Gaussian Processes Model-Based Control of Underactuated Balance Robots

Kuo Chen, Jingang Yi, and Dezhen Song

Abstract— **Control of underactuated balance robot requires external subsystem trajectory tracking and internal unstable subsystem balancing with limited control authority. We present a learning-based control approach for underactuated balance robots. The tracking and balancing control is designed the controller in fast- and slow-time scales. In the slow-time scale, model predictive control is adopted to plan desired internal state profile to achieve external trajectory tracking task. The internal state is then stabilized around the planned profile in the fast-time scale. The control design is based on a learned Gaussian process (GP) regression model without need of a priori knowledge about the robot dynamics. The controller also incorporates the GP model predicted variance to enhance robustness to modeling errors. Experiments are presented using a Furuta pendulum system.**

I. Introduction

Underactuated mechanical systems are characterized as fewer number of control inputs than the number of degree-of-freedom (DOF) [1]. Underactuated balance system is introduced in [2] as a class of underactuated system with tasks of trajectory tracking for actuated subsystem and balancing *unstable* unactuated subsystem. Commonly studied underactuated balance robots include cart-pole system [3], Furuta pendulum [4]–[6], and autonomous bicycles [7] etc. Bipedal walkers also belong to underactuated balance systems because the actuated joint angles are commanded to follow desired trajectories to form certain gaits, while the unactuated floating base should be kept stable [8], [9].

Control of underactuated balance robots faces challenges of no analytical causal compensator for the non-minimum phase systems [10]. An innovative control design of underactuated balance robots is to take advantages of partition of the external and internal subsystems. For example, in [2], a concept of balance equilibrium manifold (BEM) is introduced to characterize the desired profile of internal subsystem and its dependency on tracking performance of external subsystem. A simultaneous trajectory tracking and internal balance control is then designed to stabilize the system onto BEM. Similar design is applied to bicycle/motorcycle applications in [7], [11]. A model-based control design is reported in [3] for a cart-pole system in which a singular perturbation technique is employed to regulate the system under uncertainties. Despite of the mathematical elegance, these controllers require precise dynamics models.

Recently, machine learning-based controller design has shown potential to save effort of understanding unmodeled dynamics and achieving superior performance over physical model-based control. Gaussian process (GP)-based learning is used to create flexible nonlinear nonparametric models [12] and these models have been widely applied to inverse dynamics control [13]–[16]. Gaussian process effectively predicts differentiable and closed-form mean value and covariance of output distribution, and this attractive property enables its integration with optimization-based controllers such as model predictive control (MPC) and reinforcement learning control [17]–[22]. Compared to other learning approaches, such as artificial neural network and support vector machine, GPs provide predictive covariance that can serve as a quantitative evaluation of model uncertainty.

The goal of this paper is to use the GP-based learning model to design a simultaneous tracking and balancing control for underactuated balance robots. By transforming the learned model into an external/internal convertible (EIC) form [2], we take the singular perturbation approach to design a feedback linearization control for the internal dynamics with a much faster converging rate than that of the external dynamics [3]. An MPC method is then used to design a trajectory tracking control of the external dynamics to obtain the desired internal subsystem profiles. By doing so, we reduce the computational demands for MPC design for real-time applications. Since the robot model is learned from experiment data with GPs, the proposed control approach takes advantage of the predicted Gaussian distribution and incorporates the model uncertainty into the design. The presented method and analysis are demonstrated through experiments on a Furuta pendulum. For high-DOF systems, GPs are used to identify the dynamics of each dimension of the system independently and the proposed design is readily applied to these systems.

The presented work are different with those in [23] for cart-pole system. First, the work in [23] assume prior knowledge of the physical model, while we need no a priori model information here. Second, the feedback controller incorporates the uncertain levels from the learning model and these properties are not considered in [23]. Compared with the model-based singular perturbation design in [3], we use GP-based control without need of robot's model knowledge. Unlike the learning model construction in [24]

The work was supported in part by the US NSF under award CMMI-1762556.

K. Chen and J. Yi are with the Department of Mechanical and Aerospace Engineering, Rutgers University, Piscataway, NJ 08854 USA (email: kc625@rutgers.edu; jgyi@rutgers.edu).

D. Song is with the Department of Computer Science and Engineering, Texas A&M University, College Station, TX 77843 USA (email: dzsong@tamu.edu).

