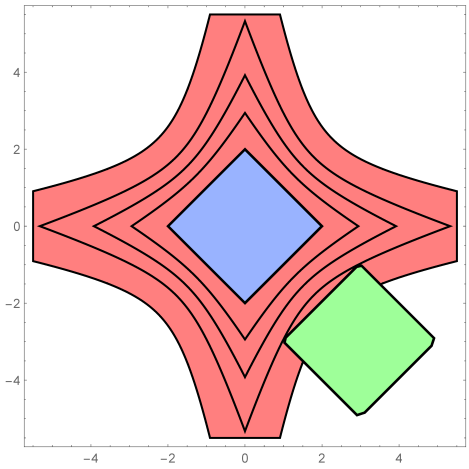
In this section we generalize algorithm 1 to handle multiple obstacles, analyze the resulting computational complexity, and show that it provides a tighter bound on the probability of collision than previous work. We refer to the collision probability as



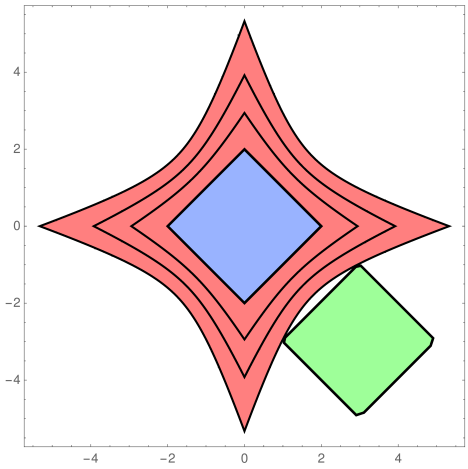
(a) First iteration of the bisection search



(b) Second iteration of the bisection search



(c) Third iteration of the bisection search. Since it overshot in this iteration (detected as a collision between the shadow and robot by the collision checker) it will try a smaller shadow in the next iteration



(d) Fourth and final iteration of the bisection search as it has found a shadow that does not collide within the desired numerical precision of the optimal one.

Figure 3-2: An visualization of the execution of algorithm 1

Algorithm 1 Finding an Optimal Shadow

```
1: function FindOptimalShadow(X, μ, Σ, ε)
2:     ε_min ← 0
3:     ε_max ← 1
4:     while ε_max − ε_min ≤ ε do
5:         ε ← (ε_max − ε_min)/2
6:         S ← GenerateShadow(μ, Σ, ε)
7:         if Intersects(S,X) then
8:             ε_min ← ε
9:         else
10:            ε_max ← ε
11:        end if
12:    end while
13:    return ε_max
14: end function
```

risk throughout this section.

In order to generalize algorithm 1 we search for a separate shadow for each obstacle and then sum up the probabilities for each shadow (justified by a union bound). The search can be done for each obstacle in parallel.

Algorithm 2 Finds a bound on the risk of the set X intersecting a set of random obstacles.

```
1: function AggregateRiskBound(X, μ_{1...n}, Σ_{1...n}, ε)
2:     for i ← 1...n do
3:         ε_i ← FindOptimalShadow(X, μ_i, Σ_i, ε/n)
4:     end for
5:     return Σ_i ε_i
6: end function
```

After compensating for the increased numerical precision necessary for multiple obstacles, we get the computational complexities presented in table 3.1.

Table 3.1: Computational complexity of shadow finding in algorithm 2 for numerical precision ε, minimum probability ϵ and n obstacles

	serial	parralel
collision checker calls	$O(n \log n \log 1/\epsilon)$	$O(\log n \log 1/\epsilon)$
number of threads	1	$O(n)$

We note that the parallel algorithm is work-efficient—it does not do any more

work than the serial algorithm.

It is important to compare the complexity of the given algorithm compared to that of a collision check with n objects. Our algorithm is able to bound the probability of collision with only a $O(\log n \log 1/\epsilon)$ slowdown compared to a collision check with obstacles of a known location. This suggests that robots can be made behave safely and reason about uncertainty without greatly increasing the required computation.

3.4 Verification of Safety Certificates

We note that the low computational complexity of algorithm 2 implies that it can be run on an embedded system with limited computational resources. This allows an low power system to verify that actions suggested by another system, perhaps with significantly more computational resources, is safe. It also allows us to verify that the trajectories found by an algorithm dependent on heuristics or machine learning is truly safe.

For systems that are even more computationally constrained we note that verification is possible with a single collision check. If we stores the probabilities for the individual shadows in algorithm 2, we can verify that a trajectory is safe with very little communication and computational overhead. This would only require a single collision check by the low-power system. In terms of communication, it requires only the transmission of these probabilities, the states visited by the trajectory, and the parameters of the obstacle distribution.

Chapter 4

Algorithms for Finding Safe Plans

In chapter 3 we demonstrated that the computational complexity of bounding the probability of collision for a trajectory is asymptotically not not much more expensive than a traditional deterministic collision check. It does not seem that a similar result is possible for either minimum risk or risk constrained planning with the shadows used in 3. In this chapter we present a NP-hardness result for the risk constrained planning. In a sense our hard our NP-hardness result is unsatisfying as it requires that the obstacles live in a space of dimension $O(\log n)$. In practice robots operate in a space of dimension 3 which means this proof does not preclude the existence of an efficient algorithm for safe planning. It does suggest that if an efficient algorithm were to exist, it would have to use the properties of a space of fixed dimension. For completeness, a naïve, exponential time algorithm is described in this chapter as well.

4.1 Motivation

While finding trajectories that can avoid obstacles with high probability has become increasingly important, the safe motion planning problem lacks certain structures that make the deterministic motion planning problem comparably easy to solve. This ends up making it difficult to efficiently provide the same theoretical guarantees to safe motion planning that are available in the deterministic setting.

