Sampling-Based Gaussian Estimation of Probability of Collision for Safe Planning

Ajaay Chandrasekaran,
Department of Electrical and Computer Engineering
University of Michigan
Ann Arbor, Michigan 48109, U.S.A.
Email: ajaay@umich.edu

***Abstract*—This is the abstract.**

***Index Terms*—collision probability, motion planning, safety, uncertainty.**

I. Introduction

THe use of robots in elderly care is becoming increasingly likely throughout the world, as the population of elderly people is beginning to overtake the number of potential caregivers.

II. Related Work

III. Estimating Probability of Collision

A. Problem Statement

We consider a robot operating in an environment with obstacles.The motion model of the robot is given as:

$$x_{t+1} = g(x_{t-1}, u_{t-1}, m), m \sim \mathcal{N}(0, R_t)$$

where $x_t \in \mathcal{C}$ is the configuration of the robot at time t, $u_t \in R^{n_u}$ is the applied control input, and m is a zero-mean Gaussian noise variable with variance R_t.

During execution of the motion plan, the robot obtains measurements of surrounding landmarks. The sensor model of the robot is given as:

$$z_t = h(x_t, l, n), n \sim \mathcal{N}(0, Q_t)$$

where $z_t \in R^{n_z}$ is the measurement obtained at time t, $l \in R^{n_l}$ is the location of the landmark that is observed, and n is a zero-mean Gaussian noise variable with variance Q_t.

We assume the existence of a nominal plan computed by a motion planner. The nominal plan is defined by:

$$[x_0^*, u_0^*, ..., x_N^*, u_N^*]$$

where $x_t^* = g(x_{t-1}^*, u_{t-1}^*, 0)$ for $0 < t \leq N$, where N is the number of configurations in the motion plan.

With the preliminary notation established, we express our problem statement. Our goal is to estimate the probability of collision p_c for the nominal plan. More formally, this probability is represented as:

$$p_c = p\left(\bigvee_{t=0}^{N} x_t^* \notin \mathcal{C}_F\right)$$

where $\mathcal{C}_F \subset \mathcal{C}$ represents the space of non-colliding configurations.

B. Formulation and Assumptions

Much of existing literature on this subect assumes that we can represent obstacles in the environment as known constraints on the robot's feasible locations in the workspace. However, we do not make that assumption, and we assume that there instead exists a collision checker $\phi : \mathcal{C} \rightarrow \{TRUE, FALSE\}$, which can determine if a configuration is in collision. If $x \notin \mathcal{C}_F$, $\phi(x) = TRUE$; otherwise $\phi(x) = FALSE$ [1].

The assumption of an available collision checker contrasts our formulation from other literatures, which tend to only consider point robots through the formulation of known robot location constraints based on obstacles. Our formulation lets us consider the possibility of non-point robots, whose locations in the workspace may cause collisions with obstacles due to the robots having 3D-geometries and being situated with complex orientations.

We assume that the Extended Kalman Filter (EKF) localization algorithm [4] is used to compute $\hat{x_t}$, an estimate of the robot's true configuration, x_t, at time t. The covariance of the EKF estimate of x_t is given by Σ_t. We assume that there is a set of already known landmarks in the environment, $L = l_1, ..., l_{n_l}$, where n_l is the number of landmarks, and that data association between sensor measurements and landmarks is known.

Due to the Gaussian noise in the motion model, we expect that the robot will deviate from the nominal motion plan during its execution. To compensate for the motion uncertainty, we assume that the robot operates with a closed-loop feedback controller [3]. We assume that there exists a linear feedback control law that operates on the EKF estimate of the robot configuration and attempts to keep the robot close to the nominal plan. This feedback policy is given as:

$$\bar{u_t} = L_{t+1}(\hat{x_t} - x_t^*)$$

where L_t is the control gain matrix, which is contingent on the choice of feedback controller [3] and $\bar{u_t}$ is the necessary deviation from the nominal control input u_t^* to account for the robot's deviation from the nominal plan. It follows that $u_t = \bar{u_t} + u_t^*$, where u_t represents the true control input to be executed by the robot–given the deviation of the robot's estimated configuration $\hat{x_t}$ from the nominal configuration x_t^*.

