JMLR: Workshop and Conference Proceedings 24:1–14, 2012 European Workshop on Reinforcement Learning

Solving Nonlinear Continuous State-Action-Observation POMDPs for Mechanical Systems with Gaussian Noise

Marc Peter Deisenroth MARC@IAS.TU-DARMSTADT.DE
Jan Peters PETERS@IAS.TU-DARMSTADT.DE

Abstract

In this paper, we introduce a novel model-based approach to solving the important subclass of partially observable Markov decision processes (POMDPs) with Gaussian noise in continuous states, actions, and observations. This kind of POMDP frequently appears in robotics and many other real-world control problems. However, except for the linear quadratic Gaussian case, no efficient ways of computing optimal controllers are known. We propose a novel method for efficiently approximating optimal solutions of nonlinear stochastic continuous state-action-observation POMDPs in high dimensions. We use Gaussian processes (GPs) to model both the latent transition dynamics and the measurement mapping. By explicit marginalization over the GP posteriors our method is robust to model errors and can be used for principled belief space inference, policy learning, and policy execution.

Keywords: continuous POMDPs, policy search, Gaussian processes

1. Introduction

Robust and scalable algorithms for planning and decision making under uncertainty are crucial for the development of autonomous systems. In principle, partially observable Markov decision processes (POMDPs) offer one of the most powerful mathematical frameworks for decision making in uncertain environments. However, despite their generality, the lack of efficient solvers for high-dimensional continuous POMDPs has been the key bottleneck for their application in real-world situations.

To date, the vast majority of POMDP solvers are specific for domains with discrete states, actions, or observations (Kaelbling et al., 1998; Ng and Jordan, 2000; Pineau et al., 2003; Poupart et al., 2006; Porta et al., 2006; Ross et al., 2008; Kurniawati et al., 2008). Provably optimal *general* solutions for continuous domains are known only for linear systems with quadratic cost and Gaussian noise (Bertsekas, 2005). For most other cases, approximate solutions are needed.

A standard approach to solving a POMDP is to recast it as a completely observable MDP with a state space that consists of information states. The *information state* consists of either a complete history of actions and observations or the corresponding sufficient statistic, the *belief state* (Kaelbling et al., 1998). Hence, MDP solvers applied to the information space can be used for finding optimal controls/actions in POMDPs.

In the last decade, policy search methods have offered efficient solutions to many continuous real-world MDPs, e.g., in robotics and control (Bagnell and Schneider, 2001; Kober and Peters, 2011; Deisenroth and Rasmussen, 2011). Policy search methods often cope better with large state spaces and imperfect state estimators than value function methods, making them promising for solving large POMDPs (Aberdeen, 2009; Ng and Jordan, 2000). However, learning is generally relatively sample inefficient.

The pilco framework proposed by Deisenroth and Rasmussen (2011) is a model-based policy search method for MDPs that achieved unprecedented sample efficiency. Based on long-term planning, policy parameters are determined using analytic policy gradient computations. Pilco uses a probabilistic Gaussian process (GP) forward dynamics model that explicitly describes model uncertainties. The key to pilco's sample efficiency is that during policy evaluation/planning, the model uncertainties are marginalized out, which makes learning robust to model errors. Due to its robustness to model errors, pilco learns from scratch, i.e., with an uninformative initialization.

The contributions of this paper are fourfold: (i) Based on the pilco framework for MDPs, we introduce an efficient solver for POMDPs with Gaussian noise and continuous states, actions, and observations. GP priors on the transition and measurement functions are used for the purposes of flexibility, robustness to model errors, and computational advantages. We use efficient approximate inference in belief space to compute long-term plans and analytic policy gradients *for learning prior policies*. (ii) We analyze two approximate methods for inference in belief space in terms of computational demand and learning performance. (iii) We use GP-Bayes filters to compute POMDP posterior policies while applying the policy. (iv) Compared to ground-truth optimal solutions, our POMDP solver achieves near-optimal performance and scales well to high dimensions, i.e., 10 or more.

The POMDP solver by Dallaire et al. (2009) is similar to ours and also uses GP dynamics and observation models. However, instead of maintaining a belief state *distribution*, Dallaire et al. (2009) represent the belief state by its MAP estimate. Moreover, model uncertainty is ignored. Additionally, only myopic controllers in finite action spaces are learned, i.e., controls are applied that minimize the immediate cost. Controllers that require long-term planning cannot be learned.

The paper is structured as follows: In Sec. 2, we define the problem and provide background information. In Sec. 3, we present our novel POMDP solver. Its scalability and optimality are analyzed in Sec. 4.

2. Problem Setup and Background

In a POMDP it is assumed that the transition dynamics are Markovian, but the underlying state $\mathbf{x}$ cannot be observed directly. Instead, a probability distribution over latent states is maintained. This belief state summarizes the history of the process and can be computed recursively Bertsekas (2005). In the following, we consider POMDPs with transition and measurement models given by

$$\mathbf{x}_t = f(\mathbf{x}_{t-1}, \mathbf{u}_{t-1}) + \mathbf{w}\,, \qquad \mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_w)\,, \tag{1}$$

$$\mathbf{z}_t = g(\mathbf{x}_t) + \mathbf{v}\,, \qquad \mathbf{v} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_v)\,, \tag{2}$$

where $\mathbf{x} \in \mathbb{R}^D$ is the state, $\mathbf{z} \in \mathbb{R}^E$ is the measurement, and $\mathbf{u} \in \mathbb{R}^F$ is the action/control signal. The transition dynamics f and the measurement function g are often assumed to be known. In this paper, we relax this assumption to having probability distributions over the latent state transitions and the measurement mappings in the form of GPs instead.

Given an initial state distribution $p(\mathbf{x}_0)$, our objective is to find a mapping π from belief states $p(\mathbf{x})$ to distributions over controls $\mathbf{u}$ that minimizes the expected finite-horizon cost $J^\pi = \sum_{t=0}^{T} \mathbb{E}[c(\mathbf{x}_t)]$ where c is an immediate cost function. The mapping π is referred to as a *state-feedback controller* or *policy* and parametrized by θ.

In our POMDP setup, we compute a posterior policy $p(\pi(\mathbf{x}_t, \theta)|\mathbf{z}_{1:i})$ conditioned on i measurements $\mathbf{z}_1, \ldots, \mathbf{z}_i$, where $i \in \{\emptyset, 1, \ldots, T\}$. In the POMDP planning phase, we compute/learn a *prior policy* $p(\pi(\mathbf{x}_t, \theta)|\mathbf{z}_\emptyset)$ since no measurements are available during planning, see Sec. 3.1. These unavailable measurements are implicitly integrated out during planning. In the POMDP execution phase, i.e., when the controller is being applied, measurements $\mathbf{z}_t$ are obtained, and the posterior policies $p(\pi(\mathbf{x}_t, \theta)|\mathbf{z}_{1:t})$ are sequentially determined as described in Sec. 3.2.