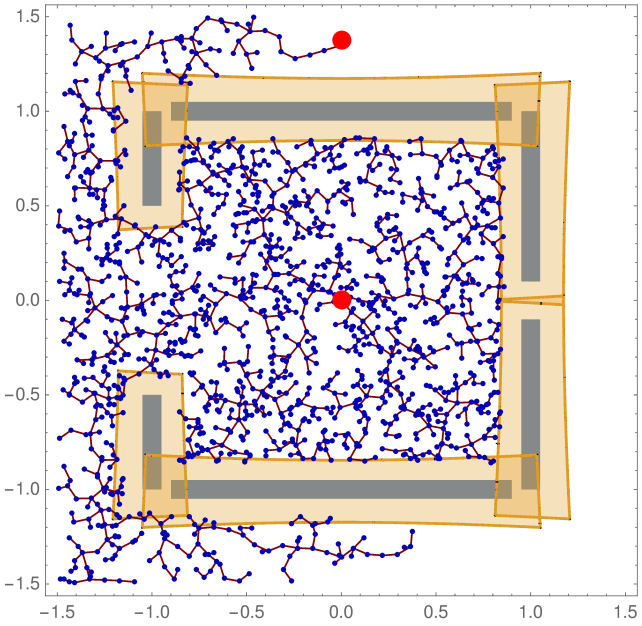
One way of evaluating a deterministic motion planner is asking whether it is prob-

abilistically complete—does the probability of finding a plan, if one exists, approach 1 as the amount of allowed computation approaches infinity? While sampling based motion planning algorithms are often probabilistically complete under mild conditions, they cannot be applied directly to the safe motion planning case without paying a high computational cost.

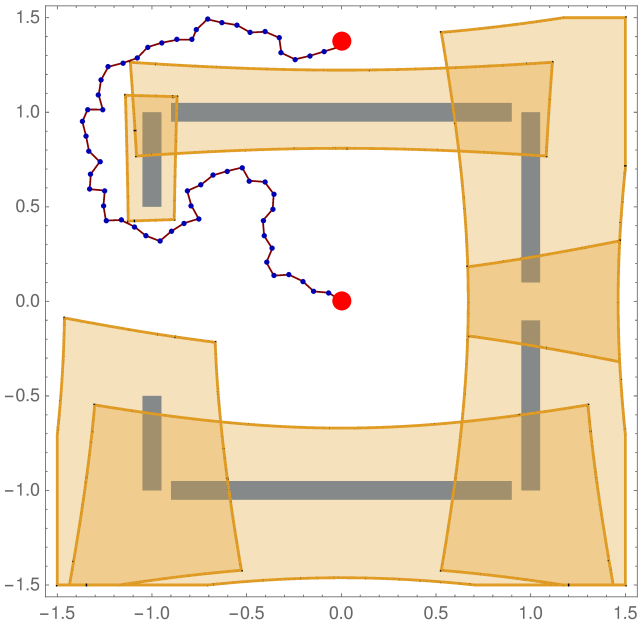
Safe motion planning does not satisfy the Bellman principle of optimality. In other words it lacks the important Markov-like Dynamic Programming structure that allows sampling-based methods like the Rapidly-exploring Random Tree (RRT) family of motion planning algorithms to be probabilistically complete. When evaluating completeness for deterministic motion planning the trajectory used to reach a particular state, only that it is reachable. In the safe motion planning problem the trajectory taken to reach a particular configuration is critical—it is exactly what determines the probability of collision. Furthermore, since the probability of collision at different points along a trajectory is not independent, the reachability of future configurations under a risk constraint is dependent on the previous trajectory. Since the RRT family of planners generally maintain only one path to every configuration (thus the term random *tree*), they may not find the safe path if multiple feasible paths exist. This suggests that simply adding a risk constraint to the standard RRT algorithm is not probabilistically complete. An execution of an RRT to only generate safe plans is shown in figure 4-1 to illustrate the issue.

This suggests that a different approach is necessary for approaching the safe motion planning problem. In approaching the safe motion planning problem, we propose a naïve algorithm based on a Rapidly-exploring Random Graph (RRG) developed for optimal deterministic motion planning [14]. This naïve algorithm will require exponentially more computation than the original RRG algorithm. This algorithm will be probabilistically complete under the standard assumptions for deterministic motion planning. While the algorithm requires a lot of computation to find a safe plan once that plan is present in the graph found by the RRG, it suggests that the method of approximating paths in the robot's configuration space can remain unchanged.

We will then show that finding a modification to the above algorithm that makes



(a) An execution of an RRT modified to only find safe plans. Uniformly sized shadows are shown for reference.



(b) The trajectory found by the RRT between the start and destination configuration with optimal shadows found by algorithm 2. Note that the RRT tends to choose suboptimal trajectories (this can be shown to happen with probability 1). While this suboptimality does not affect it's ability to find trajectories in a deterministic setting, it does affect its ability to find trajectories in a risk constrained setting since it cannot reduce total risk "taken" to get to a particular configuration.

Figure 4-1: Safe RRT

it polynomial time is difficult. We provide several proof outlines for variants of the hardness result. First we construct a reduction from 3SAT to the safe planning problem where the obstacles are in a space of dimension $O(n)$. We then describe how the result can be strengthened in several ways. We discuss how an Application of the Johnson-Lindenstrauss lemma can allow us to use a space of dimension $O(\log n)$. We also show how in the reduced space we can strengthen our result to prove hardness of multiplicative approximation to a $1 \pm \Theta\left(\frac{1}{\epsilon}\right)$ factor.

4.2 A Probabilistically Complete Planner for Safe Motion Planning

In this section we propose a probabilistically complete safe planner. In other words, if there exists a safe trajectory τ and a $\delta > 0$ such that the δ-inflation of the trajectory $\{(x,t) \mid d(x,\tau(t)) \leq \delta\}$ is safe, the probability that the planner finds a safe trajectory approaches 1 as the number of iterations the algorithm is allowed to run goes to infinity. We note that this notion of δ-inflation is equivalent to saying that if the robot was slightly inflated and took the same trajectory, the resulting trajectory would still be safe. This is a slightly stronger condition than allowing small deviations from the original trajectory at any point.

