Chance-Constrained Motion Planning using Modeled Distance-to-Collision Functions

Jacob J. Johnson¹, IEEE Student Member and Michael C. Yip¹, IEEE Senior Member

Abstract—This paper introduces Chance Constrained Gaussian Process-Motion Planning (CCGP-MP), a motion planning algorithm for robotic systems under motion and state estimate uncertainties. The paper’s key idea is to capture the variations in the distance-to-collision measurements caused by the uncertainty in state estimation techniques using a Gaussian Process (GP) model. We formulate the planning problem as a chance constraint problem and propose a deterministic constraint that uses the modeled distance function to verify the chance-constraints. We apply Simplicial Homology Global Optimization (SHGO) approach to find the global minimum of the deterministic constraint function along the trajectory and use the minimum value to verify the chance-constraints. Under this formulation, we can show that the optimization function is smooth under certain conditions and that SHGO converges to the global minimum. Therefore, CCGP-MP will always guarantee that all points on a planned trajectory satisfy the given chance-constraints. The experiments in this paper show that CCGP-MP can generate paths that reduce collisions and meet optimality criteria under motion and state uncertainties. The implementation of our robot models and path planning algorithm can be found on GitHub.

I. INTRODUCTION

In the past few decades, a profusion of work has focused on the motion planning problem for an assortment of tasks such as car navigation around obstacles [1]–[3], constrained robotic manipulation [4], [5], and surgical robot automation [6]. However, most motion planning research has focused on demonstrating examples where environments are highly structured, and uncertainties in sensing are overlooked. In reality, robots in the real world will face different sources of uncertainties: 1. errors in system model and sensor measurements, 2. ambiguity in the position of obstacles in the space, and 3. varying physical properties of the environment itself. Motion planning algorithms that consider the collision probability, i.e., chance constraints [7], perform better than previous methods in such unstructured environments [8], [9].

Previous works on planning under uncertainty have not guaranteed that states along a given trajectory are collision-free but rather verify that discrete states satisfy the chance constraints. Furthermore, they bound the obstacles and the robot to make the optimization tractable, making the probabilistic estimates overly conservative, leading to winding trajectories. Finally, estimating collision probabilities utilizing Monte Carlo methods is computationally expensive since the shortest distance from the robot to the obstacle, i.e., distance-to-collision, has to be evaluated for a large number of samples.

Ultimately, our goal is to find optimal trajectories that continuously meet chance constraints along an entire trajectory. To this end, we propose the Chance Constrained Gaussian Process-Motion Planning algorithm (CCGP-MP) that addresses those issues. We use a Gaussian Process (GP) to model the distribution of distance-to-collision measures for noisy robotic systems. In turn, we integrate this model with traditional sampling-based planners to generate trajectories that satisfy a given chance constraint. Our formulation ensures both the sampled states in a path satisfy the chance-constraint as well as all the points that lie along the edges connecting the sampled states. Thus, the main contributions of this paper are:

1) Propose a GP model to capture the continuous, probabilistic distribution of functions describing distance-to-collision for a stochastic system
2) Employ a global optimization technique to verify collision constraints along a given path without discretizing the states along the path.
3) Generate low-risk paths for systems with motion and sensor noise using sampling-based planners.

II. RELATED WORKS

Many existing planning methods are based on using collision probability for planning under uncertainty [8]–[13], while
other solutions rely on Markov Decision Process (MDP) [14] or Partially Observable MDPs (POMDPs) [15]. MDP and POMDP often need discretization of the state space, and solving an MDP can quickly become computationally intractable for continuous planning domains. In the following section, we will review a few of the works on estimating collision probability for planning and recent techniques used for distance estimation.

In [8], the authors find a path by formulating the planning problem as an optimization problem where the planned states have to satisfy user-defined chance-constraints while minimizing a cost. Further development of this algorithm [9] ensured that inter-node trajectories satisfied the chance-constraint but only for obstacles represented as linear functions. In [11], the authors propose tighter bounds over ellipsoidal obstacles. For these methods, the number of constraints to solve increases linearly with the number of obstacles, and for higher dimensions, the number of constraints grows exponentially. Using a GP model to capture the distance-to-collision function, we avoid the need to convexify the environment and robot, and irrespective of the environment’s complexity, a single equation represents the chance constraints.

