Provably Safe Robot Navigation with Obstacle Uncertainty

Brian Axelrod, Leslie Pack Kaelbling and Tomás Lozano-Pérez

***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 examine how to guarantee that a trajectory or policy is safe with only imperfect observations of the environment. We 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. 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 then use these safety checking methods to design a safe variant of the RRT planning algorithm.**

I. Introduction

A. Motivation

Safe and reliable operation of a robot in a cluttered environment can be difficult to achieve due to noisy and partial observations of the state of both the world and the robot. As autonomous systems leave the factory floor and become more pervasive in the form of drones and self-driving cars, it is becoming increasingly important to understand how to design systems that will not fail under these real-world conditions. While it is important that these systems be safe, it is also important they do not operate so conservatively as to be ineffective. They must have a strong understanding of when they take risks so they can avoid them, but still operate efficiently.

While most previous work focuses on robot state uncertainty, this paper focuses on safe navigation when the locations and geometries of these obstacles are uncertain. We focus on algorithms that find safety "certificates"—easily verifiable proofs that the trajectory or policy is safe. We examine two implications of the algorithms. First, the computational complexity of reasoning about uncertainty can be quite low. Second, the mathematics surrounding robot safety can have surprising behavior. We demonstrate how these tools can be used to design a motion planner guaranteed to give only safe plans, and inform the design of more general systems that make decisions under uncertainty.

B. Problem Formulation

We consider two settings. In the *off-line* setting we have a fixed set of information about the environment and are searching for an open-loop trajectory. In the *on-line* setting the robot has access to a stream of observations and can change its trajectory as a function of new information; the problem is to find a policy, a function from observations to actions, that allow the robot to adapt to changing circumstances. We show that different notions of safety are required for the two cases to ensure that the robot can guarantee a low probability of collision throughout its entire execution.

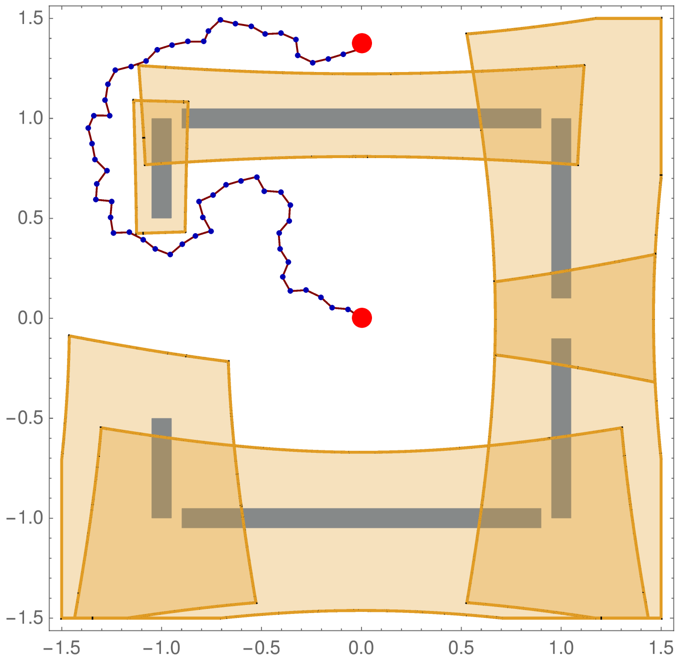


Fig. 1: The desired trajectory found by the planner shown with its specialized shadows that certify the probability of collision as less than 0.26%.

Safety in the offline setting amounts to staying out of regions likely to be contained within obstacles, and can be analyzed by computing geometric bounds on obstacles for which we have only partial information. Safety in the online setting builds on offline safety by requiring that the robot respect a contract with respect to the aggregate lifetime risk of operation while always having a guaranteed safe trajectory available to it.

We develop a general framework for analyzing safety and provide an example of applying this framework to a specific model of random geometry. We wish to emphasize that this framework can be applied to a wide variety of models beyond the example shown here.

Our framework operates in generality in $\mathbb{R}^n$ and assumes that obstacles are polytopes in $\mathbb{R}^n$. In this paper we focus on examples of typical robotic domains in $\mathbb{R}^2$ and $\mathbb{R}^3$. We ensure safety by verifying that the swept volume of a robot's trajectory is unlikely to collide with any obstacles. These swept volumes can be computed geometrically, or, for dynamical systems, via SOS programs [11].

We say that a trajectory, a map from time to robot configurations $\mathcal{Q}$, $\tau : [0, \infty) \rightarrow \mathcal{Q}$ is $\epsilon-$safe if the swept volume of the robot along trajectory τ intersects an obstacle with probability less than ϵ. Formally, if A is the event that the swept volume of τ intersects an obstacle, then τ is $\epsilon-$*safe* in the offline sense if $P(A \mid \tau) \leq \epsilon$.

We say that a policy, a map from observation history O, state history H and time to a trajectory τ, $\pi : O \times H \times [0, \infty) \rightarrow \tau$ is $\epsilon-$safe if, under all sequences of observations, $P(A \mid \pi) \leq \epsilon$. This notion of safety will be referred to as *policy safety*; it is a departure from previous models of robot safety, capturing the notion of a contract that the total risk over the lifetime of the system always be less than ϵ.

The requirement that the safety condition hold under all observations sequences is more conservative than the natural definition of $P(A \mid \pi)$. It crucial to prevent undesirable behavior that can "cheat" the definition of safety; this is discussed in detail in section IV.

C. Related Work

Planning under uncertainty has been studied extensively. Some approaches operate in generality and compute complete policies [8] while operate online, computing a plausible open-loop plan and updating it as more information is acquired [14].

