Fast Safe Mission Plans for Autonomous Vehicles

Abstract

Guaranteeing safety is a key problem that needs to be addressed in order to enable the real-world deployment of robots and autonomous cyber-physical systems (CPS). While there is a lot of interest in deploying sensors and predictors that would identify obstacles and unsafe situations, there is little research on how to use such learned systems to plan and execute missions safely and efficiently. Recent research on safe planning and control not only admits to simplistic constraints, most of them assumed to be known a priori, but attempting such synthesis often results in large optimization problems which are often impractical to solve given real time constraints of such systems. In this work we propose a novel combination of sampling-based motion planning with safe control synthesis methods for generating safe high-level plans in real-time. The distinguishing aspect of our work is that it provides a natural framework of incorporating sensor data and the associated prediction about the obstacles to quickly determine the safe mission plan. We showcase this approach with autonomous car scenarios.

1 Introduction

Robotic and cyber-physical systems are proliferating at a breakneck pace. Semi-autonomous and fully autonomous cars and unmanned aerial vehicles (UAVs) are already reality and are expected to integrate more closely with humans in the near future [Urmson *et al.*, 2008]. A key technological hurdle in this process is to ensure the safety of such systems at all times especially within the proximity of humans and while carrying out mission-critical tasks. While there has been a push in identifying obstacles and unsafe situations via sensors and machine learned predictors [Dey *et al.*, 2015; Aoude *et al.*, 2013], the task of embedding such information to determine safe course while obeying rules-of-the-road is non-trivial [Plaku and Karaman, 2015]. Further, the uncertainty and noise in prediction together with near real-time requirements under bounded computation resources makes this problem very challenging [Horvitz, 2001].

This work proposes an architecture for fast and safe planning of autonomous missions. The key idea behind this work is to consider recent research on optimization based safe controllers and then incorporate fast sampling based procedures to generate in real-time the mission plans for complex scenarios. Specifically, we build upon the recent work in Probabilistic Signal Temporal Logic (PrSTL) [Sadigh and Kapoor, 2015] that synthesizes provably safe controllers that take into account the noisy sensor readings and the associated uncertainty in learned classifier or regressor predictions. Currently, the state-of-the-art solution for PrSTL requires solving Mixed Integer Semi-Definite Programs (MISDPs), which quickly become infeasible to solve in reasonable time as the number of constraints grow. Further, PrSTL needs the description of the mission goal and the required safety invariants as logical formulations and often expressing such objectives and constraints for long horizons and complicated rules-of-the-road remain non-trivial at best. Finally, the near real-time planning requirements under constrained computational resources makes such approaches impractical.

We alleviate these problems by combining PrSTL with random sampling based planners. We propose using Rapidly-exploring Random Trees (RRT) [Lavalle and Kuffner Jr, 2000] and associated variants like RRT* [Karaman *et al.*, 2011] to simplify computation by first efficiently sampling feasible points in the robot's configuration space and then generating trajectories by connecting them via safe control. Such fast sampling of the feasible trajectories effectively reduces the optimization from a MISDP to a sequence of Second Order Cone Programs (SOCP), which being convex, can be solved much more efficiently. Furthermore, the RRT/RRT* framework only needs a procedure/function to check the violation of safety constraints. Consequently there is no requirement enforcing that all the invariants must be written as logical programs. Instead we can use only the convex subset of invariants in PrSTL to encode safety constraints. Specifically, our contributions in this paper are:

- A framework for fast and safe mission planning under uncertainty.
- Combining RRT* with control for PrSTL to generate adaptive plans such that the resulting trajectories satisfy both the safety and the PrSTL specifications using SOCP.

- A toolbox implementing the framework and experiments in autonomous driving and control of quadrotors.

In the rest of this paper, we first discuss some of the related work and preliminaries, we then describe our framework and show our solution along with some experimental results which shows the large speedup obtained in long range planning without sacrificing safety.

2 Background

2.1 Planning and Control

Most autonomous systems today are implemented as hybrid hierarchical systems with a mission planner at the top level that gives low level controllers smaller primitives (e.g. trajectories) to execute [Urmson *et al.*, 2008; Cowlagi, 2011]. The objective of the mission planner is to satisfy the mission-level objectives while observing the rules-of-the-road and avoiding obstacles. In contrast controllers at lower levels of the hierarchy (e.g. PID [Ziegler and Nichols, 1942], LQR [Li and Todorov, 2004], H-infinity [Zames, 1981]) are responsible for executing trajectories handed to them and actually making the system reach the goal state [Zames, 1981].