Another class of methods estimates the probability of collision along a path by obtaining the robot states’ distribution and choosing one with the least likelihood of a collision. Linear-Quadratic Gaussian Motion Planning (LQG-MP) [10] method derives a distribution for the states along a path associated with using a Linear-Quadratic Gaussian (LQG) controller to stabilize the robot. Although this method can obtain a trajectory that reduces the probability of collision, as suggested in [16], there is no guarantee that LQG-MP may find a path because of the finite number of paths generated by RRT. In [17], the authors propose linear constraints on the distribution of states to obtain a tighter collision probability. [18] extends the LQG-MP for higher DOF robots, but like LQG-MP, the method does not have a user-defined chance constraint parameter. All these methods guarantee safety for discrete states but are intractable to verify safety for all points on a trajectory.

In [12], [13], the authors use Monte-Carlo simulations to get a more accurate distribution of states and a more precise collision probability estimate. In [12], the authors use importance sampling to be more data-efficient in their simulation and reduces the variance of estimates using control variate. [13] extends this work to non-linear systems for verification of trajectory in an online planning setting. Although these methods estimate collision probability along a path precisely, they rely on meta planning algorithms to generate an initial path and, as such, cannot incorporate additional optimality criteria into the planning problem. CCGP-MP integrates with optimal planners such as RRT* to solve planning problems with optimality criteria.

Many methods are proposed that use geometric sensors and modeling techniques to estimate the distance-to-collision [19], [20], and for a brief review, readers can refer to [21], but none considers the measurement models in unstructured environments. Das and Yip [21] proposed one of the first techniques that included uncertainties to the distance measure model. The authors added Gaussian noise to the distance measure but did not consider the effect of state estimation uncertainty while modeling the distribution.

III. CCGP-MOTION PLANNING

In this section, we define our problem and the assumption we make and introduce the building blocks of CCGP-MP.

A. Problem Definition

Let the state, control, and observation space be defined as $\mathcal{X} \subseteq \mathbb{R}^{n_x}$, $\mathcal{U} \subseteq \mathbb{R}^{n_u}$, and $\mathcal{Z} \subseteq \mathbb{R}^{n_z}$ respectively. For a given start position ($x_{start} \in \mathcal{X}$) and goal region ($\mathcal{X}_{goal} \subseteq \mathcal{X}$), the objective is to find a trajectory that satisfies a user-defined collision constraint. A trajectory $\Pi$, defined as a sequence of states $\{x_0, x_1, \ldots, x_N\}$, is considered a solution if $x_0 = x_{start}$, $x_n \in \mathcal{X}_{goal}$, and all the points that connect state $x_i$ and $x_{i+1}$ also satisfy the given collision chance constraint. We assume that for planning the states are sampled in a subspace $\mathcal{X} \subset \mathcal{X}$, where the system’s velocities are zero. The problem can be further expanded by enforcing optimality criteria to the sequence of states, such as reducing path length. In the subsequent sections, we detail our solution for this problem.

B. Motion and Observation Model

Throughout this paper, we will suppose that we have a nonlinear dynamics and observation model given by:

$$x_{t+1} = f(x_t, u_t) + v(m_t) \quad m_t \sim \mathcal{N}(0, M) \quad (1a)$$

$$z_{t+1} = h(x_t) + n_t \quad n_t \sim \mathcal{N}(0, N) \quad (1b)$$

where $x_t, x_{t+1} \in \mathcal{X}$, $u_t \in \mathcal{U}$, $z_t \in \mathcal{Z}$, $m_t$ and $n_t$ are the process noise sampled from a Gaussian distribution with variance $M \in \mathbb{R}^{n_x \times n_x}$ and $N \in \mathbb{R}^{n_z \times n_z}$ respectively, and $v_t$ additve noise to the model motion at time $t$. Standard filter techniques are used to keep track of the state estimate $\hat{x}_t$ of the true state $x_t$. In [22], the authors show that under certain conditions, for a system given by $[\Pi]$, an LQG controller can drive the system to any point $x \in \mathcal{X}$ starting from any Gaussian distribution. The authors also show that the estimated distribution of states converges to a unique deterministic stationary covariance. We use such a controller for trajectory tracking.

C. Gaussian Process Distance Model

The shortest distance to a collision is modeled probabilistically over the entire state-space using a GP. Given a model of the environment, a GP is constructed from sampled data by randomly moving the robot model around in the environment model and searching for the distance-to-collision using a geometric method for each estimated state. In a similar fashion to [21], the distance-to-collision is evaluated using the Gilbert–Johnson–Keerthi (GJK) method, though any available geometric method can be used in practice.