2.1. GP Dynamics and Measurement Models

Pilco's probabilistic dynamics model is implemented as a GP, where we use tuples $(\mathbf{x}_{t-1}, \mathbf{u}_{t-1}) \in \mathbb{R}^{D+F}$ as training inputs and differences $\Delta_t = \mathbf{x}_t - \mathbf{x}_{t-1} + \mathbf{w} \in \mathbb{R}^D$, $\mathbf{w} \sim \mathcal{N}(0, \boldsymbol{\Sigma}_w)$, $\boldsymbol{\Sigma}_w = \text{diag}([\sigma^2_{w_1}, \ldots, \sigma^2_{w_D}])$, as training targets. The GP yields one-step predictions

$$p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_{t-1}) = \mathcal{N}(\mathbf{x}_t \,|\, \mu^x_t, \boldsymbol{\Sigma}^x_t)\,, \tag{3}$$

$$\mu^x_t = \mathbf{x}_{t-1} + \mathbb{E}_f[\Delta_t]\,, \quad \boldsymbol{\Sigma}^x_t = \text{var}_f[\Delta_t]\,. \tag{4}$$

In this paper, we consider a prior mean function $m \equiv 0$ and the sum of a Gaussian kernel with automatic relevance determination and a noise kernel, i.e.,

$$k(\tilde{\mathbf{x}}_p, \tilde{\mathbf{x}}_q) = \sigma_f^2 \exp\big(-\tfrac{1}{2}\|\tilde{\mathbf{x}}_p - \tilde{\mathbf{x}}_q\|^2_{\boldsymbol{\Lambda}^{-1}}\big) + \sigma_w^2\delta_{pq} \tag{5}$$

with $\tilde{\mathbf{x}} \coloneqq [\mathbf{x}^\top \mathbf{u}^\top]^\top$ being the control-augmented state. In Eq. (5), we define σ_f^2 as the variance of the latent function f and $\boldsymbol{\Lambda} \coloneqq \text{diag}([\ell_1^2, \ldots, \ell_D^2])$, which depends on the characteristic length-scales ℓ_i. There are n training inputs $\tilde{\mathbf{X}} = [\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_n]$ and corresponding training targets $\mathbf{y} = [\Delta_1, \ldots, \Delta_n]^\top$.

A univariate posterior predictive distribution $p(\Delta_t|\tilde{\mathbf{x}}_{t-1})$ at a test input $\tilde{\mathbf{x}}_{t-1}$ is Gaussian where the mean and variance, see Eq. (4), are explicitly given as

$$\mathbb{E}_f[\Delta_{t-1}] = \mu_\Delta = \mathbf{k}_*^\top(\mathbf{K} + \sigma_w^2\mathbf{I})^{-1}\mathbf{y} = \mathbf{k}_*^\top\beta\,, \tag{6}$$

$$\text{var}_f[\Delta_*] = \sigma^2_\Delta = k_{**} - \mathbf{k}_*^\top(\mathbf{K} + \sigma_w^2\mathbf{I})^{-1}\mathbf{k}_*\,, \tag{7}$$

$$\beta \coloneqq \mathbf{K}^{-1}\mathbf{y}\,, \tag{8}$$

respectively, with $\mathbf{k}_* \coloneqq k(\tilde{\mathbf{X}}, \tilde{\mathbf{x}}_{t-1})$, $k_{**} \coloneqq k(\tilde{\mathbf{x}}_{t-1}, \tilde{\mathbf{x}}_{t-1})$, and the entries of $\mathbf{K}$ are $K_{ij} = k(\tilde{\mathbf{x}}_i, \tilde{\mathbf{x}}_j)$.

For the *measurement model* we also use a prior mean function $m_g \equiv 0$ and the covariance function given in Eq. (5). The training inputs are states $\mathbf{x}_i$, $i = 1, \ldots, n$, the training targets are observations $\mathbf{z}_i = g(\mathbf{x}_i) + \mathbf{v}$.

The posterior GP hyper-parameters (length-scales, signal variances σ_f^2, σ_g^2, noise variances σ_w^2, σ_v^2) of the dynamics and measurement GPs are learned using evidence maximization Rasmussen and Williams (2006).

For multivariate targets, we train conditionally independent GPs for each target dimension, i.e., the GPs are independent for deterministically given test inputs. For *uncertain* inputs, the target dimensions covary (Rasmussen and Williams, 2006).

3. Policy Search in POMDPs

In this paper, we generalize the PILCO framework by Deisenroth and Rasmussen (2011) to an efficient POMDP solver with continuous states, actions, and observations. In POMDPs, we distinguish between inference during planning and inference in an execution phase when the learned controller is applied. In both cases, we plan in belief/information space. In the planning phase, no state measurements are available, and we learn a prior policy using policy search. In the execution phase, we take current measurements into account to compute posterior policies.

3.1. Planning in Belief Space: Prior Policy

To compute the long-term expected cost $J(\theta)$, we iteratively compute long-term predictive distributions $p(\mathbf{x}_1|\mathbf{z}_\emptyset), \dots, p(\mathbf{x}_T|\mathbf{z}_\emptyset)$ given a policy π. External measurements $\mathbf{z}$ are not available, indicated by $\mathbf{z}_\emptyset$. To compute the predictive state distributions, we compute $p(\mathbf{x}_0|\mathbf{z}_\emptyset) \to p(\mathbf{u}_0|\mathbf{z}_\emptyset) \to p(\mathbf{u}_0, \mathbf{x}_0|\mathbf{z}_\emptyset) \to p(\mathbf{x}_1|\mathbf{z}_\emptyset) \to p(\mathbf{u}_1|\mathbf{z}_\emptyset) \to p(\mathbf{u}_1, \mathbf{x}_1|\mathbf{z}_\emptyset) \to \dots \to p(\mathbf{x}_T|\mathbf{z}_\emptyset)$. The predictive belief state distribution $p(\mathbf{x}_t|\mathbf{z}_\emptyset)$ at time t is given by

$$p(\mathbf{x}_t|\mathbf{z}_\emptyset) = \int p(\mathbf{x}_t|\tilde{\mathbf{x}}_{t-1}) p(\tilde{\mathbf{x}}_{t-1}|\mathbf{z}_\emptyset)\, \mathrm{d}\tilde{\mathbf{x}}_{t-1} \tag{9}$$

for $t = 1, \dots, T$. The conditional probability $p(\mathbf{x}_t|\tilde{\mathbf{x}}_{t-1}) = p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_{t-1})$ is given in Eq. (3). Note that we implicitly integrate out observations $\mathbf{z}_1, \dots, \mathbf{z}_T$, which are not available during planning. The integral in Eq. (9) cannot be computed analytically and requires approximations. Once the distributions $p(\mathbf{x}_1|\mathbf{z}_\emptyset), \dots, p(\mathbf{x}_T|\mathbf{z}_\emptyset)$ are determined, the expected long-term cost $J(\theta)$ and its gradients with respect to the policy parameters