Generating plans that provide formal non-collision (safety) guarantees when the environment is uncertain has proven difficult. Many methods use heuristic strategies to try to ensure that the plans they generate are unlikely to collide. One way of ensuring that a trajectory is safe is simply staying sufficiently far away from obstacles. If the robot's pose is uncertain this can be achieved by planning with a robot whose shape is grown by an uncertainty bound [3]. Alternatively, if the obstacle geometry is uncertain, the area around estimated obstacles can be expanded into a shadow whose size depends on the magnitude of the uncertainty [7, 10].

Another approach focuses on evaluating the probability that a trajectory will collide. Monte-Carlo methods can evaluate the probability of collision by sampling, but can be computationally expensive when the likelihood of failure is very small [6]. When the uncertainty is restricted to Gaussian uncertainty on the robot's pose, probabilistic collision checking can yield notable performance improvements [17][13][12].

Another perspective is finding a plan that is safe by construction. If the system is modeled as a Markov Decision Process, formal verification methods can be used to construct a plan that is guaranteed to be safe [4][5]. Recent work on methods that are based on signal temporal logic (STL) model have also uncertainty in obstacle geometry. With PrSTL Sadigh and Kapoor [16] explicitly model uncertainty in the environment to help generate safe plans but offer weaker guarantees than our work.

D. Contributions

This paper makes three contributions. the first is a formal definition of online safety that provides risk bounds on the entire execution of a policy.

The second contribution is an algorithm for efficiently verifying offline safety with respect to polytopes with Gaussian distributed faces (PGDFs) that is then generalized to the online case. In comparison to previous methods, the quality of the resulting bound is not dependent on the number of obstacles in the environment. The presented algorithms produce a certificate, which allows another system to efficiently verify that the actions about to be taken are safe. For a maximal collision probability of ϵ, the runtime of the algorithm grows as $\log \frac{1}{\epsilon}$ making it efficient even for very small ϵ's.

The third contribution is a modification to the RRT algorithm that generates safe plans. For any fixed ϵ, the resulting planner is guaranteed to only return trajectories for which the probability of failure is less than ϵ. We note that for n obstacles, the runtime of the RRT is increased only by a $\log n \log \frac{1}{\epsilon}$ factor, which suggests that reasoning about uncertainty can come at a low computational cost. A result of running this algorithm is shown in figure 1.

II. Model for Random Geometry

Previous work on planning under uncertainty has often relied on the notion of a shadow [7, 10], which is a volume that represents an uncertain estimate of the pose of an object and the space that it may occupy. A proper shadow is likely to contain the true object; and even if the exact location of the object is not known, it is sufficient to avoid the obstacle's shadow in order to avoid the true obstacle. A shadow is essentially the geometric equivalent of a confidence interval.

In order to provide strong guarantees about the safety of robot trajectories we first formalize this notion of a shadow.

Definition 1. *A set X is said to be an $\epsilon-$*shadow *of a random obstacle O if $P(O \subset X) \geq 1 - \epsilon$.*

To be able to generate shadows with desired properties, we need to place some restrictions on the class of distributions from which obstacles are drawn.

One way to arrive at a distribution on the shape and position of obstacles is to imagine that sensor data is obtained in the form of point clouds in which the points are segmented according to the face of the obstacle to which they belong. Then, the points belonging to a particular obstacle face can be used to perform a Bayesian linear regression on the parameters of the plane containing the face; given a Gaussian prior on the face-plane parameters and under reasonable assumptions about the noise-generation process, the posterior on the face-plane parameters will also be Gaussian [2, 15].

Recalling that a polytope X is the intersection of halfspaces, we can define a distribution over the parameters of polytopes

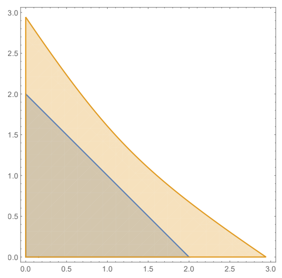


Fig. 2: An example of the region defined by theorem 1. The darker region is the "mean" obstacle and the orange region contains 95% of the obstacles generated by these parameters.

with a fixed number of faces. For $\boldsymbol{x}$ represented in homogeneous coordinates, a polytope X can be represented as

$$X = \bigcap \boldsymbol{n}_i^T \boldsymbol{x} \leq 0 \ .$$

When these normal vectors $\boldsymbol{n}_i$ are drawn from a Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$, we will call this a polytope with Gaussian distributed Faces (PGDF) with parameters $\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i$. We do not assume that the normal vectors are drawn independently and show that an independence assumption would yield little additional tightness to our bounds.

Using the PGDF model, we will be able to identify shadow regions that are guaranteed to contain the obstacle with a probability greater than $1-\epsilon$ in most cases of interest. There are degenerate combinations of values of ϵ, μ, and Σ for which there is no well-defined shadow (consider the case in which the means of all the face-planes go through the origin, for example). In addition, there are some cases in which the bounds we use for constructing regions are not tight and so although a shadow region exists, we cannot construct it. Details of these cases are discussed in the proofs in the appendix of the extended paper[1].

If we consider a single face, theorem 1 identifies a shadow region likely to contain the corresponding half-space.

