Approximately Optimal Continuous-Time Motion Planning and Control via Probabilistic Inference

Mustafa Mukadam, Ching-An Cheng, Xinyan Yan, and Byron Boots

***Abstract*— The problem of optimal motion planing and control is fundamental in robotics. However, this problem is intractable for continuous-time stochastic systems in general and the solution is difficult to approximate if non-instantaneous nonlinear performance indices are present. In this work, we provide an efficient algorithm, PIPC (Probabilistic Inference for Planning and Control), that yields approximately optimal policies with arbitrary higher-order nonlinear performance indices. Using probabilistic inference and a Gaussian process representation of trajectories, PIPC exploits the underlying sparsity of the problem such that its complexity scales linearly in the number of nonlinear factors. We demonstrate the capabilities of our algorithm in a receding horizon setting with multiple systems in simulation.**

I. Introduction

A fundamental goal in robotics is to efficiently compute trajectories of actions that drive a robot to achieve some desired behavior. We seek a control policy in a multi-stage decision problem [1] that can maximize performance indices that describe, for example, the smoothness of motion, energy consumption, or the likelihood of avoiding an obstacle.

Hierarchical planning and control has been used to solve this problem in practice [2]. The idea is to first generate a desired state sequence [3]–[9] without considering full system dynamics, and then design a robust low-level controller for tracking. Because the dynamic constraints are relaxed, it becomes possible for an algorithm to plan a trajectory that satisfies complicated, higher-order performance indices [8]–[10], offering greater flexibility in system design. Sampling-based planning techniques can even provide formal guarantees such as probabilistically complete solutions [3], [4]. However, recent work has started to challenge this classical viewpoint by incorporating more dynamic constraints within trajectory planning in search of solutions with improved optimality [11], [12].

A theoretically elegant approach would be to address both the planning and control problems within a stochastic optimal control framework. Unfortunately, since the states and actions are coupled through system dynamics, exact solutions become intractable with the exception of simple cases known as linearly solvable problems [13].[1]

These challenges have motivated researchers to find approximate solutions rather than directly approximating the original problems with hierarchical approaches. One simple approach is direct policy search [14], [15], which uses first-order information to find a locally optimal policy. To improve the convergence rate, differential dynamic programming (DDP) has been widely adopted as the foundation of locally optimal algorithms [16]–[18], which solve local linear-quadratic Gaussian (LQG) subproblems and iteratively improve these suboptimal solutions. However, for continuous-time systems, these algorithms would require inefficient high-frequency sampling to construct the LQG subproblems, *even* when the given problem is close to a LQG (e.g. a performance index with only a small set of nonlinear factors, or dynamics with a small amount of nonlinearity). Compared with the hierarchical approach, these algorithms impose a strict structural assumption: they are only applicable to problems that measure performance as an integral of instantaneous functions.

In this paper, we propose a novel approximately optimal approach to continuous-time motion planning and control that can handle costs expressed as arbitrary higher-order nonlinear factors and exploit a problem's underlying *sparse* structure. Specifically, we consider problems with a performance index expressed as the product of an exponential-

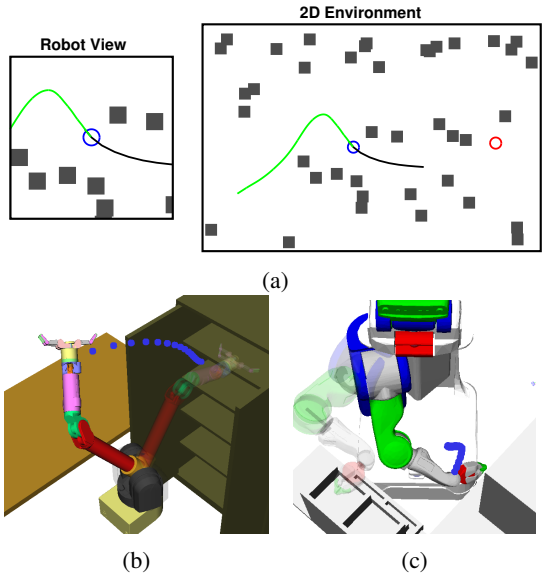


Fig. 1: PIPC used on (a) a 2D holonomic robot (blue) to reach goal (red) in a 2D environment with dynamic obstacles, where executed trajectory is in green and current planned horizon is in black, (b) a 7-DOF WAM arm, and (c) a PR2's right arm where the semitransparent arm is the goal configuration and dotted blue end effector trajectory is the current planned horizon.

Mustafa Mukadam, Ching-An Cheng, Xinyan Yan, and Byron Boots are affiliated with the Institute for Robotics and Intelligent Machines, Georgia Institute of Technology, Atlanta, GA 30332, USA. {mmukadam3, cacheng, xyan43}@gatech.edu, bboots@cc.gatech.edu.

[1]Affine systems with quadratic instantaneous control cost, or fully controllable discrete-time systems.

quadratic factor for instantaneous costs and a finite number of possibly higher-order nonlinear factors, and provide an algorithm that has linear complexity in the number of nonlinear factors. Moreover, we show the approximately optimal policy can be computed by posterior inference on a probabilistic graphical model, which is a dual to the performance index.

We convert these theoretical results into a practical algorithm called Probabilistic Inference for Planning and Control (PIPC) that recursively updates the approximately optimal policy as more information is encountered. To evaluate our approach, we employ PIPC on both Markov decision processes (MDPs) and partially-observable MDPs (POMDPs) in dynamic environments with multiple simulated systems (see Fig. 1).

A. Related Work

Our algorithm contributes to a growing set of research that seeks to reframe planning and control problems as probabilistic inference [19]. Work in this area has formed a new class of approximately optimal algorithms that leverage tools from approximate probabilistic inference, including expectation propagation [20] and expectation maximization [21], [22]. A common framework based on KL-minimization [23], [24] summarizes the above algorithms as well as approaches like path-integral control [13].

We contribute to this field in the following ways. First, we extend the performance index for control algorithms to incorporate nonlinear factors with arbitrary higher-order connections in time. In contrast to our approach, the methods mentioned above generally assume that the performance indices factor into instantaneous terms, and thus require dense sampling to solve continuous-time problems. Second, we provide a new approach to derive a Gaussian approximation based on Laplace approximation and Gaussian processes. Third, we define a new class of optimal control problems, called gLEQG (generalized Linear-Exponential-Quadratic-Gaussian), that are solvable after being transformed into their dual probabilistic representation. In particular, we show that gLEQG admits a solution given by posterior inference. This theoretical result, discussed in Section III-C, closes the gap in the duality between optimal control and inference.

This rest of the paper is structured as follows. We begin in Section II by defining the objective function in joint planning and control problems. Then, in Section III, we present our main results in approximately optimal motion planning and control. In Section IV, these theoretical results are summarized into an online algorithm PIPC that can perform simultaneous planning and control for partially observable stochastic linear systems in dynamic environments. To validate our algorithm, we present the implementation details and experimental results in Section V and Section VI. Finally, Section VII concludes the paper.