$$\frac{\mathrm{d}J(\theta)}{\mathrm{d}\theta} = \sum_{t=0}^{T} \left(\frac{\partial \mathbb{E}[c(\mathbf{x}_t)|\mathbf{z}_\emptyset]}{\partial \mu^x_{t|\emptyset}} \frac{\mathrm{d}\mu^x_{t|\emptyset}}{\mathrm{d}\theta} + \frac{\partial \mathbb{E}[c(\mathbf{x}_t)|\mathbf{z}_\emptyset]}{\partial \boldsymbol{\Sigma}^x_{t|\emptyset}} \frac{\mathrm{d}\boldsymbol{\Sigma}^x_{t|\emptyset}}{\mathrm{d}\theta} \right)$$

can be computed and used for policy learning. For details on the gradient computations, we refer to (Deisenroth and Rasmussen, 2011).

In the following, we detail two deterministic Gaussian approximations to Eq. (9), the moment-matching approximation that is used in the PILCO MDP framework (Deisenroth and Rasmussen, 2011) and an approximation based on linearization of the posterior GP mean function proposed by Ko and Fox (2008).[1] We assume that a Gaussian approximation

$$\mathcal{N}\left(\begin{bmatrix} \mathbf{x}_{t-1} \\ \mathbf{u}_{t-1} \end{bmatrix} \middle| \begin{bmatrix} \mu^x_{t-1|\emptyset} \\ \mu^u_{t-1|\emptyset} \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}^x_{t-1|\emptyset} & \boldsymbol{\Sigma}^{xu}_{t-1|\emptyset} \\ \boldsymbol{\Sigma}^{ux}_{t-1|\emptyset} & \boldsymbol{\Sigma}^u_{t-1|\emptyset} \end{bmatrix} \right) \tag{10}$$

1. Both approximations allow for the computation of analytic policy gradients within the PILCO framework. Thus, we can deal with thousands of policy parameters. Standard policy search methods rely on finite differences and sampling for gradient estimation (Williams, 1992; Peters and Schaal, 2008), which does not scale well to high-dimensional parameter vectors.

of the control-augmented state $\tilde{\mathbf{x}}_{t-1}$ is known. The objective of either approximate inference method is to compute the mean $\mu^x_{t|\emptyset}$ and the covariance $\boldsymbol{\Sigma}^x_{t|\emptyset}$ of $p(\mathbf{x}_t|\emptyset)$, which are given by

$$\mu^x_{t|\emptyset} = \mu^x_{t-1|\emptyset} + \mu_{\Delta|\emptyset} \tag{11}$$

$$\boldsymbol{\Sigma}^x_{t|\emptyset} = \boldsymbol{\Sigma}^x_{t-1|\emptyset} + \boldsymbol{\Sigma}_{\Delta|\emptyset} + \boldsymbol{\Sigma}^x_{t-1,\Delta|\emptyset} + \boldsymbol{\Sigma}^x_{\Delta,t-1|\emptyset} \tag{12}$$

$$\boldsymbol{\Sigma}^x_{t-1,\Delta|\emptyset} = \mathrm{cov}[\mathbf{x}_{t-1}, \mathbf{u}_{t-1}|\mathbf{z}_\emptyset](\boldsymbol{\Sigma}^u_{t-1|\emptyset})^{-1}\mathrm{cov}[\mathbf{u}_{t-1}, \boldsymbol{\Delta}_t|\mathbf{z}_\emptyset]\,. \tag{13}$$

The cross-covariance $\boldsymbol{\Sigma}^x_{t-1,\Delta|\emptyset}$ depends on the policy parametrization but can often be determined analytically. Since $\tilde{\mathbf{x}}_{t-1}$ and $\boldsymbol{\Delta}_t$ are both random variables, we apply the laws for the expectation and the covariance of the sum of random variables in Eqs. (11)–(12).

3.1.1. Moment Matching

Following Deisenroth and Rasmussen (2011), we will now summarize how to approximate the predictive distribution $p(\mathbf{x}_t)$ in Eq. (9) using moment matching. Hence, we analytically compute the mean μ_t and the covariance $\boldsymbol{\Sigma}_t$ of $p(\mathbf{x}_t)$, see Eqs. (11)–(12).

Using the law of iterated expectations, we obtain

$$\mu_{\Delta|\emptyset} = \mathbb{E}_{\tilde{\mathbf{x}}_{t-1}}\left[\mathbb{E}_f[f(\tilde{\mathbf{x}}_{t-1})|\tilde{\mathbf{x}}_{t-1}]|\mathbf{z}_\emptyset\right] = \mathbb{E}_{\tilde{\mathbf{x}}_{t-1}}[m_f(\tilde{\mathbf{x}}_{t-1})|\mathbf{z}_\emptyset]\,,$$

where m_f is the posterior mean function of the dynamics GP. For target dimension $a = 1, \dots, D$, we obtain

$$\mu^a_{\Delta|\emptyset} = \mathbf{q}_a^\top \beta_a\,, \tag{14}$$

$$q_{a_i} = \frac{\sigma_f^2}{\sqrt{|\tilde{\boldsymbol{\Sigma}}_{t-1|\emptyset}\boldsymbol{\Lambda}_a^{-1}+\mathbf{I}|}} \exp\left(-\tfrac{1}{2}\nu_i^\top(\tilde{\boldsymbol{\Sigma}}_{t-1|\emptyset}+\boldsymbol{\Lambda}_a)^{-1}\nu_i\right),$$

$$\nu_i \coloneqq (\tilde{\mathbf{x}}_i - \tilde{\mu}_{t-1|\emptyset}) \tag{15}$$

for $i = 1, \dots, n$, where β_a is defined in Eq. (8).

