Cross-Entropy based Stochastic Optimization of Robot Trajectories using Heteroscedastic Continuous-time Gaussian Processes

Luka Petrović[a,*], Juraj Peršić[a], Marija Seder[a], Ivan Marković[a]

[a] *University of Zagreb Faculty of Electrical Engineering and Computing, Department of Control and Computer Engineering, Laboratory for Autonomous Systems and Mobile Robotics, Unska 3, HR-10000, Zagreb, Croatia,*

Abstract

High dimensional robot motion planning has recently been approached with trajectory optimization methods that efficiently minimize a suitable objective function in order to generate robot trajectories that are both optimal and feasible. However, finding a globally optimal solution is often an insurmountable problem in practice and state-of-the-art trajectory optimization methods are thus prone to local minima, mainly in cluttered environments. In this paper, we propose a novel trajectory planning algorithm that employs stochastic optimization in order to find a collision-free trajectory generated from a continuous-time Gaussian process (GP). The contributions of the proposed motion planning method stem from introducing the heteroscedasticity of the GP, together with exploited sparsity for efficient covariance estimation, and a cross-entropy based stochastic optimization for importance sampling based trajectory optimization. We evaluate the proposed method on three simulated scenarios: a maze benchmark, a 7 DOF robot arm planning benchmark and a 10 DOF mobile manipulator trajectory planning example and compare it to a state-of-the-art GP trajectory optimization method, namely the Gaussian process motion planner 2 algorithm (GPMP2). Our results demonstrate the following: (i) the proposed method yields a more thorough exploration of the solution space in complex environments than GPMP2, while having comparable execution time, (ii) the introduced heteroscedasticity generates GP priors better suited for collision avoidance and (iii) the proposed method has the ability to efficiently tackle high-dimensional trajectory planning problems.

Keywords: robot motion planning, trajectory optimization, continuous-time gaussian processes, stochastic optimization, cluttered environments

1. Introduction

Motion planning is an indispensable tool for robots that aspire to navigate through an environment without collisions. Motion planning algorithms attempt to generate trajectories through the robot's configuration space that are both feasible and optimal based on some performance criterion that may vary depending on the task, robot or environment. Generally, feasibility is congruous to a robot satisfying constraints such as being collision-free while reaching the desired goal and optimality corresponds to the cost efficiency, often referring to the smoothness of the trajectory. The initial attempts to solve motion planning problems included grid-based methods such as the A* algorithm [1], reactive planners such as bug algorithms [2], combinatorial planning methods originating from computational geometry such as cell decompositions [3], visibility graphs [4] and Voronoi diagrams [5]. All of the mentioned methods are complete, meaning that they find a solution if it exists and report failure otherwise. They present elegant solutions for low dimensional configuration spaces with static obstacles. However, bug algorithms produce unnecessarily long paths, while grid-based methods and combinatorial approaches suffer from the so-called curse of dimensionality, i.e. they quickly become computationally intractable with the increase of the configuration space dimension. That also means that replanning is impossible, making those approaches ineffective in dynamic environments.

The increasing complexity of robots and the environments that they operate in has spurred the need for high-dimensional motion planning. Consider, for instance, a personal robot operating in a cluttered household environment or a humanoid robot performing navigation and manipulation tasks in an unstructured environment. Efficient motion planning is important to enable these high degree-of-freedom (DOF) robots to perform tasks, subject to motion constraints while avoiding collisions with obstacles in the environment. Processing time is especially important in dynamic environments where replanning is necessary. Those considerations lead to the development of grid-based methods with ameliorated efficiency [6, 7, 8, 9] and to the development of sampling-based motion planning algorithms [10] which offer weaker guarantees than combinatorial methods but are more efficient. Since grid-based methods are complete, they are often used for planning in environ-

This research has been supported by the European Regional Development Fund under the grant KK.01.1.1.01.0009 (DATACROSS).

*Corresponding author: Luka Petrović

Email addresses: luka.petrovic@fer.hr (Luka Petrović), juraj.persic@fer.hr (Juraj Peršić), marija.seder@fer.hr (Marija Seder), ivan.markovic@fer.hr (Ivan Marković)

ments featuring bottlenecks and other such narrow passages. The main drawback of grid-based methods is their computational complexity, as even the state-of-the-art methods suffer from the curse of dimensionality, and become computationally intractable for high DOF systems such as humanoid robots or mobile manipulators.

The central tenet of sampling-based methods [11, 10, 12] is the idea of connecting points randomly sampled from the free configuration space. Such approaches abandon the concept of explicitly characterizing the configuration space – they use a collision detection algorithm to probe the configuration space to determine whether some configuration lies in free space or not. Sampling-based methods are probabilistically complete, meaning that the probability they will produce a solution approaches one as more time is spent [12]. Sampling-based methods have become popular in the domain of high-dimensional motion planning, including planning for manipulation. Perhaps the most commonly used algorithms are the probabilistic roadmap (PRM) [11] and rapidly exploring random trees (RRT) [13, 14]. Both methods were shown to be well suited for path planning in configuration spaces with many DOFs and with kinodynamic constraints [15, 16]. Sampling based methods typically do not explicitly optimize an objective function, although variants of PRM and RRT which are provably asymptotically optimal have been proposed in [12]. However, sampling based methods can still be computationally inefficient for high-dimensional problems with challenging constraints. The main disadvantage of sampling-based planning methods is that the obtained paths often exhibit jerky and redundant motion and therefore require post processing to smooth and shorten the computed trajectories. Furthermore, considerable computational effort is expended in sampling the portions of the configuration space that might not be relevant to the task.

All of the discussed techniques so far aim at capturing the connectivity of free configuration space into a graph. Recently, trajectory optimization methods, appropriate for very high DOF robots, have become a subject of increased interest. In these approaches the trajectory is encoded as a sequence of states and controls, which offers several advantages for robot motion planning. First, they can be used to smooth and shorten trajectories computed by other planning methods such as sampling-based planners. Second, they can be used to compute collision-free trajectories from scratch by starting with naive trajectory initializations, that might be in collision with obstacles [17], and then minimize an appropriate objective function. Those methods are computationally efficient; however, unlike sampling-based planners, they only find a locally optimal solution. A seminal work in the modern trajectory optimization is the covariant Hamiltonian optimization for motion planning (CHOMP) [18, 19]. CHOMP revived the interest in trajectory optimization by demonstrating the effectiveness on several robotic platforms, including a mobile manipulation platform and a quadruped. It utilizes a precomputed signed distance field for collision checking and covariant gradient descent to minimize obstacle and smoothness costs. Inspired by CHOMP, stochastic trajectory optimization for motion planning (STOMP) algorithm [20] was introduced, which samples a series of noisy trajectories to explore the space around an initial trajectory which are then combined to produce an updated trajectory with lower cost. The key trait of STOMP is its ability to optimize non-differentiable constraints. An important shortcoming of CHOMP and STOMP is the need for many trajectory states to reason about fine resolution obstacle representations and to find feasible solutions when there are many constraints. To address this shortcoming, the TrajOpt [21, 17] algorithm formulates trajectory optimization as sequential quadratic programming. The key feature of TrajOpt is the ability to solve complex motion planning problems with few states, since swept volumes are considered to ensure continuous-time safety. However, if the smoothness is required in the output trajectory, either a densely parameterized trajectory or post-processing of the trajectory might still be needed, thus increasing the computation time.

The Gaussian process (GP) motion planning family of algorithms [22, 23, 24, 25] employs continuous-time trajectory representation in order to overcome the computational cost incurred by using large number of states. The GPMP algorithm [22] parameterizes the trajectory with few support states and then uses efficient GP interpolation to query the trajectory at any time of interest. Its extension, GPMP2 [23], represents trajectories as samples from a continuous-time GP and then formulates the planning problem as probabilistic inference. It exploits the sparsity of the underlying system by using preexisting efficient optimization tools developed by the simultaneous localization and mapping (SLAM) community [26] to generate fast solutions. A useful property of GPMP2 is its extensibility and applicability to a wide range of problems. For example, combined learning from demonstration and motion planning [27] presented an efficient approach to skill learning and generalizable skill reproduction. In [28], authors provided a framework for avoidance of singular robot configurations as manipulability maximization, while a unified probabilistic framework for trajectory estimation and planning was provided in [29].

Although trajectory optimization methods generate fast solutions in high-dimensional spaces, they have limited exploration ability, and in complex environments often converge to infeasible local minima, since they rely on gradient-based optimization techniques. In general optimization problems, gradient-free stochastic optimization is often employed in problems with plethora of local minima, and is the approach that we will rely on in the current paper. In motion planning, this is a relatively unexplored direction with only a few existing works. A recently proposed particle swarm filter for trajectory optimization [30] showed better local minima avoidance than gradient-based methods; however, it did not demonstrate effectiveness in complex, highly cluttered environments and carries the aforementioned problems associated with discrete-time trajectory representations. In [31], the authors proposed a general randomized path planning method based on sampling in the space of trajectories. At its core lies the cross-entropy method for estimation of rare-event probabilities. However, this approach relies on sampling from a probability distribution over the set of feasible paths – sampling whole trajectories that are collision-free, which can become infeasible in cluttered environments or high-dimensional spaces.

In this paper, we propose a gradient-free stochastic optimization method for trajectory planning with continuous time GP trajectory representations. The proposed method is the extension of our preliminary work presented in a conference paper [32]. The contributions of the current method stem from introducing the heteroscedastic GP, together with exploited sparsity for efficient covariance estimation, and a cross-entropy based stochastic optimization for importance sampling based trajectory optimization for motion planning. The main advantages of the proposed method are: (i) continuous time representation leading to smooth trajectories that can be queried at any time of interest, (ii) powerful heteroscedastic priors better suited for collision avoidance in motion planning problems, (iii) estimation of the collision-free trajectories distribution for a given environment allowing for faster convergence, and (iv) stochastic trajectory optimization allowing for better exploration while being less prone to local minima than gradient based methods. While our method belongs to the trajectory optimization approaches, it relies on random trajectory samples which raises a connection to the sampling based planning. The proposed method is an example of bridging the gap between sampling based and trajectory optimization approaches in order to generate fast solutions in high dimensional spaces while retaining the ability to thoroughly explore the environment. We evaluated our method on three simulation scenarios: a maze benchmark, a 7 DOF robot arm planning benchmark and a 10 DOF mobile manipulator trajectory planning example. In the benchmark examples we compared the proposed method to GPMP2, a state-of-the-art gradient-based method, while the mobile manipulator example showcases the applicability of the method in a high DOF planning problem. The results show that the proposed method yields a higher success rate in complex environments with comparable execution time.

