Multi-Task Learning of System Dynamics with Maximum Information Gain

Jose F. Zubizarreta-Rodriguez and Fabio Ramos
Australian Centre for Field Robotics, School of Information Technologies
The University of Sydney, Australia
{f.zubizarreta, f.ramos}@acfr.usyd.edu.au

***Abstract*— This paper introduces a new approach to adaptively learn the dynamics of a robotic system. The methodology is based on maximizing the information gain from new observations while modeling the dynamics with a Multiple Output Gaussian Process (MOGP). High-dimensional state-action spaces with unknown dependencies between inputs and outputs can be highly computationally expensive to learn. Gaussian process modeling is a Bayesian technique that naturally overcomes one of the most difficult problems in machine learning known as over-fitting. This makes it very appealing for on-line problems where testing multiple hypothesis is difficult. The computational cost of the learning task is reduced by having a smaller dataset of informative training points. Therefore we introduce a learning strategy capable of determining the most informative training set for the MOGP. This method can be implemented for learning the behavior of dynamic systems where due to their complexity and disturbances are infeasible to be analytically defined. The benefits of our approach are verified in two experiments: learning the dynamics of a cart-pole system in simulation and the dynamics of a robotic blimp.**

I. INTRODUCTION

On-line learning of system dynamics using non-parametric Bayesian techniques has significant advantages over traditional parametric modeling for control tasks. By exploring elements of statistical learning theory, a stochastic dynamic model that accurately predicts the next state given a previous state-action pair can be obtained where state transition uncertainty and sensor noise are naturally estimated. Traditional model-based control fully described by ordinary differential equations (ODE) requires extensive knowledge of the system and usually assumes stationary system dynamics. Ignoring environment complexities such as unknown disturbances could lead to a limited model of the robotic system. Conversely, a flexible learning methodology, able to learn complex dynamics from scratch and update itself to the current dynamic conditions can potentially be more robust than traditional approaches. However, learning from high-dimensional state-action spaces could be challenging specially when the dynamic conditions change.

In this paper we use a Multiple Output Gaussian Process (MOGP) as described in [1] in conjunction with a search action-state strategy to learn complex dynamics. Depending on the exploration of the dependencies between the outputs, the MOGP can require less training data than the conventional Gaussian Process (GP). As a consequence, on-line learning of complex robotic system dynamics is performed more efficiently. Our search strategy is based on maximizing the information gain [2] between the current model and a model after adding a new state-action pair observation. This method efficiently selects points from an extremely large data set by updating itself in an incremental manner.

We derive an algorithm in which the dynamics learning strategy uses a training set containing state-action observations as inputs, and the resultant state in the next time step as outputs. Based on the information gain criteria, state-action pairs are selected. In an incremental fashion, the algorithm explores the system dynamics using the current model to reach the desired state-action pairs. However, as the dynamics knowledge is limited and the system stochastic, nearby states might be reached. The strategy adopted incorporates all these observations while the information gain is continuously recomputed. The algorithm stops searching for new observations when a threshold indicating that sufficient knowledge of the system dynamics is achieved.

We performed experiments on a simulated cart pole, a simulated blimp and a real blimp. In the case of the simulations we employ a theoretical model-based ODE for obtaining the training data. To simulate the stochasticity of real dynamic systems we add Gaussian noise. For the case of the real system, a small sized robotic blimp was used, and the vertical state dimensions height and speed were inferred. An experiment comparing the predicted states and the resultant actual states was performed for evaluating the prediction of states of the trained MOGP.

This paper is organized as follows. In section II we present related work of learning system dynamics using GPs and efficient exploration strategies for training. In section III a theoretical description of GP and MOGP modeling is presented. In section IV we describe the search strategy for getting state-action observations by maximizing the information gain. In section V we present experiments demonstrating our approach in both simulation and real dynamic systems. Section VI concludes the paper.

II. RELATED WORK

GPs have been used in learning discrete-time dynamic processes in [3]. The authors used the difference between consecutive states, conditioned on the previous state-action pair, as outputs. Each output dimension is defined as an independent task with a single output GP. Then a trajectory

is defined by a set of consecutive state-action pairs and their respective state difference. The prediction reliability of the GP depends on sampling a reasonable number of trajectories for ensuring reliable correlation in the training data. The method is implemented in a robotic blimp which explores a set of trajectories for yaw control. Since the performance of this process depends on a large exploration task, it can be very time consuming. As a consequence, they assume a time invariant (stationary) model, and do not account for changes in the dynamics of the model or in the environment. As the environment always changes, autonomous systems need to constantly adapt to new circumstances. In [4] the authors propose to model the uncertainty of the environment by a continuous interaction with it in a reinforcement learning (RL) approach. [5] proposes a utility function for Bayesian active learning. Training data is selected according to the expected utility for maximizing the information gain.