For the predictive covariance matrix $\boldsymbol{\Sigma}^x_{t|\emptyset}$ in Eq. (12), we focus on computing $\boldsymbol{\Sigma}_{\Delta|\emptyset}$. Using the law of iterated variances, the entries of $\boldsymbol{\Sigma}_{\Delta|\emptyset}$ for target dimensions $a, b = 1, \dots, D$ are

$$\sigma^2_{aa} = \mathbb{E}_{\tilde{\mathbf{x}}_{t-1}}\left[\mathrm{var}_f[\Delta_a|\tilde{\mathbf{x}}_{t-1}]|\mathbf{z}_\emptyset\right] + \mathbb{E}_{f,\tilde{\mathbf{x}}_{t-1}}[\Delta_a^2|\mathbf{z}_\emptyset] - (\mu^a_{\Delta|\emptyset})^2, \tag{16}$$

$$\sigma^2_{ab} = \mathbb{E}_{f,\tilde{\mathbf{x}}_{t-1}}[\Delta_a\Delta_b|\mathbf{z}_\emptyset] - \mu^a_{\Delta|\emptyset}\mu^b_{\Delta|\emptyset}\,, \quad a \neq b\,, \tag{17}$$

respectively, where μ^a_Δ is known from Eq. (14) and $\Delta_a \coloneqq \mathbf{x}_t^{(a)} - \mathbf{x}_{t-1}^{(a)}$. The off-diagonal terms σ^2_{ab} do not contain an additional term $\mathbb{E}_{\tilde{\mathbf{x}}_{t-1}}[\mathrm{cov}_f[\Delta_a, \Delta_b|\tilde{\mathbf{x}}_{t-1}]|\mathbf{z}_\emptyset]$ because of the conditional independence assumption used for GP training: Target dimensions do not covary for a *given* $\tilde{\mathbf{x}}_{t-1}$.

For the term common to both σ^2_{aa} and σ^2_{ab}, we obtain

$$\mathbb{E}_{f,\tilde{\mathbf{x}}_{t-1}}[\Delta_a\Delta_b|\mathbf{z}_\emptyset] = \beta_a^\top \mathbf{Q}\beta_b\,, \tag{18}$$

where the entries Q_{ij} of $\mathbf{Q} \in \mathbb{R}^{n\times n}$ are given as

$$Q_{ij} = \frac{k_a(\tilde{\mathbf{x}}_i, \tilde{\mu}_{t-1|\emptyset})k_b(\tilde{\mathbf{x}}_j, \tilde{\mu}_{t-1|\emptyset})}{\sqrt{|\mathbf{R}|}} \exp\left(\tfrac{1}{2}\mathbf{z}_{ij}^\top \mathbf{R}^{-1}\tilde{\boldsymbol{\Sigma}}_{t-1|\emptyset}\mathbf{z}_{ij}\right) \tag{19}$$

with $\mathbf{R} := \tilde{\boldsymbol{\Sigma}}_{t-1|\emptyset}(\boldsymbol{\Lambda}_a^{-1} + \boldsymbol{\Lambda}_b^{-1}) + \mathbf{I}$ and $\mathbf{z}_{ij} := \boldsymbol{\Lambda}_a^{-1}\nu_i + \boldsymbol{\Lambda}_b^{-1}\nu_j$ with ν_i taken from Eq. (15). Hence, the *off-diagonal* entries σ_{ab}^2 of $\boldsymbol{\Sigma}_{\Delta|\emptyset}$ are fully determined by Eqs. (14)–(15) and (17)–(19).

From Eq. (16), we see that the *diagonal* entries σ_{aa}^2 of $\boldsymbol{\Sigma}_{\Delta|\emptyset}$ contain an additional term

$$\mathbb{E}_{\tilde{\mathbf{x}}_{t-1}}\big[\mathrm{var}_f[\Delta_a|\tilde{\mathbf{x}}_{t-1}]|\mathbf{z}_\emptyset\big] = \sigma_{f_a}^2 - \mathrm{tr}\big(\mathbf{K}_a^{-1}\mathbf{Q}\big) + \sigma_{w_a}^2 \tag{20}$$

with $\mathbf{Q}$ given in Eq. (19). This concludes the computation of $\boldsymbol{\Sigma}_{\Delta|\emptyset}$ and $\boldsymbol{\Sigma}_{t|\emptyset}^x$ is computed using Eq. (12).

The moment-matching approximation minimizes the KL divergence $\mathrm{KL}(p||q)$ between the true distribution p and an approximate Gaussian distribution q. This is generally a conservative approximation, i.e., q has probability mass where p has mass (Bishop, 2006).

3.1.2. Linearizing the GP Mean Function

An alternative way of approximating the predictive GP distributions $p(\mathbf{x}_1|\mathbf{z}_\emptyset), \dots, p(\mathbf{x}_T|\mathbf{z}_\emptyset)$ is to linearize the posterior GP mean function Ko and Fox (2009). Given this linearized function, we apply standard results for mapping Gaussian distributions through linear models. Linearizing the posterior GP mean function yields to a predicted mean that corresponds to the posterior GP mean function evaluated at the mean of the input distribution, i.e.,

$$\mu_{\Delta|\emptyset}^a = \mathbb{E}_f[f_a(\mu_{t-1})|\mathbf{z}_\emptyset] = \mathbf{r}_a^\top \beta_a \tag{21}$$

$$r_{a_i} = \sigma_{f_a}^2 \exp\big(-\tfrac{1}{2}(\tilde{\mathbf{x}}_i - \tilde{\mu}_{t-1|\emptyset})^\top \boldsymbol{\Lambda}_a^{-1}(\tilde{\mathbf{x}}_i - \tilde{\mu}_{t-1|\emptyset})\big) \tag{22}$$

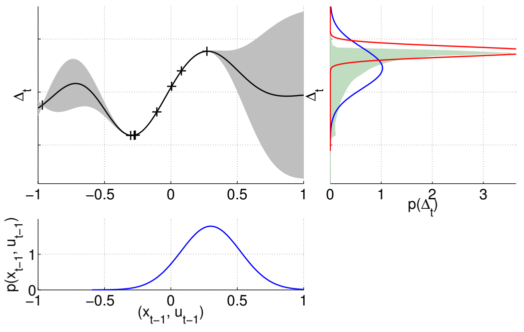
for $i = 1, \dots, n$ and target dimensions $a = 1, \dots, D$, where β_a is given in Eq. (8). The covariance matrix $\boldsymbol{\Sigma}_{\Delta|\emptyset}$ of the GP difference prediction is

$$\boldsymbol{\Sigma}_{\Delta|\emptyset} = \mathbf{V}\tilde{\boldsymbol{\Sigma}}_{t-1|\emptyset}\mathbf{V}^\top + \boldsymbol{\Sigma}_w\,, \mathbf{V} = \frac{\partial \mu_{\Delta|\emptyset}}{\partial \tilde{\mu}_{t-1|\emptyset}} = \beta_a^\top \frac{\partial \mathbf{r}_a}{\partial \tilde{\mu}_{t-1|\emptyset}}\,, \tag{23}$$

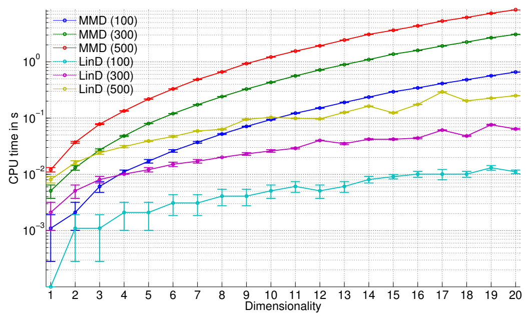
where $\mathbf{r}_a$ is given in Eq. (22). In Eq. (23), $\boldsymbol{\Sigma}_w$ is a diagonal matrix whose entries are the model uncertainty plus the noise variance evaluated at $\tilde{\mu}_{t-1}$. This means "model uncertainty" no longer depends on the density of the data points. Instead it is assumed constant.