II. The Problem of Motion Planning and Control

We begin by introducing some notation. Let x_t, u_t, and z_t be the state, action, and observation of a continuous-time partially-observable system at time t, and let $\boldsymbol{h}_t = \{z_0, u_0, z_{\delta t}, \cdots, z_t\}$ be the history of observations and actions until time t.[2] As shorthand, we use boldface to denote the time trajectory of a variable, and $\boldsymbol{\pi}(\boldsymbol{u}|\boldsymbol{h})$ to denote the collection of time-varying causal (stochastic) policies $\pi_t(u_t|\boldsymbol{h}_t)$ for all t.

We formulate the motion planning and control problem as a finite-horizon stochastic optimization problem over $\boldsymbol{\pi}$. Let $p_{\boldsymbol{\pi}}$ be the distribution of $\boldsymbol{x}$ and $\boldsymbol{u}$ under the stochastic policy $\boldsymbol{\pi}$ and system dynamics, and $\mathcal{S}$ be a finite set of time indices. Here the goal is to find an optimal policy $\boldsymbol{\pi}$ to maximize the performance index

$$\max_{\boldsymbol{\pi}} J(x_0) = \max_{\boldsymbol{\pi}} \mathbb{E}_{p_{\boldsymbol{\pi}}} \left[\psi(\boldsymbol{x}, \boldsymbol{u}) \prod_{S \in \mathcal{S}} \phi_S(x_S, u_S) \right]. \quad (1)$$

The objective function in (1) is defined as the expectation of the product of two types of factors: a Gaussian process factor $\psi(\cdot)$ and a higher-order nonlinear factor $\phi_S(\cdot)$. These two factors, described below, cover many interesting behaviors that are often desired in planning and control problems.

A. Higher-order Nonlinear Factors $\phi_S(\cdot)$

We define factors of the form

$$\phi_S(\cdot) = \exp(-\|f_S(\cdot)\|^2), \quad (2)$$

to model nonlinear, higher-order couplings frequently used in planing problems, where $f_S(\cdot)$ is a differentiable nonlinear function defined on a finite number of time indices $S \in \mathcal{S}$. The structure of $\phi_S(\cdot)$ can model many performance indices in planning: for example, a simple nonlinear cost function at a single time instance, a penalty based on the difference between the initial and the terminal states/actions, a penalty to enforce consistency across landmarks in time, or the cost of a robot-obstacle collision. As each factor $\phi_S(\cdot)$ depends only on a finite number of states or actions, we refer to the corresponding states x_S and actions u_S as *support* states or *support* actions.

B. Gaussian Process Factors $\psi(\cdot)$

The Gaussian process factor $\psi(\cdot)$ is a generalization of the exponential-of-integral cost function in the optimal control literature [25]. To illustrate, here we consider a special case $\psi(\cdot) = \psi(\boldsymbol{u})$. A joint factor between $\boldsymbol{x}$ and $\boldsymbol{u}$ as in (1) can be defined similarly.

Let $\mathcal{GP}_u(u_t|m^u_t, \mathcal{K}^u_{t,t'})$ be a Gaussian process [26], where $\forall t, t' \in \mathbb{R}$, $\mathbb{E}[u_t] = m^u_t$ and $\mathbb{C}[u_t, u_{t'}] = \mathcal{K}^u_{t,t'}$. Let $\mathcal{P}^u_{t,t'}$ be the (positive definite) Green's function of $\mathcal{K}^u_{t,t'}$ satisfying, $\forall t, t' \in \mathbb{R}$, $\delta_{t,t'} = \int \mathcal{K}^u_{t,s} \mathcal{P}^u_{s,t'} \mathrm{d}s$, where δ is the Dirac delta distribution and the integral is over the length of the trajectory. We define the Gaussian process factor $\psi(\boldsymbol{u})$ as

$$\psi(\boldsymbol{u}) = \exp\left(- \iint (u_s - m^u_s)^T \mathcal{P}^u_{s,s'} (u_{s'} - m^u_{s'}) \mathrm{d}s \mathrm{d}s' \right). \quad (3)$$

Loosely speaking, we call (3) the *probability* of a trajectory $\boldsymbol{u}$ from $\mathcal{GP}_u(u_t|m^u_t, \mathcal{K}^u_{t,t'})$. Note that this notation

[2]Here we assume the measurements z_t are taken in discrete time at time t with sampling interval δt, and u_t is a constant continuous-time trajectory in time $[t, t + \delta t)$.

does not necessarily imply that $\boldsymbol{u}$ is a sample path of $\mathcal{GP}_u(u_t|m_t^u, \mathcal{K}_{t,t'}^u)$; rather, we use (3) as a metric between $\boldsymbol{u}$ and $\boldsymbol{m}^u$. Intuitively, the maximization in (1) encourages $\boldsymbol{u}$ to be close to $\boldsymbol{m}^u$ in terms of the distance weighted by $\mathcal{P}_{t,t'}^u$.

Solving a stochastic optimization problem with (3) in the objective function is intractable in general, because $\mathcal{P}_{t,t'}^u$ is only implicitly defined. However, here we show that when $\mathcal{GP}_u$ is the sum of a Gaussian white noise process and a linearly transformed Gauss-Markov process, the problem is not only tractable but can also extend the classical exponential-of-integral cost to model higher-order behaviors.

This is realized by defining $\mathcal{GP}_u(u_t|m_t^u, \mathcal{K}_{t,t'}^u)$ through a linear stochastic differential equation (SDE). Let y_t be the hidden state of u_t (e.g. its higher-order derivatives) and $p(y_0) = \mathcal{N}(y_0|m_0^y, \mathcal{K}_0^y)$ be its prior. We set $\mathcal{GP}_u(u_t|m_t^u, \mathcal{K}_{t,t'}^u)$ as the solution to

$$\begin{aligned} dy_t &= (Dy_t + \eta)dt + Gd\omega \\ u_t &= Hy_t + r_t + \nu_t \end{aligned} \tag{4}$$

in which D, η, G, H are (time-varying) system matrices, r_t is control bias, $d\omega$ is a Wiener process, and ν_t is a Gaussian white noise process $\mathcal{GP}_\nu(0, Q_\nu \delta_{t,t'})$. In other words, the Gaussian process $\mathcal{GP}_u(u_t|m_t^u, \mathcal{K}_{t,t'}^u)$ has mean and covariance functions:

$$m_t^u = r_t + Hm_t^y \tag{5}$$

$$\mathcal{K}_{t,t'}^u = Q_\nu \delta_{t,t'} + H\mathcal{K}_{t,t'}^y H^T \tag{6}$$

in which $\mathcal{GP}_y(m_t^y, \mathcal{K}_{t,t'}^y)$ is another Gaussian process with

$$m_t^y = \Phi_y(t, t_0)m_0^y + \int_{t_0}^{t} \Phi_y(t, s)\eta_s \mathrm{d}s \tag{7}$$

$$\mathcal{K}_{t,t'}^y = \Phi_y(t, t_0)\mathcal{K}_0^y \Phi_y(t', t_0)^T + \int_{t_0}^{\min(t,t')} \Phi_y(t, s)G_s G_s^T \Phi_y(t', s)^T \mathrm{d}s \tag{8}$$

and $\Phi_y(t, s)$ is the state transition matrix from s to t with respect to D. For derivations, please refer to [27] and therein.