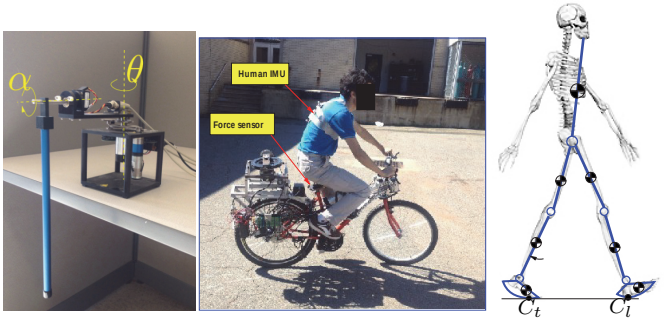
where successful balance demonstrations are required, the proposed approach takes unbalance excitation data for model learning.

The contribution of this work lies in three aspects. First, we extend the EIC form and the inversion-based BEM solver in [2] to an MPC-based trajectory planner and a trajectory stabilizer. The planning and control framework reduces the MPC computational cost. Second, the proposed learning model design is efficient and does not require successful balance demonstration. Finally, the proposed design explicitly incorporates the GPs predicted model uncertainty to enhance control robustness. This feature guarantees feasibility of achieved tracking and balancing tasks under model uncertainties.

The remainder of the paper is organized as follows. In Section II, we overview the models and control of underactuated robotic systems. Section III presents the GP-based control design. The experimental results of Furuta pendulum are demonstrated in Section IV. Finally, we present the concluding summary and discuss future work directions in Section V.

II. Underactuated Balance Robot Models and Control

Fig. 1 shows three examples of the underactuated balance robots [23]. The Furuta pendulum (Fig 1(a)) has the actuated base joint (i.e., θ) and the unactuated roll angle α. Similarly, the bikebot (Fig. 1(b)) has the (actuated) vehicle position and the (unactuated) platform roll angle [7], [25]. The 7-link bipedal walker (Fig. 1(c)) has only six actuated joints and the absolute angle of the upper body is unactuated [8], [9]. We consider a general model and control for these underactuated balance robots.



(a) (b) (c)

Fig. 1. A few example of underactuated balance robotic systems. (a) Furuta pendulum system. (b) Bikebot system. (c) Human bipedal walker.

A. Underactuated balance robot models

The underactuated balance robots can be modeled by Lagrangian equations as follows.

$$\boldsymbol{D}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{H}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{B}(\boldsymbol{q})\boldsymbol{u}, \tag{1}$$

where $\boldsymbol{q} \in \mathbb{R}^{m+n}$ is the generalized coordinate of the system, $\boldsymbol{u} \in \mathbb{R}^m$ is the control input, $\boldsymbol{D}(\boldsymbol{q})$ is the mass matrix, $\boldsymbol{H}(\boldsymbol{q}, \dot{\boldsymbol{q}})$ contains the centripetal, Coriolis and the gravitational terms and $\boldsymbol{B}(\boldsymbol{q})$ is the input mapping matrix. Furthermore, $\boldsymbol{q} = [\boldsymbol{\theta}_1^T\ \boldsymbol{\alpha}_1^T]^T$ is decomposed into actuated generalized position $\boldsymbol{\theta}_1 \in \mathbb{R}^m$ and unactuated generalized position $\boldsymbol{\alpha}_1 \in \mathbb{R}^n$. Without loss of generality, it is assumed that $m \geq n$. We define generalized velocities $\boldsymbol{\theta}_2 = \dot{\boldsymbol{\theta}}_1$ and $\boldsymbol{\alpha}_2 = \dot{\boldsymbol{\alpha}}_1$ so that $\dot{\boldsymbol{q}} = [\boldsymbol{\theta}_2^T\ \boldsymbol{\alpha}_2^T]^T$. Equation (1) is then partitioned into actuated and unactuated subsystems as

$$\boldsymbol{D}(\boldsymbol{q}) \begin{bmatrix} \dot{\boldsymbol{\theta}}_2 \\ \dot{\boldsymbol{\alpha}}_2 \end{bmatrix} + \begin{bmatrix} \boldsymbol{H}_1(\boldsymbol{q}, \dot{\boldsymbol{q}}) \\ \boldsymbol{H}_2(\boldsymbol{q}, \dot{\boldsymbol{q}}) \end{bmatrix} = \begin{bmatrix} \boldsymbol{B}_1(\boldsymbol{q}) \\ \boldsymbol{0}_{n\times m} \end{bmatrix} \boldsymbol{u}, \tag{2}$$

where $\boldsymbol{B}_1(\boldsymbol{q}) \in \mathbb{R}^{m\times m}$ has full rank. By inverting the mass matrix $\boldsymbol{D}(\boldsymbol{q})$ in (2), we obtain

$$\begin{bmatrix} \dot{\boldsymbol{\theta}}_2 \\ \dot{\boldsymbol{\alpha}}_2 \end{bmatrix} = \boldsymbol{D}^{-1}(\boldsymbol{q}) \begin{bmatrix} \boldsymbol{B}_1(\boldsymbol{q})\boldsymbol{u} - \boldsymbol{H}_1(\boldsymbol{q}, \dot{\boldsymbol{q}}) \\ -\boldsymbol{H}_2(\boldsymbol{q}, \dot{\boldsymbol{q}}) \end{bmatrix}. \tag{3}$$