Examples of high level mission planners include state-lattice motion primitives [Pivtoraiko and Kelly , 2005], incomplete planners like CHOMP [Ratliff *et al.*, 2009] and complete sampling-based motion planning like Probabilistic Road Maps (PRM) [Kavraki *et al.*, 1996], Rapidly-Exploring Random Tree (RRT) [Lavalle and Kuffner Jr, 2000] and variants like RRT* [Karaman *et al.*, 2011]. In the past few years sampling-based motion planners have shown great improvements in solving high dimensional kinodynamic motion planning problems for a wide variety of robotic systems from high-degrees-of-freedom manipulators [Berenson *et al.*, 2009] to mobile robot navigation [Kuwata *et al.*, 2008]. They are relatively simple to implement and overcome the curse of dimensionality problem by cleverly leveraging the lower dimensionality of the task space as compared to the configuration (joint) space of robots. RRT [Lavalle and Kuffner Jr, 2000] is probabilistically complete in the sense that if there exists a feasible path, RRT is guaranteed to find that path given enough time. Nonetheless, RRT was proven to converge to sub-optimal solutions and the proposed alternative RRT* [Karaman and Frazzoli, 2011] is provably asymptotically optimal albeit at the cost of increased computational complexity.

A major limitation of such planners is that they assume that the environment is perfectly known beforehand. In practice this is rarely the case and most autonomous robots have onboard sensors (e.g cameras, radars, lidars) which perceive the environment in the immediate vicinity with some uncertainty. Additionally there may be uncertainty in dynamics as well. Along with the growing proximity to humans of such systems such as autonomous cars and drones, constructing plans in real-time which are provably safe is becoming a challenge of growing importance. While there have been previous efforts to incorporate such uncertainty into planning [Melchior and Simmons, 2007; Dey *et al.*, 2015; Van Den Berg *et al.*, 2011], safety guarantees have been difficult to provide in such settings especially under constrained computational budget. One of the closest approaches to our work is that of [Luders *et al.*, 2010] who propose a chance-constrained RRT (CC-RRT) where uncertainty in dynamic obstacles and sensing is propagated down the tree and only those paths in the tree are kept such that they satisfy a real-time constraint. In contrast to CC-RRT, our approach uses PrSTL in the steering function, and allows a much richer class of Boolean and temporal constraints to be specified in addition to probabilistic constraints. Furthermore, using our approach with PrSTL constraints instead of CC-RRT provides a natural way of expressing uncertainties present from Bayesian classifiers that update their beliefs by inference.

There has been recent developments in synthesizing controllers, inspired from tools and techniques in program verification and artificial intelligence where the goal is to synthesize control policies that satisfy temporal properties, disjunctions, conjunctions or negations of user-specified predicates. Examples of such constraints include "The robot must *always* stay 2 meters away from all obstacles" or "First go to X and then do Y". Several temporal logic specification languages have been developed and adapted for synthesizing controllers such as Linear Temporal Logic (LTL) [Kress-Gazit *et al.*, 2009; Tabuada and Pappas, 2004], Metric Temporal Logic (MTL) [Karaman and Frazzoli, 2008], Probabilistic Temporal Logic (PTL) [Yoo *et al.*, 2012] and Signal Temporal Logic (STL) [Raman *et al.*, 2015; Raman *et al.*, 2014; Maler and Nickovic, 2004]. These approaches can be used for task planning [Plaku and Karaman, 2015], where a system designer a priori specifies logical specifications composed of disjunctions, conjunctions, negations as well as temporal permutations of those combinations. Previous work has also proposed methods for combining sampling-based motion planners with such specification languages to do joint task and motion planning for autonomous robots [Karaman and Frazzoli, 2009; Karaman and Frazzoli, 2008]. However, these approaches are limited in their capacity to both express the constraints as well as the capability to account for uncertainty in sensors and dynamics.

Temporal logics like STL are amenable to specification of stochastic properties of continuous signals. These signals could be functions of the robot state, environment and other safety parameters. Recent work has proposed probabilistic logical specification (PrSTL) [Sadigh and Kapoor, 2015] that introduces random variables in logical formulae to express uncertainty in the robot state, environment and other exogenous variables. Such probabilistic formulation enables embedding of Bayesian classifiers and predictors in the specification language, thereby allowing the systems to operate in environments that are only partially observed. In our framework we propose a new method that builds upon RRT* and uses PrSTL as a steering function. This method combines the positive aspects of both the techniques: (1) PrSTL enables us to specify safety invariants and allows embedding of machine learning predictors operating on real-time signals. (2) The RRT* framework allows fast computation of strategies circumventing the need to solve computationally difficult problems that usually arise in logical specification based control synthesis methods. We next provide more details on PrSTL.

2.2 Probabilistic Signal Temporal Logic

Probabilistic Signal Temporal Logic (PrSTL) allows expressing stochastic properties over real-valued, dense-time signals. Given the capabilities that PrSTL provides, we can formally define temporal properties over uncertainties that are present in sensors and classifiers of the system. For example, we can express PrSTL formulas that represent probability that the output of a Bayesian predictor would lie in a desired range for time steps in the future.

