Stochastic Optimization for Trajectory Planning with Heteroscedastic Gaussian Processes

Luka Petrović, Juraj Peršić, Marija Seder, Ivan Marković*

***Abstract*—Trajectory optimization methods for motion planning attempt to generate trajectories that minimize a suitable objective function. Such methods efficiently find solutions even for high degree-of-freedom robots. However, a globally optimal solution is often intractable in practice and state-of-the-art trajectory optimization methods are thus prone to local minima, especially in cluttered environments. In this paper, we propose a novel motion planning algorithm that employs stochastic optimization based on the cross-entropy method in order to tackle the local minima problem. We represent trajectories as samples from a continuous-time Gaussian process and introduce heteroscedasticity to generate powerful trajectory priors better suited for collision avoidance in motion planning problems. Our experimental evaluation shows that the proposed approach yields a more thorough exploration of the solution space and a higher success rate in complex environments than a current Gaussian process based state-of-the-art trajectory optimization method, namely GPMP2, while having comparable execution time.**

***Index Terms*—motion planning, trajectory optimization, gaussian processes, stochastic optimization**

I. Introduction

Motion planning is an indispensable skill 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 dependent on the task, robot or environment. Algorithms that can be executed in real time are highly encouraged, mostly because they allow fast replanning in response to environment changes. The majority of methods in the domain of high-dimensional motion planning can be roughly divided into two categories: sampling-based approaches and trajectory optimization approaches.

The central tenet of sampling-based approaches [1]–[3] is the idea of connecting points randomly sampled from the free configuration space. Due to the underlying random sampling, these approaches exhibit probabilistic completness and fast exploration of the environment. However, sampling based planners can be computationally inefficient for high-dimensional problems with challenging constraints and often require a post-processing step to smooth and shorten the computed trajectories. Furthermore, considerable computational effort is spent on exploring the portions of the configuration space that might not be relevant to the task.

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

*Authors are with the University of Zagreb Faculty of Electrical Engineering and Computing, Laboratory for Autonomous Systems and Mobile Robotics, Croatia. {luka.petrovic, juraj.persic, marija.seder, ivan.markovic}@fer.hr

A significant amount of recent work has focused on trajectory optimization and related problems. Trajectory optimization methods start with an initial trajectory and then minimize an objective function in order to optimize the trajectory. Covariant Hamiltonian optimization for motion planning (CHOMP) [4], [5] is the seminal work in modern trajectory optimization. It utilizes a precomputed signed distance field for fast collision checking and uses covariant gradient descent to minimize obstacle and smoothness costs. Stochastic trajectory optimization for motion planning (STOMP) algorithm [6] 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 for reasoning about fine resolution obstacle representations and finding feasible solutions when there are many constraints. TrajOpt [7], [8] algorithm formulates motion planning 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 parametrized trajectory or post-processing of the trajectory might still be needed thus increasing computation time.

The Gaussian process (GP) motion planning family of algorithms [9]–[12] employs continous-time trajectory representation in order to overcome the computational cost incurred by using large number of states. The GPMP algorithm [9] parametrizes the trajectory with a few support states and then uses GP interpolation to query the trajectory at any time of interest. The GPMP2 algorithm [10] 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 optimization tools developed by the simultaneous localization and mapping (SLAM) community [13] to generate fast solutions.

Altough trajectory optimization methods generate fast solutions in high-dimensional spaces, they have limited exploration ability and in complex environments often converge to the infeasible local minima. In this paper, we propose a gradient-free stochastic optimization method for trajectory planning with continous time GP trajectory representations. We consider a trajectory as a sample from a GP and introduce heteroscedasticity to generate powerful trajectory priors better suited for collision avoidance in motion planning problems. The proposed optimization method relies on importance sampling

and is a derivative of the cross-entropy optimization method [14]. 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 throughly explore the environment. We evaluated our method in simulations and compared it to GPMP2 – a state-of-the-art gradient-based, in constrast to the proposed gradient-free, trajectory optimization method. The results show that the proposed method yields a higher success rate in complex environments with comparable execution time.