The definitions (5) and (6) contain the exponential-of-integral cost [25]

$$\psi(\boldsymbol{u}) = \exp\left(-\int (u_s - r_s)^T Q_\nu^{-1}(u_s - r_s)\mathrm{d}s\right)$$

as a special case, which can be obtained by setting $H = 0$ (i.e. $\mathcal{P}_{t,t'}^u = Q_\nu^{-1}$). In general, it assigns the action $\psi(\boldsymbol{u})$ to be close to $\boldsymbol{r}$, even in terms of higher-order derivatives (or their hidden states). This leads to a preference toward smooth control signals. By extension, a joint factor between $\boldsymbol{x}$ and $\boldsymbol{u}$ would also encourage smooth state trajectories (i.e. smaller higher-order derivatives of the state).

Constructing the Gaussian process factor by SDE results in one particularly nice property: If we consider the joint Gaussian process of y_t and u_t, then its Green's function is *sparse*. To see this, let $\theta_t = (u_t, y_t)$ and $\boldsymbol{\theta} = \{\theta_1, \theta_2, \ldots, \theta_N\}$ and define $\psi(\boldsymbol{\theta})$ as its Gaussian process factor similar to (3). Then the double integral in $\psi(\boldsymbol{\theta})$ can be broken down into the sum of smaller double integrals, or factorized as

$$\psi(\boldsymbol{\theta}) = \tilde{\psi}(\theta_0) \prod_{i=1}^{N-1} \tilde{\psi}(\theta_i, \theta_{i+1}) \tag{9}$$

where $\tilde{\psi}(\cdot)$ has a similar exponential-quadratic form but over a smaller time interval $[t_i, t_i + \delta t]$. In other words, if we treat each θ_i as a coordinate, then the exponent of $\psi(\boldsymbol{\theta})$ can be written as a quadratic function with a tridiagonal Hessian matrix (please see [27] for details). This sparse property will be the foundation of the approximation procedure and algorithm proposed in Section III and IV.

III. Approximate Optimization as Inference

The mixed features from both planning and control domains in (1) present two major challenges: the optimization over continuous-time trajectories and the higher-order, nonlinear factors $\phi_S(\cdot)$. The former results in an infinite-dimensional problem, which often requires a dense discretization. The latter precludes direct use of algorithms based on Bellman's equation, because the factors may not factorize into instantaneous terms.

In this work, we propose a new approach inspired by approximate probabilistic inference. The goal here is to derive an approximation to the problem in (1), in the form

$$\max_{\boldsymbol{\pi}} \mathbb{E}_{\hat{p}_{\boldsymbol{\pi}}}\left[\psi(\boldsymbol{x}, \boldsymbol{u}) \prod_{S \in \mathcal{S}} \hat{\phi}_S(x_S, u_S)\right], \tag{10}$$

where $\hat{\phi}_S(\cdot)$ is a local exponential-quadratic approximation of $\phi_S(\cdot)$ and $\hat{p}_{\boldsymbol{\pi}}$ is a Gaussian process approximation of $p_{\boldsymbol{\pi}}$. We call the problem in (10) "gLEQG" as it generalizes LEQG (Linear-Exponential-Quadratic-Gaussian) [25] to incorporate higher-order exponentials in the form of (3).

In the rest of this section, we show how gLEQG can be derived by using the probabilistic interpretation [20] of the factors in (1). Further, we show this problem can be solved in linear time $O(|\mathcal{S}|)$ and its solution can be written in closed-form as posterior inference.

A. Probabilistic Interpretation of Factors

We begin by representing each factor in (1) with a probability distribution [21]. First, for $\phi_S(\cdot)$, we introduce additional fictitious observations e_S such that $p(e_S|x_S, u_S) \propto \phi_S(x_S, u_S)$. These new variables e_S can be interpreted as the events that we wish the robot to achieve and whose likelihood of success is reflected proportionally to $\phi_S(\cdot)$. Practically, they help us keep track of the message propagation over the support state/action in later derivations. Second, we rewrite the Gaussian process factor $\psi(\boldsymbol{u})$ to include the hidden state y_t in (4), as a joint Gaussian process factor $q(\boldsymbol{u}, \boldsymbol{y})$.[3] With the introduction of y_t, the joint Gaussian process $q(\boldsymbol{u}, \boldsymbol{y})$ has the sparse property in (9) that we desired.

Now, we rewrite the stochastic optimization (1) in the new notation. Let $\boldsymbol{e}_{\mathcal{S}} = \{e_S\}_{S \in \mathcal{S}}$ and $\xi = (x, y, u)$,

[3]This step can be carried similarly as the construction of $\psi(\boldsymbol{u})$.

and let $p(\boldsymbol{x}|\boldsymbol{u})$ and $p(\boldsymbol{z}|\boldsymbol{x})$ be the conditional distributions defined by the system dynamics and the observation model, respectively. It can be shown that (1) is equivalent to

$$\max_{\boldsymbol{\pi}} \int q(\boldsymbol{z}, \boldsymbol{\xi}|\boldsymbol{e}_{\mathcal{S}})\boldsymbol{\pi}(\boldsymbol{u}|\boldsymbol{h})\mathrm{d}\boldsymbol{\xi}\mathrm{d}\boldsymbol{z} \tag{11}$$

in which we define a joint distribution

$$q(\boldsymbol{z}, \boldsymbol{\xi}, \boldsymbol{e}_{\mathcal{S}}) = q(\boldsymbol{\xi})p(\boldsymbol{z}|\boldsymbol{x}) \prod_{S\in\mathcal{S}} p(e_S|x_S, u_S) \tag{12}$$

with likelihoods $p(\boldsymbol{z}|\boldsymbol{x})$ and $p(e_S|x_S, u_S)$, and a prior on the continuous-time trajectory $\boldsymbol{\xi}$

$$q(\boldsymbol{\xi}) = p(\boldsymbol{x}|\boldsymbol{u})q(\boldsymbol{u}, \boldsymbol{y}). \tag{13}$$

Before proceeding, we clarify the notation we use to simplify writing. We use q to denote the *ad hoc* constructed Gaussian process factor (e.g. in (3)) and use p to denote the probability distribution associated with the real system. As such, q does not always define an expectation, so the integral notation (e.g. in (11)) denotes the expectation over p and π that are well-defined probability distributions. But, with some abuse of notation, we will call them both Gaussian processes, since our results depend rather on their algebraic form.

B. Gaussian Approximation

Let $\boldsymbol{\xi}_{\mathcal{S}} = \{\xi_S\}_{S\in\mathcal{S}}$ and $\bar{\boldsymbol{\xi}}_{\mathcal{S}} = \boldsymbol{\xi}\backslash\boldsymbol{\xi}_{\mathcal{S}}$. To derive the gLEQG approximation to (1), we notice, by (12), $q(\boldsymbol{z}, \boldsymbol{\xi}|\boldsymbol{e}_{\mathcal{S}})$ in (11) can be factorized into

$$q(\boldsymbol{z}, \boldsymbol{\xi}|\boldsymbol{e}_{\mathcal{S}}) = q(\boldsymbol{z}, \bar{\boldsymbol{\xi}}_{\mathcal{S}}|\boldsymbol{\xi}_{\mathcal{S}})q(\boldsymbol{\xi}_{\mathcal{S}}|\boldsymbol{e}_{\mathcal{S}}) \tag{14}$$

where have used the Markovian property in Section III-A i.e. given $\boldsymbol{\xi}_{\mathcal{S}}$, $\boldsymbol{e}_{\mathcal{S}}$ is conditionally independent of other random variables. Therefore, if $q(\boldsymbol{z}, \bar{\boldsymbol{\xi}}_{\mathcal{S}}|\boldsymbol{\xi}_{\mathcal{S}})$ and $q(\boldsymbol{\xi}_{\mathcal{S}}|\boldsymbol{e}_{\mathcal{S}})$ can be reasonably approximated as Gaussians, then we can approximate (1) with (10).