2. Heteroscedastic Gaussian Processes for Motion Planning

A continuous-time GP robot trajectory representation allows efficient reasoning about collisions and querying the robot state at any time of interest. In this section, we present the GP trajectory representation framework for motion planning based on [25], which includes constructing a GP prior suited for motion planning and an efficient GP interpolation procedure. We introduce the heteroscedasticity in the time-domain of the proposed GP and discuss its benefits with regards to motion planning problems. We also provide a principled way to draw trajectory samples from the pertaining GP. For an in-depth treatment of GP trajectory representations in robotics, including the detailed derivations of the stated equations, we refer the reader to [25, 33, 34].

2.1. The Gaussian Process Trajectory Representations

Consider a continuous-time trajectory as a sample from a continuous-time Gaussian process [35]

$$\boldsymbol{\theta}(t) \sim \mathcal{GP}(\boldsymbol{\mu}(t), \mathcal{K}(t, t')) \tag{1}$$

that is parameterized with N *support states* at discrete time instants, $\boldsymbol{\theta}_i \in \mathrm{R}^D$, $i \in N$, where D is the state dimensionality. This implies that, for any collection of times $\boldsymbol{t} = \{t_0, \ldots, t_N\}$, $\boldsymbol{\mu}(t)$ is a vector-valued mean function and $\mathcal{K}(t, t')$ is a matrix-valued covariance function [25], the two defined as

$$\boldsymbol{\mu} = [\boldsymbol{\mu}(t_0) \ldots \boldsymbol{\mu}(t_N)]^T,\ \mathcal{K} = [\mathcal{K}(t_i, t_j)]|_{i,j,0\leq i,j\leq N}. \tag{2}$$

A vector-valued GP provides a principled representation of continuous-time trajectories, where trajectories are regarded as functions that map time to robot state. We employ a structured kernel belonging to a special class of GP priors generated by a linear time-varying stochastic differential equation (LTV-SDE)

$$\dot{\boldsymbol{\theta}}(t) = \boldsymbol{F}(t)\boldsymbol{\theta}(t) + \boldsymbol{\nu}(t) + \boldsymbol{L}(t)\boldsymbol{w}(t), \tag{3}$$

where $\boldsymbol{F}$ and $\boldsymbol{L}$ are system matrices and $\boldsymbol{\nu}$ is a known exogenous input. The white noise process $\boldsymbol{w}(t)$ itself is a GP with zero mean value

$$\boldsymbol{w}(t) \sim \mathcal{GP}(\boldsymbol{0}, \boldsymbol{Q}_c(t)\delta(t - t')), \tag{4}$$

where $\boldsymbol{Q}_c(t)$ is a positive semi-definite time-varying power-spectral density matrix. A similar dynamical system has been utilized to generate trajectory distributions in estimation [36, 37], calibration [38] and planning [25, 29]. However, the crucial difference in our approach is that the covariance $\boldsymbol{Q}_c(t)$ is time-varying and consequently generates a heteroscedastic GP [27], i.e. the covariance $\boldsymbol{Q}_c(t)$ of the white noise process $\boldsymbol{w}(t)$ is non-constant through time. We discuss benefits of this approach in Section 2.6.

The mean and the covariance of the GP generated by the LTV-SDE given in (3) evaluate to

$$\widetilde{\boldsymbol{\mu}}(t) = \boldsymbol{\Phi}(t, t_0)\boldsymbol{\mu}_0 + \int_{t_0}^{t} \boldsymbol{\Phi}(t, s)\boldsymbol{\nu}(s)\,\mathrm{d}s, \tag{5}$$

$$\widetilde{\mathcal{K}}(t, t') = \boldsymbol{\Phi}(t, t_0)\mathcal{K}_0\boldsymbol{\Phi}(t', t_0)^T + \int_{t_0}^{\min(t,t')} \boldsymbol{\Phi}(t, s)\boldsymbol{L}(s)\boldsymbol{Q}_c(s)\boldsymbol{L}(s)^T\boldsymbol{\Phi}(t', s)^T\,\mathrm{d}s, \tag{6}$$

where $\boldsymbol{\mu}_0$ and $\mathcal{K}_0$ are the initial mean and covariance of the first state, and $\boldsymbol{\Phi}(t, s)$ is the state transition matrix [36].

2.2. GP Prior for Motion Planning

Due to the Markov property of the LTV-SDE in (3), the precision matrix $\widetilde{\mathcal{K}}^{-1}$ is exactly sparse block tridiagonal [36]

$$\widetilde{\mathcal{K}}^{-1} = \widetilde{\boldsymbol{F}}^{-T}\widetilde{\boldsymbol{Q}}^{-1}\widetilde{\boldsymbol{F}}^{-1}, \tag{7}$$

where

$$\widetilde{\boldsymbol{F}}^{-1} = \begin{bmatrix} \mathbf{1} & 0 & \ldots & 0 & 0 \\ -\boldsymbol{\Phi}(t_1, t_0) & \mathbf{1} & \ldots & 0 & 0 \\ 0 & -\boldsymbol{\Phi}(t_2, t_1) & \ddots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \mathbf{1} & 0 \\ 0 & 0 & \ldots & -\boldsymbol{\Phi}(t_N, t_{N-1}) & \mathbf{1} \end{bmatrix} \tag{8}$$

and

$$\widetilde{\boldsymbol{Q}}^{-1} = \mathrm{diag}(\mathcal{K}_0^{-1}, \boldsymbol{Q}_{0,1}^{-1}, ..., \boldsymbol{Q}_{N-1,N}^{-1}) \tag{9}$$

with

$$\boldsymbol{Q}_{a,b} = \int_{t_a}^{t_b} \boldsymbol{\Phi}(t_b, s)\boldsymbol{L}(s)\boldsymbol{Q}_c(s)\boldsymbol{L}(s)^T\boldsymbol{\Phi}(t_b, s)^T \,\mathrm{d}s. \tag{10}$$

The GP defined by mean and covariance in (5) and (6) is well suited for estimation problems. However, in motion planning problems there exists a desired fixed goal state. Given that, we need to condition this GP with a fictitious observation on the goal state with mean $\boldsymbol{\mu}_N$ and covariance $\boldsymbol{\mathcal{K}}_N$. Since we do not want trajectories to deviate from the goal state, the covariance of the goal state $\boldsymbol{\mathcal{K}}_N$ should be relatively small thus keeping the desired goal state fixed. This can be accomplished while still preserving the sparsity of the precision matrix [25]

$$\begin{aligned}\boldsymbol{\mathcal{K}}^{-1} &= \begin{bmatrix} & \widetilde{\boldsymbol{F}}^{-1} & \\ \boldsymbol{0} & \dots \; \boldsymbol{0} & \boldsymbol{\mathrm{I}} \end{bmatrix}^{\top} \begin{bmatrix} \widetilde{\boldsymbol{Q}}^{-1} & \\ & \boldsymbol{\mathcal{K}}_N^{-1} \end{bmatrix} \begin{bmatrix} & \widetilde{\boldsymbol{F}}^{-1} & \\ \boldsymbol{0} & \dots \; \boldsymbol{0} & \boldsymbol{\mathrm{I}} \end{bmatrix} \\ &\coloneqq \boldsymbol{G}^T\boldsymbol{Q}^{-1}\boldsymbol{G}. \end{aligned} \tag{11}$$

While the mean vector $\boldsymbol{\mu}$ and the precision matrix $\boldsymbol{\mathcal{K}}^{-1}$ fully determine a continuous-time GP in (1), in some situations the covariance matrix $\boldsymbol{\mathcal{K}}$ might still be needed, i.e. for data visualization or when optimizing trajectories while learning from demonstration [27]. A naive approach is to compute $\boldsymbol{\mathcal{K}}$ by inverting the precision matrix. However, due to the exact sparsity of the precision matrix $\boldsymbol{\mathcal{K}}^{-1}$, the covariance matrix $\boldsymbol{\mathcal{K}}$ can be computed efficiently [39]. First we carry out a sparse lower-diagonal-upper decomposition,

$$\boldsymbol{\mathcal{K}}^{-1} = \boldsymbol{L}\boldsymbol{D}\boldsymbol{L}^T, \tag{12}$$

where $\boldsymbol{D}$ is diagonal and $\boldsymbol{L}$ is lower-triangular with ones on the main diagonal. Since $\boldsymbol{\mathcal{K}}$ is symmetric, only the calculation of the main diagonal and the lower-half blocks is required. Blocks of the covariance matrix indexed with j, k can then be computed through backward substitution

$$\boldsymbol{\mathcal{K}}_{N,N} = \boldsymbol{D}_{N,N}^{-1}, \tag{13}$$

$$\boldsymbol{\mathcal{K}}_{N,N-1} = -\boldsymbol{\mathcal{K}}_{N,N}\boldsymbol{L}_{N,N-1}, \tag{14}$$

$$\boldsymbol{\mathcal{K}}_{N-1,N-1} = \boldsymbol{D}_{N-1,N-1}^{-1} - \boldsymbol{\mathcal{K}}_{N-1,N}\boldsymbol{L}_{N,N-1}, \tag{15}$$

$$\vdots \tag{16}$$

$$\boldsymbol{\mathcal{K}}_{j,k} = \delta(j,k)\boldsymbol{D}_{j,k}^{-1} - \sum_{l=k+1}^{N} \boldsymbol{\mathcal{K}}_{j,l}\boldsymbol{L}_{l,k} \quad (j \geq k). \tag{17}$$

For a more in-depth treatment of this covariance matrix calculation procedure, the reader is referred to [40, 39].