II. Heteroscedastic Gaussian Processes for Motion Planning

A. The Gaussian Process Trajectory Representations

Consider a continuous-time trajectory as a sample from a vector-valued continuous-time Gaussian process (GP)

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

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. We employ a structured kernel belonging to a special class of GP priors generated by a linear time-varying stochastic differential equation (LTV-SDE)

$$\dot{\boldsymbol{\theta}}(t) = \boldsymbol{F}(t)\boldsymbol{\theta}(t) + \boldsymbol{v}(t) + \boldsymbol{L}(t)\boldsymbol{w}(t), \tag{2}$$

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

$$\boldsymbol{w}(t) \sim \mathcal{GP}(\boldsymbol{0}, \boldsymbol{Q}_c(t)\delta(t - t')), \tag{3}$$

where $\boldsymbol{Q}_c(t)$ is an isotropic time-varying power-spectral density matrix, $\boldsymbol{Q}_c(t) = Q_c(t)\boldsymbol{I}$. A similar dynamical system has been utilized in estimation [15], [16], calibration [17] and planning [12], [18]. However, the crucial difference in our approach is that the covariance $\boldsymbol{Q}_c(t)$ is time-varying and consequently generates a heteroscedastic GP [19]. We discuss benefits of this approach in Section II-E.

The mean and the covariance of the GP generated by the LTV-SDE given in (2) evaluate to

$$\widetilde{\boldsymbol{\mu}}(t) = \boldsymbol{\Phi}(t, t_0)\boldsymbol{\mu}_0 + \int_{t_0}^{t} \boldsymbol{\Phi}(t, s)\boldsymbol{v}(s)\,\mathrm{d}s, \tag{4}$$

$$\widetilde{\boldsymbol{\mathcal{K}}}(t, t') = \boldsymbol{\Phi}(t, t_0)\boldsymbol{\mathcal{K}}_0\boldsymbol{\Phi}(t', t_0)^T + \int_{\mathrm{t_0}}^{\min(t,t')} \boldsymbol{\Phi}(t, s)\boldsymbol{L}(s)\boldsymbol{Q}_c(s)\boldsymbol{L}(s)^T\boldsymbol{\Phi}(t', s)^T\,\mathrm{d}s, \tag{5}$$

where $\boldsymbol{\mu}_0$ and $\boldsymbol{\mathcal{K}}_0$ are the initial mean and covariance of the first state, and $\boldsymbol{\Phi}(t, s)$ is the state transition matrix [15].

B. GP Prior for Motion Planning

Due to Markov property of the LTV-SDE in (2), the inverse kernel matrix $\widetilde{\boldsymbol{\mathcal{K}}}^{-1}$ is exactly sparse block tridiagonal [15]:

$$\widetilde{\boldsymbol{\mathcal{K}}}^{-1} = \widetilde{\boldsymbol{F}}^{-T}\widetilde{\boldsymbol{Q}}^{-1}\widetilde{\boldsymbol{F}}^{-1}, \tag{6}$$

where

$$\widetilde{\boldsymbol{F}}^{-1} = \begin{bmatrix} \mathbf{1} & 0 & ... & 0 & 0 \\ -\boldsymbol{\Phi}(t_1, t_0) & \mathbf{1} & ... & 0 & 0 \\ 0 & -\boldsymbol{\Phi}(t_2, t_1) & \ddots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \mathbf{1} & 0 \\ 0 & 0 & ... & -\boldsymbol{\Phi}(t_N, t_{N-1}) & \mathbf{1} \end{bmatrix} \tag{7}$$

and

$$\widetilde{\boldsymbol{Q}}^{-1} = \mathrm{diag}(\boldsymbol{\mathcal{K}}_0^{-1}, \boldsymbol{Q}_{0,1}^{-1}, ..., \boldsymbol{Q}_{N-1,N}^{-1}) \tag{8}$$

with

$$\boldsymbol{Q}_{a,b} = \int_{t_a}^{t_b} \boldsymbol{\Phi}(t_b, s)\boldsymbol{L}(s)\boldsymbol{Q}_c\boldsymbol{L}(s)^T\boldsymbol{\Phi}(t_b, s)^T\,\mathrm{d}s. \tag{9}$$