Using linearization, the approximation optimality of the moment matching is lost. Fig. 1(*a*) illustrates some differences between approximate predictions using moment matching and linearization. In particular, it is shown that the predictive distribution based on linearization can be too tight. These errors vanish when the input distribution is relatively peaked.

Fig. 1(*b*) shows the computational demand of approximate inference and derivative computations that are needed for policy search of a single time step, using moment matching (MMD) and linearization (LinD) for data sets of size $n = 100, 300, 500$ and dimensions $D = 1, \dots, 20$. Average performances and twice the standard errors are shown. Especially in high dimensions, linearization is computationally substantially cheaper. This speedup is largely due to the simplified treatment of model uncertainty.



(*a*) Predictive distributions based on moment matching (blue) and linearization (red). Using linearization for approximate inference can lead to predictive distributions that are too tight.



(*b*) Computational demand of approximate inference and derivative computations for a single time slice using moment matching (MMD) and linearization (LinD) for $n = 100, 300, 500$ data points and $D = 1, \dots, 20$ dimensions.

Figure 1: Approximate inference in GPs using moment matching and linearization of the posterior mean. While moment matching does not tend to predictive distributions that are too tight (left), linearizing the mean is substantially faster (right).

3.2. Execution Phase: Posterior Policy

Unlike in MDPs, we do not have direct access to the latent states when controlling the system. Instead, we have access to noisy observations $\mathbf{z}_t = g(\mathbf{x}_t)+\mathbf{v}, t = 1, \dots, T, \mathbf{v} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_v)$ of the latent state. Nevertheless, the state-feedback controller applies controls based on potential values of the latent state $\mathbf{x}$. To compute a posterior policy $p(\pi_t(\theta^*)|\mathbf{z}_{1:t})$, we first compute a posterior (belief state) distribution $p(\mathbf{x}_t|\mathbf{z}_{1:t})$ on $\mathbf{x}_t$ using filtering techniques. The posterior policy distribution

$$p(\pi_t(\theta^*)|\mathbf{z}_{1:t}) = \int p(\pi(\mathbf{x}_t, \theta^*)|\mathbf{x}_t) p(\mathbf{x}_t|\mathbf{z}_{1:t})\, \mathrm{d}\mathbf{x}_t \tag{24}$$

is determined based on the belief state. Without loss of generality, we assume that we obtain a measurement $\mathbf{z}_t$ at each time step $t = 1, \dots, T$ of the execution horizon. To compute the posterior distribution $p(\mathbf{x}_t|\mathbf{z}_{1:t})$, we apply GP-Bayes filters introduced by Ko and Fox (2008); Deisenroth et al. (2009). Note that Eq. (24) is similar to the time update of a Kalman filter, but it does not correspond to an average over Q-functions. Averaging over policies as in Eq. (24) is appropriate for unimodal distributions. In the LQG case, it is optimal (Bertsekas, 2005). However, a Gaussian approximation is very poor in discrete POMDPs, which we do not deal with in this paper.

3.2.1. GP-Bayes Filters for State Estimation

The GP-Bayes filters approximate $p(\mathbf{x}_t|\mathbf{z}_{1:t})$ by first computing a Gaussian approximation

$$\mathcal{N}\left(\begin{bmatrix}\mathbf{x}_t\\ \mathbf{z}_t\end{bmatrix} \middle| \begin{bmatrix}\mu^x_{t|t-1}\\ \mu^z_{t|t-1}\end{bmatrix}, \begin{bmatrix}\mathbf{\Sigma}^x_{t|t-1} & \mathbf{\Sigma}^{xz}_{t|t-1}\\ \mathbf{\Sigma}^{zx}_{t|t-1} & \mathbf{\Sigma}^z_{t|t-1}\end{bmatrix}\right) \tag{25}$$

to the joint distribution $p(\mathbf{x}_t, \mathbf{z}_t | \mathbf{z}_{1:t-1})$, either based on moment matching (GP-ADF) or linearization (GP-EKF), see Secs. 3.1.1 and 3.1.2, respectively. The computation of this joint distribution is sufficient for Gaussian filtering (Deisenroth and Ohlsson, 2011). Second, the mean and the covariance of a Gaussian approximation to the desired posterior $p(\mathbf{x}_t|\mathbf{z}_{1:t})$ are

$$\mu^x_{t|t} = \mu^x_{t|t-1} + \mathbf{\Sigma}^{xz}_{t|t-1}(\mathbf{\Sigma}^z_{t|t-1})^{-1}(\mathbf{z}_t - \mu^z_{t|t-1})\,, \tag{26}$$
$$\mathbf{\Sigma}^x_{t|t} = \mathbf{\Sigma}^x_{t|t-1} - \mathbf{\Sigma}^{xz}_{t|t-1}(\mathbf{\Sigma}^z_{t|t-1})^{-1}\mathbf{\Sigma}^{zx}_{t|t-1}\,. \tag{27}$$

by Gaussian conditioning on Eq. (25)

3.2.2. Control Selection

The belief state distribution $p(\mathbf{x}_t|\mathbf{z}_{1:t})$ causes the learned policy π^* to output a control *distribution* $p(\mathbf{u}_t|\mathbf{z}_{1:t})$, see Eq. (24). In the execution phase, we have to decide on a single control and choose the mean $\mu^u_{t|t} = \mathbb{E}_{\mathbf{x}_t}[\pi(\mathbf{x}_t, \theta^*)|\mathbf{z}_{1:t}]$ of the posterior control distribution. Note that for nonlinear policies, this control selection differs from a certainty-equivalence controller. We do not consider the case of actuator noise. Thus, a Gaussian approximation to the joint distribution $p(\tilde{\mathbf{x}}_t) = p(\mathbf{x}_t, \mathbf{u}_t)$ is now given as

$$\mathcal{N}\left(\begin{bmatrix}\mathbf{x}_t\\ \mathbf{u}_t\end{bmatrix} \,\middle|\, \begin{bmatrix}\mu^x_{t|t}\\ \mu^u_{t|t}\end{bmatrix}, \begin{bmatrix}\mathbf{\Sigma}^x_{t|t} & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{bmatrix}\right)\,. \tag{28}$$

Note that this Gaussian approximation during interaction differs from the joint distribution during planning, see Eq. (10): During planning, we map control *distributions* through the dynamic system, whereas we decide on a single control signal in the execution phase.