Our algorithm relies on a probability of collision oracle, a function

$$\text{ProbCollision} : X \to [0,1]$$

that computes the probability of collision. We note that the algorithms presented in chapter 3 can be used in this algorithm, as can a monte-carlo method with $O\left(\frac{1}{\epsilon}\right)$ complexity. We note that the naïve algorithm will be exact if the ProbCollision oracle is exact, and approximate if it is approximate. We also note that if a randomized algorithm not guaranteed to be correct with probability 1 is used to implement this oracle, slight modifications may be necessary to use it in the naïve algorithm and ensure that the result is correct with high probability.

The δ-inflation of the trajectory guarantees that with probability 1, an RRG will contain a safe trajectory in a limit. At each iteration of the RRG we simply brute force test all of the paths in the RRG to see if any are safe. This algorithm is presented in Algorithm 3 and is only a slight modification from the original RRG algorithm. Any valid steering function from the motion planning literature may be used inside algorithm 3.

Algorithm 3 A brute force safe planning algorithm based on an RRG

Precondition: exists a safe δ-inflation of a trajectory τ that is safe for some $\delta > 0$.

```
1: function NaiveSafeRRG(x_goal, ε, n)
2:     G.V ← {x_init, x_goal}
3:     G.E ← ∅
4:     for i ← 1 to n do
5:         x_rand ← Sample(i)
6:         G ← SafeExtend_RRG(G, x_rand, ε)
7:         for all path ∈ G, path.start = x_init, path.end = x_goal do
8:             if ProbCollision(path) ≤ ε then
9:                 return path
10:            end if
11:        end for
12:    end for
13:    return ∅
14: end function
```

Theorem 7. *If a safe δ-inflated trajectory exists then, as the number of iterations, $n \to \infty$, the probability that algorithm 3 finds a safe trajectory is* 1.

Proof. The existence of a safe $\delta-$inflated trajectory guarantees that the RRG will eventually contain a safe path. Once the RRG contains a path, it will be enumerated and the algorithm will identify it as safe using the ProbCollision oracle and return it. □

We note however, that Algorithm 3 involves a step which enumerates all paths, and is at worst exponential in the size of the graph. This means that Algorithm 3 can be exponentially slower than a RRG for a similar deterministic motion planning problem.

4.3 Hardness

We characterize the hardness of the approximate safe planning problem with PGDF planning problem via a reduction to 3-SAT. The reduction is relatively robust–it does not matter if we use the shadows used in chapter 3 or exact shadows. This section contains only a proof outline.

The outline of the proof is as follows:

1. Construct a gadget to capture information information about n variables using $O(n)$ obstacles and adding $O(n)$ nodes to the graph and $O(n)$ obstacles in a space of dimension $O(n)$.

2. Construct a gadget to capture information about m clauses adding $O(m)$ nodes to the graph.

3. Use a Johnson-Lindenstrauss embedding to embed, the graph and obstacles in a space of dimension $O(\log n)$

4. Use the gap in cost between good solutions and bad solutions to the problem is not only hard to compute exactly, but also hard to approximate to within a $\frac{1}{\epsilon}$ factor.

Theorem 8. *Determining the existence of a path of probability less than ϵ in a graph of size $O(n)$ and $O(n)$ PGDF obstacles is NP-hard if the obstacles live in a space of dimension $O(\log n)$.*

We use two lemmas in the proof. One is the classic Johnson-Lindenstrauss Lemma.

Lemma 2 (Johnson-Lindenstrauss Lemma)**.** *Given an $\epsilon \in (0, 1)$ and a set X of m points in $\mathbb{R}^N$ and $n = o(\frac{\log m}{\epsilon^2})$, there exists a matrix $A : \mathbb{R}^N\mathbb{R}^n$ such that for all points $x, y \in X$*

$$(1 - \epsilon)||x - y|| \leq ||Ax - Ay|| \leq (1 + \epsilon)||x - y||$$

We also use the fact that an embedding that approximately preserves distances also approximately preserves orthogonality.

Lemma 3. *Given two orthogonal vectors x, y, and a corresponding Johnson-Lindenstrauss embedding matrix A for parameter ϵ, and θ the angle between Ax, Ay then $|\cos\theta| = O(\epsilon)$*

We will construct a graph and a set of obstacles that will capture a specific 3-SAT problem. We have one axis that corresponds to the gadget number.

First we construct a gadget for capturing information about variables in the 3-SAT formula. For convenience we place the "start“ node of the graph at the origin.

For each variable we assign a dimension and a variable node centered along this new axis. We add one obstacle at the $+1$ coordinate on that dimension and one the -1 coordinate. We add a node close to each obstacle. We connect these two nodes to the previous and next variable nodes. This construction is illustrated in figure 4-2.

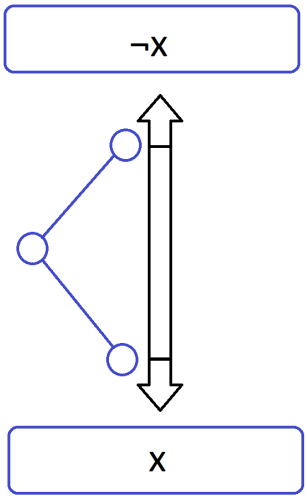


Figure 4-2: An example of a variable gadget

Now we add a gadget for every clause. We add a clause node at the next gadget index. For every variable that appears unnegated we set the corresponding index closer to $+1$. For each variable that appears negated we set the corresponding index to -1.

Since the risk is paid for each obstacle only once, choosing a path through the graph is equivalent to choosing an assignment of variables. If the risk is only paid for as many obstacles as there are variables the 3SAT is satisfiable. Otherwise one variable must be both true in some clauses and false in others.

Chapter 5

Safety in the Online Setting

So far in this thesis we have focused on guaranteeing safety in the following paradigm:

1. The episode begins
2. The robot gets information about its environment
3. The robot decides what action/trajectory to execute
4. The robot executes the action/trajectory
5. The episode ends

However, this paradigm rarely matches reality exactly. Instead of planning once, robots often replan as they either deviate from the original trajectory or receive additional sensor information.