An efficient exploration approach for adaptive learning algorithms using sparsification was proposed in [6]. To reduce the size of a training dataset, an observation is selected based on a minimum distance threshold, which works properly for uniformed spaced data points. Since a smooth stationary kernel function is used, the algorithm might be suboptimal when modeling highly non-linear functions if not enough training data is used. As the method requires a sequence of points equally spaced, it can potentially struggle to model regions with high variation in the state-space (output). [7] describes different sparse approximation methods for GPs and propose a unifying framework. The authors suggest sparse approximations to the GP prior on training and test data, and use conditional independence between training and test data given inducing variables, which leads to simplify the training data selection.

Another approach for reducing the size of training datasets for GPs uses the differential entropy score [8]. The score is used to select training points for an active dataset while jointly optimizing the model parameters. An analogue learning task using information gain for path planning of a robot in environmental surveillance is proposed in [9]. In this work a new training data point is selected when there is information gain with the addition of a new observation. An exploration path for the robot is obtained from this method, which selects the most informative locations for the surveillance of the environment.

Multi-task learning [10] can improve the generalization properties of machine learning algorithms by exploring the dependencies between related tasks. In [11], the inverse dynamics of a robot is evaluated as a multi-task learning problem. A robot manipulator is controlled while holding different loads defined as contexts. Each context represents an inverse dynamics function. For achieving higher control performance, the authors proposed to use a multi-task GP for exploiting the inter-task similarities among contexts. [1] derives a similar approach where multi-task covariance functions for GPs are constructed through kernel convolutions. We explore this construction to define valid covariance functions for the MOGP.

III. Methodology

GP regression can model transitions between consecutive states $s(k) \rightarrow s(k+1)$ given an action $a(k)$ while estimating prediction uncertainties given a training dataset $\mathcal{D}$. Next, we briefly describe single GP regression followed by the Multiple Output GP. We then generalize GP regression to multiple outputs with MOGP to account for dependencies in the output dimensions.

A. Gaussian Process Regression

A GP is a non-parametric Bayesian technique that places a multivariate Gaussian prior distribution over the space of functions $f(x)$, mapping inputs to outputs [12]. In a supervised learning set up, a GP uses a training dataset $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^N$ formed by N input locations $\mathbf{x}_i \in \mathbb{R}^D$ and their respective target values $y_i \in \mathbb{R}$. After learning hyperparameters, the GP is used to compute a new target value $f(\mathbf{x}_*)$ at an unobserved location $\mathbf{x}_*$. A GP can model complex nonlinear functions and avoids over-fitting as it naturally encodes the Occam's Razor principle, balancing data fit with model complexity. The model is represented as $\mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$, where $m(\mathbf{x})$ is the mean function and $k(\mathbf{x}, \mathbf{x}')$ is the covariance (kernel) function.

Given the uncertainty in sensor measurements, we model observations as $y = f(\mathbf{x}) + \epsilon$, where ϵ is a zero mean Gaussian noise with variance σ_n^2. Assuming a zero mean function $m(\mathbf{x}) = 0$, the joint Gaussian distribution can be expressed as:

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X,X) + \sigma_n^2 & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*) \end{bmatrix}\right) \quad (1)$$

where $\mathcal{N}(\mu, \mathbf{\Sigma})$ is a Gaussian distribution with mean μ and variance $\mathbf{\Sigma}$, and K is the covariance matrix calculated for all input locations X. The predictive distribution at an unobserved point is obtained by conditioning on the observed points:

$$(f_* \mid X_*, X, \mathbf{y}) = \mathcal{N}(\mu_*, \Sigma_*) \quad (2)$$

where

$$\mu_* = K(X_*, X)[K(X,X) + \sigma_n^2 I]^{-1}\mathbf{y} \quad (3)$$

$$\mathbf{\Sigma}_* = K(X_*, X_*) - K(X_*, X)[K(X,X) + \sigma_n^2 I]^{-1}. \quad (4)$$

Note that the predictive mean μ_* is a linear combination of N kernel functions, each evaluated on an observed point $\mathbf{x}_i$, $\mu_* = \Sigma_{i=1}^N \alpha_i k(\mathbf{x}_i, \mathbf{x}_*)$, with $\alpha = [K(X,X) + \sigma_n^2 I]^{-1}\mathbf{y}$.

The covariance function is a kernel that satisfies the condition of positive-definitiveness. One of the most used covariance functions is the square exponential

$$k(x,x) = \sigma_f^2 \exp\left(-\tfrac{1}{2}(\mathbf{x} - \mathbf{x}')^T \Lambda (\mathbf{x} - \mathbf{x}')\right) \quad (5)$$

Each covariance function has a set of hyperparameters $\theta = \{\Lambda, \sigma_f\}$. $\Lambda = diag(l_1, \ldots, l_d)$ is the weight matrix with the data-length l_n of each input dimension and σ_f is a scale factor. Each covariance function models noisy observations with variance σ_n^2. The final covariance matrix is given by $K(X,X) + \sigma_n^2 I$.