The GP defined by mean and covariance in (4) and (5) 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 $\boldsymbol{\mathcal{K}}_N$. This can be accomplished while still preserving the sparsity of the kernel matrix [12]

$$\begin{aligned} \boldsymbol{\mathcal{K}}^{-1} &= \begin{bmatrix} & \widetilde{\boldsymbol{F}}^{-1} & \\ \mathbf{0} & \dots & \mathbf{0} \quad \mathbf{I} \end{bmatrix}^{\top} \begin{bmatrix} \widetilde{\boldsymbol{Q}}^{-1} & \\ & \boldsymbol{\mathcal{K}}_N^{-1} \end{bmatrix} \begin{bmatrix} & \widetilde{\boldsymbol{F}}^{-1} & \\ \mathbf{0} & \dots & \mathbf{0} \quad \mathbf{I} \end{bmatrix} \\ &= \boldsymbol{F}^T\boldsymbol{Q}^{-1}\boldsymbol{F}. \end{aligned} \tag{10}$$

The mean vector $\boldsymbol{\mu} = \begin{bmatrix}\boldsymbol{\mu}_0 \dots \boldsymbol{\mu}_N\end{bmatrix}^T$ and the kernel matrix given in (10) fully determine a continuous-time trajectory defined by (1) that we employ for motion planning.

C. Fast GP interpolation

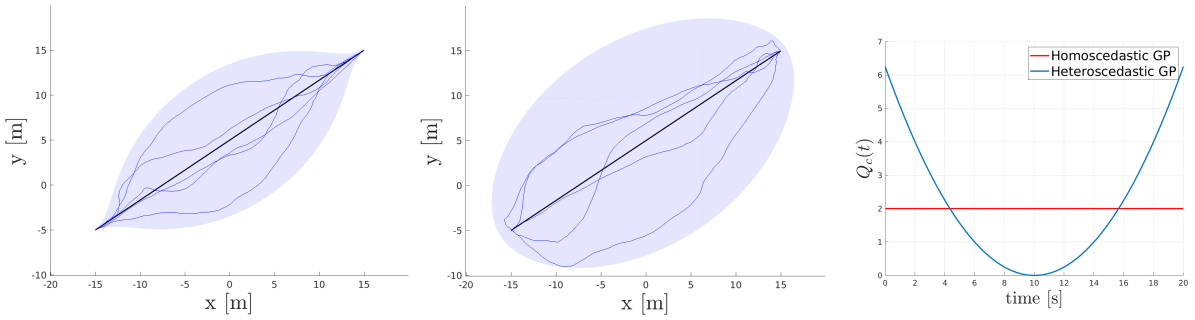
A major benefit of modelling continuous-time trajectory in motion planning with GPs is the possibility to query the planned state $\boldsymbol{\theta}(\tau)$ at any time of interest τ, and not only at discrete time instants. The kernel matrix defined in (10) allows for computationally efficient, structure-exploiting GP interpolation with $\mathcal{O}(1)$ complexity. State $\boldsymbol{\theta}(\tau)$ at $\tau \in [t_i, t_{i+1}]$ is a function only of its neighboring states [10]

$$\boldsymbol{\theta}(\tau) = \boldsymbol{\mu}(\tau) + \boldsymbol{\Lambda}(\tau)(\boldsymbol{\theta}_i - \boldsymbol{\mu}_i) + \boldsymbol{\Psi}(\tau)(\boldsymbol{\theta}_{i+1} - \boldsymbol{\mu}_{i+1}), \tag{11}$$

$$\boldsymbol{\Lambda}(\tau) = \boldsymbol{\Phi}(\tau, t_i) - \boldsymbol{\Psi}(\tau)\boldsymbol{\Phi}(t_{i+1}, t_i), \tag{12}$$

$$\boldsymbol{\Psi}(\tau) = \boldsymbol{Q}_{i,\tau}\boldsymbol{\Phi}(t_{i+1}, \tau)^T\boldsymbol{Q}_{i,i+1}^{-1}, \tag{13}$$

where $\boldsymbol{Q}_{a,b}$ is given in (9). 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.