However, $q(\boldsymbol{z}, \bar{\boldsymbol{\xi}}_{\mathcal{S}}|\boldsymbol{\xi}_{\mathcal{S}})$ and $q(\boldsymbol{\xi}_{\mathcal{S}}|\boldsymbol{e}_{\mathcal{S}})$ have notably different topologies. $q(\boldsymbol{z}, \bar{\boldsymbol{\xi}}_{\mathcal{S}}|\boldsymbol{\xi}_{\mathcal{S}})$ is a distribution over continuous-time trajectories, whereas $q(\boldsymbol{\xi}_{\mathcal{S}}|\boldsymbol{e}_{\mathcal{S}})$ is a density function on finite number of random variables. Therefore, to approximate (1), we need to find a Gaussian *process* $\hat{q}(\boldsymbol{z}, \bar{\boldsymbol{\xi}}_{\mathcal{S}}|\boldsymbol{\xi}_{\mathcal{S}})$ and a Gaussian *density* $\hat{q}(\boldsymbol{\xi}_{\mathcal{S}}|\boldsymbol{e}_{\mathcal{S}})$.

1) Gaussian Process Approximation: We derive the Gaussian process approximation $\hat{q}(\boldsymbol{z}, \boldsymbol{\xi})$ to $q(\boldsymbol{z}, \boldsymbol{\xi})$. With this result, the desired conditional Gaussian process $\hat{q}(\boldsymbol{z}, \bar{\boldsymbol{\xi}}_{\mathcal{S}}|\boldsymbol{\xi}_{\mathcal{S}})$ is given closed-form.

First we need to define the system dynamics $p(\boldsymbol{x}|\boldsymbol{u})$ and the observation model $p(\boldsymbol{z}|\boldsymbol{x})$. For now, let us assume that the system is governed by a linear SDE

$$\begin{aligned} dx &= (Ax + Bu + b)dt + Fdw \\ z &= Cx + v \end{aligned} \tag{15}$$

in which A, B, b, F, C are (time-varying) system matrices, dw is a Wiener process, and v is Gaussian noise with covariance Q_v. When a prior is placed on x_0 (similar to Section II-B) it can be shown that the solution to (15) $p(\boldsymbol{x}, \boldsymbol{z}|\boldsymbol{u}) = p(\boldsymbol{z}|\boldsymbol{x})p(\boldsymbol{x}|\boldsymbol{u})$ is a Gaussian process. Since $q(\boldsymbol{u}, \boldsymbol{y})$ is also Gaussian process, we have a Gaussian process prior on $\boldsymbol{z}$ and $\boldsymbol{\xi}$:

$$q(\boldsymbol{z}, \boldsymbol{\xi}) = p(\boldsymbol{x}, \boldsymbol{z}|\boldsymbol{u})q(\boldsymbol{u}, \boldsymbol{y}), \tag{16}$$

In this case, no approximation is made and therefore $\hat{q}(\boldsymbol{z}, \bar{\boldsymbol{\xi}}_{\mathcal{S}}|\boldsymbol{\xi}_{\mathcal{S}}) = q(\boldsymbol{z}, \bar{\boldsymbol{\xi}}_{\mathcal{S}}|\boldsymbol{\xi}_{\mathcal{S}})$.

In the case of nonlinear systems, one approach is to treat (15) as its local linear approximation and derive $\hat{q}(\boldsymbol{z}, \boldsymbol{\xi}) = \hat{p}(\boldsymbol{x}, \boldsymbol{z}|\boldsymbol{u})q(\boldsymbol{u}, \boldsymbol{y})$, where $\hat{p}(\boldsymbol{x}, \boldsymbol{z}|\boldsymbol{u})$ is the solution to the linearized system. Alternatively, we can learn the conditional distribution $\hat{p}(\boldsymbol{x}, \boldsymbol{z}|\boldsymbol{u})$ from data directly through Gaussian process regression [26]. However, since our main purpose here is to show the solution when $\hat{p}(\boldsymbol{x}, \boldsymbol{z}|\boldsymbol{u})$ is available, from now on we will assume the system is linear and given by (15).

2) Gaussian Density Approximation: Unlike $q(\boldsymbol{z}, \bar{\boldsymbol{\xi}}_{\mathcal{S}}|\boldsymbol{\xi}_{\mathcal{S}})$, the approximation to $q(\boldsymbol{\xi}_{\mathcal{S}}|\boldsymbol{e}_{\mathcal{S}})$ is more straightforward. First, because $q(\boldsymbol{\xi}_{\mathcal{S}}|\boldsymbol{e}_{\mathcal{S}})$ may not be available in closed form, we approximate $q(\boldsymbol{\xi}_{\mathcal{S}}|\boldsymbol{e}_{\mathcal{S}})$ with $\tilde{q}(\boldsymbol{\xi}_{\mathcal{S}}|\boldsymbol{e}_{\mathcal{S}})$

$$\begin{aligned} q(\boldsymbol{\xi}_{\mathcal{S}}|\boldsymbol{e}_{\mathcal{S}}) &\propto q(\boldsymbol{\xi}_{\mathcal{S}}) \prod_{S\in\mathcal{S}} p(e_S|x_S, u_S) \\ &\approx \hat{q}(\boldsymbol{\xi}_{\mathcal{S}}) \prod_{S\in\mathcal{S}} p(e_S|x_S, u_S) \propto \tilde{q}(\boldsymbol{\xi}_{\mathcal{S}}|\boldsymbol{e}_{\mathcal{S}}) \end{aligned} \tag{17}$$

where $\hat{q}(\boldsymbol{\xi}_{\mathcal{S}})$ is the marginal distribution of $\hat{q}(\boldsymbol{z}, \boldsymbol{\xi})$, found in the previous section. Given (17), we then find a Gaussian approximation $\hat{q}(\boldsymbol{\xi}_{\mathcal{S}}|\boldsymbol{e}_{\mathcal{S}})$ of $\tilde{q}(\boldsymbol{\xi}_{\mathcal{S}}|\boldsymbol{e}_{\mathcal{S}})$ via a Laplace approximation [28].

For the nonlinear factor from (2), a Laplace approximation of $\tilde{q}(\boldsymbol{\xi}_{\mathcal{S}}|\boldsymbol{e}_{\mathcal{S}})$ amounts to solving a nonlinear least-squares optimization problem. Using the sparsity of the structured Gaussian processes defined by SDEs, the optimization can be completed using efficient data structures in $O(|\mathcal{S}|)$ [9]. For space constraints, we omit the details here; please see Appendix A in [29] and [9] for details.