Given a prior number of samples $\mathcal{N}$, the set of states $\mathcal{X} = \{x_1, x_2, \ldots, x_N\}$, and the corresponding distance-to-collision from the estimated states, $d = \{d_1, d_2, \ldots, d_N\}$,
are used to model the GP. Note that the subscript does not represent a sequence in time, rather a random mix of samples from various trajectories. This data captures the variation in distance-to-collision due to uncertainty in the motion and observation model (See Fig. 2 for an example of a trajectory in X and corresponding distance-to-collision distribution). Given data points X and associated distance measure d, for a state x∗ the distribution of distance-to-collision, d∗, is given by:

\[
\begin{align*}
d^* & \sim N(\mathbb{E}[d^*], \mathbb{V}[d^*]) \\
\mathbb{E}[d^*] & = \mathcal{K}(x^*, X)[\mathcal{K}(X, X) + \sigma^2 I]^{-1} d \\
\mathbb{V}[d^*] & = \mathcal{K}(x^*, x^*) \\
& - \mathcal{K}(x^*, X)[\mathcal{K}(X, X) + \sigma^2 I]^{-1} \mathcal{K}(X, x^*)
\end{align*}
\]

where \(\mathcal{K}\) is a stationary kernel function and \(\sigma^2\) represents the variance of the observed distance measure. For the sake of brevity, we represent the vector \(\mathcal{K}(x^*, X)\) as \(k(x^*)\), the matrix \(\mathcal{K}(X, X)\) as \(K\), and the scalar \(\mathcal{K}(x^*, x^*)\) as \(k^*\). The kernel function \(\mathcal{K}\) belongs to the class of covariance functions [23, Chapter 4] and is chosen based on the application. A Radial Basis Function (RBF) kernel is popular in most applications, though [24] demonstrates that a forward kinematics kernel provides sparser and more accurate models for robot manipulators.

D. Chance Constraints

The collision probability constraint for the state, \(x^* \in \mathcal{X}\), of the robot can be represented using the distance measure \(d^*\), where

\[
\mathcal{P}(x^* \text{ is in collision}) \leq \delta \iff \mathcal{P}(d^* < 0) \leq \delta
\]

This probabilistic constraint can be converted to a deterministic constraint as given in [8], where

\[
\mathcal{P}(d^* < 0) \leq \delta \iff \frac{\mathbb{E}[d^*]}{\sqrt{2\mathbb{V}[d^*]}} \geq c
\]

\[
c = \text{erf}^{-1}(1 - 2\delta)
\]

and where \(\text{erf}(z)\) is the Gaussian error function. The ratio of mean to standard deviation in (6) is defined as function \(g\) given by:

\[
g(x^*) \triangleq \frac{k(x^*)^T[K + \sigma^2 I]^{-1} d}{2(k^* - k(x^*))^T[K + \sigma^2 I]^{-1} k(x^*))^{1/2}}
\]

E. CONNECT Function

In sampling-based planners, which include popular approaches such as the many variations of Rapidly Exploring Random Trees (RRTs) and Probabilistic Roadmaps (PRMs), there exists a function that verifies if an edge can connect two nodes by checking if the points on the edge satisfy a set of constraints. In our work, we are calling these functions CONNECT functions. In traditional planners, the CONNECT function checks for collision by subsampling the edge and evaluating if each point is collision-free. For probabilistic planners, the CONNECT function needs to ensure that all the points on the edge satisfy the chance constraints. To verify if an edge from state \(x_1\) to state \(x_2\) meets the given constraints, the condition defined in (5) must hold for all points on the edge. We can verify this by checking if the global minimum of (8) satisfies the constraint from (6) for the path segment:

\[
\hat{c} \geq c, \quad \hat{c} = \inf_{x^* \in (x_1, x_2)} g(x^*)
\]

and \(s(x_1, x_2)\) represents the trajectory between \(x_1\) and \(x_2\).

F. Simplicial Homology Global Optimization

To find the global minima of the function \(g(x^*)\) we use the Simplicial Homology Global Optimization (SHGO) algorithm as proposed in [25]. SHGO is a global optimization technique that exploits the objective function’s topography to identify sub-domains where the global minimum may lie.