2.3. Fast GP interpolation

A major benefit of modelling continuous-time trajectories in motion planning with GPs, is the possibility to query the planned state $\boldsymbol{\theta}(\tau)$ at any time of interest τ, and not only at the discrete *support state* time instants. The precision matrix defined in (11) allows for computationally efficient, structure-exploiting GP interpolation with $O(1)$ complexity. State $\boldsymbol{\theta}(\tau)$ at $\tau \in [t_i, t_{i+1}]$ is a function only of its neighboring states [23]

$$\boldsymbol{\theta}(\tau) = \boldsymbol{\mu}(\tau) + \boldsymbol{\Lambda}(\tau)(\boldsymbol{\theta}_i - \boldsymbol{\mu}_i) + \boldsymbol{\Psi}(\tau)(\boldsymbol{\theta}_{i+1} - \boldsymbol{\mu}_{i+1}), \tag{18}$$

$$\boldsymbol{\Lambda}(\tau) \coloneqq \boldsymbol{\Phi}(\tau, t_i) - \boldsymbol{\Psi}(\tau)\boldsymbol{\Phi}(t_{i+1}, t_i), \tag{19}$$

$$\boldsymbol{\Psi}(\tau) \coloneqq \boldsymbol{Q}_{i,\tau}\boldsymbol{\Phi}(t_{i+1}, \tau)^T\boldsymbol{Q}_{i,i+1}^{-1}, \tag{20}$$

where $\boldsymbol{Q}_{a,b}$ is given in (10). Efficient GP interpolation can be exploited for reasoning about small obstacles while keeping a relatively small number of *support states* which reduces the incurred computational burden. It can also be utilized for providing a dense output trajectory that a robot can execute without any post-processing.

2.4. Constant-velocity motion model

Robot dynamics are represented with the double integrator linear system with white noise injected in acceleration. The trajectory is thus generated by the LTV-SDE (3), where the Markovian state $\boldsymbol{\theta}(t)$ consists of position and velocity in configuration space with the following system matrices

$$\boldsymbol{F}(t) = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{\mathrm{I}} \\ \boldsymbol{0} & \boldsymbol{0} \end{bmatrix}, \; \boldsymbol{L}(t) = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{\mathrm{I}} \end{bmatrix}. \tag{21}$$

The state transition matrix $\boldsymbol{\Phi}(t, s)$ can then be evaluated as [33]

$$\boldsymbol{\Phi}(t, s) = \exp\{(t-s)\boldsymbol{F}\} = \begin{bmatrix} \boldsymbol{1} & (t-s)\boldsymbol{1} \\ \boldsymbol{0} & \boldsymbol{1} \end{bmatrix}. \tag{22}$$

This formulation generates a constant velocity GP prior which is centered around a zero acceleration trajectory. Applying this motion model minimizes actuator acceleration in the configuration space, thus minimizing the energy consumption and providing the physical meaning of smoothness [25].

2.5. Drawing trajectory samples

A sample trajectory can be generated using [41]

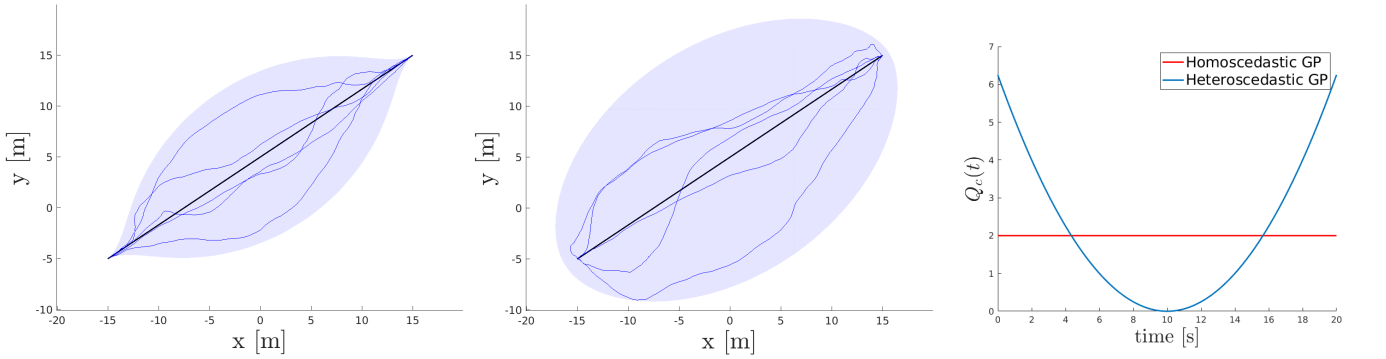
$$\boldsymbol{\theta} = \boldsymbol{\mu} + \boldsymbol{A}\boldsymbol{Z}, \tag{23}$$

where $\boldsymbol{A}$ is a lower triangular matrix obtained by Cholesky decomposition of the covariance matrix, $\boldsymbol{\mathcal{K}} = \boldsymbol{A}\boldsymbol{A}^T$, and $\boldsymbol{Z}$ is a vector of N standard normal variables $\boldsymbol{Z} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\mathrm{I}})$.

The sampling process in (23) requires knowledge of the covariance matrix $\boldsymbol{\mathcal{K}}$. While calculation of the covariance matrix $\boldsymbol{\mathcal{K}}$ employing the procedure described in (13)-(17) is efficient in comparison to the straightforward inversion of the precision matrix, it can still be computationally expensive in higher-dimensional state-spaces with dense parameterization of the GP. Alternatively, a sample trajectory can be generated without computing the covariance matrix [41]

$$\boldsymbol{\theta} = \boldsymbol{\mu} + \boldsymbol{B}^{-T}\boldsymbol{Z}, \tag{24}$$

where $\boldsymbol{B}$ is a lower triangular matrix obtained by Cholesky decomposition of the precision matrix, $\boldsymbol{\mathcal{K}}^{-1} = \boldsymbol{B}\boldsymbol{B}^T$, and $\boldsymbol{Z}$ is a vector of N standard normal variables $\boldsymbol{Z} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\mathrm{I}})$. Since the precision matrix $\boldsymbol{\mathcal{K}}^{-1}$ has block-tridiagonal structure, its Cholesky decomposition leads to a lower bidiagonal matrix which can be efficiently inversed with exploited sparsity.



(a) Homoscedastic GP (b) Heteroscedastic GP (c) White noise process $\boldsymbol{w}(t)$ covariance through time

Figure 1: Comparison of the homoscedastic and the proposed heteroscedastic GP priors which depend on the covariance of the white noise process $\boldsymbol{w}(t)$

2.6. Benefits of proposed heteroscedasticity

State-of-the-art GP based methods [23, 25] work by minimizing the sum of two costs: the obstacle cost and the smoothness cost. The obstacle cost measures how close the trajectory is to obstacles, while the smoothness cost measures the deviation of the trajectory from the GP prior mean. Those methods use covariance as a parameter in optimization, with smaller values of $\boldsymbol{\mathcal{K}}$ penalizing trajectory deviation from the prior mean. For therein employed homoscedastic GPs, small constant values of $\boldsymbol{Q}_c$ allow high variance of states near the midpoint of trajectory, while states close to the trajectory start or goal have relatively small variance. If the prior mean of those states passes through an obstacle, the Levenberg-Marquardt optimization, utilized in [23, 25], will have difficulty escaping the collision. The collision happens when the incurred smoothness cost becomes relatively large in comparison to obstacle cost thus creating a local minimum. Figure 1a shows that sample trajectories drawn from a homoscedastic GP do not have large deviation from the mean near the first and the last state. With larger constant values of $\boldsymbol{Q}_c$ this problem diminishes; however, states near the trajectory midpoint then have too high variance and trajectories lose the desirable smoothness property.

By introducing heteroscedasticity of the underlying white noise process, we can design GPs that are better suited for motion planning problems and help alleviate the problem of local minima. Ideally, the GP should be able to generate trajectories that maneuver around the obstacles near start and goal states, while retaining the smoothness property in the middle. With careful selection of the proposed time-varying white noise power-spectral density matrix $\boldsymbol{Q}_c(t)$, we are able to model GPs that achieve the stated goal. We propose modelling $\boldsymbol{Q}_c(t)$ as an isotropic matrix $\boldsymbol{Q}_c(t) = Q_c(t)\boldsymbol{I}$ with parameter $Q_c(t)$ modelled as a parabola. A parabola has high values at the beginning and end, while the lowest value lies at the temporal midpoint of the trajectory, leading to GPs that have the aforementioned desirable features. Note that the GP covariance $\boldsymbol{\mathcal{K}}(t, t')$ is obtained by propagating $Q_c(t)$ with the underlying motion model, as defined in (7), and thus low (or even zero) value of $Q_c(t)$ at a particular time instant does not imply small GP covariance. An example of such $Q_c(t)$ is depicted in Figure 1c and it leads to a heteroscedastic GP shown in Figure 1b. Notably, sample trajectories drawn from the example heteroscedastic GP have large deviation from the mean near the first and the last state. The example GP would be able to thoroughly explore the environment and generate trajectories that bypass obstacles near the first and the last state. Note that any non-negative function can be used for $Q_c(t)$, depending on the specific context of the motion planning problem at hand. For example, one could model $Q_c(t)$ as a monotonically decreasing function, resembling the aim of exploring more at the beginning and less towards the end.

3. Proposed Stochastic Trajectory Optimization

In this section we describe the proposed gradient-free stochastic optimization method for motion planning. First, we formalize the motion planning problem as trajectory optimization with trajectories considered as samples from the heteroscedastic GP proposed in Section 2. Then, we present the cross-entropy optimization method adapted to the motion planning problem setting and derive the GP mean and covariance update rules. Finally, we state the proposed algorithm steps and give remarks regarding its computational efficiency.

3.1. Problem formulation

Formally, the goal of trajectory optimization for motion planning is to find a smooth, collision-free trajectory through the robot's configuration space between the start and end points. Prior work in this area [18, 21, 23] models the cost of a trajectory using two terms: a prior term, which usually encodes smoothness that minimizes higher-order derivatives of the robot states, and an arbitrary state-dependent term, which usually measures the cost of being near obstacles. However, our optimization criteria consists solely of an arbitrary state-dependent cost term. We reason that our trajectory carries an inherent property of smoothness since we model it as a sample from the GP defined in (1) with the underlying constant-velocity model described in Section 2.4. Therefore, our method starts with the following optimization problem:

$$\begin{aligned} \underset{\boldsymbol{\theta}(t)}{\text{minimize}} \quad & f[\boldsymbol{\theta}(t)] \\ \text{subject to} \quad & \boldsymbol{\theta}(t) \sim \mathcal{GP}(\boldsymbol{\mu}(t), \boldsymbol{\mathcal{K}}(t, t')). \end{aligned} \tag{25}$$

The state-dependent cost term $f[\boldsymbol{\theta}(t)]$ can include any cost function corresponding to the desired trajectory properties. In the scope of this work we consider only collision avoidance. Other than collision avoidance, the state-dependent cost term can incorporate task-space constraints (e.g. holding the cup of coffee upright) [23], motor torques [20] and manipulability [28].