3) Summary: The above approximations allow us to approximate (12) with a Gaussian distribution

$$\hat{q}(\boldsymbol{z}, \boldsymbol{\xi}, \boldsymbol{e}_{\mathcal{S}}) = \hat{p}(\boldsymbol{z}, \boldsymbol{x}|\boldsymbol{u})q(\boldsymbol{u}, \boldsymbol{y}) \prod_{S\in\mathcal{S}} \hat{p}(e_S|x_S, u_S). \tag{18}$$

In (18), $\hat{p}(\boldsymbol{z}, \boldsymbol{x}|\boldsymbol{u})$ is the Gaussian process approximation of the system, which is exact when the system is linear, and $\hat{p}(e_S|x_S, u_S)$ is proportional to the exponential-quadratic factor $\hat{\phi}_S(x_S, u_S)$ in (10). Moreover, it can be shown that $\hat{q}(\boldsymbol{z}, \boldsymbol{\xi}, \boldsymbol{e}_{\mathcal{S}})$ is a Laplace approximation of $\hat{p}(\boldsymbol{z}, \boldsymbol{x}|\boldsymbol{u})q(\boldsymbol{u}, \boldsymbol{y}) \prod_{S\in\mathcal{S}} p(e_S|x_S, u_S)$ in terms of continuous-time trajectory $\boldsymbol{z}$ and $\boldsymbol{\xi}$.

C. Finding an Approximately Optimal Policy

Substituting the results in Section III-B into (11), we have the approximated optimization problem

$$\max_{\boldsymbol{\pi}} \int \hat{q}(\boldsymbol{z}, \boldsymbol{\xi}|\boldsymbol{e}_{\mathcal{S}})\boldsymbol{\pi}(\boldsymbol{u}|\boldsymbol{h})\mathrm{d}\boldsymbol{\xi}\mathrm{d}\boldsymbol{z}. \tag{19}$$

By (18), one can show that (19) is equivalent to the problem in (10), but expressed in probabilistic notation.

However, by writing the problem probabilistically, we can avoid the algebraic complications arising from attempting to solve the Bellman's equation of (10), which, because of higher-order factors, requires additional state expansion. This simplicity is reflected in the optimality condition for (19):

$$\begin{aligned}\pi_t^*(u_t|\boldsymbol{h}_t) &= \delta(u_t - u_t^*(\boldsymbol{h}_t)) \\ u_t^*(\boldsymbol{h}_t) &= \underset{u_t}{\operatorname{argmax}} \int \hat{q}(\boldsymbol{z}, \boldsymbol{\xi}|\boldsymbol{e}_{\mathcal{S}}) \boldsymbol{\pi}^*(\bar{\boldsymbol{u}}_t|\boldsymbol{h}) \mathrm{d}\boldsymbol{x} \mathrm{d}\boldsymbol{y} \mathrm{d}\boldsymbol{z} \mathrm{d}\bar{\boldsymbol{u}}_t \\ &= \underset{u_t}{\operatorname{argmax}}\, \hat{q}(u_t|\boldsymbol{h}_t, \boldsymbol{e}_{\mathcal{S}}) \end{aligned} \tag{20}$$

in which $\bar{\boldsymbol{u}}_t$ denotes $\boldsymbol{u} \backslash \{u_t\}$ and δ is Dirac delta distribution. From the last equality in (20), we see that the solution to the maximization problem coincides with the mode of the posterior distribution $\hat{q}(u_t|\boldsymbol{h}_t, \boldsymbol{e}_{\mathcal{S}})$. As a result, the optimal policies for time t can be derived *forward* in time, by performing inference without solving for the future policies first. Please see Appendix B in [29] for the proof.

We call this property the duality between gLEQG and inference. This result seems surprising, but similar ideas can be traced back to the duality between the optimal control and estimation [17], [20], in which the optimal value function of a linear quadratic problem is computed by backward message propagation without performing maximization.

Compared with previous work, a stronger duality holds here: gLEQG is dual to the inference problem on the *same* probabilistic graphical model defined by the random variables in Section III-A. This nice property is the result of the use of an exponential performance index, and enables us to handle higher-order factors naturally without referring to *ad hoc* derivations based on dynamic programming on extended states.

Our posterior representation of the policy can also be found in [20], [30], or can be interpreted as one step of posterior iteration [24]. In [20], this results from the approximation of the optimal value function, but its relationship to the overall stochastic optimization is unclear. In [30], the posterior representation is reasoned from the notion of a predictive policy representation without further justification of its effects on the whole decision process. Here we derive the policy based on the assumption that the associated distribution of (1) can be approximated by a Gaussian (18). Therefore, the condition on which the approximate policy remains valid can be more easily understood or even enforced, as discussed later in Section IV-B.

IV. Probabilistic Motion Planning and Control

In Section III, we show that if $q(\boldsymbol{z}, \boldsymbol{\xi}|\boldsymbol{e})$ can be approximated well by a Gaussian distribution, the stochastic optimization in (1) can be approximately solved as posterior inference (20). This representation suggests that the approximately optimal policy can be updated recursively through Kalman filtering.

A. Recurrent Policy Inference as Kalman Filtering

The approximately optimal policy in (20) can be viewed as the belief about the current action u_t given the history $\boldsymbol{h}_t$ and the fictitious events $\boldsymbol{e}_{\mathcal{S}}$. Here we exploit the Markovian structure underlying $\hat{q}(\boldsymbol{z}, \boldsymbol{\xi}|\boldsymbol{e}_{\mathcal{S}})$ to derive a recursive algorithm for updating the belief $\hat{q}(\xi_t|\boldsymbol{h}_t, \boldsymbol{e}_{\mathcal{S}})$. Given the belief, the policy can be derived by marginalization. First, for $t = 0$, we write

$$\begin{aligned}\hat{q}(\xi_0|\boldsymbol{h}_0, \boldsymbol{e}_{\mathcal{S}}) &\propto p(z_0|\xi_0) \hat{q}(\xi_0|\boldsymbol{e}_{\mathcal{S}}) \\ &= q(z_0|\xi_0) \int q(\xi_0|\boldsymbol{\xi}_{\mathcal{S}}) \hat{q}(\boldsymbol{\xi}_{\mathcal{S}}|\boldsymbol{e}_{\mathcal{S}}) \mathrm{d}\boldsymbol{\xi}_{\mathcal{S}}\end{aligned}$$

in which $q(z_t|\xi_t) = p(z_t|x_t)$ and $q(\xi_0|\boldsymbol{\xi}_{\mathcal{S}})$ is the conditional distribution defined by (16). After initialization, the posterior $\hat{q}(\xi_t|\boldsymbol{h}_t, \boldsymbol{e}_{\mathcal{S}})$ can be propagated through prediction and correction, summarized together in one step as

$$\begin{aligned}&\hat{q}(\xi_{t+\delta t}|\boldsymbol{h}_{t+\delta t}, \boldsymbol{e}_{\mathcal{S}}) \propto p(z_{t+\delta t}|\xi_{t+\delta t}) \hat{q}(\xi_{t+\delta t}|\boldsymbol{e}_{\mathcal{S}}) \\ &= q(z_{t+\delta t}|\xi_{t+\delta t}) \int \hat{q}(\xi_{t+\delta t}|\xi_t, \boldsymbol{e}_{\mathcal{S}}) \hat{q}(\xi_t|\boldsymbol{h}_t, \boldsymbol{e}_{\mathcal{S}}) \mathrm{d}\xi_t\end{aligned} \tag{21}$$