(a) Homoscedastic GP (b) Heteroscedastic GP (c) Covariance parameter of the white noise process $\boldsymbol{w}(t)$ through time

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

D. 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 (2), where the Markovian state $\boldsymbol{\theta}(t)$ consists of position and velocity in configuration space with the following system matrices

$$\boldsymbol{F}(t) = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, \boldsymbol{L}(t) = \begin{bmatrix} \mathbf{0} \\ \mathbf{I} \end{bmatrix}. \tag{14}$$

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 [12].

E. Benefits of heteroscedasticity

State-of-the-art methods based on GPs [10]–[12] minimize the sum of two costs: an obstacle cost and a smoothness cost which 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 the deviation of the trajectory from the prior mean more. 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 technique used in [10]–[12] will have difficulty escaping the collision since the incurred smoothness cost can become too large commensurate with obstacle cost. Figure 1a shows that sample trajectories drawn from the 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 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 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. In our experience, modeling $Q_c(t)$ as a parabola with high values at the beginning and end, and the lowest value at the temporal midpoint of the trajectory, leads to GPs that have these 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 (6), 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 some motion planning problem. 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.

III. Proposed Stochastic Trajectory Optimization

Formally, the goal of trajectory optimization is to find a smooth, collision-free trajectory through the configuration space between two end points. Prior work in this area 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). Therefore, our method starts with the following optimization problem:

$$\begin{aligned} \underset{\boldsymbol{\theta}(t)}{\text{minimize}} \quad & f[\boldsymbol{\theta}(t)] \\ \text{subject to} \quad & \boldsymbol{\theta}(t) \sim \mathcal{GP}(\boldsymbol{\mu}(t), \boldsymbol{\mathcal{K}}(t, t')). \end{aligned} \tag{15}$$

The state-dependent cost term $f[\boldsymbol{\theta}(t)]$ can include any cost function corresponding to the desired trajectory properties,

e.g. collision avoidance, task-space constraints, torques [6] and manipulability [20]. In this work, we consider only collision avoidance and use a precomputed signed distance field for collision checking similarly to [10], [12].

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 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. To solve (15), we employ a stochastic optimization approach stemming from the cross-entropy method [14] and with similarities to the estimation-of-distribution algorithm [21].

Our method 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 $\boldsymbol{\mathcal{K}}$ arises from the kernel matrix defined in (10). A sample trajectory is generated using

$$\boldsymbol{\theta} = \boldsymbol{\mu} + \boldsymbol{A}\boldsymbol{Z}, \tag{16}$$

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 $\boldsymbol{Z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.

Subsequently, we evaluate the cost $f[\boldsymbol{\theta}(t)]$ for each trajectory using the aforementioned hinge loss with a precomputed signed distance field. From the evaluated K trajectories we pick M best ones according to the optimization criteria, i.e. ones with the lowest cost $f[\boldsymbol{\theta}(t)]$. We then take the cost-weighted average (weighted arithmetic mean) of those M best trajectories in order to form a GP mean $\boldsymbol{\mu}$ for next iteration:

$$\boldsymbol{\mu}(t) = \frac{\sum_{m=1}^{M} w_m \boldsymbol{\theta}_m(t)}{\sum_{m=1}^{M} w_m} \tag{17}$$

where

$$w_m = 1/f[\boldsymbol{\theta}_m(t)]. \tag{18}$$

This process is repeated until a collision-free trajectory is found. The described method is summarized in Algorithm 1.

Contrary to a generic cross-entropy optimization algorithm, in our approach the covariance matrix $\boldsymbol{\mathcal{K}}$ remains unchanged through iterations. This is done because computing the new covariance matrix of the same form would significantly increase the computational burden. Furthermore, the unchanged covariance matrix allows for exhaustive exploration around the mean in each iteration. Changing only mean $\boldsymbol{\mu}$ while keeping the covariance matrix $\boldsymbol{\mathcal{K}}$ unchanged is permitted in the GP framework described in Section II, as change in $\boldsymbol{\mu}$ can be attributed to some implicitly imposed exogenous input $\boldsymbol{v}(t)$ which does not impact covariance.