Theorem 1. *Consider $\epsilon \in (0,1)$, $\boldsymbol{n} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ such that the combination of $\epsilon, \boldsymbol{\mu}, \boldsymbol{\Sigma}$ is non-degenerate. There exists a shadow S s.t. $\{\boldsymbol{x} \mid \boldsymbol{n}^T\boldsymbol{x} \geq 0\}$ is contained within S with probability at least $1-\epsilon$.*

While a detailed constructive proof is deferred to the supplemental materials, we present a sketch here. First, we identify a sufficiently high probability-mass region of half-spaces $\boldsymbol{n}$, which, by construction corresponds to a linear cone C in the space of half-space parameters. We then take the set of $\boldsymbol{x}$'s in homogeneous coordinates that these half-spaces contain. The set of points not contained by any half-space is the polar cone of C. Converting back to non-homogeneous coordinates yields conic sections as seen in figure 2.

In lemma 1 we generalize this notion to polytopes. For an obstacle with m faces, we can take the intersection of the resulting $\frac{\epsilon}{m}$-shadows. A union bound guarantees that the probability of any face not being contained in its corresponding region is less than ϵ. Thus the probability that the polytope

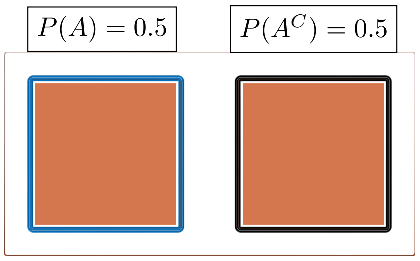


Fig. 3: Both the blue and black squares are valid 0.5-shadows, while the union of the two yellow areas is the set of points with probability at least 0.5 of being in the square.

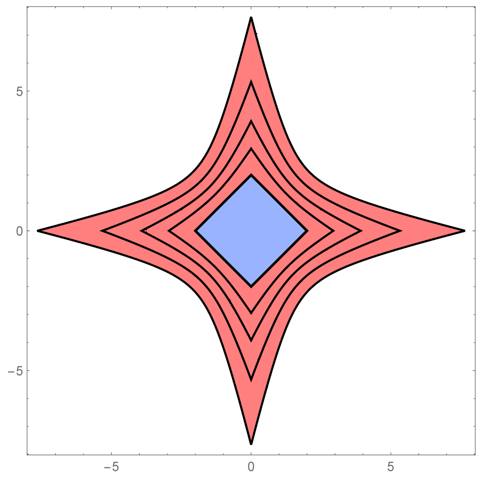


Fig. 4: The blue square represents the "mean" estimated obstacle. Each outline in the red set represents a different probability shadow of the obstacle.

is not contained in this region is less than ϵ. In other words, lemma 1 constructs an ϵ shadow.

Lemma 1. *Consider a polytope defined by $\bigcap_i \boldsymbol{n}^T\boldsymbol{x} \leq 0$. Let X_i be a set that contains the halfspace defined by $\boldsymbol{n}_i$ with probability at least $1-\epsilon_i$ (for example as in theorem 1). Then $\bigcap_i X_i$ contains the polytope with probability at least $1 - \sum_i \epsilon_i$.*

A. Computing Obstacle Shadows

We will use theorem 1 and lemma 1 to construct regions which we can prove are shadows. Before we continue, however, we note that obstacle shadows need not be unique, or correspond to the set of points with probability greater than ϵ of being inside the obstacle. Consider the case where a fair coin flip determines the location of a square as shown in figure 3. While there have been previous attempts to derive unsafe regions for normally-distributed faces [16], this lemma is stronger in that it constructs a set that is likely contain the obstacle as opposed to identifying points which are individually likely to be inside the obstacle.

Recall that a PGDF obstacle is a random polytope with m sides, that is, the intersection of m halfspaces. We can provide a shadow for each halfspace using theorem 1, and use lemma

1 to combine their intersection into a shadow for the estimated obstacle. The result is a $\epsilon-$shadow for a PGDF obstacle. An sequence of such increasingly tight $\epsilon-$shadows is shown in figure 4.

Lemma 2. *If an obstacle is PGDF with nondegenerate parameters and m sides, we can construct an ϵ-shadow as the intersection of the $\frac{\epsilon}{m}$ shadows of each of its sides.*

B. Shadows as Safety Certificates

Above we showed that for a given ϵ, we can easily compute a shadow for an obstacle under our model. Since a shadow is likely to contain the real object, the non-intersection of an $\epsilon-$shadow with the swept volume of a trajectory guarantees that the probability of colliding with the given obstacle is less than ϵ. Theorem 2 generalizes this notion to multiple obstacles using a union bound.

Theorem 2. *Let X be the volume of space that the robot may visit during its lifetime. If for a given set of obstacles, indexed by i, and their corresponding ϵ_i shadows, X does not intersect any shadow, then the probability of collision with any obstacle over the lifetime of the system is at most $\sum \epsilon_i$.*

Proof: Please see the supplementary material in the appendix of the extended paper[1].

It is important that theorem 2 does not depend on anything but the intersection of the swept volume with the obstacle shadows; it is independent of the trajectory's length and of how close to the shadow it comes. The guarantees of the theorem hold regardless of what the robot chooses to do in the space not occupied by a shadow.

Computing a shadow requires solving a relatively simple system of equations; then given a set of shadows and their corresponding ϵ's, a collision check between the visited states and the shadows is sufficient to verify that the proposed trajectory is safe. This implies that safety is easy to verify computationally, potentially enabling redundant safety checks without much computational power. Alternatively, a secondary system can verify that the robot is fulfilling its safety contract.

One potential concern is the tightness of theorem (2). If the union bound is loose, it may force us to fail to certify trajectories that are safe in practice. The union bound used in the proof of theorem (2) assumes the worst case correlation. If we assume that the system is to certify that collisions are rare events, and the events are independent, the union bound here ends up being almost tight.

