Gaussian Process Motion Planning

Mustafa Mukadam, Xinyan Yan, and Byron Boots

***Abstract*-Motion planning is a fundamental tool in robotics, used to generate collision-free, smooth, trajectories, while satisfying task-dependent constraints. In this paper, we present a novel approach to motion planning using Gaussian processes. In contrast to most existing trajectory optimization algorithms, which rely on a discrete state parameterization in practice, we represent the continuous-time trajectory as a sample from a Gaussian process (GP) generated by a linear time-varying stochastic differential equation. We then provide a gradient-based optimization technique that optimizes continuous-time trajectories with respect to a cost functional. By exploiting GP interpolation, we develop the Gaussian Process Motion Planner (GPMP), that finds optimal trajectories parameterized by a small number of states. We benchmark our algorithm against recent trajectory optimization algorithms by solving 7-DOF robotic arm planning problems in simulation and validate our approach on a real 7-DOF WAM arm.**

I. INTRODUCTION & RELATED WORK

Motion planning is a fundamental skill for robots that aspire to move through an environment without collision or manipulate objects to achieve some goal. We consider motion planning from a trajectory optimization perspective, where we seek to find a trajectory that minimizes a given cost function while satisfying any given constraints.

A significant amount of previous work has focused on trajectory optimization and related problems. Khatib [1] pioneered the idea of potential field where the goal position is an attractive pole and the obstacles form repulsive fields. Various extensions have been proposed to address problems like local minimum [2], and ways of modeling the free space [3]. Covariant Hamiltonian Optimization for Motion Planning (CHOMP) [4], [5] utilizes covariant gradient descent to minimize obstacle and smoothness cost functionals, and a precomputed signed distance field for fast collision checking. STOMP [6] is a stochastic trajectory optimization method that can work with non-differentiable constraints by sampling a series of noisy trajectories. An important drawback of CHOMP and STOMP is that a large number of trajectory states are needed to reason about small obstacles or find feasible solutions when there are many constraints. Multigrid CHOMP [7] attempts to reduce the computation time of CHOMP when using a large number of states by starting with a low-resolution trajectory and gradually increasing the resolution at each iteration. Finally, TrajOpt [8] formulates motion planning as sequential quadratic programming. Swept volumes are considered to ensure continuous-time safety, enabling TrajOpt to solve more problems with fewer states.

II. A GAUSSIAN PROCESS MODEL FOR CONTINUOUS-TIME TRAJECTORIES

Motion planning traditionally involves finding a smooth, collision-free trajectory from a start state to a goal state. Similar to previous approaches [4][8], we treat motion planning as an optimization problem and search for a trajectory that minimizes a predefined cost function (Section III). In contrast to these previous approaches, which consider finely discretized discrete time trajectories in practice, we consider continuous-time trajectories such that the state at time t is

$$\xi(t) = \left\{\xi^d(t)\right\}_{d=1}^{D} \quad \xi^d(t) = \left\{\xi^{dr}(t)\right\}_{r=1}^{R} \tag{1}$$

where D is the dimension of the configuration space (number of joints), and R is the number of variables in each configuration dimension. Here we choose R = 3, specifying joint positions, velocities, and accelerations, ensuring that our state is Markovian. Using the Markov property of the state in the motion prior, defined in Eq. 6 below, allows us to build an exactly sparse inverse kernel (precision matrix) [16], [22] useful for efficient optimization and fast GP interpolation (Section II-B)

Continuous-time trajectories are assumed to be sampled from a vector-valued Gaussian process (GP):

$$\xi(t) \sim \mathcal{GP}(\mu(t), \mathcal{K}(t,t')) \quad t_0 < t, t' < t_{N+1} \tag{2}$$

where $\mu(t)$ is a vector-valued mean function whose components are the functions $\left\{\mu^{dr}(t)\right\}_{d=1,r=1}^{D,R}$ and the entries $\mathcal{K}(t,t')_{dr,d'r'}$ in the matrix-valued covariance function $\mathcal{K}(t,t')$ corresponding to the covariances between $\xi^{dr}(t)$ and $\xi^{d'r'}(t')$. Based on the definition of vector-valued Gaussian processes [23], any collection of N states on the trajectory has a joint Gaussian distribution,

$$\begin{aligned} \xi \sim \mathcal{N}(\mu, \mathcal{K}), \quad \xi = \xi_{1:N}, \quad \mu = \mu_{1:N} \\ \mathcal{K}_{ij} = \mathcal{K}(t_i, t_j), \quad t_1 \leq t_i, t_j \leq t_N \end{aligned} \tag{3}$$

The start and goal states, ξ_0 and ξ_{N+1}, are excluded because they are fixed. ξ_i denotes the state at time t_i.