This chapter examines how to guarantee safety in this more complicated setting.

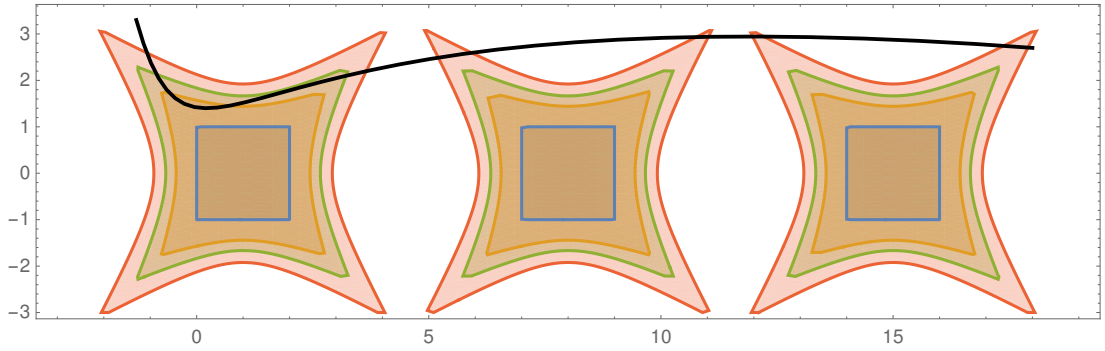
5.1 Motivation

The safety guarantees developed earlier in this thesis, and in many other works, only guarantees that during a single episode, the probability of collision between steps 1 and 5 is less an some fixed $\epsilon \in (0, 1)$. While these assumptions make it easier to develop mathematical formalisms guaranteeing safety, they do not necessarily match

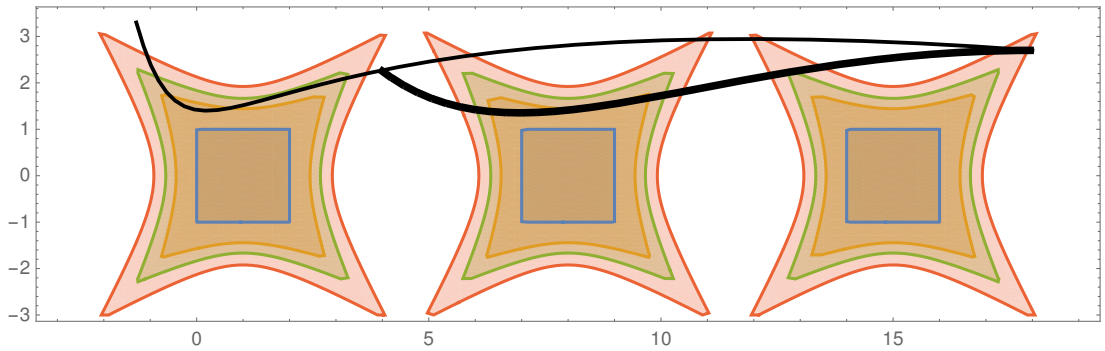
robot operation in practice. Ideally, we would guarantee that the lifetime probability of collision of the robot is less than ϵ while giving the robot the flexibility to change its desired trajectory upon receiving additional information. In fact we can show that a system that obeys a safety guarantee in the simple paradigm can be made to collide with probability arbitrarily close to 1 when used in a situation where it is allowed to re-plan. Simply always following paths that are currently ϵ-safe is not sufficient to guarantee that the lifetime probability of collision remains low. This example is illustrated in figure 5-1 and explained below.

First the robot finds a trajectory with collision probability ϵ and starts executing it. As it travels by the first obstacle, it gets more informative observations regarding the location of the obstacle. It now knows for sure it will not collide. Once it passes the first obstacle it searches for a new trajectory with probability of collision ϵ of collision and starts executing this. It repeats this process until it has reached its destination or collides. It is important to note that this example is not convoluted and is not far from how practical robot systems behave. It is quite common that robots obtain better estimates of obstacles as they get closer and then re-plan less conservative trajectories. It is also common for systems to merely require that any action they take be safe instead of explicitly proving that the lifetime sequence of actions they will take will be safe (especially when this sequence of actions is unknown when execution begins). In doing so, these systems repeatedly take small risks without keeping track of them, allowing the systems to collide more frequently than the safety guarantee indicates.

Computation of the real probability of collision (approaching 1) in the scenario presented in figure 5-1 follows quite directly. Say that during each step the probability that the robot will collide with the first obstacle is $\alpha\epsilon$ and with the remaining obstacles $1 - \alpha\epsilon$. Let C_i denote the event that the robot collides in the execution of step i.



(a) The robot finds a trajectory with probability ϵ of collision and begins execution.



(b) After having passed the first obstacle the robot uses all the new information it obtained to find a new trajectory with probability of collision ϵ. Since the robot now knows that the first segment of the trajectory was collision free, the fast and future portion of the trajectory have a combined collision probability of ϵ.

Figure 5-1: An illustration about how a robot can be always taking an action that is safe, but still be guaranteed to eventually collide.

$$
\begin{aligned}
P(collision) &= P(C_1) + P(C_1^C)P(C_2|C_1^C) + ... \\
&= \alpha\epsilon + (1-\alpha\epsilon)\alpha\epsilon + ... + (1-\alpha\epsilon)^n\alpha\epsilon \\
&\text{In the limit of number of episodes this becomes} \\
&= \frac{\alpha\epsilon}{1-(1-\alpha\epsilon)} \\
&= 1
\end{aligned}
$$

Since we clearly do not want our robots to collide with the environment with probability one we need to understand what constitutes safe behavior in this "online" setting where the robot is receiving a constant stream of information and is constantly making decisions.

In this chapter of the thesis we develop a notions of *policy safety* and *absolute safety*—a way of guaranteeing that the lifetime collision probability is less than a fixed ϵ, even when the robot has access to a constant stream of information and is allowed to react to the new information.