A general state-space representation of (3) is formulated as

$$\begin{cases} \Sigma_e : \dot{\boldsymbol{\theta}}_1 = \boldsymbol{\theta}_2,\ \dot{\boldsymbol{\theta}}_2 = \boldsymbol{f}_\theta(\boldsymbol{\theta}, \boldsymbol{\alpha}, \boldsymbol{u}), \\ \Sigma_i : \dot{\boldsymbol{\alpha}}_1 = \boldsymbol{\alpha}_2,\ \dot{\boldsymbol{\alpha}}_2 = \boldsymbol{f}_\alpha(\boldsymbol{\theta}, \boldsymbol{\alpha}, \boldsymbol{u}), \end{cases} \tag{4}$$

where $\boldsymbol{\theta} = [\boldsymbol{\theta}_1^T\ \boldsymbol{\theta}_2^T]^T$, $\boldsymbol{\alpha} = [\boldsymbol{\alpha}_1^T\ \boldsymbol{\alpha}_2^T]^T$, and $\boldsymbol{f}_\theta(\cdot)$ and $\boldsymbol{f}_\alpha(\cdot)$ are nonlinear vector functions. The goal of the control system is to allow the external subsystem Σ_e to track desired trajectory $\boldsymbol{\theta}_d = [\boldsymbol{\theta}_{d1}^T\ \boldsymbol{\theta}_{d2}^T]^T$, $\boldsymbol{\theta}_{d2} = \dot{\boldsymbol{\theta}}_{d1}$, while the internal unstable subsystem Σ_i to be kept balanced.

In (4), the external subsystem Σ_e and internal subsystem Σ_i are coupled and considered dual relationship [2]. For example, letting

$$\boldsymbol{v} = \boldsymbol{f}_\alpha(\boldsymbol{\theta}, \boldsymbol{\alpha}, \boldsymbol{u}), \tag{5}$$

dynamics Σ_i is feedback linearized as $\dot{\boldsymbol{\alpha}}_2 = \boldsymbol{v}$. Because $\boldsymbol{v} \in \mathbb{R}^n$ and $\boldsymbol{u} \in \mathbb{R}^m$, only a subspace of $\boldsymbol{u}$ is obtained by inverting (5). Letting $\boldsymbol{u} = [\boldsymbol{u}_d^T\ \boldsymbol{u}_f^T]^T$, $\boldsymbol{u}_d \in \mathbb{R}^n$ and $\boldsymbol{u}_f \in \mathbb{R}^{m-n}$, $\boldsymbol{u}_d$ is obtained by the inverse dynamics model

$$\boldsymbol{u}_d = \boldsymbol{f}_\alpha^{-1}(\boldsymbol{\theta}, \boldsymbol{\alpha}, \boldsymbol{v}, \boldsymbol{u}_f), \tag{6}$$

while $\boldsymbol{u}_f$ is freely designed. System (4) under (6) becomes

$$\begin{cases} \Sigma_e : \dot{\boldsymbol{\theta}}_1 = \boldsymbol{\theta}_2,\ \dot{\boldsymbol{\theta}}_2 = \boldsymbol{f}_\theta\left(\boldsymbol{\theta}, \boldsymbol{\alpha}, \boldsymbol{u}(\boldsymbol{v}, \boldsymbol{u}_f)\right), \\ \Sigma_i : \dot{\boldsymbol{\alpha}}_1 = \boldsymbol{\alpha}_2,\ \dot{\boldsymbol{\alpha}}_2 = \boldsymbol{v}. \end{cases} \tag{7}$$

In (7), Σ_i is directly controlled by $\boldsymbol{v}$ and not affected by Σ_e, while Σ_e is affected by both inputs $\boldsymbol{u}_f$ and $\boldsymbol{v}$.