Learning the hyperparameters of a Gaussian process is performed by maximizing the log marginal data likelihood of the observed training data $\mathcal{D}$,

$$\theta_{max} = \underset{\theta}{\text{argmax}}\{\log(p(\mathbf{y}|X,\theta))\}. \tag{6}$$

The marginal likelihood is expressed as:

$$\begin{aligned}\log(p(\mathbf{y}|X)) = -\frac{1}{2}\mathbf{y}^T(K(X,X)+\sigma_n^2 I)^{-1}\mathbf{y} \\ -\frac{1}{2}\log|K(X,X)+\sigma_n^2 I| - \frac{n}{2}\log 2\pi.\end{aligned} \tag{7}$$

The computational cost of the inversion of the covariance matrix $(K(X,X)+\sigma_n^2 I)$ is $\mathcal{O}(n^3)$ [12] where n is the number of training points. Evaluating a new point test on the covariance matrix requires a matrix multiplication with a vector which has a cost $\mathcal{O}(n^2)$.

B. Modeling Two Dependent Outputs

The multiple output formulation for GPs can be obtained through process convolutions. We describe the specific case for two outputs, the general multiple output case is straightforward from these definitions. Following [1], a single output GP can be described as the sum of a process convolution with a smoothing kernel h and a noise term ϵ, Figure 1a. The output V is obtained by evaluating the input X through the parametrized smoothing kernel h. In Figure 1b a Multiple Output GP is modeling two dependent outputs y_1 and y_2. X_0 is the shared training data set between the outputs, X_1 is a training data set that only is used for y_1, and X_2 only is used for y_2. Each output is defined as the sum of two process convolutions $y_n = V_n + U_n$ and a noise term ϵ_n. h_1, h_2, k_1 and k_2 are parametrized smoothing kernels. In the case of Gaussian kernels, $k_1(x) = v_1 \exp\left(-\frac{1}{2}x^T\Lambda_1 x\right)$, $k_2(x) = v_2 \exp\left(-\frac{1}{2}x^T\Lambda_2 x\right)$ and $h_i(x) = w_i \exp\left(-\frac{1}{2}x^T\beta_i x\right)$.

The covariance function $K_{ij}^Y(x,x')$ represents the auto-covariance $(i=j)$ and models relationships between output points for the same task y_i. The cross-covariance $(i \neq j)$ models relationships between output points for tasks y_i and y_j. Solving a convolution integral [1] with Gaussian smoothing kernels leads to a close form for $K_{ij}^Y(x,x')$. The result is analogue to the single output GP where the covariance function is fully described by the hyperparameters and noise variances σ_1^2 and σ_2^2 as:

$$\begin{aligned}K_{11}^Y(x,x') &= K_{11}^U(x,x') + K_{11}^V(x,x') + \delta_{ab}\sigma_1^2 \\ K_{22}^Y(x,x') &= K_{22}^U(x,x') + K_{22}^V(x,x') + \delta_{ab}\sigma_2^2 \\ K_{12}^Y(x,x') &= K_{12}^U(x,x') \\ K_{21}^Y(x,x') &= K_{21}^U(x,x')\end{aligned}$$

where

$$K_{ii}^U(x,x') = \frac{\pi^{\frac{p}{2}}v_i^2}{\sqrt{|\Lambda_i|}}\exp\left(-\frac{1}{4}(x-x')^T\Lambda_i(x-x')\right)$$

$$K_{12}^U(x,x') = \frac{2\pi^{\frac{p}{2}}v_1v_2}{\sqrt{|\Lambda_1+\Lambda_2|}}\exp\left(-\frac{1}{2}(x-x')^T\Sigma(x-x')\right)$$

$$K_{21}^U(x,x') = \frac{2\pi^{\frac{p}{2}}v_1v_2}{\sqrt{|\Lambda_1+\Lambda_2|}}\exp\left(-\frac{1}{2}(x-x')^T\Sigma(x-x')\right)$$

$$K_{ii}^V(x,x') = \frac{\pi^{\frac{p}{2}}w_i^2}{\sqrt{|\beta_i|}}\exp\left(-\frac{1}{4}(x-x')^T\beta_i(x-x')\right)$$

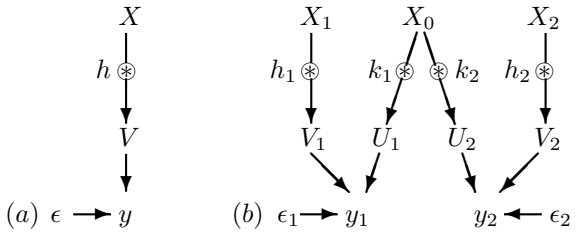


Fig. 1: *a)* Model of the Single Output GP. *b)* Model of two dependent outputs

and
$\Sigma = \Lambda_1(\Lambda_1+\Lambda_2)^{-1}\Lambda_2 = \Lambda_2(\Lambda_1+\Lambda_2)^{-1}\Lambda_1$. In the above, the hyperparameters are

$$\Theta = \{v_1, v_2, w_1, w_2, \Lambda_1, \Lambda_2, \beta_1, \beta_2, \sigma_1, \sigma_2\}.$$