5.2 Absolute Safety

First we formalize how our robotic system makes decisions. Let O_t be the set of observations by the robot (eg. sensor data) up to, but not including time t. Let A_t be the action that the robot commits to at t. If there is a processing delay, this action might only be executed at a time $t' > t$.

Definition 3 (policy)**.** *A policy $\pi : O_t \to A_t$ is a function from observations to actions.*

This allows us to define a natural notion of safety for policies. Let U be the event that something unsafe (ex. a collision with an obstacle) occurs.

Definition 4 (Policy Safety)**.** *A policy π is ϵ-policy safe with respect to an event U*

if the probability of U under the policy is less than ϵ.

$$P(U|\pi) \leq \epsilon$$

In order to understand policy safety, it is important to be clear on how the policy is allowed to use randomness. Consider the policy that uses randomness internally illustrated in figure 5-2.

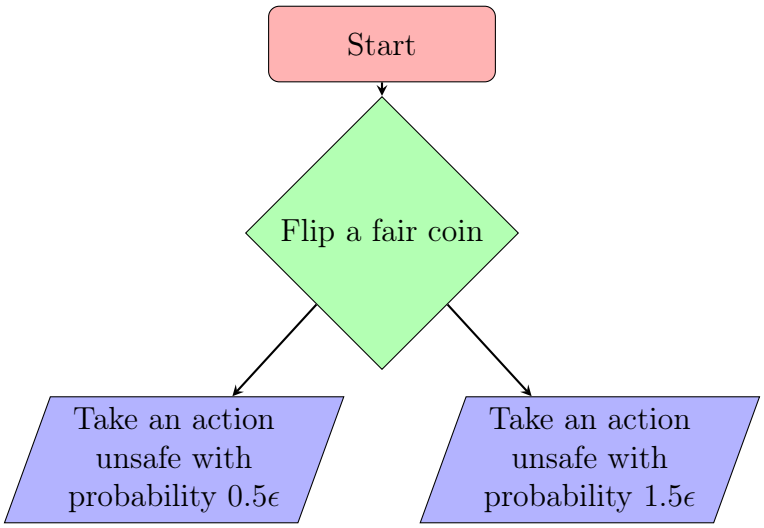


Figure 5-2: A flow chart for a policy that can cheat a natural definition of safety using internal randomness

While the overall probability of failure in the policy of figure 5-2 is still ϵ, after the policy flips the coin, it knowingly takes an action that is unsafe with probability 1.5ϵ. If we want to prevent such behavior, it is also not sufficient to merely ban the policy from using randomness. Not only is randomness useful for computation, but banning internal randomness ends up making little difference. The observations often contain sufficient randomness to enable to policy to emulate having an internal random number generator.

We define *absolute safety* as a notion of safety that forces the policy to always commit to a plan that is sufficiently safe given the current sensor observations.

Definition 5 (Absolute Safety)**.** *We say that a policy π is $\epsilon-$absolutely safe if at*

every point in time t the following holds true:

$$\int_0^\infty E[p_t|\pi(O_t)]dt = \tag{5.1}$$

$$\int_0^t p_t dt + \int_t^\infty E[p_t|\pi(O_t)]dt \leq \epsilon \tag{5.2}$$

where p_t is the probability of an unsafe event at time t, given the information up to the last point in time where the policy could change the action affecting the probability of collision.

We note that the second term in equation (5.2) is exactly the probability of collision given the current information.

5.2.1 Absolute Safety vs traditional

The definition of absolute safety is somewhat different than is traditionally used to guarantee safety. In order to better understand this condition, we develop some comparisons to these more traditional mathematical conditions used to guarantee safety.

First we note the second term in equation (5.2) is exactly the probability of collision given the current information. This allows an alternative interpretation of the condition of absolute safety. At every time the following must hold true: the sum of the future risk, and some accounting of past risk must always be less than ϵ. This a strictly stronger condition than requiring that future actions have risk less than ϵ—an often used condition that is broken by the counterexample presented in the beginning of the chapter.

Comparing absolute safety to policy safety is more complicated. In order to be able to better understand the difference we introduce the notion of an information adversary. An information adversary is a particular kind of adversary that tries to make the system unsafe by turning off the system's sensor stream.

Definition 6 (Information Adversary)**.** *An information adversary is an agent that has access to the internal state of the robot and can stop the sensor stream at any*

point. The information adversary attempts to maximize the probability of an unsafe event such as collision. In the notation used earlier in the chapter, an information adversary can at any time t ensure that $O_{t'} = O_t$ for all $t' \geq t$.

The notion of an information adversary is what allows us to compare absolute safety to policy safety. We use two theorems to compare the two notions of safety.

Theorem 9. *ϵ-absolute safety guarantees ϵ-policy safety.*

Theorem 10. *A system that is ϵ-policy safe at all times guarantees ϵ-absolute safety.*

More informally, absolute safety is a stronger condition than policy safety that requires the system's actions always be safe under the current information set.

The proof of the theorem 9 follows almost by definition.

Proof. Recall that we use U to denote the unsafe event. We simply consider time 0, the beginning of the episode. Absolute safety gives us the following conditions:

$$\int_0^\infty E[p_t|O_t]dt = P(U \mid \pi) \leq \epsilon \quad \square$$

$\square$

The proof of theorem 10 is by contradiction. If the theorem did not hold true, we could explicitly construct a set of actions by the information adversary that would make the policy unsafe. Absolute safety is stronger than policy safety in two ways:

1. No hedging bets. Absolute safety does not allow the robot to choose a risky path half the time and a safe path half the time, even if the expected risk is acceptable.

2. Relying on future information to make decisions. Absolute safety does not allow a system to rely on receiving future observations.