Lemma 3. *Given n obstacles and their shadows, if*

- *the events that each obstacle is not contained in its shadow are independent,*
- *the probability that obstacles are not contained in their shadows is less than ϵ, and*
- $\epsilon = O(\frac{\sqrt{\delta}}{n})$

then the difference between the true probability of a shadow not containing the object, and the union bound in theorem 2 is less than δ.

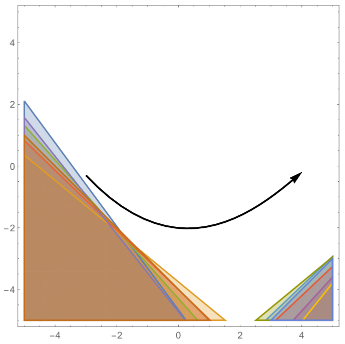


Fig. 5: This figure overlays several possible instances of the real obstacles given a single observation. The trajectory is much more likely to collide with a draw of the bottom left obstacle than the bottom right one. The two obstacles do not affect the safety of the trajectory equally.

Proof: Please see the supplementary material in the extended paper[1].

Another way of interpreting lemma 3 is that if the probability of failure is small, the union bound is close to tight.

Theorem 2 highlights a problem with systems that do not search for a set of optimal shadows. If we allocate equal ϵ's for every obstacle, then we are forced to pay as much for an object far away and irrelevant to the trajectory as we are for obstacles close to the trajectory. Figure 5 suggests that it is optimal to have large shadows with very small ϵ's for such irrelevant obstacles, and larger ϵ's for obstacles that may present significant risk. Doing so requires finding a good ϵ-shadow pair for every obstacle. We present an algorithm that finds this optimal certificate in the next section.

III. Algorithm For Finding Optimal Shadows

We will verify that trajectories are safe by finding a set of shadows that proves the swept volume of the trajectory is unlikely to collide with an obstacle. In order to minimize the number of scenarios in which a trajectory is actually safe, but our system fails to certify it as such, we will search for the optimal set of shadows for the given trajectory, allowing shadows for distant obstacles to be larger than those for obstacles near the trajectory. In order to understand the search for shadows of multiple obstacles we first examine the case of a single obstacle.

A. Single Obstacle

For a single obstacle with index i, we want to find the smallest ϵ_i risk bound, or equivalently, largest shadow that contains the estimate but not the volume of space that the robot may visit. That is, we wish to solve the following optimization problem:

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

If we restrict ourselves to the shadows obtained by lemma 2, a shadow with a larger ϵ is strictly contained in a shadow with

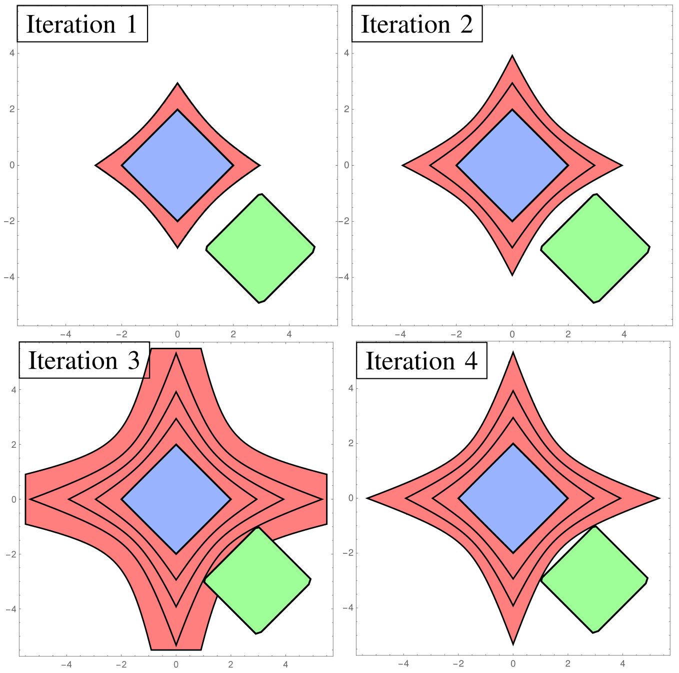


Fig. 6: A line search for the maximal shadow. The shadow "grows" and "shrinks" until it contacts the green space visited by the robot.

a smaller ϵ. This implies that the intersection is monotone in ϵ, allowing us to solve the above problem with a line search as shown in figure 6. While we restrict our attention to the general case, in certain cases, such as where X is a collection of points, this optimization can be solved analytically.

Essentially we are growing the size of the shadow until it almost touches the states that the robot can visit, X.

We define FIND_MAXIMAL_SHADOW$(\epsilon_p, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i, V)$, which takes the precision ϵ_p, PGDF parameters $\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i$, and swept volume V, and uses a standard bisection search to find and return the largest ϵ for which the shadows are non-intersecting with V. This requires $O(\log 1/\epsilon_p)$ calls of intersection—proportional to the number of digits of precision required. The runtime grows very slowly as the acceptable probability of collision goes to zero.

B. Multiple Obstacles

In order to extend the algorithm to multiple obstacles we imitate the union bound in theorem 2. We run the line search to determine the largest allowable ϵ for every obstacle, and sum the resulting ϵ's to get the ultimate bound on the risk. The psuedocode is presented in algorithm 1.