Temporarily ignoring the tracking task of $\boldsymbol{\theta}$ for Σ_e, we design a proportional-differential (PD) controller to force $\boldsymbol{\alpha}$ to converge to desired trajectory $\boldsymbol{\alpha}_d = [\boldsymbol{\alpha}_{d1}^T\ \boldsymbol{\alpha}_{d2}^T]^T$, $\boldsymbol{\alpha}_{d2} = \dot{\boldsymbol{\alpha}}_{d1}$,

$$\boldsymbol{v}_{pd} = \dot{\boldsymbol{\alpha}}_{d2} - \frac{k_d}{\epsilon}\boldsymbol{e}_{\alpha 2} - \frac{k_p}{\epsilon^2}\boldsymbol{e}_{\alpha 1}, \tag{8}$$

where errors $\boldsymbol{e}_{\alpha 1} = \boldsymbol{\alpha}_1 - \boldsymbol{\alpha}_{d1}$, $\boldsymbol{e}_{\alpha 2} = \boldsymbol{\alpha}_2 - \boldsymbol{\alpha}_{d2}$, $\boldsymbol{e}_\alpha = [\boldsymbol{e}_{\alpha 1}^T\ \boldsymbol{e}_{\alpha 2}^T]^T$, $\epsilon > 0$ is a small constant, and $k_p, k_d > 0$ are constant gains. To enforce the tracking task for Σ_e, we design the desired trajectory $\boldsymbol{\alpha}_d(\boldsymbol{\theta}_d, \boldsymbol{\theta})$ to be dependent on $(\boldsymbol{\theta}_d, \boldsymbol{\theta})$ such that $\boldsymbol{\theta} \to \boldsymbol{\theta}_d$. Balance equilibrium manifold (BEM) is introduced in [2] to capture such dependency. BEM is defined as $\mathcal{E} = \{\boldsymbol{\alpha}_d = \boldsymbol{\alpha}_d^0 : \boldsymbol{\alpha}_{d1}^0 = \boldsymbol{\alpha}_{d1}(\boldsymbol{\theta}_d, \boldsymbol{\theta}), \boldsymbol{\alpha}_{d2}^0 = \boldsymbol{0}\}$ from inverting implicit function

$$\boldsymbol{f}_\theta(\boldsymbol{\theta}, \boldsymbol{\alpha}_d^0) = \dot{\boldsymbol{\theta}}_{d2} - k_d\boldsymbol{e}_{\theta 2} - k_p\boldsymbol{e}_{\theta 1}, \tag{9}$$

where errors $\boldsymbol{e}_{\theta 1} = \boldsymbol{\theta}_1 - \boldsymbol{\theta}_{d1}$, $\boldsymbol{e}_{\theta 2} = \boldsymbol{\theta}_2 - \boldsymbol{\theta}_{d2}$, and $\boldsymbol{e}_\theta = [\boldsymbol{e}_{\theta 1}^T \, \boldsymbol{e}_{\theta 2}^T]^T$.

Inverting (9) suffers accuracy issue for a learned model of $\boldsymbol{f}_\theta$ [23]. We instead take an MPC approach to solve $\boldsymbol{\alpha}_d^0$ and obtain $\mathcal{E}$ under tracking design of $\boldsymbol{\theta}_d$. We do not directly apply MPC to (7) to solve $\boldsymbol{v}$ because in that case the controlled Σ_i might not be stable. We take a singular perturbation approach.

B. Decoupled dynamics with singular perturbation

Applying controller (8) to (7), the error dynamics then become

$$\begin{cases} \dot{\boldsymbol{\theta}}_1 = \boldsymbol{\theta}_2, \ \dot{\boldsymbol{\theta}}_2 = \boldsymbol{f}_\theta(\boldsymbol{\theta}, \boldsymbol{\alpha}_d + \boldsymbol{e}_\alpha, \boldsymbol{u}(\boldsymbol{v}_{pd}, \boldsymbol{u}_f)) \\ \dot{\boldsymbol{e}}_{\alpha 1} = \boldsymbol{e}_{\alpha 2}, \ \dot{\boldsymbol{e}}_{\alpha 2} = -\frac{k_p}{\epsilon^2}\boldsymbol{e}_{\alpha 1} - \frac{k_d}{\epsilon}\boldsymbol{e}_{\alpha 2}. \end{cases} \tag{10}$$