Proof. Assume for the sake of contradiction that there exists a set of observations O and a time t for which absolute safety does not hold. That is to say that conditioned

on these observations

$$\int_0^t p_t dt + \int_t^\infty E[p_t \mid \pi] dt > \epsilon$$

Let the information adversary stop the flow of information to the robot at time t. Let A_1 de note the event that the system fails during times $(0, t]$ and A_2 denote the event that the sy stems fails during times (t, ∞). We note that a system can fail at most once, so A_1, A_2 are exclusive.

Utilizing the fact that $\int_t^\infty E[p_t|O]dt = P(A_2|O)$ and $\int_0^t p_t \mid Odt = P(A_1 \mid O)$, we get that the probability of failure $P(A_1 \cup A_2) = P(A_1) + P(A_2) \geq \epsilon$ violating our assumption that the system is policy safe under the set of observations O. □

□

5.2.2 Absolute Safety using Shadows

In this section we examine how to augment the algorithm in chapter 3 to guarantee absolute safety. Recall that absolute safety is the following condition:

$$\int_0^t p_t dt + \int_t^\infty E[p_t \mid \pi] dt \leq \epsilon$$

Alternatively using the fact that the second term is simply the future probability of collision given the current information we get the following condition: (Note that U is the collision event)

$$\int_0^t p_t dt + P(U) \leq \epsilon$$

$P(U)$ is exactly the probability of collision computed in algorithm 2. The integral in the first part can be computed using information available at the current time.

This leads to a modification that can allow us to certify that a system is safe under replanning. This leads to the following modification to algorithm 2 that guarantees

ϵ-absolute safety:

1. Compute $\epsilon_1 = \int_0^t p_t dt$
2. Verify that any new proposed trajectory has probability of collision less than $\epsilon - \epsilon_1$

If the trajectory chosen always passes the above safety check, the policy is guaranteed to be ϵ-absolutely safe.

5.3 Conclusion

Guaranteeing safety in the online setting can be rather nuanced and identifying desired behavior can be difficult. For example, policy safety allows the system to take actions that are known to be risky. Absolute safety and the information adversary model on the other hand, can be too restrictive. We hope that future work will take care to define safe behavior in a manner appropriate for the application.

Chapter 6

Conclusion

The contributions of this thesis can be split into two halves. The first half consists of contributions of a mathematical nature. The thesis presents a way of quantifying the quality of estimated shapes through models of random geometry and shadows. It presents the PGDF model of random polytopes and describes how to compute shadows of PGDF polytopes. For arbitrary distributions over half-spaces we show how to find probability of intersection. We characterize the existence and uniqueness of ϵ-shadows for arbitrary distributions as well.

The second half presents algorithms that utilize the mathematics developed in the first half. We present an efficient algorithm that an be used to bound the probability of collision for a trajectory. We also discuss issues with generalizing our algorithm to guarantee safe operation in a setting with a continuous flow of sensor data. We present a general method that can be used to guarantee safety in such settings.

6.1 Future Work Related to Safe Navigation

While the algorithm we presented for bounding probability of collision is exponentially faster than the state of the art Monte-Carlo based methods, much work remains to be done. The bounds computed by the algorithm are not tight, and there is hope that they can be improved.

The algorithm presented in chapter 3 does not compute actual motion plans. We

hope that a practical algorithm for finding safe plans can be obtained. Finally, the PGDF has some surprising behavior that was highlighted in chapter 2. We hope that future work will develop better estimation methods and statistical models in order to avoid the degenerate behavior shown in chapter 2.

6.2 General Future work

The main contribution of the work is towards developing and applying a theory of random geometry. Many systems, robotic and otherwise, must interact with geometry in the environment. While we have many methods that estimate geometry based on noisy observations, we often lack a rigorous understanding of how these methods perform as statistical estimators [7, 19]. Understanding the mathematics behind random geometry will not only help us understand how our current methods perform, but also how to design better methods.

As demonstrated in this thesis, a better understanding of random geometry allows us to efficiently reason about uncertainty in our estimates of geometry. In this thesis we used this information to compute the probability that a particular robot trajectory would collide with a set of estimated obstacles. This information could be used for other applications as well. If we desired to improve our estimates of obstacles, this information could inform us as to which observations would yield the most valuable information. Perhaps this information can be used to devise grasping strategies that are robust to the uncertainty in the estimates of the geometry.

Finally this thesis demonstrated that reasoning about uncertainty in this framework can be done efficiently. Many problems in robotics that require reasoning about uncertainty are provably hard. Restricting the problem in question to a specific distribution of obstacles, and examining the underlying mathematics led to a very simple, efficient approximation algorithm. In fact, the proposed algorithm is barely slower than its equivalent for the deterministic setting. Hopefully understanding the mathematics behind random geometry will give us insight to what actually makes these problems computationally hard, and suggest ways to get around the fundamental

computational restrictions at little practical cost.

Appendix A

Geometry and Vector Space Primer

The methods of bounding random geometry presented in this thesis require understanding the relationship between geometry in the space that robots operate in (usually $\mathbb{R}^3$) and the space of half-space parameters of geometry over $\mathbb{R}^n$ (the dual space).

Duality in convex geometry studies precisely this when examining half-spaces (sets of the form $\alpha^T x \leq 0$). The dual of the vector space $\mathbb{R}^n$ is defined by the map that sends the half-space $\alpha^T x \leq 0$ to α. The space of parameters α of half-spaces over $\mathbb{R}^n$ is called the dual space.

Throughout this thesis we use homogeneous coordinates, embedding the point $< x_1, ...x_n > \in \mathbb{R}^n$ as $< x_1, ...x_n, 1 > \in \mathbb{R}^{n+1}$. We can undo this mapping by sending any point $< x_1....x_n, x_{n+1} >$ to $\frac{1}{x_{n+1}} < x_1, ..., x_n >$. This proves convenient as it allows us write an equation such as $\alpha^T x \leq b$ compactly as $< \alpha_1, ..., \alpha_n, -b >^T \; x \leq 0$ and describe affine transformations on the original coordinates as linear transformations (as matrices) on homogeneous coordinates.