This algorithm is embarrassingly parallel because every ϵ_i can be computed independently without increasing the total amount of required computation. To obtain a total accumulated numerical error less than δ we only need to set $\epsilon_p = \delta/n$. If ω is the complexity of a single call of intersection, our algorithm runs in $O(\omega n \log n \log 1/\delta)$ time. However, since the search for shadows can be done in parallel in a work-efficient manner, the algorithm can run in $O(\omega \log n \log 1/\delta)$ time on $\Theta(n)$ processors.

Algorithm 1 FIND_MAXIMAL_SHADOW_SET

Input: $\epsilon_p, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i, V$
Output: ϵ, s.t. the path generating volume V is at least ϵ safe and each shadow is less than ϵ_p away from the minimal ϵ for which this class of bound may be obtained.

```
1: for i = 1...n do
2:     $\epsilon_i$ = FIND_MAXIMAL_SHADOW($\epsilon_p, \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i, V$)
3: end for
4: return $\sum \epsilon_i$
```

If the intersection check is implemented with a collision checker then finding a safety certificate is only log factors slower than running a collision check–suggesting that systems robust to uncertainty do not necessarily have to have significantly more computational power.

Furthermore, since the algorithm computes a separate ϵ for every obstacle, obstacles with little relevance to the robot's actions do not significantly affect the resulting risk bound. This allows for a much tighter bound than algorithms which allocate the same risk for every obstacle.

C. Experiments

We can illustrate the advantages of a geometric approach by certifying a trajectory with a probability of failure very close to zero. For an allowable chance of failure of ϵ, the runtime of sample-based, Monte-Carlo methods tends to depend on $\frac{1}{\epsilon}$ as opposed to $\log 1/\epsilon$. Monte-Carlo based techniques rely on counting failed samples requiring them to run enough simulations to observe many failed samples. This means that they have trouble scaling to situations where ϵ approaches zero and failed samples are very rare. For example, Janson et al.'s method takes seconds to evaluate a simple trajectory with $\epsilon = 0.01$, even with variance reduction techniques [6].

We demonstrate our algorithm on a simple domain with $\epsilon = 2.2 \times 10^{-5}$. Our algorithm required just 6 calls to a collision checker for each obstacle. We also demonstrate that our algorithm can certify trajectories which cannot be certified as safe with shadows of equal sizes. Figures 7 and 8 show the problem domain. Figure 7 shows that the trajectory cannot be certified as safe with a uniform risk assigned to each obstacle. Figure 8 shows the shadows found by our algorithm that prove the trajectory is safe.

IV. Online Safety

The bounds in the previous section do not immediately generalize to a setting where the robot acquires more information over time and can be allowed to change its desired trajectory. Additional care must be taken to ensure that the system cannot "trick" the notion of safety used, and not honor the desired contract on aggregate lifetime risk of the execution instance. Consider the case where, if a fair coin turns up as heads the robot takes a path with a 1.5ϵ probability of failure and it takes a trajectory with a 0.5ϵ probability of failure otherwise. This policy takes an action that is unsafe, but the probability of

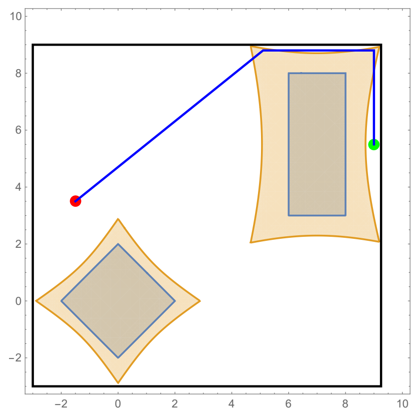


Fig. 7: Computing the optimal equal allocation of probabilities fails to certify the safety of the path.

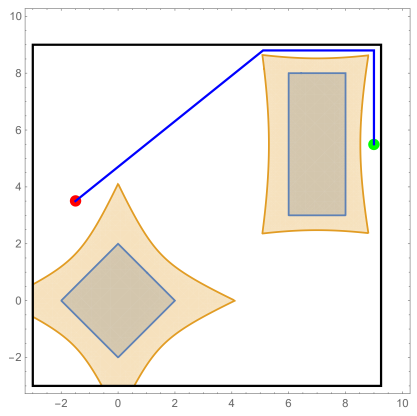
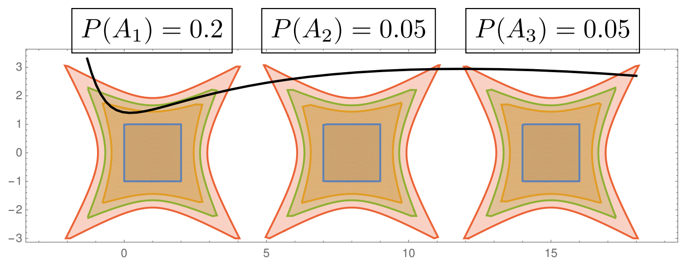


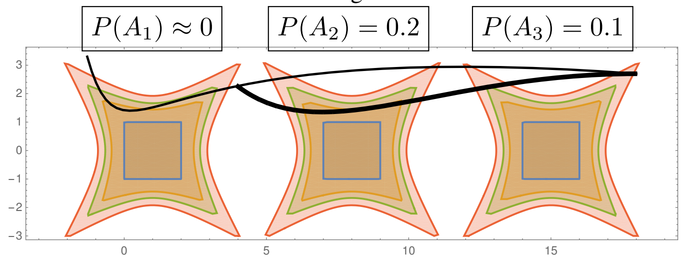
Fig. 8: Computing the optimal probability for each shadow allows us to successfully verify that the trajectory is safe.

failure of the policy is still less than ϵ. Furthermore, the history of actions is also important in ensuring aggregate lifetime safety. In figure 9 we illustrate a example of how the robot can always be committed to some trajectory that is ϵ-safe but have more than an ϵ probability of collision over the lifetime of the execution.