Similarly to GPMP2 [23], the robot body is represented as a set of spheres. The obstacle cost function for any robot configuration $\boldsymbol{\theta}_i$ is then evaluated by computing the hinge loss for each sphere and collecting it into a single scalar

$$f[\boldsymbol{\theta}_i] = \sum_{j=1}^{S} c[d(\boldsymbol{x}(\boldsymbol{\theta}_i, S_j), \mathcal{O})] \tag{26}$$

where $\boldsymbol{x}$ is the forward kinematics of a robot, $\mathcal{O}$ is the set of obstacles in the environment, d is the Euclidean signed distance function, c is the hinge loss function and S is the number of spheres that accurately represent the robot model.

Forward kinematics $\boldsymbol{x}(\boldsymbol{\theta}_i, S_j)$ maps any configuration $\boldsymbol{\theta}_i$ to the robot's task space in order to find the center position of a sphere S_j of the robot model. Given a sphere radius and its center position, we compute the Euclidean signed distance $d(\boldsymbol{x}, \mathcal{O})$ from the sphere at position $\boldsymbol{x}$ to the closest obstacle surface in the task space. The pertaining distance is easy to calculate with a robot represented as set of spheres, since it corresponds to the distance between a sphere center and its closest obstacle surface minus the sphere radius. A Euclidean signed distance field (SDF) can be precomputed with a desired resolution and stored in a voxel grid. The signed distance of any position in the task space can then be efficiently queried by using trilinear interpolation on the voxel grid. The aforementioned hinge loss function is defined as

$$\boldsymbol{c}(d) = \begin{cases} \varepsilon - d & \text{if } d \leq \varepsilon \\ 0 & \text{if } d > \varepsilon \end{cases}, \tag{27}$$

where d is the calculated signed distance between the sphere and the closest obstacle, and ε is the parameter that can be seen as a safety distance indicating the boundary of the area near obstacle surfaces that is considered dangerous [25]. Even if a given trajectory is collision-free but passes too close to the obstacles, its incurred obstacle cost would be non-zero. The optimization thus guides the robot to stay a minimum distance ε away from obstacles.

3.2. Cross-entropy optimization method

Most of the state-of-the-art approaches use gradient-based methods which find locally optimal trajectories. In this work, we instead optimize using a derivative-free cross-entropy stochastic optimization method. This allows for better exploration while being less prone to local minima. It also enables optimization of arbitrary costs which are non-differentiable or non-smooth. Herein, we provide a brief introduction to the stochastic framework for solving the optimization problem in (25) by sampling trajectories from a GP in (1). Our development follows [42], but with a GP instead of a general parametric distribution and with notation adapted to our setting.

Consider the optimization problem in (25). Let us denote its minimum by γ^*. We can thus write

$$f[\boldsymbol{\theta}^*(t)] = \quad \gamma^* = \min_{\boldsymbol{\theta}(t) \sim \mathcal{GP}(\boldsymbol{\mu}, \mathcal{K})} f[\boldsymbol{\theta}(t)]. \tag{28}$$

The starting point of the cross-entropy method is associating an estimation problem with the optimization problem in (28). First, a collection of indicator functions $I_{\{f[\boldsymbol{\theta}] \leq \gamma\}}$ for various thresholds $\gamma \in \mathbb{R}$ is defined

$$I_{\{f[\boldsymbol{\theta}] \leq \gamma\}} = \begin{cases} 1 & \text{if } f[\boldsymbol{\theta}] \leq \gamma \\ 0 & \text{if } f[\boldsymbol{\theta}] > \gamma \end{cases}. \tag{29}$$

For a certain GP defined by its mean $\boldsymbol{\mu}$ and covariance $\mathcal{K}$ let us associate the optimization problem in (28) with the problem of estimating the probability that a cost function $f(\boldsymbol{\theta})$ has smaller value than a given threshold γ

$$\begin{aligned} \iota(\gamma) &\coloneqq \mathbb{P}_{\mu,\mathcal{K}}(f[\boldsymbol{\theta}] \leq \gamma) \\ &= \int_{\boldsymbol{\theta}} I_{\{f[\boldsymbol{\theta}] \leq \gamma\}} p(\boldsymbol{\theta}; \boldsymbol{\mu}, \mathcal{K}) d\boldsymbol{\theta} \\ &= \mathbb{E}_{\boldsymbol{\mu},\mathcal{K}}[I_{\{f[\boldsymbol{\theta}] \leq \gamma\}}], \end{aligned} \tag{30}$$

where $\mathbb{P}_{\mu,\mathcal{K}}$ denotes the probability measure under which the random trajectory $\boldsymbol{\theta}$ has probability density function $p(\boldsymbol{\theta}; \boldsymbol{\mu}, \mathcal{K})$ and $\mathbb{E}_{\mu,\mathcal{K}}$ is the corresponding expectation operator. To see how the problems in (30) and (28) are associated, suppose that a certain threshold γ is close to the optimal value γ^*. In a typical case the probability of a randomly sampled trajectory $\boldsymbol{\theta}$ being near the optimal one is rather small, thus $f[\boldsymbol{\theta}] \leq \gamma$ is a rare event. Note that, for a globally optimal $\boldsymbol{\theta}^*$, $\iota(\gamma^*) = p(\boldsymbol{\theta}^*; \boldsymbol{\mu}, \mathcal{K})$. Therefore, for $\gamma = \gamma^*$ a natural way to estimate $\iota(\gamma)$ would be to use the rare-event estimation framework [43] which can solve this problem efficiently by making adaptive changes to the cross-entropy and thus iteratively generate GPs that are *steered* in the direction of the theoretically optimal density.

This is achieved by employing the likelihood-ratio estimator [42] with reference parameters $\{\boldsymbol{\mu}, \mathcal{K}\}$ given by

$$\boldsymbol{\mu}^*, \mathcal{K}^* = \operatorname*{argmax}_{\boldsymbol{\mu},\mathcal{K}} \mathbb{E}_{\mu,\mathcal{K}}[I_{\{f[\boldsymbol{\theta}] \leq \gamma\}} \ln p(\boldsymbol{\theta}; \boldsymbol{\mu}, \mathcal{K})]. \tag{31}$$

In other words, we try to find the GP parameters $\boldsymbol{\mu}^*, \mathcal{K}^*$ that maximize the expectation of finding the optimal trajectory $\boldsymbol{\theta}^*$ which minimizes the cost function $f(\boldsymbol{\theta})$. In motion planning context, our approach can be seen as trying to find a GP that is the best approximation (in a Kullback-Leibler sense) of the underlying distribution of collision-free trajectories for a given environment. In implementation, cross-entropy method adapted for motion planning starts with drawing K sample trajectories from the GP defined in (1), where the mean $\boldsymbol{\mu}$ is initialized as a constant-velocity straight line in configuration space and covariance matrix $\mathcal{K}$ arises from the exactly sparse precision matrix defined in (11). We sample trajectories from the GP with (24). Subsequently, for each sampled trajectory we evaluate the cost using the aforementioned hinge loss with a precomputed signed distance field defined in (26), with the cost of k-th trajectory being $\gamma^k = f[\boldsymbol{\theta}^k(t)]$. From the evaluated K trajectories,

we pick the M best ones according to the optimization criteria, i.e. ones with the lowest cost $f[\boldsymbol{\theta}(t)]$. In a cross-entropy optimization framework, this can be seen as setting the indicator function threshold as $f[\boldsymbol{\theta}] \leq \gamma^M$. Using the sampled trajectories and evaluated costs, the GP parameters $\boldsymbol{\mu}^*, \boldsymbol{\mathcal{K}}^*$ for the next iteration can be computed with (31). For Gaussian sampling distributions, it was shown that the problem in (31) admits a closed-form solution given by the sample (empirical) mean and covariance [44, 31].

3.3. Updating the mean and the covariance

We modify the cross-entropy optimization method described in previous section by modifying the indicator function in (29). A similar modification was done in [44]. After sampling, the cross-entropy method takes into account M best trajectories according to the computed cost and their contribution to the new mean and covariance is equal regardless of their cost. In motion planning context, if we pick M best trajectory samples in a given highly cluttered environment and one of them has significantly lower cost than others (i.e. it is closer to being collision-free), the computed mean will average it out together with the other $M-1$ and valuable information will be lost. Since we try to find collision-free trajectories, it makes sense to value more the trajectory with the smallest cost. Therefore, we introduce an alternative, *soft* indicator function

$$I_{\{f[\boldsymbol{\theta}]\leq\gamma\}} = \begin{cases} 1/f[\boldsymbol{\theta}] & \text{if } f[\boldsymbol{\theta}] \leq \gamma \\ 0 & \text{if } f[\boldsymbol{\theta}] > \gamma \end{cases}. \tag{32}$$

With the introduced alternative indicator function, trajectories with lower cost posses a higher *indicator value*. The solution to the problem in (31) then corresponds to the weighted sample mean and covariance [44]. The cost-weighted arithmetic sample mean of the M best trajectories forms a new GP mean $\boldsymbol{\mu}$ for next iteration

$$\boldsymbol{\mu}(t) = \sum_{m=1}^{M} \lambda_m \boldsymbol{\theta}^m(t) \tag{33}$$

where weights λ_m correspond to the normalized *indicator values* of given trajectories

$$\lambda_m = \frac{1/f[\boldsymbol{\theta}^m(t)]}{\sum_{m=1}^{M} 1/f[\boldsymbol{\theta}^m(t)]}. \tag{34}$$

When computing the GP covariance $\boldsymbol{\mathcal{K}}$ or precision $\boldsymbol{\mathcal{K}}^{-1}$ matrix for the next iteration from the provided M best trajectories, we need to take into account the pertaining sparse structure of the precision matrix and the underlying motion model. Since trajectories are sampled at discrete time instances, we only have access to the support states $\boldsymbol{\theta}_i$. A discrete version of the LTV-SDE in (3) is defined as

$$\boldsymbol{\theta}_{i+1} = \boldsymbol{\Phi}(t_{i+1}, t_i)\boldsymbol{\theta}_i + \boldsymbol{v}_{i,i+1} + \boldsymbol{w}_{i,i+1}, \tag{35}$$

where

$$\boldsymbol{w}_{i,i+1} \sim \mathcal{N}(0, \boldsymbol{Q}_{i,i+1}). \tag{36}$$

Mean trajectory of the discrete LTV-SDE in (35) can be written as

$$\boldsymbol{\mu}_{i+1} = \boldsymbol{\Phi}(t_{i+1}, t_i)\boldsymbol{\mu}_i + \boldsymbol{v}_{i,i+1}. \tag{37}$$