in which the transition is given by

$$\begin{aligned}\hat{q}(\xi_{t+\delta t}|\xi_t, \boldsymbol{e}_{\mathcal{S}}) &\propto \hat{q}(\xi_k, \xi_{t+\delta t}|\boldsymbol{e}_{\mathcal{S}}) \\ &= \int q(\xi_k, \xi_{t+\delta t}|\boldsymbol{\xi}_{\mathcal{S}}) \hat{q}(\boldsymbol{\xi}_{\mathcal{S}}|\boldsymbol{e}_{\mathcal{S}}) \mathrm{d}\boldsymbol{\xi}_{\mathcal{S}}\end{aligned} \tag{22}$$

and $q(\xi_k, \xi_{t+\delta t}|\boldsymbol{\xi}_{\mathcal{S}})$ is given by (16) [31], [32]. Because of the Markovian structure in (9), the integral (22) only depends on two adjacent support states/actions of t and can be computed in constant time. Note, in $\hat{q}(\xi_t|\boldsymbol{h}_t, \boldsymbol{e}_{\mathcal{S}})$ in (21), the action u_t is actually conditioned on the action taken $u_t^*(\boldsymbol{h}_t)$. This notation is adopted to simplify the writing.

Thus, we can view (21) as Kalman filtering with transition dynamics $\hat{q}(\xi_{t+\delta t}|\xi_t, \boldsymbol{e}_{\mathcal{S}})$ and observation process $q(z_t|\xi_t)$. This formulation gives us the flexibility to switch between open-loop and closed-loop policies. That is, before a new observation $z_{t+\delta t}$ is available, (22) provides a continuous-time open-loop action trajectory during the interval $(t, t+\delta t)$.

This recurrent policy inference is based on the assumption that $\hat{q}(\boldsymbol{z}, \boldsymbol{\xi}|\boldsymbol{e}_{\mathcal{S}})$ is accurate. Although this assumption is not necessarily true in general, it is a practical approximation if the belief about current state $p(x_t|\boldsymbol{h}_t)$ is concentrated and the horizon within which (20) applies is short.

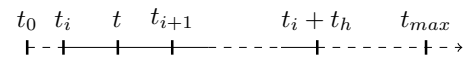


Fig. 2: Time-line with PIPC, where a system that started at t_0, is currently at time $t \in [t_i, t_{i+1}]$ between support points t_i and t_{i+1} in δt resolution. In a receding horizon setting, t_+ represents the receding horizon window $[t_i, t_i + t_h]$, and t_{max} is the (infinite) final time when the algorithm terminates. In a finite horizon setting, $t_h = t_{max} - t_i$.

B. Online Motion Planning and Control

We now summarize everything into the PIPC algorithm. Let $t_i \in \mathcal{S}$. We compensate for the local nature of (20) by recomputing a new Laplace approximation $q(\boldsymbol{z}_{t_+}, \boldsymbol{\xi}_{t_+}|\boldsymbol{h}_t, \boldsymbol{e}_{\mathcal{S}})$ whenever $t \in \mathcal{S}$, and applying filtering to update the policy by (21) for $t \in (t_i, t_{i+1})$, in which the subscript t_+ denotes the future trajectory from t (see Fig. 2). This leads to an

Algorithm 1 Receding Horizon PIPC

Input: horizon t_h, start time t_0, initial belief $q(\xi_{t_0})$
Output: success/failure

while not STOP_CRITERIA **do**
  $\hat{q}(\boldsymbol{\xi}_S|\boldsymbol{e}_S, \boldsymbol{h}_{t_i-\delta t}, u_{t_i-\delta t})$ = getLaplaceApprox(t_i, t_h, $q(\xi_t|\boldsymbol{h}_{t_i-\delta t}, u_{t_i-\delta t})$, ENVIRONMENT)
  for $t \in [t_i, t_{i+1}]$ **do**
    z_t = makeObservation()
    $\hat{q}(\xi_t|\boldsymbol{h}_t, \boldsymbol{e}_S)$ = filterPolicy(z_t, $\hat{q}(\xi_{t-\delta t}|\boldsymbol{h}_{t-\delta t}, \boldsymbol{e}_S)$, $\hat{q}(\boldsymbol{\xi}_S|\boldsymbol{e}_S, \boldsymbol{h}_{t_i-\delta t}, u_{t_i-\delta t})$)
    executePolicy($u_t = u_t^*(\boldsymbol{h}_t)$)
    $q(\xi_{t+\delta t}|\boldsymbol{h}_t, u_t)$ = filterState(z_t, u_t, $q(\xi_t|\boldsymbol{h}_{t-\delta t}, u_{t-\delta t})$)
  end for
end while
return checkSuccess()

iterative framework which solves for the new approximation with up-to-date knowledge about the system.

We can apply this scheme to MDP/POMDP problems in both finite and receding horizon cases.[4] When facing a dynamic environment, PIPC updates environmental information in the new Laplace approximation in Section III-B.2.

The details of the receding horizon approach are summarized in Algorithm 1 and can be derived similarly for the finite horizon case. First, at any time step t_i, PIPC computes the Laplace approximation for the current horizon window $[t_i, t_i + t_h]$ with the latest information about the system and the environment, where $t_h \geq t_{i+1} - t_i$ is length of the preview horizon. Second, for $t \in (t_i, t_{i+1})$, PIPC recursively updates the policy using the most current observation with a resolution of δt. These two steps repeat until the set criteria are met or the execution fails (for example, the robot is in collision).

V. Implementation Details

We perform experiments with the receding horizon version of PIPC in four different setups, including both MDP and POMDP scenarios: MDP-CL and POMDP-CL execute the receding horizon PIPC in Algorithm 1; MDP-OL and POMDP-OL ignore the policy filtering step, but instead recursively apply the open-loop policy given as the mode found in the Laplace approximation. This open-loop baseline can be viewed as the direct generalization of [9] to include action trajectories.

The Laplace approximation is implemented using GPMP2[5] and the GTSAM[6] C++ library, which solves posterior maximization as a nonlinear least-squared optimization defined on a factor graph with the Levenberg-Marquardt algorithm. Note that in implementation we consider $y_t = u_t$ (i.e. $\xi_t = (x_t, u_t)$) and a constant time difference Δt between any two support states or actions.

We evaluate our algorithms on three different systems: a 2D holonomic robot, a 7-DOF WAM, and a PR2 arm. The state dynamics, following (15), is defined as a double integrator with the state consisting of position and velocity and

$$A = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}, B = \begin{bmatrix} \mathbf{0} \\ \mathbf{I} \end{bmatrix}, b = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, FF^T = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & Q_x\mathbf{I} \end{bmatrix}$$

and following (4) we define the Gaussian process factor by

$$H = \mathbf{0}, D = \mathbf{0}, \eta = 0, GG^T = Q_u\mathbf{I}$$

where $\mathbf{0}$ and $\mathbf{I}$ are $d \times d$ zero and identity matrices, where $d = 2$ for the 2D holonomic robot and $d = 7$ for the 7-DOF WAM arm and PR2 arm, and Q_x and Q_u are positive scalars. The observation process in the POMDP is modeled as a state observation with additive zero-mean Gaussian noise with covariance $Q_v = \sigma_m \mathbf{I}_{2d\times 2d}$. The state dynamics for both the arms are assumed to be feedback linearized. On a real system, the control would to be mapped back to real torques using inverse dynamics.