Figure 9 highlights the need to ensure low probability of failure under all sets of observations. If this scenario is run multiple times the failure rate will be much greater than acceptable. In order to propose an algorithm that allows the robot to change the desired trajectory as a function of a stream of information, we develop an alternative criteria for safety that accounts for risks as they are about to be incurred. We let p_t denote the probability of collision at time t, given the information available at time t, given that we follow the trajectory currently predicted by the policy π. We note that since the information itself is random, p_t is a random variable for future times. We say that a policy π is *absolutely safe* if for all times t, equation (1) is satisfied. The expectation in the integral is with respect to the information available at the



(a) At first the robot chooses the thin black trajectory which is has a probability of collision of only 0.3. It is more likely to collide with the first obstacle than the remaining obstacles.



(b) After observing that it had not collided with the first obstacle, it readjusts the plan to follow the bold trajectory so that the probability of collision is still less than 0.3. However, a system that follows this policy will collide with probability 0.51 even though at every point it was following a trajectory with probability of collision less than 0.3. In other words, for this set of observations O, $P(A|\pi, O) > 0.3$. A system that is allowed to change its action after seeing additional information (in this case the fact that it did not collide) must properly account for the risk already taken.

Fig. 9: Tricking "safety" by changing paths once more information is acquired. Note that even knowing that the system did not collide can serve as information.

current time t.

$$\int_0^\infty E[p_t \mid \pi]dt = \int_0^t p_t dt + \int_t^\infty E[p_t \mid \pi]\; dt \le \epsilon \qquad (1)$$

We note that the $\int_0^t p_t dt$ can be evaluated as an accumulation with standard numerical techniques for evaluating integrals.

The second term, $\int_t^\infty E[p_t \mid \pi]\; dt$, is exactly the probability that the remaining part of the trajectory will collide and can be evaluated with the method for solving the offline safety problem.

A. Absolute Safety vs Policy Safety

Algorithm 2 provides a method for performing safe online planning in the case that the PGDF parameters are updated during execution. While it shows that absolute safety can be verified efficiently, it is not clear how to efficiently verify policy safety. However, unlike absolute safety, policy safety (introduced in section I-B) is a very direct condition on aggregate lifetime probability of collision and can be easier to interpret. In this section we compare policy safety to absolute safety in order to identify when they are equivalent.

Algorithm 2 FIND_MAXIMAL_SHADOWS_ONLINE

```
Input: intersects_i, $\epsilon_p$, $t$, $\{p_{t'} | \forall t' \in [0, t]\}$
Output: $\epsilon$, s.t. $\epsilon$ is greater than the sum cumulate risk taken
    before the current time $t$ and the future risk. $\epsilon$ is at most
    $\epsilon_p$ away from the minimal $\epsilon$ for which a bound of this
    class may be obtained.
1: $\epsilon_1 = \int_0^t p_t dt$
2: $\epsilon_2 =$ FIND_MAXIMAL_SHADOW_SET($\epsilon_p, \boldsymbol{\mu}_{1...n}, \boldsymbol{\Sigma}_{1...n}$)
3: return $\epsilon_1 + \epsilon_2$
```

First we show that absolute safety is a strictly stronger condition than policy safety in theorem 3. This comes by integrating the probability of failure over time to get the total probability of failure. Since the absolute safety condition in equation (1) must always be satisfied, regardless of the observation set, the probability of failure for that information set will always be sufficiently small.

Theorem 3. *If a policy is absolutely safe, then it is also safe in the policy safety sense.*

Proof: Please see the supplementary material in the extended paper[1].

Absolute safety, however, is not always equivalent to policy safety. The key difference lies in how the two conditions allow future information to be used. Absolute safety requires that the system always designate a safe trajectory under the current information while policy safety allows the robot to postpone specifying a complete, safe trajectory if it is certain it will acquire critical information in the future.

In order to formalize when policy safety and absolute safety are equivalent, we introduce the notion of an information adversary. An information adversary is allowed to (1) see the observations at the same time as the agent, (2) access the policy used by the agent, and (3) terminate the agent's information stream at any point. Policy safety under an information adversary is guaranteed by the policy safety conditions if the information stream can stop naturally at any point. Theorem 4 shows that policy safety with an information adversary is equivalent to absolute safety.

Theorem 4. *A policy that is safe at all times under an information adversary is also absolutely safe.*

Proof: Please see the supplementary material of the extended paper[1].

B. Experiments

We demonstrate a simple replanning example based on the domain presented in figure 8. Once the robot gets halfway through, it will receive a new observation that helps it refine its estimate of the larger, second obstacle. This allows it to shrink the volume of the shadow corresponding to the same probability and certify a new, shorter path as safe. This new path is shown in figure 10. It takes this new trajectory without taking an unacceptable amount of risk.