The covariance matrices K_{11}, K_{12}, K_{21} and K_{22} can be written as $K_{ij}^Y(x)$ in

$$\mathbf{K}_{ij} = \begin{bmatrix} K_{ij}^Y(x_{i,1},x_{j,1}) & \cdots & K_{ij}^Y(x_{i,1},x_{j,N_j}) \\ \vdots & \ddots & \vdots \\ K_{ij}^Y(x_{i,N_i},x_{j,1}) & \cdots & K_{ij}^Y(x_{i,N_i},x_{j,N_j}) \end{bmatrix}. \tag{8}$$

Then, the covariance matrix $\mathbf{K}$ is expressed for the combined data $\mathcal{D}$ as

$$\mathbf{K} = \begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix}. \tag{9}$$

Analogue to equation 7, the log marginal likelihood is expressed as:

$$\log(p(\mathbf{y}|X)) = -\frac{1}{2}\log \mid \mathbf{K} \mid -\frac{1}{2}\mathbf{y}^T\mathbf{K}^{-1}\mathbf{y} - \frac{N_1+N_2}{2}\log 2\pi \tag{10}$$

where $\mathbf{y}^T = [y_{1,1}\cdots y_{1,N_1} \quad y_{2,1}\cdots y_{2,N_2}]$.

Similar to equation 2, we have

$$(f_* \mid x_*, X, \mathbf{y}) = \mathcal{N}(\mu_*, \Sigma_*)$$

where μ_* and Σ_* for output i, at point x_* are given by

$$\begin{aligned}&\mu_* = \mathbf{k}^{\mathbf{T}}\mathbf{K}^{-1}\mathbf{y} \\ &\Sigma_* = k - \mathbf{k}^T\mathbf{K}^{-1}\mathbf{k} \\ \text{where} \quad &k = K_{ii}^Y(0) = v_i^2 + w_i^2 + \sigma_i^2 \\ \text{and} \quad &\mathbf{k} = \begin{bmatrix} K_{i1}^Y(x_*,x_{1,1})\cdots K_{i1}^Y(x_*,x_{1,N_1}) \\ K_{i2}^Y(x_*,x_{2,1})\cdots K_{i2}^Y(x_*,x_{2,N_2}) \end{bmatrix}^T\end{aligned}$$

The computational cost of the inversion of the covariance matrix for the MOGP with m tasks is $\mathcal{O}(m^3n^3)$, since we have a covariance matrix formed by $m \times m$ sub-matrices (auto-covariance and cross-covariance) as shown in equation 9. Evaluating a new point test on the covariance matrix requires a matrix multiplication with a vector which has a cost $\mathcal{O}(m^2n^2)$.

IV. INFORMATION GAIN STRATEGY

In this section we present the information gain strategy to perform efficient exploration of the state-action space. The state-action pairs $[s_t, a_t]$ and their resultant states $s_{t+1} =$

$f(s_t, a_t)$ are denoted as $\mathbf{X}$ and $\mathbf{y}$ respectively. The Linear Quadratic Regulator (LQR) [13] is used to control the platform and reach a nearby state $s'_{t+1} \sim s_{t+1}$. We briefly review the LQR in the next subsection.

A. Information gain based on posterior entropy

The information gain strategy adopted is based on the Informative Vector Machine in [8]. A greedy algorithm is used to select the next training point that maximizes a differential entropy score:

$$\Delta_j \triangleq H[p(f_j)] - H[p^{new}(f_j)], \tag{11}$$

where $H[p(f_j)]$ is the entropy of the Gaussian process prediction at point $\mathbf{x}_j$ and $H[p^{new}(f_j)]$ is the entropy at this point after observing a new point.

Let the variance at $\mathbf{x}_j$ be v_j before the observation inclusion. The entropy of a Gaussian distribution with variance v_j is expressed as

$$H[p(f_j)] = \log(2\pi e v_j).$$

Since $p(f_j|\mathbf{y_I}, y_j) \propto p(f_j|\mathbf{y_I})\mathcal{N}(y_j|f_j, \sigma^2)$ where $\mathbf{y}_I$ is the set of training output observations thus far, the variance after including the new observation is $(v_j^{new})^{-1} = v_j^{-1} + \sigma^{-2}$. Then the entropy score is

$$\Delta_j = \log\left(1 + \frac{v_j}{\sigma^2}\right) \tag{12}$$

which is maximized by selecting

$$\mathbf{x}_{new} = \underset{\mathbf{x}_{new} \in \mathrm{R}^D}{\operatorname{argmax}} \Delta_j. \tag{13}$$

Note that this is equivalent to selecting observations with the highest variance as predicted by the GP. This method is defined for a single dimensional output y. In the multiple output case, we need to evaluate the variance $v_{f,n}$ for each output dimension y_n. Therefore we select n new observations per step of the algorithm. This incremental update procedure, where new observations are added, is repeated until the information gain is smaller than a predefined threshold. The threshold defines an acceptable quality for the model and depends on the application.