4. Results

In the evaluations, we will show the following properties of our POMDP solver: (i) It works near optimally compared to the optimal solution for LQG systems, i.e., the only problem class which we know how to solve exactly. (ii) When only minimally altering this problem class, the optimal solution for the linear system breaks down while our method still works well. (iii) We show that our POMDP solver even works for highly nonlinear problems where general solvers are unknown.

In our experiments, we applied the following steps until convergence. After initializing the policy parameters, random controls were applied to collect an initial data set, and the GP dynamics and measurement models were trained.[2] Based on these models, policy search using the results from Sec. 3.1 returned optimized policy parameters θ^*. When executing the policy $\pi(\mathbf{x}, \theta^*)$, we might have real-time constraints and, hence, cannot identify the latent states after each time step. Thus, we maintained a posterior distribution over the latent state by performing GP filtering, see Sec. 3.2.1. The controls $\mathbf{u}_t$, $t = 1, \ldots, T$ to be applied were determined based on the posteriors $p(\mathbf{u}_t|\mathbf{z}_{1:t})$, see Sec. 3.2.2. After a single rollout, the recorded trajectory was used to update the GP models and the policy.

2. We employed the idealizing assumption that in a (potentially time consuming) post-processing step the latent variables can be identified such that GP training is a supervised learning problem.

Table 1: Average trajectory-loss ratios of LQGC over both POMDP-pilco and an uncontrolled system.

System	Controller	Dimensionality								
		2	3	4	5	6	7	8	9	10
Linear	POMDP pilco	0.97	0.99	0.96	0.97	0.89	0.98	0.97	0.94	0.86
	Uncontrolled	0.9	0.71	0.16	0.52	0.47	0.42	0.23	0.24	0.01
Tanh	POMDP pilco	1.00	0.98	0.97	0.95	0.93	0.95	$\mathbf{1.5 \times 10^7}$	$\mathbf{1.9 \times 10^8}$	$\mathbf{3.9 \times 10^6}$
	Uncontrolled	0.83	0.41	0.10	0.18	0.19	0.16	3.5×10^6	5.1×10^7	1.1×10^6

4.1. Comparison to Ground Truth: LQG

In the following, we compare our approach to a provably optimal controller in linear POMDPs with Gaussian noise and quadratic costs. We consider time-invariant systems with incomplete state information

$$\mathbf{x}_t = \mathbf{A}\mathbf{x}_{t-1} + \mathbf{B}\mathbf{u}_{t-1} + \mathbf{w}\,, \quad \mathbf{w} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_w) \tag{29}$$

$$\mathbf{z}_t = \mathbf{C}\mathbf{x}_t + \mathbf{v}\,, \quad \mathbf{v} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_v)\,, \tag{30}$$

where $\mathbf{x} \in \mathbb{R}^D, \mathbf{z} \in \mathbb{R}^E, \mathbf{u} \in \mathbb{R}^F$. The matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}$ are of appropriate dimensions. We assume that the noise covariance matrices $\boldsymbol{\Sigma}_w, \boldsymbol{\Sigma}_v$ are diagonal.

We used a quadratic cost $c(\mathbf{x}_t) = \mathbf{x}_t^\top \mathbf{Q}_0 \mathbf{x}_t$, where $\mathbf{Q}_0$ is symmetric positive definite and $\mathbf{x}_{\text{target}} = \mathbf{0}$. The optimal controller for the system in Eqs. (29)–(30) is given by $\mathbf{u}_t^* = -\mathbf{L}_t \mu_{t|t}^x$, where $\mu_{t|t}^x = \mathbb{E}[\mathbf{x}_t|\mathbf{z}_{1:t}]$ and $\mathbf{L}_t$ is an optimal feedback gain matrix (Bertsekas, 2005).

We *randomly generated LQG problems*, see Eqs. (29)–(30), with $D = E = F = 2, \ldots, 10$. This means, we randomly generated the matrices $\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{Q}_0, \boldsymbol{\Sigma}_w, \boldsymbol{\Sigma}_v$ and the mean of the prior distribution $p(\mathbf{x}_0) = \mathcal{N}(\mu_0, \mathbf{I})$, where $\mu_0 \sim \mathcal{N}(\mathbf{0}, 5\mathbf{I})$. All LQG problems were both controllable and observable (Bertsekas, 2005). The planning horizon was $T = 20$ time steps.

Despite the linearity of the system, we trained GP dynamics and measurement models using nonlinear Gaussian covariance functions. For the controller π, we chose a linear parametrization, i.e., $\pi(\mathbf{x}) = \boldsymbol{\Phi}\mathbf{x}$, where $\theta = \boldsymbol{\Phi} \in \mathbb{R}^{F \times D}$ are the policy parameters to be learned. In this section, only the moment matching approximation (see Sec. 3.1.1) is considered.

POMDP-pilco performed 15 policy searches. Thus, the learned GP models were based on only up to 320 data points (15 controlled rollouts plus a single initial random rollout). The controllers were applied to control the system starting from 1000 initial states randomly sampled from $p(\mathbf{x}_0)$. This learning and test procedure was repeated $N = 4$ times with different random LQG problems. The upper half of Tab. 1 describes the average performance ratios of the optimal ground-truth LQG controller (LQGC) with respect to both the learned POMDP-pilco controller and an uncontrolled system, i.e., $\mathbf{u} = \mathbf{0}$. Even in 10D, our learned data-driven controller is competitive with a performance loss of 14% and performs substantially better than the uncontrolled system. However, we also noticed a small performance drop from dimension six onward. This drop is a result of relatively small data set used for GP training. Further, the number of policy parameters scales quadratically with the dimensionality, which makes their optimization more prone to local minima due to the use of nonlinear kernels.

4.2. Limits of LQGC

In the following, we will show that when minimally altering the LQG problem class, the optimal LQGC applied to a linearized POMDP can degrade. On the other hand, our POMDP pilco controller appropriately deals with the nonlinear POMDP and can learn controllers superior to LQGC. We consider the moderately nonlinear system

$$\mathbf{x}_t = \tanh(\mathbf{A}\mathbf{x}_{t-1}) + \mathbf{B}\mathbf{u}_{t-1} + \mathbf{w}\,, \tag{31}$$

$$\mathbf{z}_t = \tanh(\mathbf{C}\mathbf{x}_t) + \mathbf{v}\,, \tag{32}$$

where $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, 0.1\mathbf{I})$ and $\mathbf{v} \sim \mathcal{N}(\mathbf{0}, 0.1\mathbf{I})$. To apply LQGC, we globally linearize Eqs. (31)–(32) at the origin. This approximation is good in the non-saturating regime of $\tanh(\cdot)$. We initialized our POMDP solver *identically* to the linear case, see Sec. 4.1. Again, we generated random POMDPs for $D = E = F = 2, \dots, 10$ and evaluated them as in Sec. 4.1.