Let $x(t)$ denote a real-valued signal at time t, then $(x, t) \models \varphi$ specifies that the signal x satisfies the PrSTL formula φ at time t. A PrSTL formula φ consists of temporal and Boolean properties over atomic predicates represented as $\lambda_{\alpha_t}^{\epsilon_t}$. Such predicates are defined over time-varying random variables α_t drawn from a distribution at every time step. Furthermore, $\epsilon_t \in [0, 0.5]$ represents a tolerance level for satisfaction of the predicate. Therefore, satisfaction of this atomic predicate translates to:

$$(x, t) \models \lambda_{\alpha_t}^{\epsilon_t} \iff P\big(\lambda_{\alpha_t}(x(t)) < 0\big) > 1 - \epsilon_t, \quad (1)$$

where $\lambda_{\alpha_t}(x(t))$ is a stochastic function of the signal, which can express uncertainties regarding sensors, classifiers, etc. For example, if α_t represent parameters of a classifier then computing the stochastic function simply corresponds to application of the classifier to $x(t)$. Consequently, the atomic predicate described above signifies that only those trajectories for which the condition $\lambda_{\alpha_t}(x(t)) < 0$ holds with a high probability should be considered valid. PrSTL allows nesting of temporal and Boolean properties over the probabilistic predicates. Similar to other temporal logics, PrSTL provides the capability of expressing rich properties such as safety, response, surveillance, etc. in addition to preserving the uncertainties inherent in sensors as part of the formula. The syntax of PrSTL is defined as follows:

$$\varphi ::= \lambda_{\alpha_t}^{\epsilon_t} \,|\, \tilde{\neg}\lambda_{\alpha_t}^{\epsilon_t} \,|\, \varphi \wedge \psi \,|\, \varphi \vee \psi \,|\, \mathbf{G}_{[a,b]}\psi \,|\, \varphi\, \mathbf{U}_{[a,b]}\psi \,|\, \mathbf{F}_{[a,b]}\psi.$$

Here, φ is constructed as a probabilistic predicate $\lambda_{\alpha_t}^{\epsilon_t}$, its negation $\tilde{\neg}\lambda_{\alpha_t}^{\epsilon_t}$, the Boolean conjunction or disjunction of two PrSTL formulae, or temporal operators applied over PrSTL formulae. The temporal operators consist of $\mathbf{G}$ (*globally*), $\mathbf{F}$ (*eventually*) and $\mathbf{U}$ (*until*). For example, $\mathbf{G}_{[5,7]}(P(\lambda_{\alpha_t}(x(t)) < 0) > 0.8)$ is a formula indicating that the stochastic function $\lambda_{\alpha_t}(x(t))$ must be less than zero with 0.8 confidence for all times in the interval $t \in [5, 7]$.

The satisfaction of each temporal or propositional formula is then defined as follows:

$$\begin{array}{lcl}
(\xi, t) \models \lambda_{\alpha_t}^{\epsilon_t} & \Leftrightarrow & P(\lambda_{\alpha_t}(\xi(t)) < 0) > 1 - \epsilon_t \\
(\xi, t) \models \tilde{\neg}\lambda_{\alpha_t}^{\epsilon_t} & \Leftrightarrow & P(-\lambda_{\alpha_t}(\xi(t)) < 0) > 1 - \epsilon_t \\
(\xi, t) \models \varphi \wedge \psi & \Leftrightarrow & (\xi, t) \models \varphi \wedge (\xi, t) \models \psi \\
(\xi, t) \models \varphi \vee \psi & \Leftrightarrow & (\xi, t) \models \varphi \vee (\xi, t) \models \psi \\
(\xi, t) \models \mathbf{G}_{[a,b]}\varphi & \Leftrightarrow & \forall t' \in [t + a, t + b], (\xi, t') \models \varphi \\
(\xi, t) \models \mathbf{F}_{[a,b]}\varphi & \Leftrightarrow & \exists t' \in [t + a, t + b], (\xi, t') \models \varphi \\
(\xi, t) \models \varphi\, \mathbf{U}_{[a,b]}\, \psi & \Leftrightarrow & \exists t' \in [t + a, t + b] \text{ s.t. } (\xi, t') \models \psi \\
 & & \wedge \forall t'' \in [t, t'], (\xi, t'') \models \varphi.
\end{array}$$

The PrSTL formulas can be treated as constraints in receding horizon optimization problems that synthesize safe controllers. Also note that as the robot begins to traverse the trajectory, it has the opportunity to observe new data and update its beliefs over the parameters α_t. Such evolution of constraints allow the autonomous system to safely operate in environment thats partially observed.

It is shown that given α_t is drawn from a Gaussian distribution, the control synthesis problem under PrSTL constraints can be solved as a mixed integer semi-definite program (MISDP). While solving an MISDP is NP-complete, there exists a subset of PrSTL called *Convex PrSTL* that is recursively defined over the predicates using only conjunctions or the globally operator. As the name implies the optimization problem reduces to second order cone programming (SOCP) and is convex. One of the advantages of the framework proposed in this paper is that instead of solving a general mission planning task as computationally intensive PrSTL, it uses RRT* to decompose the problem into a sequence of simpler convex optimization tasks.

3 Approach

In this section, we detail our approach for the general scenario where a robot is tasked with navigating from a start state to a goal state and an *incomplete* map of the environment is available. This means that there might be additional obstacles and other latent variables on the map, which are unknown in the beginning but as the robot navigates, onboard sensors (noisily) detect them.