As ϵ goes to 0, both $\boldsymbol{e}_{\alpha 1}$ and $\boldsymbol{e}_{\alpha 2}$ converge to zero exponentially with a rate of $-\frac{1}{\epsilon}$. The $\boldsymbol{\theta}$ dynamics is slow, while $\boldsymbol{e}_\alpha$ dynamics is referred as fast one. It can be shown that $\boldsymbol{\theta}(t,\epsilon) - \hat{\boldsymbol{\theta}}(t) = O(\epsilon)$ or equivalently $||\boldsymbol{\theta}(t,\epsilon) - \hat{\boldsymbol{\theta}}(t)|| \leq k|\epsilon|$ [26], where $\hat{\boldsymbol{\theta}}(t) = [\hat{\boldsymbol{\theta}}_1(t)^T \, \hat{\boldsymbol{\theta}}_2(t)^T]^T$ is the solution of $\dot{\hat{\boldsymbol{\theta}}}_1 = \hat{\boldsymbol{\theta}}_2$, $\dot{\hat{\boldsymbol{\theta}}}_2 = \boldsymbol{f}_\theta(\hat{\boldsymbol{\theta}}, \boldsymbol{\alpha}_d, \boldsymbol{u}(\dot{\boldsymbol{\alpha}}_{d2}, \boldsymbol{u}_f))$.

Since estimating $\hat{\boldsymbol{\theta}}$ takes much less computational effort than obtaining $\boldsymbol{\theta}$ by (10), we formulate the MPC problem for $\hat{\boldsymbol{\theta}}$ to follow $\boldsymbol{\theta}_d$, for which (7) is considered as

$$\dot{\hat{\boldsymbol{\theta}}}_1 = \hat{\boldsymbol{\theta}}_2, \dot{\hat{\boldsymbol{\theta}}}_2 = \boldsymbol{f}_\theta(\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\alpha}}, \boldsymbol{u}(\hat{\boldsymbol{w}}, \boldsymbol{u}_f)), \dot{\hat{\boldsymbol{\alpha}}}_1 = \hat{\boldsymbol{\alpha}}_2, \dot{\hat{\boldsymbol{\alpha}}}_2 = \hat{\boldsymbol{w}} \tag{11}$$

with $\hat{\boldsymbol{\alpha}}_1 = \boldsymbol{\alpha}_{d1}$, $\hat{\boldsymbol{\alpha}}_2 = \boldsymbol{\alpha}_{d2}$, and $\hat{\boldsymbol{w}} = \dot{\boldsymbol{\alpha}}_{d2}$. The design variable of the MPC problem is the input trajectory $\hat{\boldsymbol{w}}$, $\boldsymbol{u}_f$ and the initial values $\hat{\boldsymbol{\alpha}}_1(0)$ and $\hat{\boldsymbol{\alpha}}_2(0)$. Although the form of (11) is the same as (7), $\hat{\boldsymbol{\alpha}}(0)$ in (11) is a design variable that needs to be optimized, while $\boldsymbol{\alpha}(0)$ in (7) is measured.

III. GP-based Planning and Control

A. GP-based model estimation

Controller (6) and dynamics (7) require information about $\boldsymbol{f}_\theta$ and $\boldsymbol{f}_\alpha^{-1}$. In order to use a zero-mean Gaussian distribution in model estimation, we re-write model (7) as

$$\begin{cases} \dot{\boldsymbol{\theta}}_1 = \boldsymbol{\theta}_2, \ \dot{\boldsymbol{\theta}}_2 = \boldsymbol{f}_\theta(\boldsymbol{\theta}, \boldsymbol{\alpha}, \boldsymbol{u}_d, \boldsymbol{u}_f) \\ \dot{\boldsymbol{\alpha}}_1 = \boldsymbol{\alpha}_2, \ \dot{\boldsymbol{\alpha}}_2 + \boldsymbol{\kappa}_\alpha(\boldsymbol{\theta}, \boldsymbol{\alpha}, \dot{\boldsymbol{\alpha}}_2, \boldsymbol{u}_f) = \boldsymbol{u}_d, \end{cases} \tag{12}$$

where $\boldsymbol{f}_\theta$ and $\boldsymbol{\kappa}_\alpha$ are unknown functions that need to be estimated. One benefit of writing the model into (12) is that the inverse dynamics controller becomes $\boldsymbol{u}_d = \boldsymbol{v} + \boldsymbol{\kappa}_\alpha(\boldsymbol{\theta}, \boldsymbol{\alpha}, \boldsymbol{v}, \boldsymbol{u}_f)$ with $\boldsymbol{v}$ specified in (8). By doing so, the estimate of $\boldsymbol{\kappa}_\alpha$ is obtained by zero-mean GP such that if the application data are far away from the training data inputs, the inverse dynamics control still stabilizes $\boldsymbol{\alpha}$ by (8). By (12), the learning model is formulated as