C. Sources of Difficulty

Computing the probability of collision for a motion plan in practice is a difficult task. There are three key sources of the difficulty:

1) Uncertainty in robot motion and sensor models
2) Dependent individual events of collision
3) Computational time of Monte-Carlo simulations

We proceed to describe each of these sources.

1) Model Uncertainty: Robot motion is often corrupted by noise. We find that when we command a robot to follow a nominal control input, it often does not follow the command exactly due to various factors–usually wheel slippage due to lack of uniformity of surfaces.

Furthermore, sensor measurements that are used to localize a robot are also corrupted by natural noise and innaccuracy. Using sensor measurements for localization naturally causes uncertainty in the robot's known position. When using a state estimate as an input to a feedback control loop to correct the robot's position, there is a subsequent uncertainty in the accuracy of the desired controller adjustment–despite the desired intention of following the nominal motion plan.

2) Dependence between Collision Events: A key feature of this problem is that the event of a collision not occurring at a configuration x_t of a motion plan is not independent of the event of a collision not occurring at previous configurations of the motion plan. More formally:

$$p(x_t \in \mathcal{C}_F) \neq p\left(x_t \in \mathcal{C}_F | \bigwedge_{i=0}^{t-1} x_i \in \mathcal{C}_F\right)$$

Prior approaches to solving this problem cite here often consider the events of collisions along the motion plan to be independent of each other, which can lead to an overly conservative estimate of the probability of collision for a motion plan. Incorporating the fact that the collision events (and non-collision events) are dependent on each other increases the difficultly of computing an accurate estimate of the probability of collision.

3) Computational Time of Monte-Carlo Simulations: Running a large number of Monte-Carlo simulation, each having a large number of particles, can–in theory–give the true probability of collision for a motion plan. The idea of a Monte-Carlo simulation is to simulate the motion of thousands of particles, each particle representing a potential configuration of a robot's configuration given uncertainty in the robot's true configuration x_t. By finding the proportion of particles that did not collide with obstacles at all during the entirety of a motion plan, we can have an estimate of the probability of collision.

However, a single Monte-Carlo simulation is generally not sufficient, and we would need to average the proportions of colliding-particles across hundreds or even thousands of Monte-Carlo simulations to obtain an accurate estimate of the true probability of collision for the motion plan.

Furthermore, running a single Monte-Carlo simulation with thousands of particles can be very computationally expensive due to the compounding of time required for collision checking. Running thousands of Monte-Carlo simulations compounds the computation time even further, causing this approach to be a poor idea in practice.

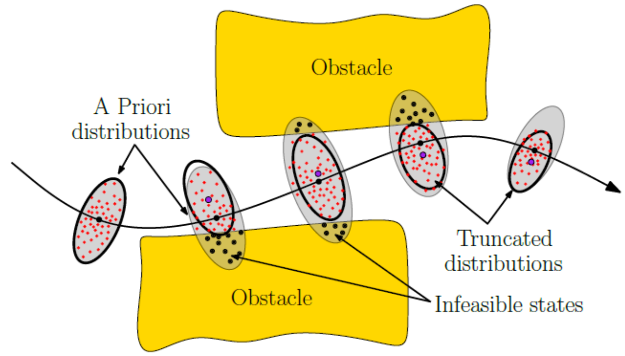

Fig. 1. Propagation and Truncation of State Estimate Distribution. Figure borrowed from Patil, van den Berg, Alterovitz 2012. "The probabilities of collision at each stage of the plan are conditioned on the previous stages being collision free. We truncate a priori distributions with respect to obstacles to discount plan executions that collide with obstacles (black disks). Propagating the truncated distributions (black ellipses) accounts for only the collision free samples (red disks), resulting in accurate estimation of the probability of collision. Using the unconditional distributions (gray ellipses) to estimate the collision probability results in a very conservative estimate."[2]

D. Metric Evaluation

Given the sources of difficulty for this problem, we primarily strive to compute an accurate estimate of the true probability of collision for a motion plan. We also seek to compute the probability of collision for the plan in an amount of time that is approximately less than or equal to the time required for a single Monte-Carlo simulation while within the error bounds of the average proportion of colliding particles that would be computed from executing hundreds of Monte-Carlo simulations with many particles in each simulation instance.

IV. Method

We propose a sampling-based method paired with the use of a Gaussian Mixture Model (GMM) to estimate the probability of collision for a motion plan. Our method adopts ideas from the the approach proposed by Patil, van den Berg, and Alterovitz [2]. However, our approach differs in that we do not assume that the environment obstacles can be represented as constraints on the robot location in the workspace, and we assume that a collision checker exists instead.