The key idea in our approach is to first sample the configuration space for valid points that satisfy the safety invariants, and then seek for a safe path or trajectory that would connect these sets of sampled points. Such safe trajectories are determined via safe control synthesis using the PrSTL framework. Given the sampled points and the safe trajectories that connect these, the framework finally chooses the shortest path from the start to the goal which minimizes the cost criterion of interest.

One big advantage of this framework is that the validity test for random samples does not need a rigid logical specification and can be expressed as an imperative procedure. Such imperative descriptions allows checking of fairly complex safety conditions, which might be very hard to evaluate using PrSTL. For example, the boundaries of a flying arena can be of arbitrary shape, and constraints on such non-parametric boundaries cannot be easily expressed as logical formulae. However, given a map of such arena it is easy to check whether a sample is valid or not. Also note that the PrSTL framework has the capability to embed an online predictor that can continuously monitor the environment. Consequently, any obstacle or unsafe conditions not known a priori can also be handled in a seamless manner.

Algorithm 1 details the main steps of our approach, which we term as PRSTL-TREE. The algorithm requires a map $\mathcal{M}$ of the environment which contains known obstacles and also encodes rules-of-the-road like no-fly regions, a start state s_{start}, a goal state s_{goal}, radius g_{radius} which describes the goal region centered around the goal state, the number of vertices to be built into the sampling-based motion planner tree n_{vertices} at each planning cycle and the number of steps n_{steps} that the robot will actually traverse each planning cycle.

Algorithm 1 PrSTL-Tree: Safe planning and control to goal.

```
Require: Map of known obstacles $\mathcal{M}$
         Start state $s_{\text{start}}$
         Goal state $s_{\text{goal}}$
         Goal region radius $g_{\text{radius}}$
         Number of vertices in tree $n_{\text{vertices}}$
         Number of steps per planning cycle $n_{\text{steps}}$
Ensure: Path traversed to goal $p = \{s_{\text{start}}, s_1, \dots, s_{\text{goal}}\}$
 1: $p = \{\}$
 2: $s_{\text{current}} = s_{\text{start}}$
 3: while dist($s_{\text{current}}, s_{\text{goal}}$) $> g_{\text{radius}}$ do
 4:     tree ← BuildSafeTree($\mathcal{M}, s_{\text{current}}, s_{\text{goal}}, n_{\text{vertices}}$)
 5:     $s_{\text{nearest}}$ ← FindNearestNeighbor(tree, $s_{\text{goal}}$)
 6:     $p_{\text{shortest}}$ ← ShortestPathToGoal(tree, $s_{\text{current}}, s_{\text{nearest}}$)
 7:     ($p_{\text{traversed}}, s_{\text{current}}$, obsv) ← TakeNSteps($p_{\text{shortest}}, n_{\text{steps}}$)
 8:     $p \leftarrow p \cup p_{\text{traversed}}$
 9:     UpdateBelief(obsv)
10: end while
```

Initially the path taken by the robot is set to the empty sequence $p = \{\}$ and the current state is set to s_{current} (lines $1-2$). While the robot is still more than g_{radius} away from the goal region the safe planner is invoked in a receding-horizon style to find a safe path to goal (lines 3 – 10). In line 4 the function BuildSafeTree invokes a sampling-based motion planner on the map $\mathcal{M}$ of known obstacles. Suitable choices for sampling-based motion planner include RRT [Lavalle and Kuffner Jr, 2000] and RRT* [Karaman and Frazzoli, 2011]. This function creates a tree so that it has n_{vertices} from the current state s_{current} of the robot towards the goal state s_{goal}. Note that it is not a requirement for the tree to reach the goal in n_{vertices}. Approaches like RRT and RRT* build a tree towards the goal state by sampling states at random, checking that they lie in free space and then connecting the sampled state to the nearest node{s} in the tree using a steering function which is responsible for producing dynamically feasible trajectories. These trajectories are then checked for collision and satisfaction of rules-of-the-road and then added to the tree. In this work, we take the approach of constructing dynamically feasible and *high-probability collision-free* trajectories for connecting states in the tree leveraging the PrSTL [Sadigh and Kapoor, 2015] framework for synthesizing trajectories. PrSTL takes sensor uncertainty and robot dynamics into account to synthesize trajectories which are probabilistically safe up to user-specified confidence. If it is not feasible to construct such a trajectory then the PrSTL routine returns an empty trajectory and the sampled state is rejected. So in line 4 the returned $tree$ has edges (trajectories) which are safe by construction.

In line 5 the nearest state s_{nearest} in the tree to the goal state is found by an efficient nearest neighbor search. Then the shortest path in the tree from root (s_{current}) to s_{nearest} is computed using A* [Hart *et al.*, 1968] or Dijkstra's shortest path algorithm [Dijkstra, 1959] in line 6 to give a path p_{shortest}.

In line 7 the robot executes n_{steps} of p_{shortest} and ends up in a new current state s_{current}. Along the way it makes observations using its onboard (noisy) sensors which then can be used to update the obstacle classifier embedded in the PrSTL framework. If the robot is not in the goal region at this time, it builds a safe tree again using its updated beliefs from its current state.