$$\begin{cases} \dot{\boldsymbol{\theta}}_1 = \boldsymbol{\theta}_2, \ \dot{\boldsymbol{\theta}}_2 \sim \boldsymbol{gp}_\theta(\boldsymbol{\theta}, \boldsymbol{\alpha}, \dot{\boldsymbol{\alpha}}_2, \boldsymbol{u}_f), \\ \dot{\boldsymbol{\alpha}}_1 = \boldsymbol{\alpha}_2, \ \boldsymbol{u}_d - \dot{\boldsymbol{\alpha}}_2 \sim \boldsymbol{gp}_\alpha(\boldsymbol{\theta}, \boldsymbol{\alpha}, \dot{\boldsymbol{\alpha}}_2, \boldsymbol{u}_f), \end{cases} \tag{13}$$

where $\boldsymbol{gp}_\theta$ and $\boldsymbol{gp}_\alpha$ are the GP distributions to estimate $\boldsymbol{f}_\theta$ and $\boldsymbol{\kappa}_\alpha$, respectively. To train these GP distributions, the inputs are tuple $\{\boldsymbol{\theta}, \boldsymbol{\alpha}, \dot{\boldsymbol{\alpha}}_2, \boldsymbol{u}_f\}$ and the outputs are $\dot{\boldsymbol{\theta}}_2$ and $\boldsymbol{u}_d - \dot{\boldsymbol{\alpha}}_2$. For each output, an individual GP model is built and the GP models for different outputs are assumed independent.

With (13), the control input $\boldsymbol{u}_d$ is obtained as

$$\boldsymbol{u}_d \sim \boldsymbol{v} + \boldsymbol{gp}_\alpha(\boldsymbol{\theta}, \boldsymbol{\alpha}, \boldsymbol{v}, \boldsymbol{u}_f), \tag{14}$$

where $\boldsymbol{gp}_\alpha(\boldsymbol{\theta}, \boldsymbol{\alpha}, \boldsymbol{v}, \boldsymbol{u}_f) \sim \mathcal{N}(\boldsymbol{\mu}_\alpha, \boldsymbol{\Sigma}_\alpha)$ is predictive Gaussian distribution, $\boldsymbol{\mu}_\alpha$ and $\boldsymbol{\Sigma}_\alpha$ [1] are input dependent and computed from (26) in Appendix. Input $\boldsymbol{v}$ is the inverse dynamics control for $\dot{\boldsymbol{\alpha}}_2$ as

$$\boldsymbol{v} = \hat{\boldsymbol{w}} - k_d(\boldsymbol{\alpha}_2 - \hat{\boldsymbol{\alpha}}_2(0)) - k_p(\boldsymbol{\alpha}_1 - \hat{\boldsymbol{\alpha}}_1(0)), \tag{15}$$

$\{\hat{\boldsymbol{w}}, \hat{\boldsymbol{\alpha}}_1(0), \hat{\boldsymbol{\alpha}}_2(0)\}$ are solution from the MPC design that will be discussed later in this section. By (14), $\boldsymbol{u}_d \sim \mathcal{N}(\boldsymbol{\mu}_d, \boldsymbol{\Sigma}_d)$ is Gaussian distribution with $\boldsymbol{\mu}_d = \boldsymbol{v} + \boldsymbol{\mu}_\alpha$ and $\boldsymbol{\Sigma}_d = \boldsymbol{\Sigma}_\alpha$. The mean value $\boldsymbol{\mu}_d$ is used as the control input.

Under (14), we briefly show that the subsystem Σ_i is stabilized to $\hat{\boldsymbol{\alpha}}$. Plugging (14) into (12), the closed-loop dynamics become

$$\dot{\boldsymbol{\alpha}}_1 = \boldsymbol{\alpha}_2, \ \dot{\boldsymbol{\alpha}}_2 = \boldsymbol{v} + \boldsymbol{\mu}_\alpha - \boldsymbol{\kappa}_\alpha(\boldsymbol{\theta}, \boldsymbol{\alpha}, \dot{\boldsymbol{\alpha}}_2, \boldsymbol{u}_f).$$

The error dynamics are then obtained as

$$\dot{\boldsymbol{e}}_\alpha = \boldsymbol{A}\boldsymbol{e}_\alpha + \boldsymbol{B}[\boldsymbol{\mu}_\alpha - \boldsymbol{\kappa}_\alpha(\boldsymbol{\theta}, \boldsymbol{\alpha}, \dot{\boldsymbol{\alpha}}_2, \boldsymbol{u}_f)], \tag{16}$$

where

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{I}_n \\ -\boldsymbol{K}_p & -\boldsymbol{K}_d \end{bmatrix}, \ \boldsymbol{B} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{I}_n \end{bmatrix}, \tag{17}$$

matrices $\boldsymbol{K}_p$ and $\boldsymbol{K}_d$ are properly chosen such that $\boldsymbol{A}$ is Hurwitz. It is assumed that the predicted GP mean value is close to the unknown underlying function, that is, $\boldsymbol{\mu}_\alpha - \boldsymbol{\kappa}_\alpha(\boldsymbol{\theta}, \boldsymbol{\alpha}, \dot{\boldsymbol{\alpha}}_2, \boldsymbol{u}_f)$ is close to zero, and therefore, the stabilization is obtained. More details can be found in [27].