While an in depth discussion we refer to Boyd et al. [3], we introduce the minimal set of dual geometry concepts necessary to understand the developments in this thesis. We discuss polar cones, dual cones, norm cones and dual norms.

A.1 Cones

Definition 7 (Convex Cone)**.** *We say that a set C is a convex cone if C is convex and closed under positive scaling. In other works C is a cone if for any $x, y \in C, \lambda_1 \geq 0, \lambda_2 \geq 0$, $\lambda_1 x + \lambda_2 y \in C$.*

Definition 8 (Dual Cone ($C^\star$))**.** *The dual cone, $C^\star$ of a set X is*

$$C^\star = \{\alpha \mid \alpha^T x \geq 0, \ \forall x \in X\}$$

In other words the dual cone of a subset of a vector space X is the set of linear functions that do not contain any point of X. If X is a set of linear functionals, it's dual cone can be interpreted as contained in all the half-spaces defined by $\alpha^T x \leq 0$ with x in the set X.

Taking the dual cone of the dual cone of X yields the minimal convex cone containing X. If X was already a convex cone then taking the dual twice yields exactly the original set.

Definition 9 (Polar Cone (C^0))**.** *The polar cone, C^0 of a set X is*

$$C^0 = \{\alpha \mid \alpha^T x \leq 0, \ \forall x \in X\}$$

The polar cone of a set X is the set of linear functions that contain every point in X.The relationship between a polar cone and the dual cone is described in theorem 11.

Definition 10 (Norm Cone)**.** *Let $||\cdot||$ be a norm (distance measure) on $\mathbb{R}^n$. The corresponding norm cone is*

$$C = \{(x_1, ...x_n, r) \mid ||x|| \leq r\}$$

We say that the dual space of $\mathbb{R}^n$ is the space of linear functionals α over $\mathbb{R}^n$, where $\alpha^T x \leq 0$

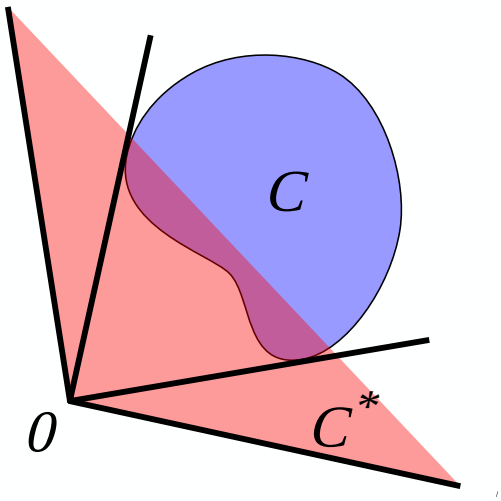


Figure A-1: The dual cone of the blue blob

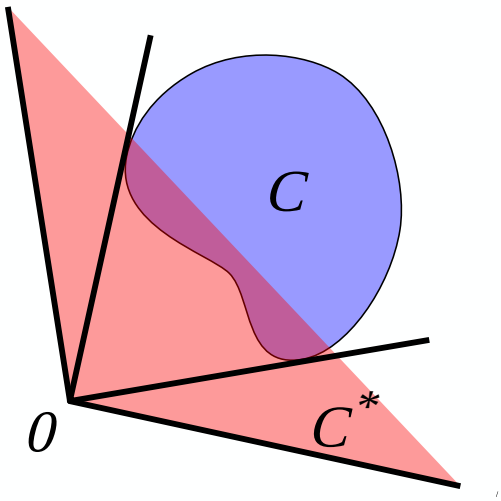


Figure A-2: The polar cone of the blue blob. Note it's relation with the dual cone.

A.2 Useful Theorems about dual and polar Cones

Theorem 11. *Let $C^\star$ be the dual cone of X, and C^0 the polar cone of X. Then $C^\star = -C^0$*

Theorem 12. *Let C be the norm cone corresponding to $||\cdot||$. Then the dual cone, $C^\star$ is the norm cone with norm $||\cdot||_\star$, the dual norm of $||\cdot||$*

A.3 Dual Norms

Definition 11 (Dual Norm ($||\cdot||_\star$))**.** *The dual norm of $||\cdot||$ is $||x||_\star = \sup_y\{y^T x \mid ||y|| \leq 1\}$*

Theorem 13. *Let $||x|| = \sqrt{\boldsymbol{x}^T\boldsymbol{\Sigma}\boldsymbol{x}}$ for $\boldsymbol{\Sigma} \succ 0$. Then $||y||_\star = \sqrt{y^T\boldsymbol{\Sigma}^{-1}\boldsymbol{y}}$*

Proof. We simply apply the KKT conditions to the definition of a dual norm.

$$||y||_\star = \sup(x^T y \mid ||x|| \leq 1)$$

Now we apply the KKT conditions

$$\begin{aligned} 0 &= \nabla(x^T y - \lambda(\sqrt{x^T\boldsymbol{\Sigma}x} - 1)) \\ 0 &= y - \frac{2\lambda\boldsymbol{\Sigma}x}{2\sqrt{x^T\boldsymbol{\Sigma}x}} \end{aligned}$$

Since our problem is of a linear function over a convex domain our solution will lie on the boundary of the domain. This implies that $||x|| = 1$ and allows us to solve for x.

$$\begin{aligned} x^\star &= \lambda\Sigma^{-1}y \\ &= \frac{\Sigma^{-1}y}{\sqrt{y^T\boldsymbol{\Sigma}^{-1}\boldsymbol{\Sigma}\boldsymbol{\Sigma}^{-1}y}} \\ &= \frac{\Sigma^{-1}y}{\sqrt{y^T\boldsymbol{\Sigma}^{-1}y}} \end{aligned}$$

Now we can evaluate the supremum

$$\begin{aligned}\sup(x^T y \mid ||x|| \leq 1) &= x^{\star T} y \\ &= \frac{y^T \mathbf{\Sigma}^{-1} y}{\sqrt{y^T \mathbf{\Sigma}^{-1} y}} \\ &= \sqrt{y^T \mathbf{\Sigma}^{-1} y}\end{aligned}$$

□

Bibliography

[1] Haoyu Bai, David Hsu, Wee Sun Lee, and Vien A Ngo. Monte carlo value iteration for continuous-state pomdps. In *Algorithmic foundations of robotics IX*, pages 175–191. Springer, 2010.