PrSTL-Tree mitigates the chief limitation of using PrSTL alone for long-horizon mission planning with arbitrarily complicated obstacle maps and rules-of-the-road: it eliminates the use of mixed-integer constraints which are necessary for accounting for obstacles and sequential waypoints in PrSTL. Since mixed-integer SOCPs or SDPs are NP-hard [Papadimitriou and Steiglitz, 1982] these are usually solved sub-optimally by branch-and-bound based algorithms and have large runtimes for non-trivial problem sizes. By relegating this difficult task of modeling known obstacles and waypoints to a sample-based planner, only the convex subset of PrSTL is needed which gives rise to SOCPs which are convex and can be solved optimally in polynomial time.

In section 4 we show via two examples in simulation the large speedup in time obtained by using PrSTL-Tree as opposed to PrSTL.

4 Case Study: Experiment with Autonomous Robot Obstacle Avoidance

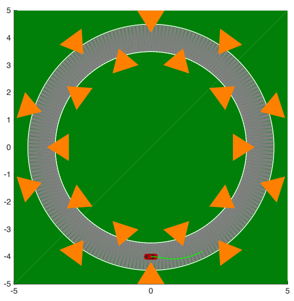
In this section, we study safe control of an autonomous ground robot under known obstacles and uncertain environments. We let the dynamics of the robot be a simple point-mass model, where the state of the robot is: $\mathbf{x} = [x \quad y \quad \theta \quad v]^\top$. Here, x and y are the coordinates of the robot, θ is the heading angle , and v is the speed. The control input of this robot is $\mathbf{u} = [u_1 \quad u_2]$, where u_1 is the steering angle and u_2 is the acceleration. Then, given m is the mass of the ground robot, the dynamics model of the robot is:

$$\begin{aligned} \dot{x} &= v\cos(\theta) \quad & \dot{y} &= v\sin(\theta) \\ \dot{\theta} &= \frac{v}{m}u_1 & \dot{v} &= u_2 \end{aligned} \tag{2}$$

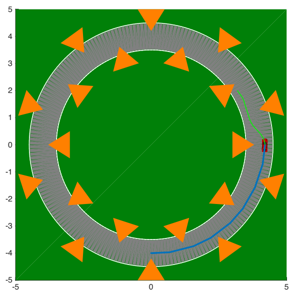
Our goal in this case study is to find control inputs for the ground robot so it reaches a final goal while avoiding obstacles and staying within the road boundaries. Figure 1a shows the initial setting of our experiment. The red car represents our ground vehicle that must stay within the boundaries of the circular road. The orange triangles represent the obstacles present in this scenario. The robot is constrained to travel on the road while avoiding the orange triangles. These obstacles can either be known a priori or only known locally based on uncertainties arising from classifiers.

In this case study, we compare two techniques for solving the safe controller synthesis problem: (i) Using receding horizon controller synthesis methods with PrSTL constraints for reaching the goal state as in [Sadigh and Kapoor, 2015]. (ii) Using our proposed method, PrSTL-Tree, which allows RRT planning as well as applying safe control for connecting the nodes of the tree.

In our experiments, we make this comparison between the two methods for both cases with known and unknown obstacles. Known obstacles refer to when the coordinates of all obstacles in the environment are known a priori, while unknown



(a) Initial state of the robot.



(b) Final state of the robot.

Figure 1: Autonomous robot reaching a final goal while avoiding obstacles. Here, the red car on the road shows the autonomous robot. The robot's goal is to travel on the circular road while avoiding obstacles and staying within boundaries. The orange triangles represent the obstacles on the road. The green line on both figures shows the computed future trajectory of the robot for the next horizon. The blue line in 1b shows the trajectory computed and taken by the robot to reach its goal.

obstacles correspond to when we run Bayesian classifiers to construct a belief of where the obstacles are located at in the safe trajectory planning.

4.1 Known Obstacles

We first consider the case when all the obstacles are known a priori. We represent avoiding each triangle constraint in Figure 1, by stating that the robot must stay outside of the triangle, which is equivalent to staying on one side of each face of the triangle. Note, obstacle avoidance is a non-convex property, which requires disjunction of properties that state the robot must stay on one side of a hyperplane.

Let $\mathbf{p}$ and $\mathbf{q}$ denote the coordinates of the two corners of a side of an obstacle as shown in Figure 2(a). We compute $\vec{\mathbf{n}}$, the normal vector to the hyperplane represented by $\overline{\mathbf{pq}}$. Then, to specify that the coordinates of a state $\mathbf{r} = [x \quad y]^\top$ must be exactly on one side of the hyperplane $\mathbf{pq}$ is equivalent to the following linear constraint:

$$\vec{\mathbf{n}} \cdot (\mathbf{r} - \mathbf{p}) \leq 0 \tag{3}$$

This constraint specifies that the inner product of the normal vector and the vector of $\vec{\mathbf{pr}}$ must be negative meaning that $\mathbf{r}$ is in the yellow region in Figure 2(a), which is outside the obstacle.