A. Computational Efficiency Remarks

In order to throughly explore the environment, our approach requires cost evaluation for relatively many drawn trajectory samples, which naturally leads to the 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 [22].

We use GP interpolation for dense collision checking, similarly to [10]. Since trajectory *support states* are temporally equidistant and each sampled trajectory is drawn from the same GP, matrices Λ and Ψ in (12) and (13) can be precomputed, instead of computing them each time interpolation is needed. This provides another significant increase of the proposed method computational efficiency.

Algorithm 1 Stochastic Trajectory Optimization with GPs

Input: Start and goal states θ_0, θ_N, a state-dependent cost function $f[\boldsymbol{\theta}_k(t)]$

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

for $1 \dots N_{iter}$ **do**
  for $1 \dots K$ **do**
    Sample trajectory $\boldsymbol{\theta}_k(t) \sim \mathcal{GP}(\boldsymbol{\mu}(t), \boldsymbol{\mathcal{K}}(t, t'))$
    Evaluate trajectory cost $f[\boldsymbol{\theta}_k(t)]$
    if $f[\boldsymbol{\theta}_k(t)] = 0$ **then**
      Return collision free trajectory $\theta_k(t)$
    end if
  end for
  From K sampled trajectories take M with lowest cost
  Compute new mean $\boldsymbol{\mu}(t) = \frac{\sum_{m=1}^{M} [w_m \boldsymbol{\theta}_m(t)]}{\sum_{m=1}^{M} w_m}$
end for

IV. Test Results

We tested the proposed method on two simulation benchmarks and compared it with the state-of-the-art trajectory optimization technique GPMP2 [10]. In Section IV-A, we quantitatively demonstrate the improvement of the proposed method over GPMP2 with random restarts in solving a 2D maze, which is a good 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 IV-B, we demonstrate 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 aims to show benefits of the proposed heteroscedastic prior since the environment was set up so that obstacles are placed near the start and goal state.

In both benchmarks, our method was always initialized with a constant-velocity straight line trajectory in the configuration space. For GPMP2, we used a straight line initialization as a baseline, and in our experiments we designate to this model as *line*. 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 [4]. In this technique, the optimizer is first initialized with a straight-line and, on failure, re-initialized with a random trajectory. Our implementation samples the random restart trajectory from a homoscedastic GP, similarly to [11]. We designate to this model as *rr*.

We used the GPMP2 C++ library [10], [23] and its respective MATLAB toolbox based on the GTSAM C++ library [13]. Experiments were performed on a system with a 2.8-GHz Intel Core i7-7700HQ processor and 16 GB of RAM.

A. The Maze Benchmark

The maze benchmark, appropriate for quantitative evaluation, consisted of 1000 synthetic environments created by the Wilson's algorithm [24], 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 $0.5\,\mathrm{m}$. For 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 $M = 3$. Altough these parameters may seem disproportionate, choosing a large K ensures exhaustive exploration, while 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 robot moves from start to goal state) to $t_{\mathrm{total}} = 20\,\mathrm{s}$, while time-varying covariance matrix of the white noise governing the heteroscedastic GP was calculated as $\boldsymbol{Q}_c(t) = (t - \frac{t_{\mathrm{total}}}{2})^2$, which generates a parabola with its vertex at the midpoint of the trajectory. For GPMP2 we used the parameters set from the Matlab toolbox 2D example. For both methods, the trajectory was parametrized with $N = 10$ *support states* and 5 interpolation steps inbetween for which the trajectory cost is evaluated. We set the maximum runtime for our algorithm and random restarts as $t_{\max} = 1\,\mathrm{s}$, with one exception where we set $t_{\max} = 2\,\mathrm{s}$ 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. The reults of the experiment are shown in Table I.