The method samples the objective function by a predetermined number of samples and constructs a simplicial complex \(\mathcal{H}\). The simplicial complex \(\mathcal{H}\) can be conceived as a directed graph, where the vertices represent the value of the objective function at the sampled points and the directed edges point towards the vertex with a higher objective value. In [28], a vertex \(v_i\) is defined as a local minimizer if all the edges connected to \(v_i\) are directed away. A minimizer set, \(\mathcal{M}\), is formed with all such local minimizers. \(\text{st}(v_i)\) defines a new space called the star of a vertex \(v_i\) as the set of points \(Q\) such that every simplex containing \(Q\) contains \(v_i\). The global minimum is found by searching through each sub-domain, \(\text{st}(v_i)\), for all \(v_i \in \mathcal{M}\). The authors prove that the cardinality of \(\mathcal{M}\) remains unchanged with increasing samples, i.e., the number of regions to search for the global minimum does not change with increasing samples. The following theorem guarantees a stationary point in each of these sub-domains:

Theorem 1: Given a minimizer \(v_i \in \mathcal{M} \subseteq \mathcal{H}\) on the surface of a continuous, Lipschitz smooth objective function \(f\) with a compact bounded domain in \(\mathbb{R}^n\) and range \(\mathbb{R}\), there exist at least one stationary point of \(f\) within the domain defined by \(\text{st}(v_i)\) [25].

Thus if the objective function is Lipschitz smooth and an adequate number of samples is given, SHGO is able to converge to the global minimum.
In our situation, to use SHGO to identify the global minimum, we show that (8) is Lipschitz smooth. (8) can be re-written as a composition of two functions, \(q(x^*) = q \circ k(x^*)\). For simplicity, we assume \(k^*\) to be 1. First, we show that the function \(q\) is Lipschitz continuous.

**Lemma 1:** For \(k \in [0, 1]^n\), \(K \geq 0\) and \(\sigma > 0\), the function \(q(k)\) given by

\[
q(k) = \frac{k^T(\sigma^2 I + K)^{-1}d}{(2(1-k^T(\sigma^2 I + K)^{-1}k))^1/2}
\]

satisfies the Lipschitz condition:

\[
L_q = \frac{1}{\sqrt{2}} \frac{\lambda_{\max}}{1 - \lambda_{\max}} \quad \text{and} \quad \lambda_{\max} \text{ is the largest eigenvalue of } (I\sigma^2 + K)^{-1}
\]

where \(\lambda_{\max}\) is the largest eigenvalue of \((I\sigma^2 + K)^{-1}\), and \(k_1, k_2 \in [0, 1]^n\).

**Proof:** For \(k_1, k_2 \in [0, 1]^n\), we can write \(\|q(k_1) - q(k_2)\|\) as:

\[
\|q(k_1) - q(k_2)\| = \frac{\left|\frac{k_1^T(\sigma^2 I + K)^{-1}d}{(2(1-k_1^T(\sigma^2 I + K)^{-1}k_1)))^{1/2}} - \left(\frac{k_2^T(\sigma^2 I + K)^{-1}d}{(2(1-k_2^T(\sigma^2 I + K)^{-1}k_2)))^{1/2}}\right)\right|}{\sqrt{2}}
\]

Let \(M = (\sigma^2 I + K)^{-1}\). Since \(K\) is a Gram matrix of a covariance function, the matrix \(M\) is positive definite and symmetric [23, Chapter 4]. Hence we can simplify (13) as:

\[
\|q(k_1) - q(k_2)\| \leq \frac{|k_1^T(\sigma^2 I + K)^{-1}d - k_2^T(\sigma^2 I + K)^{-1}d|}{\sqrt{2} \lambda_{\max}}
\]

where \(\lambda_{\max}\) is the largest eigenvalue of \((I\sigma^2 + K)^{-1}\), and \(k_1, k_2 \in [0, 1]^n\).

**Using Lemma 1** we can show that (11) is Lipschitz continuous for a Lipschitz continuous kernel function \(k\).

**Theorem 2:** For a Lipschitz continuous kernel \(k\), (8) satisfies the Lipschitz condition.

\[
\|q \circ k(x_1) - q \circ k(x_2)\| \leq L_q \|x_1 - x_2\| \quad \text{(14)}
\]

where \(L_q\) is the Lipschitz constant for the kernel function.

**IV. EXPERIMENTS AND RESULTS**

To evaluate the CCGP-MP technique’s performance, we tested it on two noisy robot models - a Linear and a Dubins Car model. We explored the planner’s performance for 200 random start and goal pairs for different \(\delta\) values on randomly generated environments of blocks and circles. Next, to investigate the effects of the increased number of obstacles in the environment on the planning time and accuracy, we evaluated CCGP-MP for the Linear system on 6 randomly generated environments for 10 random start and goal pairs. In addition to these simulated test environments, we assessed the Dubins Car model in a realistic indoor environment taken from the Gibson Environment suite [26].