VI. Evaluation

We conduct benchmark experiments[7] with our receding horizon algorithm on the 2D holonomic robot in a dynamic environment, and on the WAM arm and the PR2's right arm in a static environment (see Fig. 1). In each case, we compare the closed-loop and open-loop algorithms for both MDP and POMDP settings across different Q_x (and number of dynamic obstacles N_{obs} in the 2D case) with respect to success rate, time to reach the goal, path length, and path cost.[8] Each setting is run for K times with a unique random generator seed to account for stochasticity, which is kept the same across all four algorithms for a fair comparison. A trial is marked "successful" if the robot reaches the goal within a Euclidean distance $gdist$, and is marked "failed" if at any point the robot runs into collision or runs out of the maximum allotted time t_{max}.

A. 2D Robot Benchmark

We simulate a 2D holonomic robot (radius $= 0.5m$) in a 2D environment ($30m \times 20m$) with moving obstacles (see Fig. 1 (a)). The robot's sensor returns a limited view of a $5m \times 5m$ square centered at the robot's current position.

[4]The receding-horizon version solves a new finite-horizon problem at each iteration of the Laplace approximation.

[5]Available at https://github.com/gtrll/gpmp2

[6]Available at https://bitbucket.org/gtborg/gtsam

[7]A video of experiments is available at https://youtu.be/8rQcg1O-6aU

[8]Path cost is calculated as the negative log of the product of the factors.

TABLE I: Success rate across increasing Q_x and N_{obs} on the 2D holonomic robot.

	Q_x	10		20		30		40		50	
		CL	OL	CL	OL	CL	OL	CL	OL	CL	OL
MDP	0.01	0.975	0.975	0.85	0.85	0.7	0.675	0.4	0.375	0.25	0.325
	0.04	0.95	0.975	0.85	0.8	0.55	0.525	0.525	0.375	0.325	0.325
	0.07	0.95	0.85	0.875	0.575	0.725	0.475	0.45	0.2	0.225	0.125
POMDP	0.01	0.975	0.975	0.925	0.875	0.725	0.7	0.375	0.4	0.15	0.225
	0.04	0.95	0.975	0.875	0.825	0.525	0.475	0.425	0.45	0.4	0.25
	0.07	0.975	0.875	0.825	0.55	0.7	0.425	0.45	0.25	0.2	0.075

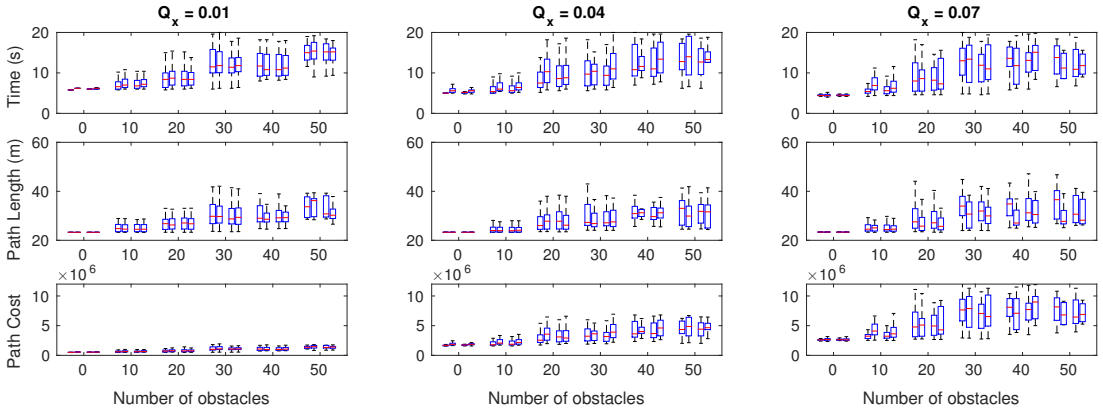


Fig. 3: Results of successful runs with increasing Q_x and N_{obs} on the 2D holonomic robot.

The moving obstacles (squares of $1m \times 1m$) start at random locations and follow a 2D stochastic jump process, where a noisy acceleration a_{obs} is uniformly sampled within $[-2.5, 2.5]m/s^2$ at every time step. Their velocities v_{obs} are restricted within, $[-1.3, 1.3]m/s$. All obstacles are confined inside the boundary during simulation.

Table I summarizes the success rates for this benchmark,[9] and Fig. 3 shows the aggregate results of successful runs. From Table I, we see that, for both MDP and POMDP cases, the closed-loop algorithms have higher success rates than the open-loop algorithms, especially in difficult problems with larger stochasticity in the system (Q_x) or increased complexity in the environment (N_{obs}). Similar increasing trends can also be observed in the difference of the success rates between the closed-loop and open-loop algorithms. The majority of failed open-loop cases arise from collision; only a few are due to hitting the maximum run time. The performance in POMDP cases are slightly worse than that in the MDP cases on average. All three metrics (time, path length, and path cost) in Fig. 3 increase in general with more noise and obstacles. It is important that these plots should be interpreted alongside the success rates, since the sample size of successful trails is comparatively sparse for the harder problems.

B. WAM and PR2 Benchmark

We demonstrate the scalability of PIPC to higher dimensional systems by performing a benchmark on the WAM and the PR2 robot arms. Here the WAM and the PR2 robot arms are set up in *lab* and *industrial* environments [6], [7], [9] respectively, in OpenRAVE. Here the task is to drive the robot arm from a given start to a goal configuration (see Fig. 1 (b) and (c)). The environments are static and fully observable at all times. We compare the algorithms with respect to increasing Q_x. Table II summarizes the success rates for this benchmark,[10] and Fig. 4 shows the aggregate results of successful runs. Similar to the 2D robot benchmark, the results show that the closed-loop algorithms have higher success rate than the open-loop ones, and all three metrics increase with noise. In particular, POMDP-CL performs even better than MDP-OL.

TABLE II: Success rate across increasing Q_x on the WAM and the PR2 robot arms.

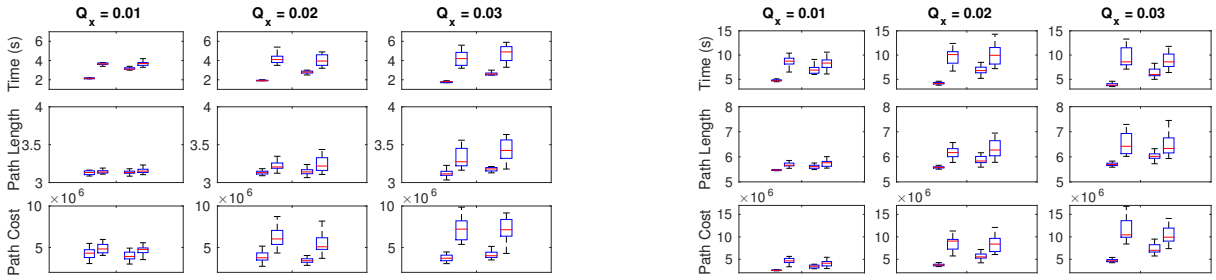
	Q_x	WAM		PR2	
		CL	OL	CL	OL
MDP	0.01	1	1	1	1
	0.02	1	1	1	0.95
	0.03	1	0.85	1	0.5
POMDP	0.01	1	1	1	1
	0.02	1	0.9	1	0.8
	0.03	0.9	0.75	1	0.8

VII. Conclusion

We consider the problem of motion planning and control as probabilistic inference, and we propose an algorithm PIPC

[9]Parameters for this benchmark are set as follows: $K = 40$, $gdist = 0.2$, $t_{max} = 20$, $\Delta t = 0.2$, $t_h = 2$, $n_{ip} = 20$, $\sigma_m = 0.01$, $\sigma_g = 1$, $\sigma_{fix} = 10^{-4}$, $Q_u = 10$, $\sigma_{obs} = 0.02$, $\epsilon = 1$.