Now we can write exogenous input $\boldsymbol{v}_{i,i+1}$ as

$$\boldsymbol{v}_{i,i+1} = \boldsymbol{\mu}_{i+1} - \boldsymbol{\Phi}(t_{i+1}, t_i)\boldsymbol{\mu}_i, \tag{38}$$

and after substituting back to (35) get the residual

$$\boldsymbol{w}^m_{i,i+1} = \boldsymbol{\theta}^m_{i+1} - \boldsymbol{\Phi}(t_{i+1}, t_i)\boldsymbol{\theta}^m_i - \boldsymbol{\mu}_{i+1} + \boldsymbol{\Phi}(t_{i+1}, t_i)\boldsymbol{\mu}_i. \tag{39}$$

The weighted sample covariance matrix between states i and $i+1$ can be estimated using the calculated mean and sampled trajectories

$$\boldsymbol{Q}^{est}_{i,i+1} = \mathbb{E}[\boldsymbol{w}_{i,i+1}\boldsymbol{w}^T_{i,i+1}] \tag{40}$$

$$= \sum_{m=1}^{M} \lambda_m \boldsymbol{w}^m_{i,i+1}\boldsymbol{w}^{mT}_{i,i+1}. \tag{41}$$

By calculating $\boldsymbol{Q}^{est}_{i,i+1}$ with (41), we implicitly solved the integral in (10), determining $Q_c(t)$ in the process. After calculating the covariance estimates $\boldsymbol{Q}^{est}_{i,i+1}$ for each pair of states, we can construct the matrix $\widetilde{\boldsymbol{Q}}^{est}$ similarly to (9)

$$\widetilde{\boldsymbol{Q}}^{est} = \text{diag}(\boldsymbol{\mathcal{K}}_0, \boldsymbol{Q}^{est}_{0,1}, ..., \boldsymbol{Q}^{est}_{N-1,N}), \tag{42}$$

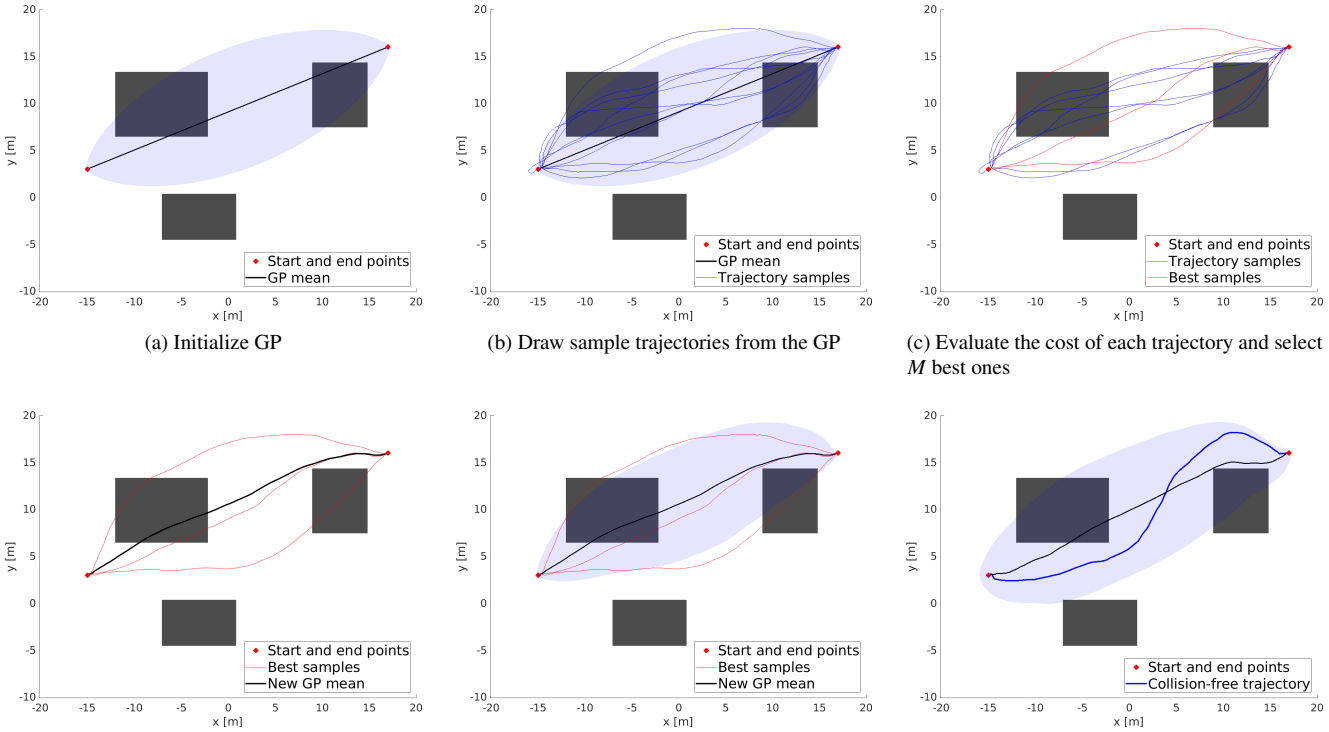
and calculate the precision matrix similarly to (11)

$$[\boldsymbol{\mathcal{K}}^{est}]^{-1} = \boldsymbol{G}^T[\boldsymbol{Q}^{est}]^{-1}\boldsymbol{G}, \tag{43}$$

thus retaining the initial and goal covariances $\boldsymbol{\mathcal{K}}_0, \boldsymbol{\mathcal{K}}_N$ in order to keep the start and goal states fixed and preserving the block-tridiagonal structure.

The constant-velocity model that underpins our GP remains intact through iterations, meaning that smoothness is retained while searching for collision-free trajectory. The shape of the estimated covariance matrix determines relatively how the trajectories sampled in next iteration will deviate from the mean, e.g. in parts of the environment with tight spaces, trajectories will only slightly deviate from the mean, while the parts of trajectories that pass through open space will deviate more. While the described covariance estimation process correctly captures the shape of the covariance, it might have trouble capturing the correct value. This problem stems from normalizing the *indicator value* - we lose information about absolute values of the cost i.e. how close our trajectories actually are to being collision free. This means that such method will have similar covariance values for problems where sampled trajectories are highly in collision and problems where only small parts of sampled trajectories are in collision. Intuitively, we know that if most parts of trajectories are in collision we want larger values of covariance to reinforce exploration, while if only small parts are in collision smaller covariance should be enough to explore around the area locally and find a collision-free trajectory. To obviate this problem, we opt to scale the estimated covariance with the cost value of the calculated mean

$$\boldsymbol{\mathcal{K}} = \alpha f[\boldsymbol{\mu}(t)]\boldsymbol{\mathcal{K}}^{est}, \tag{44}$$



(a) Initialize GP

(b) Draw sample trajectories from the GP

(c) Evaluate the cost of each trajectory and select M best ones

(d) Compute the new GP mean

(e) Compute the new GP covariance

(f) Repeat until a collision-free sample trajectory is found

Figure 2: An example illustrating the steps of the proposed algorithm in a 2D environment with three obstacles. In this simple example the number of sampled trajectories is set as $K = 10$ and the number of chosen best ones $M = 3$.

or equivalently scale the precision matrix

$$\mathcal{K}^{-1} = \frac{1}{\alpha f[\boldsymbol{\mu}(t)]}[\mathcal{K}^{est}]^{-1}, \tag{45}$$

where α is a scalar parameter that regulates the sensitivity of the covariance or precision matrix scaling. Larger mean cost imposes larger covariance values and thus encourages exploration. The current mean can be used in the next iteration as one of the sampled trajectories so that computing its cost does not introduce additional computational burden.

To summarize, we start by sampling K trajectories from a GP initialized with a constant-velocity straight line. For each sampled trajectory we compute its associated cost and select M best ones according to the cost. We then form the new GP mean and precision matrix with (33) and (45). This process is repeated until a collision-free trajectory is found. The described method is summarized in Algorithm 1 and illustrated in Figure 2.

Algorithm 1 Stochastic Trajectory Optimization with GPs

Input: Start and goal states $\boldsymbol{\theta}_0, \boldsymbol{\theta}_N$, a state-dependent cost function $f[\boldsymbol{\theta}^k(t)]$

Precompute: Initial mean $\boldsymbol{\mu}$ and covariance $\mathcal{K}$

for $1 \dots N_{iter}$ **do**
  for $1 \dots K$ **do**
    Sample trajectory $\boldsymbol{\theta}^k(t) \sim \mathcal{GP}(\boldsymbol{\mu}(t), \mathcal{K}(t, t'))$
    Evaluate trajectory cost $f[\boldsymbol{\theta}^k(t)]$
    if $f[\boldsymbol{\theta}^k(t)] = 0$ **then**
      Return collision free trajectory $\boldsymbol{\theta}^k(t)$
    end if
  end for
  From K sampled trajectories take M with lowest cost
  Compute weights λ_m using (34)
  Compute new mean $\boldsymbol{\mu}(t) = \sum_{m=1}^{M}[\lambda_m \boldsymbol{\theta}^m(t)]$
  Compute new precision matrix using (41)-(45)
end for

3.4. Computational Efficiency Remarks

In order to thoroughly explore the environment, our approach requires cost evaluation for relatively many drawn trajectory samples, which naturally leads to a slower computation than the state-of-the-art gradient based methods. However, due to the fact that cost evaluation for each trajectory is independent, the inner for loop in the proposed Algorithm 1 can be parallelized with computational efficiency scaling linearly with the number of processing cores. In our implementation, we exploit this property and parallelize the inner loop on 4 processing cores. A GPU implementation presents an interesting possibility that would allow sampling and cost evaluation for a huge number of trajectories, leading to fast environment exploration and discovering optimal trajectories allowing real-time replanning in dynamic environments [45].