In our experiments, we report the \(\delta\) values used as percentages, since from our definition in (5), it represents the upper bound of the probability measure of a state in a collision. For each robot, 2000 state-distance pairs sampled randomly in the environment, and a RBF kernel are used to define the GP model. To measure the distance to a collision for sampled states, GJK was used over the map. Each planned path’s performance was evaluated for 100 trials using a Linear Quadratic Regulator (LQR) based trajectory following controller. A trial was concluded to be successful if the robot could reach the goal region without colliding with any obstacles. We used the Open Motion Planning Library.
Fig. 3: We compare the success rate and path length of the planned paths for the Linear model. Left: The quartile plot compares the success rate for a planning problem without any optimality criteria. Center: The quartile plot compares the success rate for a planning problem with added optimality criteria for reducing path length. Right: The quartile plot compares the length for the same set of start and goal pairs for different $\delta$ values. The plans generated by CCGP-MP and CCGP-MP* are robust to motion and sensor noise, and CCGP-MP* generates shorter paths than LQG-MP.

Fig. 4: The top row shows the plans generated for different start and goal pairs for the Linear model, while the bottom row does the same for the Dubins Car model. The goals are marked using a black circle for the Linear model and a black arrow for the Dubins Car model. CCGP-MP* deviates from the RRT* planner to satisfy chance constraints.

A. Linear Model

The first system we tested was a simple 2D model (see Fig. 4 top row). As the robot is symmetric about its base, we plan in the $\mathbb{R}^2$ space. The discrete-time dynamics model of the system is given by:

$$f(x, u) = x + u + m, \quad m \sim N(0, 0.1I) \quad (15a)$$
$$z = f(x, u) + n \quad n \sim N(0, 0.01I) \quad (15b)$$

The LQG controller used a Kalman Filter for state estimation and an infinite horizon LQR for generating the control signal.

B. Dubins Model

We also tested CCGP-MP on a Dubins Car model whose dynamics is described by:

$$f(x, u) = x + \begin{bmatrix} 21 (-\sin(\theta) + \sin(\theta + \tau u)) \tau \\ 22 (\cos(\theta) - \cos(\theta + \tau u)) \tau \\ u \tau \end{bmatrix} \quad (16a)$$

where $\tau$ is the time step, and $v$ is the linear velocity of the car. The state $x = [x \ y \ \theta]$ and control $u = \omega$ where $(x, y)$ represents the position, $\theta$ the orientation, and $\omega$ the robot’s angular velocity. The noisy motion model is implemented as described in [28] with linear and angular velocity noise sampled from from $N(0, 0.1)$, and rotation noise sampled from $N(0, 5^\circ)$ . The boundary value problem of connecting the sampled state was solved using Dubins curves. An Extended Kalman Filter was used to estimate the robot state, and similar to [22], and a time-varying LQG controller was used to track the trajectory.

C. Experiment Results

We compared CCGP-MP against the RRT planner and the Linear Quadratic Gaussian-Motion Planning (LQG-MP) algorithm for the Linear system in a simulated environment. Fig. 3 (center) compares the planner’s performance with an added optimality criteria of finding the path with the shortest
The CCGP-MP and CCGP-MP* planner have an equivalent or lower standard deviation of success-rate than LQG-MP for lower thresholds. The better performance for the CCGP methods is because they ensure all intermediate points on a path satisfy the chance-constraint while LQG-MP does not have such guarantees.

Fig. [3] (right) reports the length of paths generated by the different planners for the same set of start and goal pairs. The paths generated by CCGP-MP* are shorter compared to the paths generated by LQG-MP. CCGP-MP* is able to generate optimal paths because the underlying planner of CCGP-MP* uses the RRT* algorithm, which is an asymptotically optimal planner [29] that makes no assumptions on the \texttt{CONNECT} function. Fig. 4 plots the different plans generated by RRT*, LQG-MP, and CCGP-MP* for a random start and goal point. From the image, we may infer that CCGP-MP* deviates from the RRT* plan where the chance constraints are not met, which results in better accuracy for these paths. In comparison, the paths from RRT*, although shorter, would result in multiple failures during execution because of the noisy robot motion. The paths from LQG-MP, on the other hand, although safer, are not optimal.

For the Dubins Car model, we compared CCGP-MP* with RRT*. We did not consider the LQG-MP algorithm for this experiment since it does not explicitly solve the shortest path problem. As expected, CCGP-MP* has a much lower standard deviation of the success rate than RRT* (See Fig. 5 (right)). Fig. 5 reveals the reason for this improvement. The path planned by CCGP-MP* avoids the obstacles, even with the noisy robot model, while for the RRT* plan, the robot hits the obstacle. In Fig. 4 we compare the paths generated by CCGP-MP* and RRT*, and like the Linear model, the CCGP-MP* deviates from the RRT* plan when chance-constraints are not met.