Tab. 1 shows that in low dimensions, POMDP pilco performed as well as in the linear case, compared to the linearized system. In higher dimensions, however, it was much more likely that the latent state ended up in a saturating area of tanh. Then, the LQGC failed dramatically as shown in the lower half of Tab. 1. The reason for this failure is that the linearization around the origin introduces big errors in the saturating regions of tanh resulting in very large values of the optimal feedback gain matrix $\mathbf{L}$, resulting in even worse "overshooting" at the next time step. Even an uncontrolled system performed better in this case.

4.3. The POMDP Cart-Pole Swingup

We applied our proposed method to a POMDP version of the cart-pole swingup benchmark problem, where the state of the system cannot be measured directly. In the cart-pole swingup problem, we consider a cart running on a track with a freely swinging pendulum attached to it. Initially, the pendulum hangs downward. By applying forces $u \in [-10, 10]\,\mathrm{N}$ to the cart, the objective is to swing the pendulum up and to balance it in the upright position in the center of the track. Neither myopic nor linear control can solve the task.

4.3.1. Setup

The latent state $\mathbf{x} = [x, \dot{x}, \phi, \dot{\phi}]$ evolved according to the differential equations (DEQs)

$$\ddot{x} = \frac{2ml\dot{\phi}^2 \sin\phi + 3mg \sin\phi\cos\phi + 4u(t) - 4b\dot{x}}{4(M+m) - 3m\cos^2\phi} \tag{33}$$

$$\ddot{\phi} = \frac{-3ml\dot{\phi}^2 \sin\phi\cos\phi - 6(M+m)g\sin\phi - 6(u(t) - b\dot{x})\cos\phi}{4l(m+M) - 3ml\cos^2\phi} \tag{34}$$

where x and $\dot{x}$ are the position and the velocity of the cart, respectively, ϕ is the angle of the pendulum (measured counter-clockwise from hanging down), and $\dot{\phi}$ the corresponding angular velocity. In Eqs. (33)–(34), $M = 0.5\,\mathrm{kg}$ was the cart's mass, $m = 0.5\,\mathrm{kg}$ the pendulum's mass, $b = 0.1\,\mathrm{N\,s/m}$ the coefficient of friction between the cart and the ground, $l = 0.6\,\mathrm{m}$ the length of the pendulum, and $g = 9.82\,\mathrm{m/s^2}$ the acceleration of gravity. The transition function f was $f(\mathbf{x}_t, u_t) = \int_t^{t+\Delta_t} \begin{bmatrix}\dot{x} & \ddot{x} & \dot{\phi} & \ddot{\phi}\end{bmatrix}^\top (\tau)\,\mathrm{d}\tau$, where $\Delta_t = 0.1\,\mathrm{s}$. The successor state

$$\mathbf{x}_{t+1} = \mathbf{x}_{t+\Delta_t} = f(\mathbf{x}_t, u_t) + \mathbf{w}\,, \quad \mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_w) \tag{35}$$

was computed using an ODE solver with a zero-order hold control signal $u(\tau)$. The system noise covariance in Eq. (35) was set to $\boldsymbol{\Sigma}_w = \mathrm{diag}([0.01\,\mathrm{m}^2, 0.1\,\mathrm{m}^2/\mathrm{s}^2, 0.01\,\mathrm{rad}^2, 0.1\,\mathrm{rad}^2/\mathrm{s}^2])$.

At every discrete time step Δ_t, the latent state $\mathbf{x}_t$ was measured indirectly: We observed the difference $\boldsymbol{\delta}$ and its time derivative $\dot{\boldsymbol{\delta}}$ of the (x, y)-coordinate of target position (pendulum inverted in the middle of the track) and the current (x, y)-coordinate of the tip of the pendulum in a global coordinate system. Thus, the measurement equation is nonlinear in the state:

$$\mathbf{z}_t = \begin{bmatrix}\boldsymbol{\delta}\\ \dot{\boldsymbol{\delta}}\end{bmatrix} + \mathbf{v}\,, \quad \boldsymbol{\delta} = \begin{bmatrix}x_t + l\sin\phi_t\\ -l - l\cos\phi_t\end{bmatrix}. \tag{36}$$

The measurement noise covariance was set to $\boldsymbol{\Sigma}_v = 10^{-3}\mathrm{diag}([1\,\mathrm{m}^2, 1\,\mathrm{m}^2, 1\,\mathrm{m}^2/\mathrm{s}^2, 0.1\,\mathrm{m}^2/\mathrm{s}^2])$.

We chose a saturating cost function $c(\mathbf{x}) = 1 - \exp(-\frac{1}{2}\|\boldsymbol{\delta}\|^2/\sigma_c^2)$, where $\sigma_c = 0.25\,\mathrm{m}$ is the width of the cost function. Note that the pendulum's length is $l = 0.6\,\mathrm{m}$ and for $c(\mathbf{x}) < 1$ it is required that the pendulum is above horizontal. The policy was parametrized as an RBF network, i.e., $\pi(\mathbf{x}, \theta) = \sum_{i=1}^{L} w_i F(\mathbf{x}, \mu_i)$, where $F(\mathbf{x}, \mu_i)$ are axes-aligned Gaussian-shaped basis functions located at $\mu_i \in \mathbb{R}^D$ with shared widths $\mathbf{P}$. The $L(1 + D) + D$ policy parameters θ were the weights w_i, the locations μ_i, and the D widths on the diagonal of $\mathbf{P}$. We set the number of basis functions to $L = 100$ resulting in 404 policy parameters. The latent state prior was $p(\mathbf{x}_0) = \mathcal{N}(\mathbf{0}, 10^{-2}\mathbf{I})$, the finite prediction horizon was 3 s, i.e., 30 time steps. We randomly sampled the initial policy parameters from Gaussians.

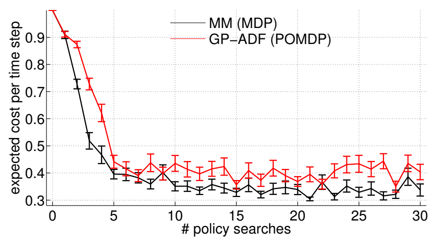
4.3.2. Analysis

We analyzed the learning performances for the two inference methods for planning described in Secs. 3.1.1–3.1.2 and their corresponding GP-Bayes filters during the execution phase, which we refer to as GP-ADF (moment matching) and GP-EKF (linearization), respectively. Further, we compared these methods to baselines, which are the pilco algorithm applied to the stochastic cart-pole swingup *MDP*, i.e., the measurement Eq. (36) is the identity mapping with $\boldsymbol{\Sigma}_v = \mathbf{0}$. Our results support two conclusions: First, the learned POMDP controllers were close to the corresponding MDP controllers that require access to the latent state to execute the learned policies. Second, approximate inference using moment matching led to better performance than approximate inference using linearization.