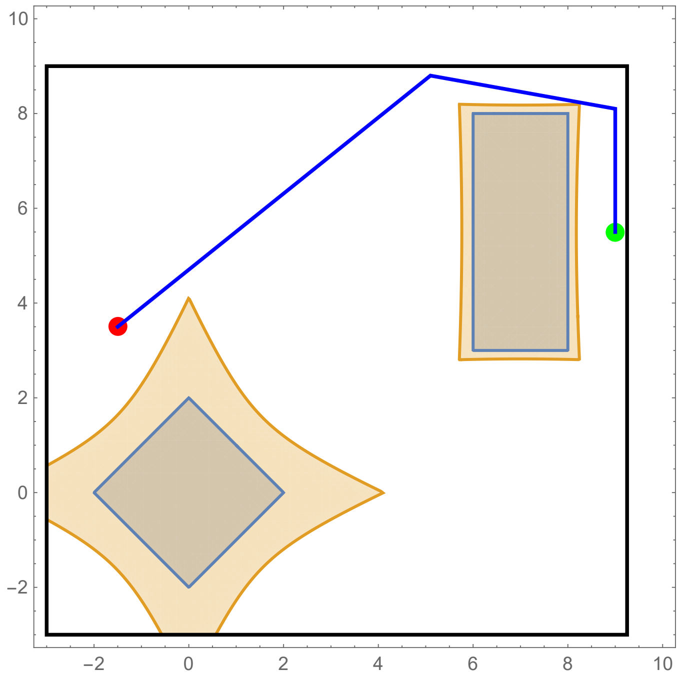


Fig. 10: Once the robot reached the top of the old trajectory it got a new observation regarding the second obstacle. This allows it to shrink the shadow around the second obstacle and take the less conservative path that is shown above.

Algorithm 3 SAFE_RRT

```
Input: $\epsilon_{safe}$, $\epsilon_p, t, q_s, q_f, \boldsymbol{\mu}_{1...n}, \boldsymbol{\Sigma}_{1...n}$
Output: A sequence of waypoints from $q_s$ to $q_f$, such that the
    trajectory going through these waypoints has a probability
    of less than $\epsilon_{safety}$ of collision.
1: $tree =$ new TREE($q_s$)
2: for iteration $= 1...n$ do
3:     $x_{rand} =$ RANDOM_STATE
4:     $x_{near} =$ NEAREST_NEIGHBOR(TREE)
5:     $x_{new} =$ EXTEND($x_{near}, x_{rand}$)
6:     $X =$ GET_TRAJECTORY($tree, x_{near}, x_{new}$)
7:     $risk =$
8:       FIND_MAXIMAL_SHADOW_SET($X, \epsilon_p, \boldsymbol{\mu}_{1...n}, \boldsymbol{\Sigma}_{1...n}$)
9:     if $risk \leq \epsilon_{safe}$ then
10:        ADD_CHILD($tree, x_{near}, x_{new}$)
11:    end if
12: end for
13: return $tree$
```

V. Enabling Safe Planning using Safety Certificates

The safety certification algorithms we presented above can be used for more than just checking safety. It can enable safe planning as well. We present a modification to the RRT algorithm that restricts output to only safe plans [9]. Every time the tree is about to be expanded, the risk of the trajectory to the node is computed. The tree is only grown if the risk of the resulting trajectory is acceptable.

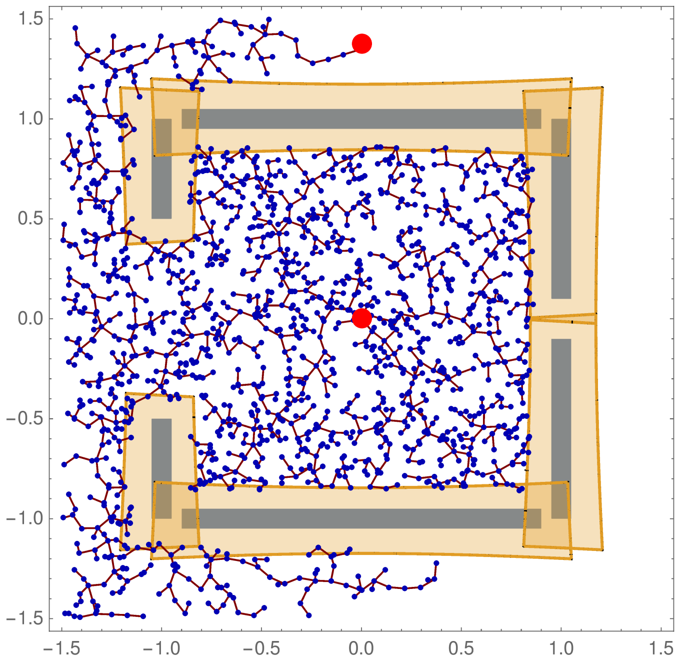


Fig. 11: A tree of safe trajectories branching from the red dot in interior of the box. Equally sized shadows are shown for reference.

We note that it is not necessary to check the safety of the entire trajectory every time the tree is extended. Since the bounds for each obstacle are determined by a single point in the trajectory, it is sufficient to consider the following two risks: the trajectory from the root of the tree to x_{near} and the trajectory from x_{near} to x_{new}. Finally we note that analyzing probabilistic completeness is quite different in the risk constrained case from the original case. We do not believe this method is probabilistically complete. Unlike the deterministic planning problem, the trajectory taken to reach a point affects the ability to reach future states—breaking down a crucial assumption required for RRTs to be probabilistically complete.

We demonstrate the safe-RRT algorithm on a point robot trying to escape a box. The box has two exits. While the robot can safely pass through the larger exit, it cannot safely pass through the smaller exit. The planner is run to only return paths with a probability of failure less than 0.5%. Figure 11 shows a safe tree from the red dot inside the box to the red dot above the box. Figure 1 shows just the ultimate trajectory with its corresponding shadows that certify the probability of failure as less than 0.26%. Note that some shadows did not extend all the way to the trajectory as their risk was already below the numerical threshold.