Then, the PrSTL formula that represents avoiding each triangle is a disjunction of staying on one side of each hyperplane of the triangle for all time steps:

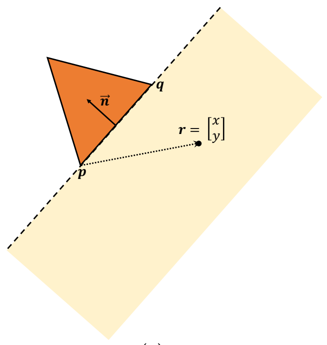
$$\mathbf{G}_{[0,\infty]} \bigvee_{i \in 1,2,3} (\vec{\mathbf{n}_i} \cdot (\mathbf{r} - \mathbf{p}_i) \leq 0) \tag{4}$$

Here, i is an index for sides of each triangle.

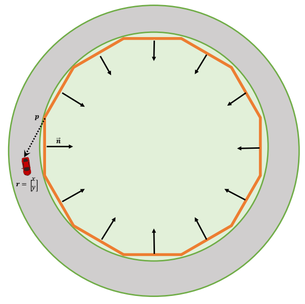
In addition to obstacle avoidance, the robot must stay within the boundaries of the road. Remaining inside the outer boundary is a convex property:

$$\mathbf{G}_{[0,\infty)}(|| [x \quad y]^\top ||_2) \leq R_{\text{out}}, \tag{5}$$

which indicates that the coordinates of the robot must stay within an outer radius R_{out}. We represent remaining outside

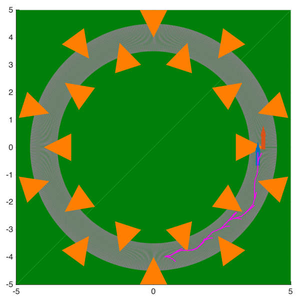


(a)

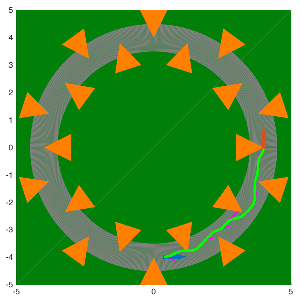


(b)

Figure 2: Obstacle avoidance by staying on one side of a hyperplane. $\mathbf{p}$ and $\mathbf{q}$ are the corners of an obstacle. We calculate $\vec{\mathbf{n}}$ as the unit normal to this hyperplane. For a state with coordinates $\mathbf{r}$ to be in the yellow region, the dot product of $\vec{\mathbf{n}}$ and vector $\vec{\mathbf{px}}$ must be negative. In (b), we show fitting a polygon to the inner circle in order to enforce staying out of the inner road boundary.



(a) The RRT generated for reaching the goal.



(b) Reaching the final state by following the tree.

Figure 3: Using PrSTL-Tree approach for reaching the final goal while staying safe. Here, all the obstacles are known a priori.

of the inner boundary by approximating the inner circle as a polygon as shown in Figure 2. Then, to enforce being outside of the inner circle is equivalent to enforce being outside of each edge of the polygon. Each of such constraints is similar to equation (3), which specifies staying on one side of a hyperplane. In our experiments, we approximated the inner circle with a 20-sided polygon, which was sufficient for the robot to stay on the road.

We then compute the trajectory that travels a quarter of the circular road shown in Figure 1b solely by using receding horizon optimization with obstacle avoidance properties as in equation (4). The blue trajectory in Figure 1b shows the trajectory computed by this method, and the green line represents the computed trajectory for the next horizon. The solution to this optimization is found in 20.9898 seconds.

Using our PrSTL-Tree method, we create an RRT as shown in Figure 3a, and use the safe controller synthesis method with PrSTL constraints to connect nodes of this tree. However, the RRT planning will handle remaining within road boundaries and obstacle avoidance as all obstacles and the road boundaries are known. Consequently, all the non-convex properties are handled through RRT without being considered as part of the safe control method. This hierarchical approach removes all the non-convex properties, and significantly speeds up the controller synthesis algorithm. The final trajectory using this approach is shown in green in Figure 3b, and finding this solution took 1.4175 seconds. This is **approximately 15 times faster** than using receding horizon optimization with PrSTL constraints.

4.2 Unknown Obstacles

Although we have shown significant timing improvement for the case of known obstacles, our PrSTL-Tree approach shows its complete power for scenarios that involve uncertainty. Uncertain scenarios, such as noisy range finders or imperfect classifiers allow us to take advantage of the expressibility of PrSTL. For this case study, we consider the same driving scenario as shown in Figure 1 with the difference that all the inner obstacles are not known a priori. This is shown in Figure 4, where the pink obstacles are the unknown obstacles whose location is learned online.