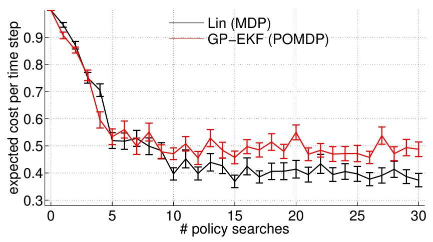
Using $N = 4$ independent random initializations, 30 policy searches were performed for each with the following setups: moment matching in an MDP (corresponds to MDP pilco), GP-ADF in a POMDP, linearization in an MDP, and GP-EKF in a POMDP. After each policy search, a single test rollout with the learned policy was performed, adding 30 data points to the GP training sets. In the MDP setup, controls were computed based on the latent state, in the POMDP case, controls were applied depending on the posterior state distributions determined by the GP-Bayes filters.

First and most importantly, all learned controllers succeeded in solving the task. To evaluate the learning performances in more detail, we applied each intermediate policy of the four learning setups 10 times to the system (MDP or POMDP) where $\mathbf{x}_0 \sim p(\mathbf{x}_0)$.

Fig. 2 summarizes the results. The figure shows the expected cost per time step of a 30-step horizon rollout after $0, \ldots, 30$ policy searches, where the 0-th policy search corresponds to the random initial rollout. Our conclusions are: (i) Both Fig. 2(*a*) and 2(*b*) demonstrate



(*a*) Performances using moment matching



(*b*) Performances using linearization.

Figure 2: Mean performance (cost per time step) of different learned controllers with standard errors. The learned POMDP controllers (red) are close to the corresponding MDP controllers (black). Moment matching (MM, left) leads to faster learning and better solutions than linearization (Lin, right).

that the learned POMDP controllers achieved performances close to the corresponding MDP controllers. The differences were largely due to the remaining belief state uncertainties $p(\mathbf{x}_t|\mathbf{z}_{1:t})$ in the inverted position: Any suboptimal control caused remarkable increase in the immediate cost. (ii) All controllers were learned with uninformative prior knowledge and very limited data: After about 10 policy searches (30 seconds of data), the task was learned by either method. (iii) Learning with moment matching required less data and led to superior performance (the vertical axes in Figs. 2(*a*) and 2(*b*) are identical). The learning speed difference was caused by the tendency of the GP-EKF to produce incoherent filter distributions $p(\mathbf{x}_t|\mathbf{z}_{1:t})$, especially in the early stages of learning when the GP model uncertainty was large.

5. Conclusions and Future Work

We introduced a novel efficient model-based solver for POMDPs with continuous state, action, and observation spaces. Our policy search approach exploits efficient GP inference and filtering techniques in belief space. Compared to ground-truth solutions, our algorithm learned near-optimal policies in up to 10 dimensions from uninformative prior knowledge and little data. Furthermore, our method reliably solved even POMDPs with unknown optimal policies. We analyzed two approximate belief-space inference methods: While linearization is computationally advantageous, moment matching yields more coherent predictions.

A promising extension to any POMDP solver is to compute the belief state posteriors not only conditioned on measurements and controls, but also on observed rewards. Recently, we started pursuing this direction. Preliminary results are promising.

A current limitation of our approach is the requirement of a data post-processing step to identify latent states for supervised training of GPs. Approaches as proposed by Wang et al. (2008); Ko and Fox (2009) and especially McHutchon and Rasmussen (2012) can be applied to learning GP models from *noisy* training inputs. In future, we will investigate these methods for learning and long-term planning in POMDPs.

Acknowledgements

The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement # 231495.

References

D. A. Aberdeen. *Policy-Gradient Algorithms for POMDPs.* PhD thesis, ANU, 2009.

J. A. Bagnell and J. G. Schneider. Autonomous Helicopter Control using Reinforcement Learning Policy Search Methods. In *ICRA*, 2001.

D. P. Bertsekas. *Dynamic Programming and Optimal Control Vol. 1.* Athena Scientific, 2005.

C. M. Bishop. *Pattern Recognition and Machine Learning.* Springer-Verlag, 2006.

P. Dallaire, C. Besse, S. Ross, and B. Chaib-draa. Bayesian Reinforcement Learning in Continuous POMDPs with Gaussian Processes. In *IROS*, 2009.

M. P. Deisenroth and H. Ohlsson. A General Perspective on Gaussian Filtering and Smoothing. In *ACC*, 2011.

M. P. Deisenroth and C. E. Rasmussen. PILCO: A Model-Based and Data-Efficient Approach to Policy Search. In *ICML*, 2011.

M. P. Deisenroth, M. F. Huber, and U. D. Hanebeck. Analytic Moment-based Gaussian Process Filtering. In *ICML*, 2009.

L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and Acting in Partially Observable Stochastic Domains. *AI*, 101, 1998.

J. Ko and D. Fox. GP-BayesFilters: Bayesian Filtering using Gaussian Process Prediction and Observation Models. In *IROS*, 2008.

J. Ko and D. Fox. Learning GP-BayesFilters via Gaussian Process Latent Variable Models. In *RSS*, 2009.

J. Kober and J. Peters. Policy Search for Motor Primitives in Robotics. *Machine Learning*, 2011.

H. Kurniawati, D. Hsu, and W. Sun Lee. SARSOP: Efficient Point-based POMDP Planning by Approximating Optimally Reachable Belief Spaces. In *RSS*, 2008.

A. McHutchon and C. E. Rasmussen. Gaussian Process Training with Input Noise. In *NIPS*. 2012.

A. Y. Ng and M. Jordan. Pegasus: A Policy Search Method for Large MDPs and POMDPs. In *UAI*, 2000.

J. Peters and S. Schaal. Natural Actor-Critic. *Neurocomputing*, 71, 2008.

J. Pineau, G. Gordon, and S. Thrun. Point-based Value Iteration: An Anytime Algorithm for POMDPs. In *IJCAI*, 2003.

J. M. Porta, N. Vlassis, M. T. J. Spaan, and P. Poupart. Point-Based Value Iteration for Continuous POMDPs. *JMLR*, 7, 2006.

P. Poupart, N. Vlassis, J. Hoey, and K.Regan. An Analytic Solution to Discrete Bayesian Reinforcement Learning. In *ICML*, 2006.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. 2006.

S. Ross, B. Chaib-draa, and J. Pineau. Bayesian Reinforcement Learning in Continuous POMDPs with Application to Robot Navigation. In *ICRA*, 2008.

J. M. Wang, D. J. Fleet, and A. Hertzmann. Gaussian Process Dynamical Models for Human Motion. *PAMI*, 30, 2008.

R. J. Williams. Simple Statistical Gradient-following Algorithms for Connectionist Reinforcement Learning. *Machine Learning*, 8, 1992.