The results of the study on environment density on planning time and accuracy are summarized in Table I. The start and goal pairs were sampled from independent normal distributions with fixed means and 0.5 standard deviations. This distribution was fixed for all the environments. Apart from planning time, we also recorded the time taken by SHGO to find the global minimum for 50 random path segments. The GP model used for each environment had an equal number of support points, resulting in almost similar edge evaluation times. The overall planning time increases with the number of obstacles in space since the planner has to search more to find a feasible solution. We also observed that path accuracy was inversely proportional to the number of objects. The drop in accuracy could be attributed to the fact that more path segments are closer to the set threshold with a denser environment, thus decreasing the overall accuracy of the path.

In addition to the simulated environment, we tested our planner on an indoor environment from the Gibson suite [26]. Fig. 6 (left) shows the trajectory generated by the RRT* and CCGP-MP* (5%) for a single start and goal pair. Fig. 6 (right) shows the minimum distance to collision for each trajectory evaluated across 500 runs. For RRT, 16.4% of the trajectories have the distance-to-collision less than zero, while for CCGP-MP, only 2.4% of the trajectories have distance-to-collision less than zero. The value for CCGP-MP* also satisfies the delta threshold set for the planner, which is 0.05.

V. CONCLUSION

In this work, we introduced the Chance Constrained Gaussian Process Motion Planning, a chance-constrained motion planning approach that uses modeled distance-to-collision
functions to plan in unstructured environments. Through the modeled distribution function, the planner guarantees that all states along the trajectory satisfy the given chance constraints. Simulation results on two robot systems showed that CCGP-MP and CCGP-MP* were able to generate paths that improved the planned path’s success rate.

One of the few limitations of our work lies in approximating distance-to-collision distribution as a Gaussian distribution. This simplification may not apply to some robotic systems. Another limitation centers around the scaling up of planning space. For larger maps, we require more support points to model our GP. For a large number of points (>10000), the kernel matrix requires a considerable amount of memory [23, Chapter 8], and the linear equations that need solving for inference becomes computationally intensive. One way to overcome these limitations is to use sparse GP models. One could even construct local GP models for larger maps similar to [30].

There are multiple directions to be investigated further for the current work. One of them is extending the models to high DoF robotic systems. Rather than using the RBF kernel, the forward kernel (FK) [24] would capture the distance function better. Another interesting avenue to investigate would be using a heteroscedastic GP to model the distance function, which might be more appropriate for time-varying robotic systems.

APPENDIX

For \( x \in [0, 1] \), the Taylor series expansion of \( \frac{1}{(1-ax)^{\frac{1}{2}}} \) about 0 is given by:

\[
\frac{1}{(1-ax)^{\frac{1}{2}}} = 1 + \sum_{m=1}^{\infty} \frac{(ax)^m}{m!} \prod_{j=0}^{m-1} \left( \frac{1}{2} + j \right) \tag{17}
\]

Using simple calculus we can show that:

\[
1 + \sum_{m=1}^{\infty} \frac{a^m}{m!} \prod_{j=0}^{m-1} \left( \frac{1}{2} + j \right) = \frac{1}{(1-a)^{\frac{1}{2}}} \tag{18}
\]

\[
\sum_{m=1}^{\infty} \frac{ma^m}{m!} \prod_{j=0}^{m-1} \left( \frac{1}{2} + j \right) = \frac{a}{2(1-a)^{\frac{1}{2}}} \tag{19}
\]

**Lemma 2:** For \( x_1, x_2 \in [0, 1]^n \) and a postive definite symmetric matrix \( M \) we have,

\[
|x_1^T M x_1 - x_2^T M x_2| \leq 2\lambda_{max} \sqrt{n} \| x_1 - x_2 \| \tag{20}
\]

where \( \lambda_{max} \) is the maximum eigenvalue of \( M \).

**Proof:** Since \( M \) is symmetric, we can expand [20] as

\[
|x_1^T M x_1 - x_2^T M x_2| = |(x_1 - x_2)^T M (x_1 + x_2)| \\
\leq \lambda_{max} \| x_1 - x_2 \| \| x_1 + x_2 \| \\
\leq 2\lambda_{max} \| x_1 - x_2 \| \sqrt{n}
\]

**Lemma 3:** For \( x_1, x_2 \in [0, 1]^n \), a symmetric positive definite matrix \( M \) and \( m \in \mathbb{N} \) we have,

\[
\|(x_1^T M x_1)^m - (x_2^T M x_2)^m\| \leq \frac{2}{\sqrt{n}} m(n\lambda_{max})^m \| x_1 - x_2 \| \tag{21}
\]
where $\lambda_{\text{max}}$ is the maximum eigen value of $M$.