We use GP interpolation for dense collision checking, sim-

ilarly to [23]. Since trajectory *support states* are temporally equidistant and each sampled trajectory is drawn from the same GP, matrices $\mathbf{\Lambda}$ and $\mathbf{\Psi}$ in (19) and (20) can be precomputed, instead of computing them each time interpolation is needed. This provides another significant increase in the proposed method's computational efficiency.

4. Test Results

In this section, we demonstrate the performance of the proposed method. We conducted three simulation experiments to evaluate the proposed method. The method was quantitatively tested in two simulated benchmarks and compared with the state-of-the-art trajectory optimization techniques GPMP2 [23] and STOMP [20]. We also demonstrate our method's capabilities in a qualitative test. In Section 4.1, we demonstrate the improvement of the proposed method over GPMP2 with random restarts and STOMP in solving a 2D maze, which is a suitable benchmark for an optimization-based planner effectiveness at finding a collision-free solution in a haystack of local minima. This experiment aims to show benefits of the proposed stochastic method, which allows for better exploration in comparison to gradient-based methods. In Section 4.2, we show the improvement of the proposed method over prior techniques in finding a collision-free trajectory for a 7 DOF manipulator in cluttered environment. This experiment attests to the benefits of the proposed heteroscedastic GP prior since the environment was set up so that obstacles are placed near the start and goal state. In Section 4.3, we qualitatively demonstrate the ability of the proposed method to find a collision-free trajectory for a 10 DOF mobile manipulator in a cluttered environment. This experiment aims to show general capabilities of the proposed method in motion planning for high DOF robots in complex environments.

In all three experiments, our method was always initialized with a constant-velocity straight line trajectory in the configuration space. Signed distance parameter was set as $\varepsilon = 0.1$ m. We keep the parameter ε, which indicates *safety distance*, relatively small since our primary goal is collision avoidance. Larger values of ε sometimes tend to penalize trajectories that are collision-free but close to obstacles more than trajectories that are in collision at one point but far from obstacles for the most part. The covariance scaling parameter was set as $\alpha = 0.5$, for which we empirically determined that it provides good balance between thorough exploration and fast convergence. In the two conducted benchmarks, along with testing the proposed method we also tested the variant without the proposed covariance estimation procedure, keeping the covariance matrix $\mathcal{K}$ unchanged through iterations, as was done in [32]. The unchanged covariance matrix allows for exhaustive exploration around the mean in each iteration. Changing only mean $\boldsymbol{\mu}$ while keeping the covariance matrix $\mathcal{K}$ unchanged is permitted in the proposed GP framework described in Section 2, as change in $\boldsymbol{\mu}$ can be attributed to some implicitly imposed exogenous input $\boldsymbol{\nu}(t)$ which does not impact covariance.

For GPMP2, we used a straight line initialization as a baseline, and in our experiments we designate to this model as *line*.

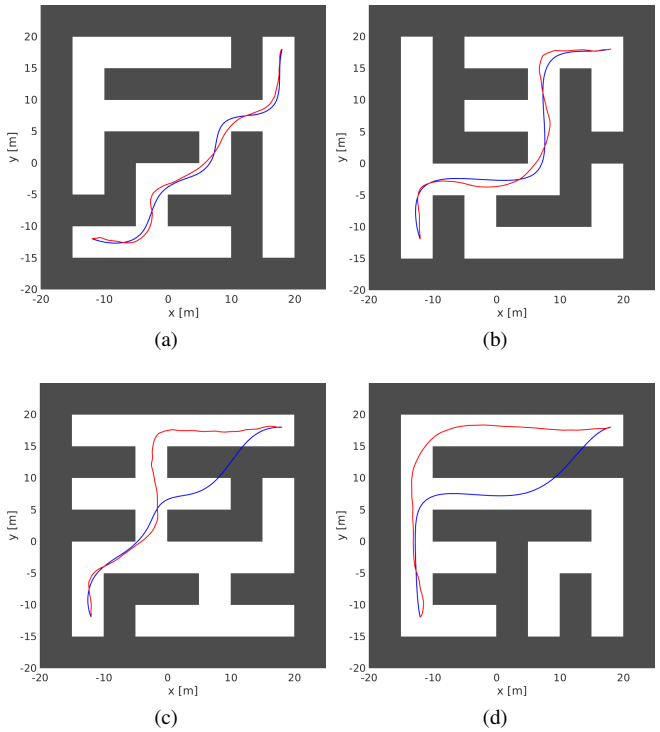


Figure 3: Examples of 4×4 mazes where the proposed approach finds a collision-free solution. Blue line represents trajectory found by GPMP2, while red line represents trajectory found by our method. Slight undulation of trajectories obtained by the proposed method is due to the criterion of finding a collision-free trajectory, unlike the GPMP2 criterion which explicitly encourages smoothness.

Since GPMP2 always converges very quickly, but often fails in cluttered environments due to infeasible local minima, we also employed random restarts, which is a commonly used method to tackle the local minima problems in gradient-based trajectory optimization methods [18]. In this technique, the optimizer is first initialized with a straight-line and then, on failure, reinitialized with a random trajectory. Our implementation samples the random restart trajectory from a homoscedastic GP, similarly to [24]. We designate to this model as *rr*. For STOMP, we also used a constant-velocity straight line initialization. The parameter that determines the number of noisy trajectories generated at each STOMP iteration was set to $K = 5$, which was demonstrated to achieve good performance in prior work [20].

We used the GPMP2 C++ library [23, 46] and its respective MATLAB toolbox based on the GTSAM C++ library [26], while we used our own implementation of STOMP. Experiments were performed on a system with a 2.8-GHz Intel Core i7-7700HQ processor and 16 GB of RAM.

4.1. The Maze Benchmark

The maze benchmark, appropriate for quantitative evaluation, consisted of 1000 synthetic environments created by the Wilson's algorithm [47], which generates uniformly sampled mazes with a single solution (i.e. perfect mazes). Mazes were generated on grids with sizes of 3×3, 4×4 and 5×5 and afterwards inflated to realistic dimensions. While the 2D maze

problem is generally suitable for grid-based or sampling-based motion planning approaches which achieve 100% success rate, it can be used to measure an optimization-based planner's effectiveness at finding the unique collision-free solution in a cluttered environment.

For each maze environment, we plan motion for a 2D holonomic circular robot with the radius of 0.5 m. For both variants of our method, we chose the number of sampled trajectories $K \in [200, 400]$, while the number of best trajectories chosen for the weighted mean in each iteration was set to $M = 3$. Although these parameters may seem disproportionate, choosing a large K ensures exhaustive exploration, while a small M induces drastic changes in the GP mean $\boldsymbol{\mu}$ between iterations, which helps in finding the solution faster in complex environments. We set the total trajectory time (i.e. the timespan in which the robot moves from the start to goal state) to $t_{\text{total}} = 20\,\text{s}$, while time-varying covariance matrix of the white noise governing the heteroscedastic GP was calculated as $\boldsymbol{Q}_c(t) = (t - \frac{t_{\text{total}}}{2})^2 \boldsymbol{I}$, which generates a parabola with its vertex at the midpoint of the trajectory. For GPMP2 we set $\epsilon = 2$ and $Q_c = 1$ according to the Matlab toolbox 2D example. To achieve the best performance on our benchmark, we tuned the GPMP2 algorithm, testing it with the parameter σ_{obs} in range $[0.01, 0.1]$ with increments of 0.01 and in range $[0.1, 1]$ with increments of 0.1. For the 3x3 benchmark, reported GPMP2 performance was achieved with $\sigma_{obs} = 0.3$, while for the 4x4 and 5x5 benchmark it was achieved with $\sigma_{obs} = 0.1$.

For the proposed method and GPMP2, the trajectory was parameterized with $N = 10$ *support states* and 5 interpolation steps in-between for which the trajectory cost is evaluated. While larger number of *support states* N would likely lead to somewhat better performance, it would negatively impact computation time. GPMP2 demonstrated that using less *support states* with more interpolated states inbetween provides optimal performance and this translates to the proposed method since we utilize similar underlying GP framework. For STOMP, the trajectory was parameterized with $N = 50$ discrete states. We set the maximum runtime for our algorithm, STOMP and random restarts as $t_{\text{max}} = 1\,\text{s}$, with one exception where we set $t_{\text{max}} = 2\,\text{s}$ in order to investigate the ability of our algorithm to find solutions given more time. We measure the number of mazes solved (success rate) and the execution time. Figure 3 depicts examples of four 4x4 mazes with obtained trajectories with the proposed method and GPMP2. We do not depict trajectories obtained with STOMP since it consistently achieved both the worst success rate and execution time in the maze experiment. Table 1 shows the experiment results in detail.

For every maze complexity level, the proposed method with covariance estimation achieved better results (higher success rate and lower execution time) than the variant without covariance estimation when the maximum algorithm runtime was set to $t_{\text{max}} = 1\,\text{s}$. While calculating the new covariance matrix in each iteration takes additional computation time, the method converges faster, i.e. it needs less iterations to find a collision-free solution. On average, it required approximately 2.5× less iterations. If a trajectory that is close to being collision-free is found, the estimated covariance is rather small and thus the focus shifts to exploring the area around that trajectory locally. Without covariance estimation, the initial covariance carries exhaustive exploration of the space in each iteration, which can be sample inefficient since the algorithm will sample trajectories through the whole environment even when the mean is close to a collision-free trajectory. In other words, covariance estimation reduces the search space and thus leads to faster convergence. However, when given additional computation time (algorithm runtime set to $t_{\text{max}} = 2\,\text{s}$), the proposed approach without covariance estimation achieves marginally better success rate in all three maze types. This stems from the aforementioned fact that the covariance estimation reduces the search space which can sometimes lead to exploring locally around local minima. The variant without covariance estimation is more reliant on "lucky" sampling, it possesses better ability to stumble upon a collision-free trajectory given enough time. It is also worth noting that, through covariance estimation, the proposed method approximates the covariance of the underlying distribution of collision-free trajectories. It is thus less reliant on finely tuned initialization than the variant without covariance estimation, where providing too large initial covariance matrix $\boldsymbol{\mathcal{K}}$ leads to degraded performance. Without covariance estimation, sampling $K = 200$ and $K = 400$ trajectories in each iteration lead to similar results; however, with covariance estimation, sampling $K = 200$ trajectories proved to find the solution significantly faster than sampling $K = 400$ trajectories. This attests to the benefits of covariance estimation where the algorithm adapts to the given environment through iterations and is more sample-efficient.