[2] Christopher M Bishop. Pattern recognition. *Machine Learning*, 128:1–58, 2006.

[3] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[4] Adam Bry and Nicholas Roy. Rapidly-exploring random belief trees for motion planning under uncertainty. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 723–730. IEEE, 2011.

[5] John Canny and John Reif. Lower bounds for shortest path and related problems. In *Proc. 28th Ann. IEEE Symp. Found. Comp. Sci*, volume 49, page 60, 1987.

[6] Gregory Dudek, Kathleen Romanik, and Sue Whitesides. Localizing a robot with minimum travel. *SIAM Journal on Computing*, 27(2):583–604, 1998.

[7] Olivier Faugeras, Quang-Tuan Luong, and Theo Papadopoulo. *The geometry of multiple images: the laws that govern the formation of multiple images of a scene and some of their applications*. MIT press, 2004.

[8] Mzuri Handlin. Conic sections beyond $\mathbb{R}^2$. https://www.whitman.edu/Documents/Academics/Mathematics/Handlin.pdf. Accessed: 2017-2-16.

[9] Kaijen Hsiao, Leslie Pack Kaelbling, and Tomas Lozano-Perez. Grasping pomdps. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 4685–4692. IEEE, 2007.

[10] Lucas Janson, Edward Schmerling, and Marco Pavone. Monte carlo motion planning for robot trajectory optimization under uncertainty. In *International Symposium on Robotics Research*, Sestri Levante, Italy, September 2015. To Appear.

[11] Lucas Janson, Edward Schmerling, and Marco Pavone. Monte carlo motion planning for robot trajectory optimization under uncertainty. *arXiv preprint arXiv:1504.08053*, 2015.

[12] Leslie Pack Kaelbling and Tomás Lozano-Pérez. Integrated task and motion planning in belief space. *The International Journal of Robotics Research*, 32 (9-10):1194–1227, 2013.

[13] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101 (1):99–134, 1998.

[14] Sertac Karaman and Emilio Frazzoli. Incremental sampling-based algorithms for optimal motion planning. *Robotics Science and Systems VI*, 104, 2010.

[15] Hanna Kurniawati, David Hsu, and Wee Sun Lee. Sarsop: Efficient point-based pomdp planning by approximating optimally reachable belief spaces. In *Robotics: Science and systems*, volume 2008. Zurich, Switzerland., 2008.

[16] Alex Lee, Yan Duan, Sachin Patil, John Schulman, Zoe McCarthy, Jur van den Berg, Ken Goldberg, and Pieter Abbeel. Sigma hulls for gaussian belief space planning for imprecise articulated robots amid obstacles. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 5660–5667. IEEE, 2013.

[17] Omid Madani, Steve Hanks, and Anne Condon. On the undecidability of probabilistic planning and infinite-horizon partially observable markov decision problems. In *AAAI/IAAI*, pages 541–548, 1999.

[18] Anirudha Majumdar and Russ Tedrake. Funnel libraries for real-time robust feedback motion planning. *arXiv preprint arXiv:1601.04037*, 2016.

[19] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*, pages 127–136. IEEE, 2011.

[20] Christos H Papadimitriou and John N Tsitsiklis. The complexity of markov decision processes. *Mathematics of operations research*, 12(3):441–450, 1987.

[21] Joelle Pineau, Geoff Gordon, Sebastian Thrun, et al. Point-based value iteration: An anytime algorithm for pomdps. In *IJCAI*, volume 3, pages 1025–1032, 2003.

[22] Carl Edward Rasmussen. Gaussian processes for machine learning. 2006.

[23] John H Reif. Complexity of the mover's problem and generalizations. In *Foundations of Computer Science, 1979., 20th Annual Symposium on*, pages 421–427. IEEE, 1979.

[24] Dorsa Sadigh and Ashish Kapoor. Safe control under uncertainty with probabilistic signal temporal logic. In *Proceedings of Robotics: Science and Systems*, Ann Arbor, Michigan, June 2016. doi: 10.15607/RSS.2016.XII.017. URL `http://www.roboticsproceedings.org/rss12/p17.html`.

[25] Edward Schmerling and Marco Pavone. Evaluating trajectory collision probability through adaptive importance sampling for safe motion planning. In *Proceedings of Robotics: Science and Systems*, Cambridge, Massachusetts, July 2017. doi: 10.15607/RSS.2017.XIII.068.

[26] David Silver and Joel Veness. Monte-carlo planning in large pomdps. In *Advances in neural information processing systems*, pages 2164–2172, 2010.

[27] Trey Smith. *Probabilistic planning for robotic exploration.* Carnegie Mellon University, 2007.

[28] Kiril Solovey and Dan Halperin. On the hardness of unlabeled multi-robot motion planning. *The International Journal of Robotics Research*, 35(14):1750–1759, 2016.

[29] Jacob Steinhardt and Russ Tedrake. Finite-time regional verification of stochastic non-linear systems. *The International Journal of Robotics Research*, 31(7):901–923, 2012.

[30] Russ Tedrake. Lqr-trees: Feedback motion planning on sparse randomized trees. 2009.

[31] Georgios Theocharous and Leslie P Kaelbling. Approximate planning in pomdps with macro-actions. In *Advances in Neural Information Processing Systems*, pages 775–782, 2004.

[32] Mark M Tobenkin, Ian R Manchester, and Russ Tedrake. Invariant funnels around trajectories using sum-of-squares programming. *IFAC Proceedings Volumes*, 44(1):9218–9223, 2011.

[33] Craig Tovey and Sven Koenig. Gridworlds as testbeds for planning with incomplete information. In *AAAI/IAAI*, pages 819–824, 2000.