B. Linear Quadratic Regulator

The Linear Quadratic Regulator (LQR) [13] was employed to control the system to follow target trajectories whose dynamics evolve according to a linear model:

$$s_{t+1} = A_t s_t + B_t a_t + \epsilon_t \tag{14}$$

where s_t and s_{t+1} represent the state at times t and $t+1$ respectively and a_t is an action at t. A_t and B_t are the system matrices and ϵ_t is the system noise. For these models LQR provides the optimal control.

When the system dynamics is not linear, LQR can still perform well on linear approximations of the system obtained thorough first order derivatives. This approximation is based on the Jacobians of the dynamics model:

$$s_{t+1} \approx \hat{f}(s_t, a_t) \tag{15}$$

where the Jacobians of $\hat{f}$ evaluated along the target trajectory are

$$\hat{A}_t = D_s \hat{f}(s, a) \mid_{s=s_t^*, a=a_t^*} \quad \hat{B}_t = D_a \hat{f}(s, a) \mid_{s=s_t^*, a=a_t^*}$$

with D_s and D_a defined as the derivatives with respect to the state and action dimensions respectively.

As in [14], this approximation is sufficient as long as the approximate model captures the derivatives of the true system well. Therefore, it has to at least specify correctly the sign of the elements of the derivatives along with the trajectory.

Algorithm 1 presents all the steps of the learning procedure. The approximate model $\hat{f}(s_t, a_t)$ is learned by the MOGP producing estimates $\hat{f}_* \mid \bar{\mathcal{D}}$. The gradients for LQR are obtained from the learned model of the system dynamics.

Algorithm 1 Information Gain Learning Strategy

Input
$\mathbf{X}$, $\mathbf{y}$, Θ
Output
$\Theta_{trained}$, $\mathbf{X}_I$ and $\mathbf{y}_I$ (active set)
Repeat until information gain is smaller than threshold
1. Optimize Θ based on $\mathbf{X}_I$ and $\mathbf{y}_I$
2. For all points in $\mathbf{X}$ and $\mathbf{y}$, compute the information gain score
   - select $\mathbf{x}_j = \operatorname{argmax}_{\mathbf{x}_j \in \mathrm{R}^D \times \mathrm{R} \setminus X_I} \Delta_j$.
3. Use LQR to reach $\mathbf{x}_j$
   - Repeat LQR actions until close enough to $\mathbf{x}_j$
4. Add $\mathbf{x}_j$ and $\mathbf{y}_j$ into $\mathbf{X}_I$ and $\mathbf{y}_I$
5. **Return**

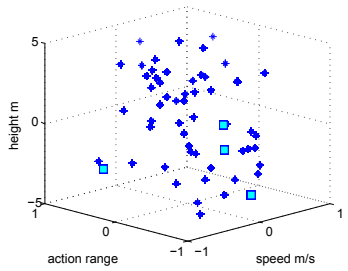
The computational cost of calculating the variance for this strategy on a single iteration for the single GP case is $\mathcal{O}(n^2)$ and for the MOGP is $\mathcal{O}(m^2n^2)$ per point.

V. EXPERIMENTS

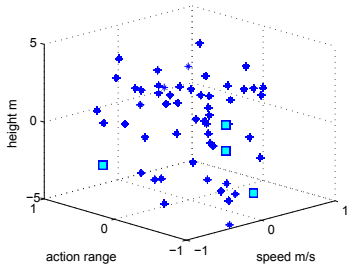
In this section we test the information gain strategy in a blimp (simulated and real) and a cart-pole system. The MOGP was trained using the proposed strategy and compared against a random point selection strategy. In all the experiments, the dynamics are defined in a discrete-time and continuous-state environment. In the synthetic data experiments the actions are continuous, whereas in the real blimp experiment the actions are discrete. For either the single GP and the MOGP models the square exponential function described in equation 5 is used.

A. Synthetic data: Blimp dynamics

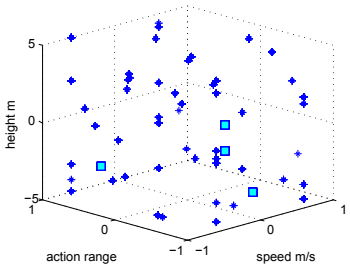
We simulate the dynamics of a robotic blimp in their vertical components $s = [x, \dot{x}]$ based on a blimp model described in [3]. The task consists of estimating the vertical position and speed of a robotic blimp using a two output MOGP. The system state at time $t+1$ is defined as $s_{t+1} = f(s_t, u_t)$. A time step $\delta_t = 0.2$ second is used for creating the learning and testing data sets. As indicated in Figure 1b,



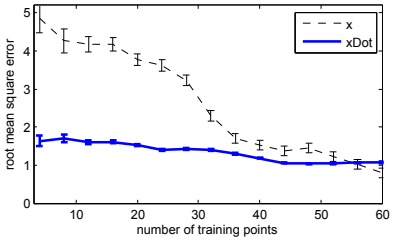
(a) Random training points for single GPs



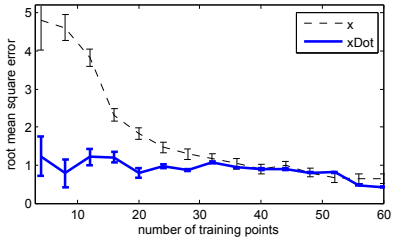
(b) Random training points for MOGP



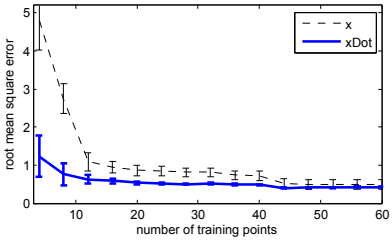
(c) Information gain strategy training points for MOGP



(d) Prediction error with single GPs model using random training points.