STOMP achieved both the worst success rate and execution time for every maze complexity level. While STOMP does not require gradient and samples noisy trajectories, it tries to combine sampled trajectories to produce an update rule that produces smooth trajectories, but often fails to explore the environment. Unlike the proposed method, if one of sampled noisy trajectories is collision-free, it is not used as a solution, but utilized to produce an update to the mean trajectory. In a complex environment such as maze, this often leads to the reuslting trajectory being in collision, even if a collision-free trajectory was found during the sampling process. While the GPMP2 without random restarts has almost an order of magnitude faster execution time than every other tested method, it has significantly worse success rate than random restarts and the proposed method for every maze complexity level. This was expected, as it is a classical gradient-based trajectory optimization algorithm meaning that it is prone to local minima which are omnipresent in a maze environment. The proposed approach also outperformed the GPMP2 with random restarts for every maze complexity level. For the mazes created from a 3×3 grid, our algorithm managed similarly high success rate with twice as fast computation. For the mazes created from a 4×4 grid our algorithm vastly outperformed random restarts, having notably higher success rate with similar reported times. The most complex mazes created from a 5×5 grid demonstrated the inability of gradient-based methods to find solutions in environments plagued with multitude of local minima, while the proposed method achieved moderate success. Note that the reported execution time for our

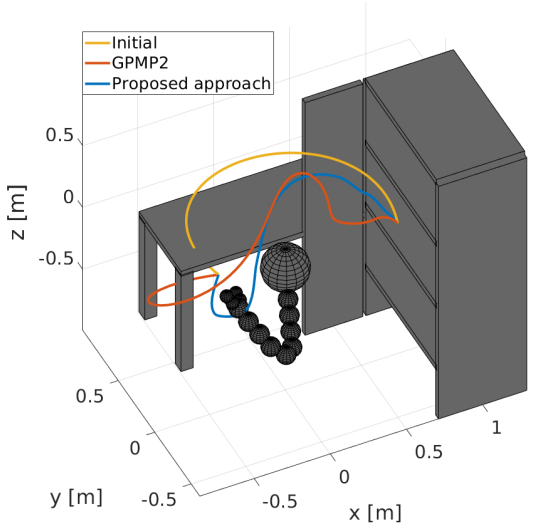


Figure 4: A simulated WAM robotic arm in an environment featuring a table and a drawer cabinet. Plotted lines depict the end effector trajectories. This is an example where the proposed approach finds a collision free solution, while GPMP2 converges to the infeasible local minimum. Initial straight-line trajectory in configuration space is also shown.

algorithm is the actual time it took to compute on all cores, and not the sum of times over them.

4.2. The Robot Arm Planning Benchmark

The robot arm planning benchmark consisted of a simulated WAM robotic arm in an environment featuring a table and a drawer cabinet. We conducted 20 unique planning experiments, all with different start and goal states with starting points being under the table and end states being inside the cabinet. This set of problems is not particularly difficult since most of the states are initially collision free; however, it was set up to accentuate the proneness of the homoscedastic GP motion planning methods to get stuck in local minima near start or goal states.

In this benchmark we tested both variants of the proposed method with heteroscedastic and homoscedastic GP priors in order to demonstrate the benefits of heteroscedasticity. For both variants of our method, we chose the number of sampled trajectories $K = 400$, while the number of best trajectories chosen for the weighted mean in each iteration was set as $M = 3$. We set the total trajectory time to $t_{\text{total}} = 20\,\text{s}$. For homoscedastic cases we chose $\boldsymbol{Q}_c = 0.02\boldsymbol{I}$, while for a heteroscedastic GPs we calculated $\boldsymbol{Q}_c(t) = 0.01(t - \frac{t_{\text{total}}}{2})^2\boldsymbol{I}$. The optimal coefficients of $\boldsymbol{Q}_c$ were found empirically, as is the case in other applications of the GP trajectory representation [22, 36]. We utilized grid search to tune the GPMP2 algorithm, searching for both ϵ in range $[0.01, 0.1]$ with increments of 0.1 and σ_{obs} in range $[0.1, 1]$ with increments of 0.1. The reported GPMP2 performance on this benchmark was achieved with parameters $\epsilon = 0.02$ and $\sigma_{obs} = 0.2$. For the proposed method and GPMP2, the trajectory was parameterized with $N = 10$ equidistant *support states* and 10 interpolation steps in-between for which the trajectory cost was evaluated. For STOMP, the trajectory was parameterized with $N = 100$ discrete states. We again set the fixed time budget for our algorithm, STOMP and random restarts as $t_{\text{max}} = 1\,\text{s}$ and measured the success rate and the execution time. Due to stochastic nature of the results, we repeated every planning experiment with our method and with STOMP 20 times to correctly assess the required computation time. We measured the average success rate and the average algorithm execution time. The results of the experiment are shown in Table 1, where we can see that while the baseline GPMP2 often fails, the stochasticity introduced by random restarts helps in achieving higher success rates. An example of a case where our method finds a solution and GPMP2 fails is depicted in Figure 4. Interestingly, when the proposed approach with covariance estimation was initialized with a homoscedastic GP, the covariance estimation guided the GP towards more exploration near the start and end points, leading to better success rate than a homoscedastic GP without covariance estimation, albeit requiring additional computational time to converge. Both variants of the proposed method with heteroscedastic GPs achieved a perfect score within the fixed time budget, thus demonstrating the advantage of the proposed heteroscedastic prior for solving problem instances with obstacles near the start or end points of the robot trajectory.

4.3. Mobile Manipulator Trajectory Planning

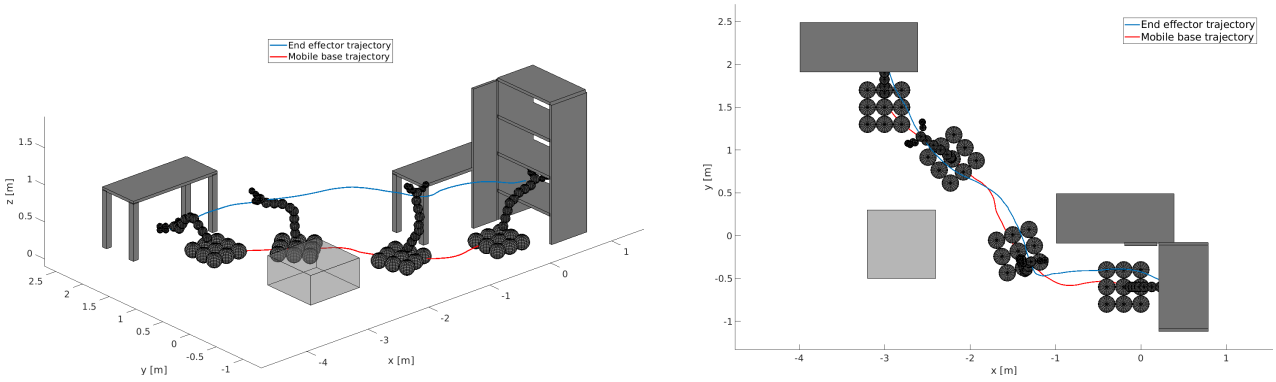
To demonstrate the ability of the proposed method to find a collision-free trajectory for high DOF robots in complex environments, we conducted a mobile manipulator trajectory planning experiment. The simulated mobile manipulator consisted of a simulated WAM robotic arm mounted on top of an omnidirectional platform in an environment featuring two tables, a drawer cabinet and a generic static obstacle. The system has 10 DOF corresponding to mobile platform's 3 DOF pose and robot arm's 7 joint angles. We planned the trajectory of the whole system, simultaneously planning the motion of the arm and the platform.

We chose the number of sampled trajectories $K = 200$, while the number of best trajectories chosen for the weighted mean in each iteration was set as $M = 3$. We set the total trajectory time $t_{\text{total}} = 10\,\text{s}$. We modelled the time-varying covariance matrix of the white noise governing the heteroscedastic GP as an anisotropic diagonal matrix. The elements of the matrix associated with the omnidirectional platform were calculated as $\boldsymbol{Q}_c(t) = 0.1(t - \frac{t_{\text{total}}}{2})^2$, while the elements associated with the robot arm states were calculated as in the robot arm planning benchmark $\boldsymbol{Q}_c(t) = 0.01(t - \frac{t_{\text{total}}}{2})^2$. The coefficients of $\boldsymbol{Q}_c(t)$ were found empirically. The noise induced for a 2D mobile platform was much smaller in comparison to the 2D maze benchmark since start and end points of this problem were much closer to each other and the environment required less exploration. In such case, smaller $\boldsymbol{Q}_c(t)$ should still lead to a collision-free solution, but the resulting trajectory will be smoother. The trajectory was parameterized with $N = 20$ equidistant *support states* and 10 interpolation steps in-between for which the trajectory cost was evaluated. To assess the required computation time, we repeated this experiment 30 times. Every take resulted in successfully finding a collision-free trajectory and the average algorithm execution time was

Table 1: Success rate (%) / average execution time (ms) on maze and robot arm planning benchmarks.

Maze	The proposed approach with covariance estimation			The proposed approach without covariance estimation			GPMP2		STOMP
	$K = 400$, $t_{max} = 2$ s	$K = 400$, $t_{max} = 1$ s	$K = 200$, $t_{max} = 1$ s	$K = 400$, $t_{max} = 2$ s	$K = 400$, $t_{max} = 1$ s	$K = 200$, $t_{max} = 1$ s	rr	line	
3x3	92 / 107	91.5 / 97	90.2 / 55	95.2 / 197	92.9 / 171	89.0 / 160	89.2 / 96	55.3 / 28	51.4 / 600
4x4	72.4 / 240	70.9 / 209	66.9 / 128	79.1 / 489	66.9 / 324	61.8 / 297	58.1 / 298	20.5 / 40	9.5 / 786
5x5	42.7 / 520	37.3 / 381	37.4 / 299	43.8 / 858	26.7 / 493	26.3 / 429	19.8 / 273	9.6 / 36	0.2 / 742

Arm	Heteroscedastic $K = 400$, $t_{max} = 1$ s	Homoscedastic $K = 400$, $t_{max} = 1$ s	Heteroscedastic $K = 400$, $t_{max} = 1$ s	Homoscedastic $K = 400$, $t_{max} = 1$ s	rr	line	
	100 / 494	90 / 680	100 / 458	75 / 569	85 / 372	65 / 84	30 / 832



(a) 3d view (b) 2d view

Figure 5: An example of a trajectory obtained with the proposed method on a simulated mobile manipulator in a static environment.