B. MPC-based trajectory planning

With controller (14), we still need to determine the desired trajectories $\hat{\boldsymbol{\alpha}}$ and $\hat{\boldsymbol{w}}$ for Σ_i. We use MPC to find these trajectories. The learned model (13) with controller (14) is in the same form as (10) except that the deterministic function $\boldsymbol{f}_\theta$ is replaced by $\boldsymbol{gp}_\theta$, namely,

$$\dot{\boldsymbol{\theta}}_1 = \boldsymbol{\theta}_2, \ \dot{\boldsymbol{\theta}}_2 \sim \boldsymbol{gp}_\theta(\boldsymbol{\theta}, \hat{\boldsymbol{\alpha}} + \boldsymbol{e}_\alpha, \hat{\boldsymbol{w}} + \dot{\boldsymbol{e}}_2, \boldsymbol{u}_f). \tag{18}$$

In the above equations, we use $\boldsymbol{\alpha} = \hat{\boldsymbol{\alpha}} + \boldsymbol{e}_\alpha$ to replace $\boldsymbol{\alpha}$ and $\dot{\boldsymbol{\alpha}}_2 = \hat{\boldsymbol{w}} + \dot{\boldsymbol{e}}_2$ to replace $\dot{\boldsymbol{\alpha}}_{\alpha 2}$ in $\boldsymbol{gp}_\theta$. By singular perturbation theory, assuming $\boldsymbol{e}_\alpha$ converges to zero rapidly, similar to (11), we obtain the reduced system dynamics as

$$\dot{\hat{\boldsymbol{\theta}}}_1 = \hat{\boldsymbol{\theta}}_2, \dot{\hat{\boldsymbol{\theta}}}_2 \sim \boldsymbol{gp}_\theta(\hat{\boldsymbol{\theta}}, \hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{w}}, \boldsymbol{u}_f), \dot{\hat{\boldsymbol{\alpha}}}_1 = \hat{\boldsymbol{\alpha}}_2, \dot{\hat{\boldsymbol{\alpha}}}_2 = \hat{\boldsymbol{w}}.$$

For presentation convenience, we use discrete-time format of the above dynamics for MPC design as follows [2].

$$\begin{cases} \Delta\hat{\boldsymbol{\theta}}_1(k) = \hat{\boldsymbol{\theta}}_2(k)\Delta t, \ \Delta\hat{\boldsymbol{\theta}}_2(k) \sim \hat{\boldsymbol{\theta}}_2(k) + \boldsymbol{gp}_\theta\Delta t, \\ \Delta\hat{\boldsymbol{\alpha}}_1(k) = \hat{\boldsymbol{\alpha}}_2(k)\Delta t, \ \Delta\hat{\boldsymbol{\alpha}}_2(k) = \hat{\boldsymbol{w}}(k)\Delta t, \end{cases} \tag{19}$$

[1] We here drop their dependency on $(\boldsymbol{\theta}, \boldsymbol{\alpha}, \boldsymbol{v}, \boldsymbol{u}_f)$ for presentation convenience.

[2] For notation clarity, we drop all arguments for GP model $\boldsymbol{gp}_\theta$.

where $\Delta\hat{\boldsymbol{\theta}}_i(k) = \hat{\boldsymbol{\theta}}_i(k+1) - \hat{\boldsymbol{\theta}}_i(k)$, $\Delta\hat{\boldsymbol{\alpha}}_i(k) = \hat{\boldsymbol{\alpha}}_i(k+1) - \hat{\boldsymbol{\alpha}}_i(k)$, $i = 1, 2$, $k \in \mathbb{N}$, and Δt is the sampling period.