While the GPMP2 without random restarts has an order of magnitude faster execution time, it has the worst success rate for every maze complexity level. For the least complex mazes, created from a 3×3 grid, the random restarts outperformed our algorithm, managing similarly high success rate with significantly faster computation. However, for the mazes created from a 4×4 grid our algorithm outperformed random restarts,

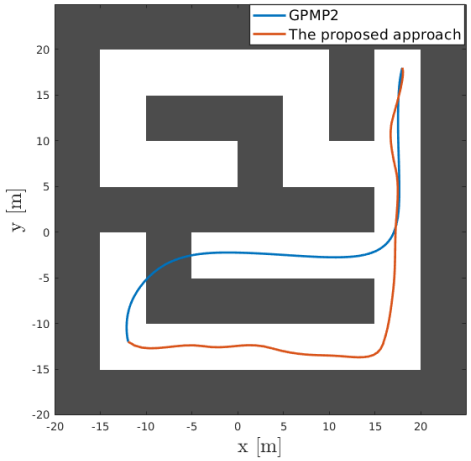


Fig. 2. Example of a 4×4 maze where the proposed approach finds a collision-free solution, while GPMP2 converges to an infeasible local minimum. Slight undulation of the trajectory obtained by the proposed method is due to the criterium of finding a collision-free trajectory, unlike the GPMP2 criterium which explicitly encourages smoothness.

TABLE I
SUCCESS RATE (PERCENTAGE) / AVERAGE EXECUTION TIME (MILISECONDS) ON MAZE AND ROBOT ARM PLANNING BENCHMARKS.

Maze	The proposed approach			GPMP2	
	$K = 400$, $t_{max} = 2\,\mathrm{s}$	$K = 400$, $t_{max} = 1\,\mathrm{s}$	$K = 200$, $t_{max} = 1\,\mathrm{s}$	rr	line
3x3	95.2 / 197	92.9 / 171	89.0 / 160	89.2 / 96	55.3 / 28
4x4	79.1 / 489	66.9 / 324	61.8 / 297	44.5 / 301	13.8 / 39
5x5	43.8 / 858	26.7 / 493	26.3 / 429	5 / 252	2.1 / 29
Arm		Heteroscedastic	Homoscedastic	rr	line
		100 / 446	75 / 548	80 / 368	10 / 45

having notably higher success rate with similar reported times. Note that the reported execution time for our algorithm is the actual time it took to compute, and not the sum over all cores. An example of a 4×4 maze is shown in Figure 2. The most complex mazes created from a 5×5 grid demonstrated the inability of gradient-based methods to find solutions in complex environments plagued with multitude of local minima.

B. 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. We conducted 20 unique experiments, all with different start and goal states with starting points being under the table and end states being inside the drawer. 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 planning methods to get stuck in local minima near start or goal states.

For 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 $M = 3$. In this benchmark we tested our optimization method with both heteroscedastic and homoscedastic GP priors in order to demonstrate the benefits of heteroscedasticity. We set the total trajectory time $t_{\mathrm{total}} = 20\,\mathrm{s}$. For homoscedastic case we chose $\boldsymbol{Q}_c = 2$, while for a heteroscedastic GP we calculated $\boldsymbol{Q}_c(t) = (t - \frac{t_{\mathrm{total}}}{2})^2$, similarly to the maze benchmark.

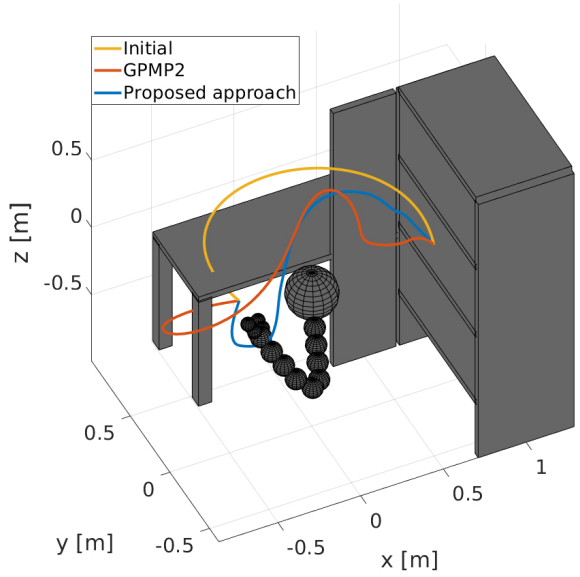


Fig. 3. A simulated WAM robotic arm in an environment featuring a table and a drawer. Plotted lines depict the end effector trajectories. This is an example where the proposed approach finds a collision free solution, while GPMP2 converges to the infeasible local minimum. Initial straight-line trajectory in configuration space is also shown.