(e) Prediction error with MOGP model using random training points.



(f) Prediction error for MOGP model using the information gain strategy.

Fig. 2: Training performance of the learning strategy for the simulated blimp dynamics.

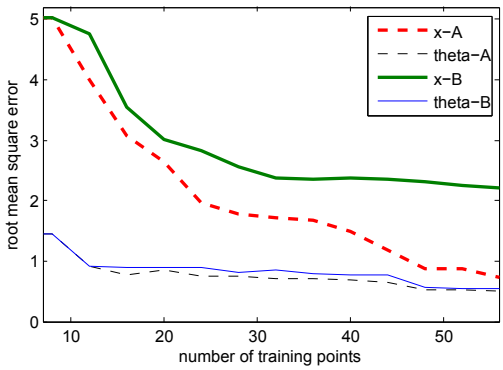
we trained the MOGP with only the shared input X_0, so the whole input dataset is shared between the height output y_h and the speed output y_s. As a performance reference, we compare the prediction error using MOGP with the information gain strategy, MOGP with randomly selected points and two single GPs (one for height, one for speed) trained with randomly selected point. Our training data $\mathbf{X}$ operating range is: $[-5m \leq h \leq 5m, -1m/s \leq s \leq 1m/s, -1 \leq a \leq 1]$ with a rate of 5 samples/s. The value of the action a is multiplied by a fixed vertical force of magnitude $F_m = 10N$ to accelerate the system.

Figure 2 presents the comparison and the selected points. The selected points (shown as asterisks) and prediction error of a single GP regression trained with 56 randomly selected points is shown in Figure 2a and Figure 2d respectively. The selected points and prediction error of an MOGP trained with 56 randomly selected training points is shown in Figure 2b and Figure 2e, and an MOGP trained with 56 points selected by our information gain learning strategy is shown in Figure 2c and Figure 2f. In each case we initialize the training data set with the same 4 points shown as squares in Figures 2a-c, so for each case we had the same starting training data. We evaluate the prediction for the single GPs and the MOGPs using for both the same test set X_* of 4 random points. The prediction error is the mean square error between the prediction y_* and the real model. x is the error height measured in meters and xDot is the speed error measured in meters/s. In the case of the MOGPs Figures 2e and 2f show that they have the same height and speed prediction error for their first 4 training points since initially we had the same training data. However, in the single GP case the starting prediction error is different since the model does not account for dependencies in output variables. The experiment was run 15 times for each model and their respective variances are shown as the error bars in Figures 2d-f.

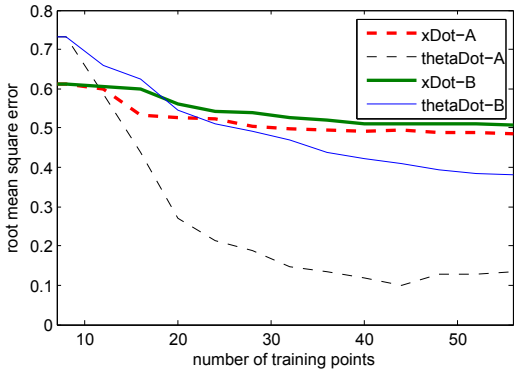
For the MOGP trained with the information gain learning strategy we started adding x_j points to the training data based on equation 13. Since the predicted variance is different for each of d output dimensions of X we obtain new d points in each iteration. We deliberately retrained the hyperparameters after the addition of 4 points, 2 points per each output dimension.

For the case of the single GPs we obtained a slower prediction error reduction compared to MOGP specially for the height as shown in Figure 2d. This is the result of modeling both outputs independently with separate GPs. To achieve the same performance as MOGP, more training points would be necessary.

The prediction performance of the MOGP trained with information gain required less training points to achieve higher accuracy than using the randomly selected points as shown in Figure 2f. The error limit is equivalent to the noise level added to the outputs in the simulated data. The faster convergence is achieved by selecting points with higher predictive variance with information gain criterion. It can be observed that the selected points are more spread through the state-action space as shown in Figure 2c compared to the randomly selected points shown in Figure 2b. The best performance is achieved by combining the information gain point selection with MOGPs. Next we describe another experiment running on a higher number of output dimensions.



(a) Horizontal position error x (cm), and pole angle (theta) in radians.



(b) Horizontal velocity xDot (cm/s) and angular velocity of the system (thetaDot) in rad/s.