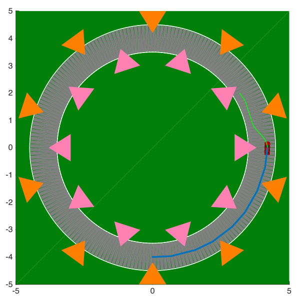
We then use a mesh of points around the robot that act as range finders that can detect obstacles. Using linear Gaussian Processes [Rasmussen and Williams, 2006], we are able to predict if a point in the space is an obstacle or not based on the learned Gaussian distribution. Therefore, the problem of obstacle avoidance translates to probabilistic constraints as follows:

$$\mathbf{G}_{[0,\infty)}\left(Pr(\mathbf{v}\cdot[x \quad y \quad 1]^\top \leq 0) \geq 1-\epsilon\right) \qquad (6)$$

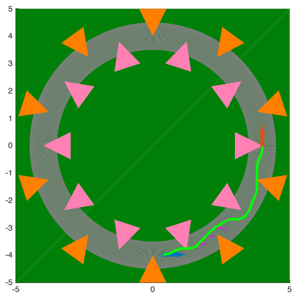
Here, $\mathbf{v}$ is a Gaussian vector learned by linear Gaussian Processes, and (x, y) are the coordinates of the robot. The inner product of $\mathbf{v}$ and $[x \quad y \quad 1]^\top$ represents the current belief of coordinates (x, y) being in an obstacle or not. We would like to enforce that the coordinates are outside of obstacles with high probability $1-\epsilon$ at all times $t \in [0,\infty)$. For our experiments, we chose $\epsilon = 0.5$, which allows an easier fit of a prediction line to triangular shaped obstacles. Although ϵ is large, the resulting trajectories in Figure 4 do not collide with any obstacles.

Note, in this case, the probabilistic constraints can equivalently be written as semi-definite programs which makes the constraints corresponding to uncertain obstacles convex. However, the known obstacles and the boundary conditions of the road are still non-convex constraints. Using the receding horizon safe control technique, we compute the optimal trajectory of traveling a quarter of the circular road in 16.9848 seconds. This is shown in Figure 4a, where the blue line shows the trajectory computed and taken by the robot and the green line is the next horizon's planned trajectory. The computation time for this example is smaller than the same example with known obstacles. Obstacle avoidance under uncertainty results in convex properties, which can help lowering the computation time by reducing the number of disjunctions in the formula.

Using our method of PrSTL-Tree, we were able to find the controller in 2.1339 seconds. This value is again **significantly (approximately 8 times) smaller** than only using the optimization based method with PrSTL constraints. Note, this computation time is larger than the same scenario but with all known obstacles. This is because, the safe control optimization that connects the nodes in the graph has to solve a more complex problem including obstacle avoidance under uncertainty.



(a) Reaching the final state by receding horizon optimization for PrSTL under uncertainty.



(b) Reaching the final state by following PrSTL-Tree under uncertainty.

Figure 4: Comparing receding horizon optimization for safe control on the left and PrSTL-Tree approach on the right for reaching the same final goal while staying safe. Here, the orange triangles represent the known obstacles and the pink ones represent the unknown obstacles.

5 Conclusion

We propose a framework for efficiently computing mission plans that are safe even under uncertain environments. The core idea of the proposed approach is to combine recent techniques in controller synthesis via Probabilistic logical specifications with sampling based mission planners. In particular, the proposed method gets around the computational and semantic limitations of PrSTL by embedding into an RRT* sampling strategy. We demonstrate the framework on the task of autonomous driving and quadrotor scenarios. Future work includes application of this work to other domains including airline flight planning and other robotic tasks.

References

[Aoude *et al.*, 2013] Georges S Aoude, Brandon D Luders, Joshua M Joseph, Nicholas Roy, and Jonathan P How. Probabilistically safe motion planning to avoid dynamic obstacles with uncertain motion patterns. *Autonomous Robots*, 35(1):51–76, 2013.

[Berenson *et al.*, 2009] Dmitry Berenson, Siddhartha S Srinivasa, Dave Ferguson, Alvaro Collet, and James J Kuffner. Manipulation planning with workspace goal regions. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 618–624. IEEE, 2009.

[Cowlagi, 2011] Raghvendra V Cowlagi. *Hierarchical motion planning for autonomous aerial and terrestrial vehicles.* PhD thesis, Georgia Institute of Technology, 2011.

[Dey *et al.*, 2015] Debadeepta Dey, Kumar Shaurya Shankar, Sam Zeng, Rupesh Mehta, M. Talha Agcayazi, Christopher Eriksen, Shreyansh Daftry, Martial Hebert, and J. Andrew Bagnell. Vision and learning for deliberative monocular cluttered flight. *Field and Service Robotics*, 2015.

[Dijkstra, 1959] Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[Hart *et al.*, 1968] Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *Systems Science and Cybernetics, IEEE Transactions on*, 4(2):100–107, 1968.

[Horvitz, 2001] Eric Horvitz. Principles and applications of continual computation. *Artificial Intelligence*, 126(1):159–196, 2001.

[Karaman and Frazzoli, 2008] S Karaman and E Frazzoli. Optimal vehicle routing with metric temporal logic specifications. In *IEEE Conference on Decision and Control*, 2008.

[Karaman and Frazzoli, 2009] Sertac Karaman and Emilio Frazzoli. Sampling-based motion planning with deterministic μ-calculus specifications. In *Decision and Control, 2009 held jointly with the 2009 28th Chinese Control Conference. CDC/CCC 2009. Proceedings of the 48th IEEE Conference on*, pages 2222–2229. IEEE, 2009.