The experiment shown in figure 11 demonstrates offline-safety. If the robot were given additional information during execution, we could use the equations for online-safety to re-run the RRT with the new estimates of obstacles while preserving the safety guarantee.

VI. Conclusion

We presented a framework to compute shadows, the geometric equivalent of a confidence interval, around observed geometric objects. Our bounds are tighter than those of previous methods and, crucially, the tightness of the bounds does not depend on the number of obstacles. In order to achieve this tightness we rely on computing a bound specific to a trajectory instead of trying to identify a generic "safe" set.

We present offline and online variants of algorithms that can verify safety with respect to the shadows identified above for both trajectories and policies. The online method highlights nuances and potential issues with a mathematical definition of safety, and presents a strong, but still computationally verifiable notion of safety. These algorithms do not have a computational complexity much larger than a collision check, and are only a $O\left(\log n \log \frac{1}{\epsilon}\right)$ factor slower than a collision check for n obstacles and an $\epsilon-$safety guarantee. Finally the output of these algorithms is easy to verify, allowing the output to serve as a safety certificate.

These safety certification algorithms are an important not only in ensuring that a given action is safe, but also in enabling the search for safe plans. We demonstrate an extension to the RRT algorithm that only outputs safe plans.

Future work includes using other models of random geometry. A model for occupancy would match the information of interest to a motion planner more closely, and may be easier to apply to real world data sets. The presented algorithms are compatible with mixture models—a potentially interesting and useful direction that should be evaluated empirically. Another direction of interest is the development of safe planning algorithms with strong theoretical guarantees such as optimality and probabilistic completeness.

Acknowledgements

BA would like to thank Gireeja Ranade, Debadeepta Dey and Ashish Kapoor for introducing him to the safe robot navigation problem during his time at Microsoft Research.

We gratefully acknowledge support from the Thomas and Stacey Siebel Foundation, from NSF grants 1420927 and 1523767, from ONR grant N00014-14-1-0486, and from ARO grant W911NF1410433. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of our sponsors.

References

[1] Brian Axelrod, Leslie Kaelbling, and Tomas Lozano-Perez. Provably safe robot navigation with obstacle uncertainty. *Robotics Science and Systems*, 13, 2017. URL http://lis.csail.mit.edu/pubs/axelrod-rss-17.pdf.

[2] Christopher M Bishop. Pattern recognition. *Machine Learning*, 128, 2006.

[3] 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. URL http://ieeexplore.ieee.org/abstract/document/5980508/.

[4] Xu Chu Ding, Alessandro Pinto, and Amit Surana. Strategic planning under uncertainties via constrained markov decision processes. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 4568–4575. IEEE, 2013. URL http://ieeexplore.ieee.org/document/6631226/.

[5] Seyedshams Feyzabadi and Stefano Carpin. Multi-objective planning with multiple high level task specifications. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 5483–5490. IEEE, 2016. URL http://ieeexplore.ieee.org/document/7487762/.

[6] 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. URL https://web.stanford.edu/~pavone/papers/JSP.ISRR15.pdf.

[7] Leslie Pack Kaelbling and Tomás Lozano-Pérez. Integrated task and motion planning in belief space. *The International Journal of Robotics Research*, page 0278364913484072, 2013. URL http://journals.sagepub.com/doi/abs/10.1177/0278364913484072.

[8] 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. URL http://dl.acm.org/citation.cfm?id=1643301.

[9] Steven M LaValle. Rapidly-exploring random trees: A new tool for path planning. 1998. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.35.1853.

[10] 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 *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5660–5667. IEEE, 2013. URL http://ieeexplore.ieee.org/document/6697176/.

[11] Anirudha Majumdar, Mark Tobenkin, and Russ Tedrake. Algebraic verification for parameterized motion planning libraries. In *American Control Conference (ACC), 2012*, pages 250–257. IEEE, 2012. URL https://arxiv.org/abs/1601.04037.

[12] Chonhyon Park, Jae Sung Park, and Dinesh Manocha. Fast and Bounded Probabilistic Collision Detection in Dynamic Environments for High-DOF Trajectory Planning. *CoRR*, abs/1607.04788, 2016. URL https://arxiv.org/abs/1607.04788.

[13] Jae Sung Park, Chonhyon Park, and Dinesh Manocha. Efficient Probabilistic Collision Detection for Non-Convex Shapes. *CoRR*, abs/1610.03651, 2016. URL https://arxiv.org/abs/1610.03651.

[14] R. Platt, R. Tedrake, L. Kaelbling, and T. Lozano-Perez. Belief space planning assuming maximum likelihood observations. In *Proceedings of Robotics: Science and Systems*, Zaragoza, Spain, June 2010. doi: 10.15607/RSS.2010.VI.037. URL http://www.roboticsproceedings.org/rss06/p37.html.

[15] Carl Edward Rasmussen. Gaussian processes for machine learning. 2006. URL http://www.gaussianprocess.org/gpml/.

[16] Dorsa Sadigh and Ashish Kapoor. Safe Control under Uncertainty with Probabilistic Signal Temporal Logic. In *Proceedings of Robotics: Science and Systems*, AnnArbor, Michigan, June 2016. doi: 10.15607/RSS.2016.XII.017. URL http://www.roboticsproceedings.org/rss12/p17.html.

[17] Wen Sun, Luis G Torres, Jur Van Den Berg, and Ron Alterovitz. Safe motion planning for imprecise robotic manipulators by minimizing probability of collision. In *Robotics Research*, pages 685–701. Springer, 2016. URL http://link.springer.com/chapter/10.1007/978-3-319-28872-7_39.