Reasoning about trajectories with GPs has several advantages. The Gaussian process imposes a prior distribution on the space of trajectories. Given a fixed set of timeparameterized states, we can condition the GP on those states to compute the posterior mean at any time of interest:

$$\bar{\xi}(\tau) = \mu(\tau) + \mathcal{K}(\tau)\mathcal{K}^{-1}(\xi - \mu) \tag{4}$$

$$\mathcal{K}(\tau) = [\mathcal{K}(\tau, t_1) \quad ... \quad \mathcal{K}(\tau, t_N)] \tag{5}$$

A consequence of GP interpolation is that the trajectory does not need to be discretized at a high resolution, and trajectory states do not need to be evenly spaced in time. Additionally,

the posterior mean is the maximum a posteriori interpolation of the trajectory, and high-frequency interpolation can be used to generate an executable trajectory or reason about the cost over the entire trajectory instead of just at the states.

A. Gaussian Processes Generated by Linear Time-Varying Stochastic Differential Equations

The choice of prior is an important design consideration when working with Gaussian processes. In this work, we consider the Gaussian processes generated by the linear time varying (LTV) stochastic differential equations (SDEs) [16]:

$$\begin{aligned} &\xi'(t) = A(t)\xi(t) + F(t)\omega(t) \\ &\omega(t) \sim \mathcal{GP}(0, Q_c\delta(t-t')), \quad t_0 < t, t' < t_{N+1} \end{aligned} \tag{6}$$

where A(t) are F(t) are time-varying system matrices, $\delta(\cdot)$ is the Dirac delta function, w(t) is the process noise modeled by a Gaussian process with zero mean, and Qc is a powerspectral density matrix, a hyperparameter that can be set using data collected on the system [16].

Given the fixed start state ξ_0 as the initial condition, and $\Phi(t,s)$, the state transition matrix that propagates the state from time s to t, the solution to the SDE is:

$$\tilde{\xi}(t) = \Phi(t,t_0)\xi_0 + \int_{t_0}^{t} \Phi(t,s)F(s)\omega(s)ds \tag{7}$$

$\tilde{\xi}(t)$ corresponds to a Gaussian process. The mean function and the covariance function can be obtained from the first and second moments:

$$\tilde{\mu}(t) = \Phi(t,t_0)\xi_0 \tag{8}$$

$$\begin{aligned} \tilde{\mathcal{K}}(t,t') &= E[(\tilde{\xi}(t) - \tilde{\mu}(t))(\tilde{\xi}(t') - \tilde{\mu}(t'))^T] \\ &= \int_{t_0}^{min(t,t')} \Phi(t,s)F(s)Q_cF(s)^T\Phi(t',s)^T ds \end{aligned} \tag{9}$$

After considering the end state ξ_{N+1}as a noise-free measurement, and conditioning $\tilde{\xi}(t)$ on this measurement, we derive the Gaussian process for the trajectories with fixed start and end states. The mean and covariance functions are:

$$\begin{aligned} \mu(t) &= \tilde{\mu}_t + \tilde{\mathcal{K}}_{N+1,t}\tilde{\mathcal{K}}^{-1}_{N+1,N+1}(\xi_{N+1} - \tilde{\mu}_{N+1}) \\ &= \Phi_{t,0}\xi_0 + W_t\Phi^T_{N+1,t}W^{-1}_{N+1}(\xi_{N+1} - \Phi_{N+1,0}\xi_0) \end{aligned} \tag{10}$$

$$\begin{aligned} \mathcal{K}(t,t') &= \tilde{\mathcal{K}}_{t,t'} - \tilde{\mathcal{K}}^T_{N+1,t}\tilde{\mathcal{K}}^{-1}_{N+1,N+1}\tilde{\mathcal{K}}_{N+1,t'}, \quad t < t' \\ &= W_t\Phi^T_{t',t} - W_t\Phi^T_{N+1,t}W^{-1}_{N+1}\Phi_{N+1,t'}W_{t'}, \quad t < t' \end{aligned} \tag{11}$$