[10]Parameters for this benchmark are set as follows: $K = 20$, $gdist = 0.065$, $t_{max} = 15$, $\Delta t = 0.1$, $t_h = 2$, $n_{ip} = 10$, $\sigma_m = 0.005$, $\sigma_g = 0.03$, $\sigma_{fix} = 10^{-4}$, $Q_u = 10$, $\epsilon = 0.1$, $\sigma_{obs} = 0.008$ (WAM), $\sigma_{obs} = 0.005$ (PR2).



(a) (b)

Fig. 4: Results of successful runs with increasing Q_x on (a) the WAM and (b) the PR2 robot arms.

for solving this problem that can exploit intrinsic sparsity in continuous-time stochastic systems. In particular, PIPC can address performance indices given by arbitrary, higher-order nonlinear factors and a general exponential-integral-quadratic factor. Despite PIPC solving a continuous-time problem, its complexity scales only linearly in the number of nonlinear factors, thus making online simultaneous planning and control possible in receding/finite horizon MDP/POMDP problems.

Acknowledgments

The authors would like to thank Jing Dong for help with the GTSAM interface. This material is based upon work supported by NSF CRII Award No. 1464219 and NSF NRI Award No. 1637758.

References

[1] D. P. Bertsekas, *Dynamic programming and optimal control*. Athena Scientific Belmont, MA, 1995, vol. 1, no. 2.

[2] R. C. Arkin, *Behavior-based robotics*. MIT press, 1998.

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

[4] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, vol. 2. IEEE, 2000, pp. 995–1001.

[5] A. Byravan, B. Boots, S. S. Srinivasa, and D. Fox, "Space-time functional gradient optimization for motion planning," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 6499–6506.

[6] 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.

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

[8] Z. Marinho, B. Boots, A. Dragan, A. Byravan, G. J. Gordon, and S. Srinivasa, "Functional gradient motion planning in reproducing kernel hilbert spaces," in *Proceedings of Robotics: Science and Systems (RSS-2016)*, 2016.

[9] J. Dong, M. Mukadam, F. Dellaert, and B. Boots, "Motion planning as probabilistic inference using Gaussian processes and factor graphs," in *Proceedings of Robotics: Science and Systems (RSS-2016)*, 2016.

[10] M. Toussaint, "Newton methods for k-order Markov constrained motion problems," *arXiv preprint arXiv:1407.0414*, 2014.

[11] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[12] R. Tedrake, I. R. Manchester, M. Tobenkin, and J. W. Roberts, "LQR-trees: Feedback motion planning via sums-of-squares verification," *The International Journal of Robotics Research*, 2010.

[13] H. J. Kappen, "Linear theory for control of nonlinear stochastic systems," *Physical review letters*, vol. 95, no. 20, p. 200201, 2005.

[14] S. Levine and V. Koltun, "Guided policy search." in *ICML (3)*, 2013, pp. 1–9.

[15] M. Deisenroth and C. E. Rasmussen, "PILCO: A model-based and data-efficient approach to policy search," in *Proceedings of the 28th International Conference on machine learning (ICML-11)*, 2011, pp. 465–472.

[16] D. Mayne, "A second-order gradient method for determining optimal trajectories of non-linear discrete-time systems," *International Journal of Control*, vol. 3, no. 1, pp. 85–95, 1966.

[17] E. Todorov and W. Li, "A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems," in *American Control Conference, 2005. Proceedings of the 2005*. IEEE, 2005, pp. 300–306.

[18] E. Todorov and Y. Tassa, "Iterative local dynamic programming," in *2009 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*. IEEE, 2009, pp. 90–95.

[19] H. Attias, "Planning by probabilistic inference," in *AISTATS*, 2003.

[20] M. Toussaint, "Robot trajectory optimization using approximate inference," in *Proceedings of the 26th annual international conference on machine learning*. ACM, 2009, pp. 1049–1056.

[21] M. Toussaint and A. Storkey, "Probabilistic inference for solving discrete and continuous state Markov decision processes," in *Proceedings of the 23rd international conference on Machine learning*. ACM, 2006, pp. 945–952.

[22] S. Levine and V. Koltun, "Variational policy search via trajectory optimization," in *Advances in Neural Information Processing Systems*, 2013, pp. 207–215.

[23] H. J. Kappen, V. Gómez, and M. Opper, "Optimal control as a graphical model inference problem," *Machine Learning*, vol. 87, no. 2, pp. 159–182, 2012. [Online]. Available: http://dx.doi.org/10.1007/s10994-012-5278-7

[24] K. Rawlik, M. Toussaint, and S. Vijayakumar, "On stochastic optimal control and reinforcement learning by approximate inference," *Proceedings of Robotics: Science and Systems VIII*, 2012.

[25] P. Kumar and J. Van Schuppen, "On the optimal control of stochastic systems with an exponential-of-integral performance index," *Journal of mathematical analysis and applications*, vol. 80, no. 2, pp. 312–332, 1981.

[26] C. E. Rasmussen, *Gaussian processes for machine learning*, 2006.

[27] S. Sarkka, A. Solin, and J. Hartikainen, "Spatiotemporal learning via infinite-dimensional Bayesian filtering and smoothing: A look at Gaussian process regression through Kalman filtering," *IEEE Signal Processing Magazine*, vol. 30, no. 4, pp. 51–61, 2013.

[28] C. M. Bishop, "Pattern recognition," *Machine Learning*, vol. 128, pp. 1–58, 2006.

[29] M. Mukadam, C.-A. Cheng, X. Yan, and B. Boots, "Approximately optimal continuous-time motion planning and control via probabilistic inference," *arXiv preprint arXiv:1702.07335*, 2017.

[30] A. Boularias, "A predictive model for imitation learning in partially observable environments," in *Machine Learning and Applications, 2008. ICMLA'08. Seventh International Conference on*. IEEE, 2008, pp. 83–90.

[31] T. Barfoot, C. H. Tong, and S. Sarkka, "Batch continuous-time trajectory estimation as exactly sparse Gaussian process regression," *Proceedings of Robotics: Science and Systems, Berkeley, USA*, 2014.

[32] X. Yan, V. Indelman, and B. Boots, "Incremental sparse GP regression for continuous-time trajectory estimation & mapping," in *Proceedings of the International Symposium on Robotics Research (ISRR-2015)*, 2015.