For GPMP2 we used the default parameters set up in the Matlab toolbox WAM planner example. For both methods the trajectory was parametrized with $N = 10$ equidistant *support states* and 10 interpolation steps inbetween for which the trajectory cost was evaluated. We again set the fixed time budget for our algorithm and random restarts as $t_{\max} = 1\,\mathrm{s}$. We again measured the success rate and the execution time. The results of the experiment are shown in Table I, where we can see that while the baseline GPMP2 fails in most cases, the stochasticity introduced by random restarts helps in achieving higher success rates. The proposed method achieved a perfect score within the fixed time budget, thus demonstrating the advantage of the proposed heteroscedastic prior.

V. Conclusion

In this paper we have presented a stochastic trajectory optimization method for motion planning. We considered each trajectory as a sample from a continuous time GP generated by a linear time-varying stochastic differential equation. 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 proposed a cross-entropy method based derivative-free optimization in order to contend with the local minima problem present in trajectory optimization methods. Through simulated experiments we demonstrated that the proposed approach ameliorated the local minima problem present in trajetory optimization approaches while having comparable execution time.

In future work, it would be interesting to exploit the parallelization capability of our algorithm with a GPU implementation. Furthermore, the strong exploration capability could be used for finding homotopy classes in the environment.

References

[1] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

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

[3] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

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

[5] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa, "Chomp: Covariant hamiltonian optimization for motion planning," *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1164–1193, 2013.

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

[7] J. Schulman, J. Ho, A. X. Lee, I. Awwal, H. Bradlow, and P. Abbeel, "Finding locally optimal, collision-free trajectories with sequential convex optimization." in *Robotics: Science and Systems*, vol. 9, 2013.

[8] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1251–1270, 2014.

[9] M. Mukadam, X. Yan, and B. Boots, "Gaussian process motion planning," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2016, pp. 9–15.

[10] J. Dong, M. Mukadam, F. Dellaert, and B. Boots, "Motion planning as probabilistic inference using gaussian processes and factor graphs," in *Robotics: Science and Systems*, vol. 12, 2016.

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

[12] M. Mukadam, J. Dong, X. Yan, F. Dellaert, and B. Boots, "Continuous-time gaussian process motion planning via probabilistic inference," *arXiv preprint arXiv:1707.07383*, 2017.

[13] F. Dellaert, "Factor graphs and gtsam: A hands-on introduction," Georgia Institute of Technology, Tech. Rep., 2012.

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

[15] T. D. Barfoot, C. H. Tong, and S. Särkkä, "Batch continuous-time trajectory estimation as exactly sparse gaussian process regression." in *Robotics: Science and Systems*, vol. 10, 2014.

[16] S. Anderson, T. D. Barfoot, C. H. Tong, and S. Särkkä, "Batch nonlinear continuous-time trajectory estimation as exactly sparse gaussian process regression," *Autonomous Robots*, vol. 39, no. 3, pp. 221–238, 2015.

[17] J. Persic, L. Petrovic, I. Markovic, and I. Petrovic, "Spatio-temporal multisensor calibration based on gaussian processes moving object tracking," *arXiv preprint arXiv:1904.04187*, 2019.

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

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

[20] F. Marić, O. Limoyo, L. Petrović, I. Petrović, and J. Kelly, "Manipulability maximization using continuous-time gaussian processes," *arXiv preprint arXiv:1803.09493*, 2018.

[21] P. Larrañaga and J. A. Lozano, *Estimation of distribution algorithms: A new tool for evolutionary computation*. Springer Science & Business Media, 2001, vol. 2.

[22] C. Park, J. Pan, and D. Manocha, "Real-time optimization-based planning in dynamic environments using gpus," in *2013 IEEE Int. Conf. on Robotics and Automation*. IEEE, 2013, pp. 4090–4097.

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

[24] D. B. Wilson, "Generating random spanning trees more quickly than the cover time," in *STOC*, vol. 96. Citeseer, 1996, pp. 296–303.