Fig. 3: Prediction performance of the learning strategy for simulated cart-pole dynamics.

B. Synthetic data: Cart-pole dynamics

In this task we trained a MOGP for learning the dynamics of a simulated cart-pole. The MOGP has 4 output dimensions $\mathbf{y} = [x, \dot{x}, \theta, \dot{\theta}]$ and 5 input dimensions $\mathbf{X} = [x, \dot{x}, \theta, \dot{\theta}, a]$, where: x is the cart postion; $\dot{x}$ is the cart velocity; θ is the pole angle; $\dot{\theta}$ is the pole angular velocity. The state-action range was $[-5cm \leq x \leq 5cm, -1cm/s \leq \dot{x} \leq 1cm/s, -\pi \leq \theta \leq \pi, -1rad/s \leq \dot{\theta} \leq 1rad/s, -1N \leq a \leq 1N]$ with a rate of 10 samples/s. For the discussion of the results shown in Figures 3a and 3b we refer to the model trained with the information gain strategy as MOGP-A and with equally spaced points as MOGP-B. The plots in Figure 3 show the mean of 15 experiments run for each of the trained models.

Initially both models were trained with the same data set of 8 training points. The training plots shown in Figure 3 are additional to those first 8 training points. A test data set $[X_*, y_s]$ composed of 10 points was used to evaluate the prediction error of the MOGPs. In Figure 3a the horizontal cart position x and pole angle θ prediction errors are shown. In Figure 3b the horizontal velocity $\dot{x}$ of the cart and the angular velocity $\dot{\theta}$ prediction errors are displayed.

In Figure 3a MOGP-A performed better and achieved higher accuracy compared with MOGP-B in all the predicted state dimensions. The prediction error of x of MOGP-A decreased significantly compared with MOGP-B, meaning that the decrease in the variance contribute more to reduce the prediction error. Referring to $\dot{\theta}$ (thetaDot), shown in Figure 3b, the prediction error using MOGP-A obtained higher accuracy after 20 training points compared with MOGP-B. Then, the prediction errors for $\dot{x}$ (xDot) in Figure 3a and θ (theta) were slightly similar but MOGP-A achieved higher accuracy for both dimensions. Given that the dynamics of the cart-pole are not linear, specially referring to the pole dynamics, the MOGP trained with the points of higher uncertainty outperformed the one with equally spaced training points. This means that a uniform spaced grid would not take into account some high variance points, which might lead to a more accurate prediction of a non-linear system.

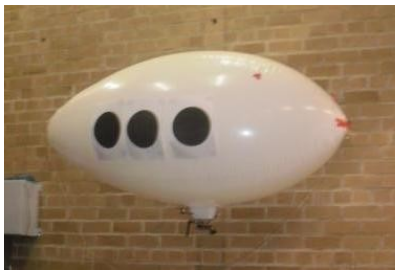
C. Real data: robotic blimp

Figure 4a shows the robotic blimp used for learning the dynamics of a real system. The blimp is 1.8 meter long and has 1 meter diameter with a gondola carrying a monocular camera and two propellers actuated by independent motors. The experiment consisted in letting the blimp to explore its state-action space using the information gain strategy (MOGP-A) and the randomly selected points (MOGP-B). We define the state-action boundaries of our blimp system as: $[-1m \leq h \leq 1m, -0.6m/s \leq s \leq 0.6m/s, -1 \leq a \leq 1]$, where: h is the height of the blimp; $\dot{h}$ is the vertical velocity; and a is the action of the propellers. This range represents physical limits in our experimental set up. The sampling rate was 4 samples/s. The vehicle could move in height in a range of 2.2 meters, maximum speed of 0.7 m/s, and motor speed ranging from -40 to 40 revolutions per second, scaled to -1 to 1 in the action range.

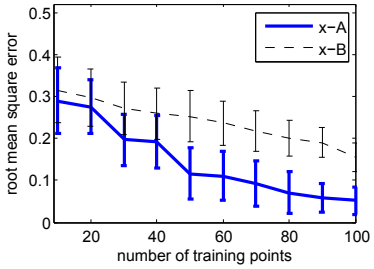
In this experiment the MOGPs were trained by iteratively adding 10 new data points over 10 iterations. As in the previous experiments, both MOGPs were trained with the same initial dataset and have the same initial prediction error. Given the rate of 4 samples per second, and that the training of the MOGPs took around 0.1 secs with a 2.4 GHz processor, we could retrain the MOGPs while the blimp was in operation every 2.5 seconds. We used an LQR to control the platform and reach the desired state-action point. The matrices A_t and B_t were computed with the gradient of an approximated model.

The experiments for each MOGP were run 10 times with the same initial training data for both cases and all iterations. The error bars in both figures represent the variance with respect the 10 iterations of the experiments. For testing data we used the same 4 randomly selected points in each iteration and computed the respective y_*. Then we compared y_* to the actual measurements to obtain the prediction errors.