where

$$W_t = \int_{t_0}^{t} \Phi(t,s)F(s)Q_cF(s)^T\Phi(t',s)^T ds \tag{12}$$

The inverse kernel matrix $\mathcal{K}^{-1}$ for the N states is exactly sparse (block-tridiagonal), and can be factored as:

$$\mathcal{K}^{-1} = B^T Q^{-1} B \tag{13}$$

where

$$B = \begin{bmatrix} 1 & 0 & \dots & 0 \\ -\Phi(t_1,t_0) & 1 & \dots & 0 \\ 0 & -\Phi(t_2,t_1) & \ddots & \vdots \\ \vdots & \vdots & \ddots & 1 \\ 0 & 0 & \dots & -\Phi(t_N,t_{N-1}) \end{bmatrix}$$

and

$$\begin{aligned} Q &= diag(Q_1, \dots, Q_{N+1}) \\ Q_i &= \textstyle\int_{t_{i-1}}^{t_i} \Phi(t_i,s)F(s)Q_cF(s)^T\Phi(t_i,s)^T ds \end{aligned}$$

Here Q^{-1} is block diagonal, and B is band diagonal.

The inverse kernel matrix, $\mathcal{K}^{-1}$, encodes constraints on consecutive states imposed by the state transition matrix, $\Phi(t,s)$.If state $\xi(t)$ includes joint angles, velocities, and accelerations, then the constraints are on all of these variables. If we take this state transition matrix and Q^{-1} to be unit scalars, then $\mathcal{K}^{-1}$ reduces to the matrix *A* formed by finite differencing in CHOMP [4].

B. Fast Gaussian Process Interpolation

An advantage of Gaussian process trajectory estimation is that any point on a trajectory can be interpolated from other points by computing the posterior mean (Eq. 4). In the case of a GP generated from the LTV-SDE in Sec. II-A, interpolation is very fast, involving only two nearby states and is performed in $O(1)$ time [16]:

$$\tilde{\xi}(\tau) = \mu(\tau) + [\Lambda(\tau)\Psi(\tau)]([\xi_i\xi_{i+1}] - [\mu_i\mu_{i+1}]) \tag{14}$$

III. CONTINUOUS-TIME MOTION PLANNING WITH GAUSSIAN PROCESSES

IV. EXPERIMENTAL RESULTS

V. DISCUSSION

From the results in Table I we see that GPMP compares favorably with recent trajectory optimization algorithms on the benchmark. GPMP is able to solve all the 24 problems in a reasonable amount of time, while solving for trajectories in a state space 3 times the size of the state for all the other algorithms (with the exception of AugCHOMP). GPMP provides a speedup over AugCHOMP by utilizing GP interpolation during optimization, such that each iteration is faster without any significant loss in the quality of the trajectory at the end of optimization. This is illustrated in a comparison on an example optimized trajectory obtained from GPMP and AugCHOMP on the same problem (Figure 2). GPMP and AugCHOMP are able to converge to better solution trajectories (solve more problems) and do so faster when compared to CHOMP. One advantage of these algorithms is that the trajectory is augmented with velocities and accelerations. In contrast to CHOMP, where velocities and accelerations are computed from finite differencing, velocities and accelerations in GPMP and AugCHOMP can be used directly during optimization. This affects the calculation of the objective cost and its gradient, resulting in better gradient steps and convergence in fewer iterations. Benchmark results show that TrajOpt is faster

than our approach and fails on only one of the problems. By formulating the optimization problem as sequential quadratic programming (SQP), TrajOpt achieves faster convergence with fewer iterations. However, GPMP offers several advantages over TrajOpt: the continuous representation allows

VI. CONCLUSIONS

We have presented a novel approach to motion planning using Gaussian processes for reasoning about continuous-time trajectories by optimizing a small number of states. We considered trajectories consisting of joint positions, velocities, and accelerations sampled from a GP generated from a LTV-SDE, and we provided a gradient-based optimization technique for solving motion planning problems. The Gaussian process machinery enabled us to query the trajectory at any time point of interest, which allowed us to generate executable trajectories or reason about the cost of the entire trajectory instead of just at the states. We benchmarked our algorithm against various recent trajectory optimization algorithms on a set of 7-DOF robotic arm planning problems, and we validated our algorithms by running them on a 7-DOF Barrett WAM arm. Our empirical results show GPMP to be competitive or superior to competing algorithms with respect to speed and number of problems solved.