Proof: The polynomial equation $x^m - y^n$ can be expressed as follows:

\[ x^m - y^n = (x - y)(x^{m-1} + x^{m-2}y + \ldots + y^{m-1}) \quad (22) \]

Let $x_i^TMx_i = \|x_i\|_M^2$ for $i \in \{0, 1\}$ we can express (21) in the same fashion as above,

\[
\|x_1\|_M^2 - \|x_2\|_M^2 = \|\lambda_1\|_M - \|\lambda_2\|_M - \sum_{j=3}^{m} \|\lambda_j\|_M \quad (23)
\]

Since $x_i \in [0, 1]^n$ we can write $\|x_i\|_M \leq \lambda_{\text{max}}$ for $i \in \{0, 1\}$, where $\lambda_{\text{max}}$ is the largest eigen value of $M$. Using this bound we can simplify (23) as follows:

\[ \|x_1\|_M^2 - \|x_2\|_M^2 \leq 2\sqrt{n}\lambda_{\text{max}}\|x_1 - x_2\|_M^2 \quad (\text{from Lemma 2}) \]

\[ \|x_i\|_M^2 - \|x_j\|_M^2 \leq 2\sqrt{n}\lambda_{\text{max}}\|x_i - x_j\|_M^2 \]

Lemma 4: For $x_1, x_2 \in [0, 1]^n$, a symmetric positive definite matrix $M$ and $m \in \mathbb{N}$ we have:

\[ \|x_1^TMx_1 - (x_2^TMx_2)\|_M^2 \leq (1+2m)(\lambda_{\text{max}})^m\|x_1 - x_2\|_M^2 \quad (24) \]

where $\lambda_{\text{max}}$ is the largest eigenvalue of $M$.

Proof: We can simplify (24) using Lemma 3:

\[ \|x_1\|_M^2 - \|x_2\|_M^2 \leq \|\lambda_1\|_M - \|\lambda_2\|_M - \sum_{j=3}^{m} \|\lambda_j\|_M \]

\[ \|x_1\|_M^2 - \|x_2\|_M^2 \leq (1+2m)(\lambda_{\text{max}})^m\|x_1 - x_2\|_M^2 \quad (\text{from Lemma 5}) \]

\[ \|x_i\|_M^2 - \|x_j\|_M^2 \leq (1+2m)(\lambda_{\text{max}})^m\|x_i - x_j\|_M^2 \]

ACKNOWLEDGMENT

We thank Vikas Dhiman, Florian Richter, and Nikhil Das for insightful discussions.

REFERENCES

[1] C. Rösmann, F. Hoffmann, and T. Bertram, “Kinodynamic trajectory optimization and control for car-like robots,” in 2017 IEEE/RSJ Int. Conf. Intell. Robots and Systems, 2017, pp. 5681–5686.

[2] J. J. Johnson, L. Li, F. Liu, A. H. Qureshi, and M. C. Yip, “Dynamically constrained motion planning networks for non-holonomic robots,” in 2020 IEEE/RSJ Int. Conf. Intell. Robots and Systems, 2020, pp. 6937–6943.

[3] L. Li, Y. Miao, A. H. Qureshi, and M. C. Yip, “MPC-MPNet: Model-predictive motion planning networks for fast, near-optimal planning under kinodynamic constraints,” 2021, arXiv:2101.06798.

[4] M. Stilman, “Task constrained motion planning in robot joint space,” in 2007 IEEE/RSJ Int. Conf. Intell. Robots and Systems, 2007, pp. 3074–3081.

[5] A. H. Qureshi, J. Dong, A. Baig, and M. C. Yip, “Constrained motion planning networks X,” 2020, arXiv:2010.08707.

[6] Y. Li, F. Richter, J. Lu, E. K. Funk, R. K. Orosco, J. Zhu, and M. C. Yip, “SuPer: A surgical perception framework for endoscopic tissue manipulation with surgical robotics,” IEEE Robot. and Automat. Lett., vol. 5, no. 2, pp. 2294–2301, 2020.

[7] Masahiro Ono and B. C. Williams, “Iterative risk allocation: A new approach to robust model predictive control with a joint chance constraint,” in 2008 47th IEEE Conf. Decis. and Control, 2008, pp. 3427–3432.

[8] I. Blackmore, M. Ono, and B. C. Williams, “Chance-constrained optimal path planning with obstacles,” IEEE Trans. on Robot., vol. 27, no. 6, pp. 1080–1094, 2011.