Figure 1 portrays the approach recognized in [2], which estimates the probability of collision for a motion plan by propagating a Kalman Filter (KF) estimate of a robot's state and truncating the distribution of the estimate with respect to obstacles. Patil et al. make the observation that the distribution of the robot's configuration at a timestep t–conditioned on the configuration $x_i \notin \mathcal{C}_F$, $\forall i, 0 \leq i < t$ being collision-free, can be represented by truncating the KF estimate at x_t with respect to the obstacles. They also note that the proportion of the distribution that has been truncated by the obstacles would represent the conditional probability of collision for that state of the motion plan.

The approach of Patil et al. is purely analytical. It truncates the KF estimate of the robot state by sequentially applying each obstacle constraint to truncate the Gaussian distribution of the state estimate using analytical techniques from existing literature. They assume that the truncation of a Gaussian distribution by the obstacle constraints subsequently yields another Gaussian distribution.

The approach of Patil et al. makes two more important assumptions that are worth noting when they truncate the KF estimate with respect to obstacles:

1) The obstacles in the environment can be expressed as linear constraints on the robot's location in the workspace
2) The free region containing the robot's configuration at time t is convex

We base our approach on the framework of [2], and attempt to release these assumptions as we define our sampling-based GMM approach to estimate the probability of collision.

Algorithm 1 GMM Estimation of Probability of Collision

Input: $[x_0^*, u_0^*, ..., x_N^*, u_N^*]$
Output: p_c

```
INIT_GMM
if some condition is true then
    do some processing
else if some other condition is true then
    do some different processing
else if some even more bizarre condition is met then
    do something else
else
    do the default actions
end if=0
```

Our approach

V. Results

VI. Discussion

VII. Conclusion

Acknowledgment

The author would like to acknowledge Dmitry Berenson, Maani Ghaffari, and Valerie Chen for their helpful feedback.

References

[1] S. LaValle. *Planning Algorithms*. Cambridge University Press, 2006.
[2] S. Patil, J. van den Berg, and R. Alterovitz, Estimating probability of collision for safe planning under Gaussian motion and sensing uncertainty, in ICRA, 2012, pp. 3238-3244.
[3] R. F. Stengel, *Optimal Control and Estimation*. Dover Publications, 1994.
[4] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, 2005.

Ajaay Chandrasekaran received the BSE degree in computer science from the University of Michigan, Ann Arbor, MI, in 2017. He is currently pursuing the MSE degree in electrical and computer engineering from the University of Michigan.

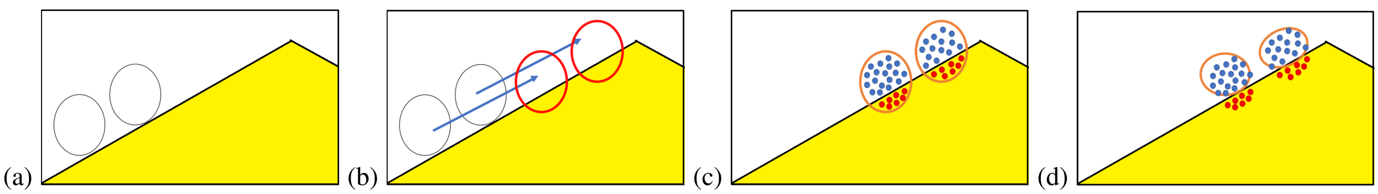

Fig. 2. Sampling-based GMM Algorithm applied to Mixture of 2 Gaussians. (a) Initialize mixture components with same initial mean x_0^* and covariance Σ_0, as well as equal weights (Means shown as different here to highlight distinction). Gaussians are represented by blue ellipses. (b) Propagate Gaussians in mixture to next state of motion plan via EKF prediction step. Updated Gaussians are represented by red ellipses. (c) Update Gaussians via EKF correction step based on measurements (orange ellipses). Then, sample N_p configurations from GMM based on weights. Check which configurations are in collision. Red dots represent colliding configurations and blue dots represent collision-free configurations. (d) Update mean and covariance of each Gaussian analytically based on the set of collision-free configurations contained in it. Update each mixture component's weight to be the proportion of the total non-colliding configurations that it contains.