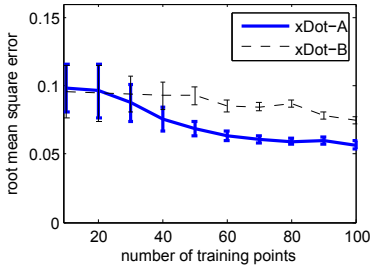
Figure 4b shows the mean prediction error for the blimp height; MOGP-A had significant higher accuracy after 30 training points compared to MOGP-B. Figure 4c shows the prediction errors for the blimp vertical velocity; MOGP-A



(a) Robotic blimp used in the experiments.



(b) Prediction mean error of height (meters).



(c) Prediction mean error of speed (meters/s).

Fig. 4: Prediction accuracy for the height and speed of the Blimp.

after 40 training points had reached a considerable improvement in accuracy compared with MOGP-B. The iterations using the information gain strategy were stopped when the the prediction error of MOGP-A for the height reach a threshold of 0.05 cm, this happened around 100 training points as shown in Figure 4b.

Even though in some occasions the LQR could not reach the desired state-action pair given the physical constrains of the actuators, it was still getting closer to that particular point through a subsequent action. Therefore, the prediction accuracy of the MOGP with the information gain strategy still performed better than the MOGP with the randomly selected points as shown in Figures 4b and 4c.

VI. CONCLUSIONS AND FUTURE WORK

An integrated method for active learning of system dynamics was presented in this paper. This method is based on multi-task dynamics learning through a multiple output Gaussian process in conjunction with an information gain learning strategy.

The Multi-Task Learning performed by the MOGP was able to achieve higher prediction accuracy with less training data than a set of independent Gaussian processes in learning the dynamics of the simulated systems and the real blimp. This prediction performance was achieved by the ability of the MOGP to correlate the output dimensions through the cross covariance functions.

We verify our information gain strategy in 3 different experiments where we select the training points that minimize the posterior variance (maximum information gain). Our learning strategy is more efficient for learning the dynamics of robotic systems and can be used to reduce the amount of training data. At each iteration of the algorithm, the prediction error is reduced significantly compared with a random selection procedure as the case of the simulated and real blimp experiments, and compared with the equally spaced points used for the cart-pole problem.

The linear quadratic regulator can be used in practical applications to control the system and obtain state-action observations as requested by the active information gain procedure. It is important to underscore that the reliability of the LQR depends on using an approximated model that captures reasonably well the derivatives of the robotic system.

As future work we plan to investigate the performance of the information gain strategy for exploring higher dimensional dynamic systems. The information gain strategy would be reformulated for selecting one single training point that represents the highest entropy decrease independent of the number of output dimensions.

ACKNOWLEDGMENTS

This work is supported by Rio Tinto Centre for Mine Automation, the ARC Centre of Excellence programme, funded by the Australian Research Council (ARC) and the New South Wales State Government, and the Secretariat of Public Education, Mexico.

REFERENCES

[1] P. Boyle and M. Fren, "Multiple output gaussian process regression," Victoria University of Wellington, Tech. Rep. CS-TR-05/2, April 2005.

[2] D. MacKay, *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.

[3] J. Ko, D. Klein, D. Fox, and D. Haehnel, "Gaussian process and reinforcement learning for identification and control of an autonomous blimp," *Proc. of the International Conference on Robotics and Automation (ICRA)*, 2007.

[4] Y. Liu, Z. Pan, D. Stirling, and F. Naghdy, "Q-learning for navigation control of autonomous blimp," *Australasian Conference on Robotics and Automation (ACRA)*, December 2009.

[5] M. P. Deisenroth, C. E. Rasmussen, and J. Peters, "Gaussian process dynamic programming," *Neurocomputing, Elseiver*, vol. 1, 2009.

[6] A. Rottmann and W. Burgard, "Adaptive autonomous control using online value iteration with gaussian processes," *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2009.

[7] J. Quionero-Candela and K. Rasmussen, "A unifying view of sparse gaussian progress regression," *Journal of Machine Learning Research*, 2005.

[8] N. Lawrence, M. Seeger, and R. Herbrich, "Fast sparse gaussian process methods: The informative vector machine," in *Advances in Neural Information Processing Systems 15 (NIPS)*. MIT Press, 2003.

[9] A. Singh, F. Ramos, H. Durrant Whyte, and W. Kaiser, "Modeling and decision making in spatio-temporal processes for environmental surveillance," *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2010.

[10] R. Caruana, "Multitask learning," Ph.D. dissertation, School of Computer Science, Carnegie Mellon University, 1997.

[11] K. Chai, C. Williams, S. Klanke, and S. Vijayakumar, "Multi-task gaussian process learning of robot inverse dynamics," *Proc. of NIPS 21*, 2008.

[12] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[13] B. D. Anderson and J. B. Moore, *Optimal Control Linear Quadratic Methods*. Prentice Hall, 1989.

[14] J. Kolter, C. Plagemann, D. Jackson, A. Ng, and S. Thrun, "A probabilistic approach to mixed open-loop and closed-loop control, with application to extreme autonomous driving," *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2010.