[Karaman and Frazzoli, 2011] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7):846–894, 2011.

[Karaman *et al.*, 2011] Sertac Karaman, Matthew R Walter, Alejandro Perez, Emilio Frazzoli, and Seth Teller. Anytime motion planning using the rrt*. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1478–1483. IEEE, 2011.

[Kavraki *et al.*, 1996] Lydia E Kavraki, Petr Švestka, Jean-Claude Latombe, and Mark H Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *Robotics and Automation, IEEE Transactions on*, 12(4):566–580, 1996.

[Kress-Gazit *et al.*, 2009] Hadas Kress-Gazit, Georgios E Fainekos, and George J Pappas. Temporal-logic-based reactive mission and motion planning. *Robotics, IEEE Transactions on*, 25(6):1370–1381, 2009.

[Kuwata *et al.*, 2008] Yoshiaki Kuwata, Gaston A Fiore, Justin Teo, Emilio Frazzoli, and Jonathan P How. Motion planning for urban driving using rrt. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 1681–1686. IEEE, 2008.

[Lavalle and Kuffner Jr, 2000] Steven M Lavalle and James J Kuffner Jr. Rapidly-exploring random trees: Progress and prospects. In *Algorithmic and Computational Robotics: New Directions*, 2000.

[Li and Todorov, 2004] Weiwei Li and Emanuel Todorov. Iterative linear quadratic regulator design for nonlinear biological movement systems. In *ICINCO (1)*, pages 222–229, 2004.

[Luders *et al.*, 2010] Brandon Luders, Mangal Kothari, and Jonathan P How. Chance constrained rrt for probabilistic robustness to environmental uncertainty. In *AIAA guidance, navigation, and control conference (GNC), Toronto, Canada*, 2010.

[Maler and Nickovic, 2004] Oded Maler and Dejan Nickovic. Monitoring temporal properties of continuous signals. In *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, pages 152–166. Springer, 2004.

[Melchior and Simmons, 2007] Nik A Melchior and Reid Simmons. Particle rrt for path planning with uncertainty. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 1617–1624. IEEE, 2007.

[Papadimitriou and Steiglitz, 1982] Christos H Papadimitriou and Kenneth Steiglitz. *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1982.

[Pivtoraiko and Kelly , 2005] Mikhail Pivtoraiko and Alonzo Kelly . Efficient constrained path planning via search in state lattices. In *The 8th International Symposium on Artificial Intelligence, Robotics and Automation in Space*, September 2005.

[Plaku and Karaman, 2015] Erion Plaku and Sertac Karaman. Motion planning with temporal-logic specifications: Progress and challenges. *AI Communications*, 2015.

[Raman *et al.*, 2014] Vasumathi Raman, Alexandre Donzé, Mehdi Maasoumy, Richard M Murray, Alberto Sangiovanni-Vincentelli, and Sanjit A Seshia. Model predictive control with signal temporal logic specifications. In *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*, pages 81–87. IEEE, 2014.

[Raman *et al.*, 2015] Vasumathi Raman, Alexandre Donzé, Dorsa Sadigh, Richard M Murray, and Sanjit A Seshia. Reactive synthesis from signal temporal logic specifications. In *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*, pages 239–248. ACM, 2015.

[Rasmussen and Williams, 2006] Carl Rasmussen and Chris Williams. Gaussian processes for machine learning. *Gaussian Processes for Machine Learning*, 2006.

[Ratliff *et al.*, 2009] Nathan Ratliff, Matt Zucker, J Andrew Bagnell, and Siddhartha Srinivasa. Chomp: Gradient optimization techniques for efficient motion planning. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 489–494. IEEE, 2009.

[Sadigh and Kapoor, 2015] Dorsa Sadigh and Ashish Kapoor. Safe control under uncertainty. *CoRR*, abs/1510.07313, 2015.

[Tabuada and Pappas, 2004] Paulo Tabuada and George J Pappas. Linear temporal logic control of linear systems. *IEEE Transactions on Automatic Control*, 2004.

[Urmson *et al.*, 2008] Chris Urmson, Joshua Anhalt, Drew Bagnell, Christopher Baker, Robert Bittner, MN Clark, John Dolan, Dave Duggins, Tugrul Galatali, Chris Geyer, et al. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics*, 25(8):425–466, 2008.

[Van Den Berg *et al.*, 2011] Jur Van Den Berg, Sachin Patil, and Ron Alterovitz. Motion planning under uncertainty using differential dynamic programming in belief space. In *Intl Symposium on Robotics Research*, 2011.

[Yoo *et al.*, 2012] Chanyeol Yoo, Robert Fitch, and Salah Sukkarieh. Probabilistic temporal logic for motion planning with resource threshold constraints. 2012.

[Zames, 1981] George Zames. Feedback and optimal sensitivity: Model reference transformations, multiplicative seminorms, and approximate inverses. *Automatic Control, IEEE Transactions on*, 26(2):301–320, 1981.

[Ziegler and Nichols, 1942] John G Ziegler and Nathaniel B Nichols. Optimum settings for automatic controllers. *trans. ASME*, 64(11), 1942.