[9] M. d. S. Arantes, C. F. M. Toledo, B. C. Williams, and M. Ono, “Collision-free encoding for chance-constrained nonconvex path planning,” IEEE Trans. on Robot., vol. 35, no. 2, pp. 433–448, 2019.

[10] J. van den Berg, P. Abbeel, and K. Goldberg, “LQG-MP: Optimized path planning for robots with motion uncertainty and imperfect state information,” Int. J. of Robot. Res., vol. 30, no. 7, pp. 895–913, 2011.

[11] H. Zhu and J. Alonso-Mora, “Chance-constrained collision avoidance for mavs in dynamic environments,” IEEE Robot. and Automat. Lett., vol. 4, no. 2, pp. 776–783, 2019.

[12] L. Janson, E. Schmerling, and M. Pavone, “Monte carlo motion planning for robot trajectory optimization under uncertainty,” in Robot. Res.: Volume 2, 2018, pp. 343–361.

[13] E. Schmerling and M. Pavone, “Evaluating trajectory collision probability through adaptive importance sampling for safe motion planning,” 2017, arXiv:1609.05399.

[14] W. Burgard, O. Brock, and C. Stachniss, “The Stochastic Motion Roadmap: A Sampling Framework for Planning with Markov Motion Uncertainty.” The MIT Press, 2008, pp. 233–240.

[15] J. van den Berg, S. Patil, and R. Alterovitz, “Motion planning under uncertainty using iterative local optimization in belief space,” The Int. J. of Robot. Res., vol. 31, no. 11, pp. 1263–1278, 2012.

[16] A. Bry and N. Roy, “Rapidly-exploring random belief trees for motion planning under uncertainty,” in 2011 IEEE Int. Conf. Robot. and Automation, 2011, pp. 723–730.

[17] S. Patil, J. van den Berg, and R. Alterovitz, “Estimating probability of collision for safe motion planning under gaussian motion and sensing uncertainty,” in 2012 IEEE Int. Conf. Robot. and Automation, 2012, pp. 3238–3244.

[18] W. Sun, L. G. Torres, J. van den Berg, and R. Alterovitz, “Safe motion planning for imprecise robotic manipulators by minimizing probability of collision,” in Robot. Res., ser. STAR, vol. 114, 2013, pp. 685–701.

[19] J. Chase Kew, B. Ichter, M. Bandari, T.-W. E. Lee, and A. Faust, “Neural collision clearance estimator for batched motion planning,” in Algorithmic Found. of Robot. XIV, 2021, pp. 73–89.

[20] Y. Zhi, N. Das, and M. Yip, “Diffico: Auto-differentiable proxy collision detection with multi-class labels for safety-aware trajectory optimization,” 2021, arXiv:2102.07413.

[21] N. Das and M. C. Yip, “Stochastic modeling of distance to collision for robot manipulators,” IEEE Robot. and Automat. Lett., vol. 6, no. 1, pp. 207–214, 2021.

[22] A. Agha-mohammadi, S. Chakravorty, and N. M. Amato, “Firm: Sampling-based feedback motion-planning under motion uncertainty and imperfect measurements,” Int. J. of Robot. Res., vol. 33, no. 2, pp. 268–304, 2014.

[23] C. E. Rasmussen and C. K. I. Williams, Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning). The MIT Press, 2005.

[24] N. Das and M. C. Yip, “Forward kinematics kernel for improved proxy collision checking,” IEEE Robot. and Automat. Lett., vol. 5, no. 2, pp. 2349–2356, 2020.

[25] S. C. Endres, C. Sandrock, and W. W. Focke, “A simplicial homology algorithm for lipschitz optimisation,” J. of Global Optim., vol. 72, no. 1, pp. 181–217, Oct 2018.

[26] F. Xia, A. R. Zamir, Z.-Y. He, A. Sax, J. Malik, and S. Savarese, “Gibson env: real-world perception for embodied agents,” in IEEE Comput. Vision and Patt. Recognit. (CVPR), 2018, IEEE, 2018.

[27] I. A. Šucan, M. Moll, and L. E. Kavraki, “The Open Motion Planning Library,” IEEE Robot. & Automat. Mag., vol. 19, no. 4, pp. 72–82, December 2012.

[28] S. Thrun, W. Burgard, and D. Fox, Probabilistic Robot. (Intell. Robot. and Autonomous Agents.), The MIT Press, 2005.

[29] S. Karaman and E. Frazzoli, “Sampling-based algorithms for optimal and Autonomous Agents), The MIT Press, 2005.