788 ms, while the worst case computation time was 1362 ms. The result of this experiment is shown in Figure 5. The experiment demonstrated the ability of the proposed approach to find collision-free trajectories for a mobile manipulator in a cluttered environment.

5. Conclusion

In this paper, we have presented a stochastic optimization method for trajectory planning, representing robot trajectories as samples from a continuous-time GP. By introducing the heteroscedasticity of the underlying GP, we were able to generate trajectory priors better suited for collision avoidance in motion planning problems. We derived a cross-entropy based derivative-free stochastic optimization method, utilizing importance sampling in order to contend with the local minima problem present in trajectory optimization methods. Due to the sparsity of the employed GP framework, we were able to efficiently estimate the GP covariance at each iteration. We evaluated our method on three simulation scenarios: a maze benchmark, a 7 DOF robot arm planning benchmark and a 10 DOF mobile manipulator trajectory planning example. In a maze benchmark, the proposed method yielded a more thorough exploration of the solution space in complex environments than GPMP2, while having comparable execution time. The robot arm planning benchmark demonstrated the benefits of heteroscedastic GP, while the mobile manipulator planning experiments showed that the proposed method is capable of efficiently tackling high-dimensional trajectory planning problems.

The subject of future research is to utilize the strong exploration capacity of our algorithm for finding different homotopy classes in the environment, which could be useful for replanning when a given trajectory becomes infeasible due to changes in the environment. Another possible direction is to exploit the parallelization capability of our algorithm with a GPU implementation. The proposed approach could also be coupled with learning from demonstration algorithms for generalized skill learning and reproduction.

6. References

[1] P. E. Hart, N. J. Nilsson, B. Raphael, A formal basis for the heuristic determination of minimum cost paths, IEEE Transactions on Systems Science and Cybernetics 4 (2) (1968) 100–107.

[2] J. Ng, T. Bräunl, Performance comparison of bug navigation algorithms, Journal of Intelligent and Robotic Systems 50 (1) (2007) 73–84.

[3] J.-C. Latombe, Robot motion planning, Vol. 124, Springer Science & Business Media, 2012.

[4] T. Lozano-Pérez, M. A. Wesley, An algorithm for planning collision-free paths among polyhedral obstacles, Communications of the ACM 22 (10) (1979) 560–570.

[5] C. Ó'Dúnlaing, C. K. Yap, A "retraction" method for planning the motion of a disc, Journal of Algorithms 6 (1) (1985) 104–111.

[6] A. Stentz, Optimal and efficient path planning for partially known environments, in: Intelligent Unmanned Ground Vehicles, Springer, 1997, pp. 203–220.

[7] A. Stentz, et al., The focussed dˆ* algorithm for real-time replanning, in: IJCAI, Vol. 95, 1995, pp. 1652–1659.

[8] S. Koenig, M. Likhachev, Fast replanning for navigation in unknown terrain, IEEE Transactions on Robotics 21 (3) (2005) 354–363.

[9] M. Likhachev, G. J. Gordon, S. Thrun, Ara*: Anytime a* with provable bounds on sub-optimality, in: Advances in neural information processing systems, 2004, pp. 767–774.

[10] S. M. LaValle, Planning algorithms, Cambridge university press, 2006.

[11] L. E. Kavraki, P. Svestka, J.-C. Latombe, M. H. Overmars, Probabilistic roadmaps for path planning in high-dimensional configuration spaces, IEEE Transactions on Robotics and Automation 12 (4) (1996) 566–580.

[12] S. Karaman, E. Frazzoli, Sampling-based algorithms for optimal motion planning, The International Journal of Robotics Research 30 (7) (2011) 846–894.

[13] J. J. Kuffner, S. M. LaValle, Rrt-connect: An efficient approach to single-query path planning, in: IEEE International Conference on Robotics and Automation (ICRA), Vol. 2, 2000, pp. 995–1001.

[14] S. M. LaValle, J. J. Kuffner Jr, Rapidly-exploring random trees: Progress and prospects, Algorithmic and computational robotics (2000).

[15] R. Kindel, D. Hsu, J.-C. Latombe, S. Rock, Kinodynamic motion planning amidst moving obstacles, in: IEEE International Conference on Robotics and Automation (ICRA), Vol. 1, 2000, pp. 537–543.

[16] S. M. LaValle, J. J. Kuffner Jr, Randomized kinodynamic planning, The international journal of robotics research 20 (5) (2001) 378–400.

[17] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, P. Abbeel, Motion planning with sequential convex optimization and convex collision checking, The International Journal of Robotics Research 33 (9) (2014) 1251–1270.

[18] N. Ratliff, M. Zucker, J. A. Bagnell, S. Srinivasa, Chomp: Gradient optimization techniques for efficient motion planning, in: IEEE International Conference on Robotics and Automation (ICRA), 2009, pp. 489–494.

[19] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, S. S. Srinivasa, Chomp: Covariant hamiltonian optimization for motion planning, The International Journal of Robotics Research 32 (9-10) (2013) 1164–1193.

[20] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, S. Schaal, Stomp:stochastic trajectory optimization for motion planning, in: IEEE International Conference on Robotics and Automation (ICRA), 2011, pp. 4569–4574.

[21] J. Schulman, J. Ho, A. X. Lee, I. Awwal, H. Bradlow, P. Abbeel, Finding locally optimal, collision-free trajectories with sequential convex optimization., in: Robotics: Science and Systems, 2013.

[22] M. Mukadam, X. Yan, B. Boots, Gaussian process motion planning, in: IEEE International Conference on Robotics and Automation (ICRA), 2016, pp. 9–15.

[23] J. Dong, M. Mukadam, F. Dellaert, B. Boots, Motion planning as probabilistic inference using gaussian processes and factor graphs, in: Robotics: Science and Systems, 2016.

[24] E. Huang, M. Mukadam, Z. Liu, B. Boots, Motion planning with graph-based trajectories and gaussian process inference, in: IEEE International Conference on Robotics and Automation (ICRA), 2017, pp. 5591–5598.

[25] M. Mukadam, J. Dong, X. Yan, F. Dellaert, B. Boots, Continuous-time gaussian process motion planning via probabilistic inference, The International Journal of Robotics Research 37 (11) (2018) 1319–1340.

[26] F. Dellaert, Factor graphs and gtsam: A hands-on introduction, Tech. rep., Georgia Institute of Technology (2012).

[27] M. A. Rana, M. Mukadam, S. R. Ahmadzadeh, S. Chernova, B. Boots, Towards robust skill generalization: Unifying learning from demonstration and motion planning, in: Conference on Robot Learning (CoRL), 2017, pp. 109–118.

[28] F. Marić, O. Limoyo, L. Petrović, T. Ablett, I. Petrović, J. Kelly, Fast manipulability maximization using continuous-time trajectory optimization, arXiv preprint arXiv:1908.02963 (2019).

[29] M. Mukadam, J. Dong, F. Dellaert, B. Boots, Simultaneous trajectory estimation and planning via probabilistic inference, in: Robotics: Science and Systems, 2017.

[30] J.-J. Kim, J.-J. Lee, Trajectory optimization with particle swarm optimization for manipulator motion planning, IEEE Transactions on Industrial Informatics 11 (3) (2015) 620–631.

[31] M. Kobilarov, Cross-entropy randomized motion planning, in: Robotics: Science and Systems, 2012.

[32] L. Petrović, J. Peršić, M. Seder, I. Marković, Stochastic optimization for trajectory planning with heteroscedastic gaussian processes, in: European Conference on Mobile Robots (ECMR), IEEE, 2019, pp. 1–6.

[33] T. D. Barfoot, State Estimation for Robotics, Cambridge University Press, 2017.

[34] P. S. Maybeck, Stochastic models, estimation, and control, Academic press, 1982.

[35] C. E. Rasmussen, Gaussian processes in machine learning, in: Summer School on Machine Learning, Springer, 2003, pp. 63–71.

[36] T. D. Barfoot, C. H. Tong, S. Särkkä, Batch continuous-time trajectory estimation as exactly sparse gaussian process regression., in: Robotics: Science and Systems, 2014.

[37] S. Anderson, T. D. Barfoot, C. H. Tong, S. Särkkä, Batch nonlinear continuous-time trajectory estimation as exactly sparse gaussian process regression, Autonomous Robots 39 (3) (2015) 221–238.

[38] J. Persic, L. Petrovic, I. Markovic, I. Petrovic, Spatio-temporal multisensor calibration based on gaussian processes moving object tracking, arXiv preprint arXiv:1904.04187 (2019).

[39] T. D. Barfoot, J. R. Forbes, D. Yoon, Exactly sparse gaussian variational inference with application to derivative-free batch nonlinear state estimation, arXiv preprint arXiv:1911.08333 (2019).

[40] F. Broussolle, State estimation in power systems: detecting bad data through the sparse inverse matrix method, IEEE Transactions on Power Apparatus and Systems (3) (1978) 678–682.

[41] J. E. Gentle, Computational statistics, Vol. 308, Springer, 2009.

[42] P.-T. De Boer, D. P. Kroese, S. Mannor, R. Y. Rubinstein, A tutorial on the cross-entropy method, Annals of operations research 134 (2005) 19–67.

[43] R. Y. Rubinstein, D. P. Kroese, The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning, Springer Science & Business Media, 2013.

[44] B. Amos, D. Yarats, The differentiable cross-entropy method, arXiv preprint arXiv:1909.12830 (2019).

[45] C. Park, J. Pan, D. Manocha, Real-time optimization-based planning in dynamic environments using gpus, in: IEEE International Conference on Robotics and Automation, 2013, pp. 4090–4097.

[46] J. Dong, B. Boots, F. Dellaert, Sparse gaussian processes for continuous-time trajectory estimation on matrix lie groups, arXiv preprint arXiv:1705.06020 (2017).

[47] D. B. Wilson, Generating random spanning trees more quickly than the cover time, in: STOC, Vol. 96, Citeseer, 1996, pp. 296–303.