At the kth step, we use $\hat{\boldsymbol{\theta}}(k+i|k)$, $i = 0, \ldots, H+1$, to denote the predicted $\boldsymbol{\theta}$ at the $(k+i)$th step, $H \in \mathbb{N}$ is the prediction horizon, and $\hat{\boldsymbol{\theta}}(k|k) = \hat{\boldsymbol{\theta}}(k)$. Using (19), we have

$$\hat{\boldsymbol{\theta}}(k+i+1|k) \sim \boldsymbol{F}\hat{\boldsymbol{\theta}}(k+i|k) + \boldsymbol{G}\boldsymbol{gp}_\theta, \tag{20}$$

where matrices

$$\boldsymbol{F} = \begin{bmatrix} \boldsymbol{I}_m & \Delta t\boldsymbol{I}_m \\ \boldsymbol{0}_m & \boldsymbol{I}_m \end{bmatrix}, \boldsymbol{G} = \begin{bmatrix} \boldsymbol{0}_m \\ \Delta t\boldsymbol{I}_m \end{bmatrix}.$$

$\hat{\boldsymbol{\theta}}(k+i+1|k)$ generally does not satisfy Gaussian distribution even if $\hat{\boldsymbol{\theta}}(k+i|k)$ is Gaussian. To make this prediction manageable, we adopt the linearization of the posterior GP mean function [19] and an approximation of $\hat{\boldsymbol{\theta}}(k+i+1|k)$ is a Gaussian distribution with the following mean and covariance.

$$\begin{cases} \boldsymbol{\mu}_{\hat{\theta}}(k+i+1|k) = \boldsymbol{F}\boldsymbol{\mu}_{\hat{\theta}}(k+i|k) + \boldsymbol{G}\boldsymbol{\mu}_{gp_\theta}, \\ \boldsymbol{\Sigma}_{\hat{\theta}}(k+i+1|k) = \boldsymbol{F}\boldsymbol{\Sigma}_{\hat{\theta}}(k+i|k)\boldsymbol{F}^T + \boldsymbol{G}\partial\boldsymbol{\Sigma}_{\hat{\theta}}\boldsymbol{G}^T, \end{cases} \tag{21}$$

where $\boldsymbol{\mu}_{gp_\theta}$ and $\boldsymbol{\Sigma}_{gp_\theta}$ are the mean value and covariance functions of $\boldsymbol{gp}_\theta$, respectively, $\partial\boldsymbol{\Sigma}_{\hat{\theta}} = \frac{\partial\boldsymbol{\mu}_{gp_\theta}}{\partial\boldsymbol{\theta}}\boldsymbol{\Sigma}_{\hat{\theta}}(k+i|k)\frac{\partial\boldsymbol{\mu}_{gp_\theta}^T}{\partial\boldsymbol{\theta}} + \boldsymbol{\Sigma}_{gp_\theta}$. Note that $\Sigma_{gp_\theta} = \Sigma_{gp_\theta}(\boldsymbol{\mu}_{\hat{\theta}}(k+i|k), \hat{\boldsymbol{\alpha}}(k+i|k), \hat{\boldsymbol{w}}(k+i), \boldsymbol{u}_f(k+i))$ is input dependent.

For the entire system specified by (16) and (18), the objective function is designed as

$$\bar{J}^k_{\hat{\theta},\hat{W}} = \sum_{i=0}^{H}\Big[\mathbb{E}\|\hat{\boldsymbol{e}}_\theta(k+i)\|^2_{Q_1} + \|\hat{\boldsymbol{w}}(k+i)\|^2_R + \|\hat{\boldsymbol{\alpha}}(k)\|^2_{Q_2} + \|\boldsymbol{u}_f(k+i)\|^2_R\Big] + \mathbb{E}\|\hat{\boldsymbol{e}}_\theta(k+H+1)\|^2_{Q_e}, \tag{22}$$

where $\hat{e}(k+i) = \hat{\boldsymbol{\theta}}(k+i|k) - \boldsymbol{\theta}_d(k+i)$, matrices $\boldsymbol{Q}_i$, $i = 1, 2, 3$, $\boldsymbol{R}$ are positive definite and norm of a vector $\boldsymbol{e}$ is defined as $\|\boldsymbol{e}\|^2_{Q_i} = \boldsymbol{e}^T\boldsymbol{Q}_i\boldsymbol{e}$. The MPC input variable is $\hat{\boldsymbol{W}} = \{\hat{\boldsymbol{\alpha}}(k), \hat{\boldsymbol{w}}(k+i), \boldsymbol{u}_f(k+i), i = 0, \ldots, H\}$. We take expectation operator in (22) because we approximate $\boldsymbol{\theta}(k+i)$ by the probabilistic variable $\hat{\boldsymbol{\theta}}(k+i|k)$ from (21).

The fact that the reduced dynamics (21) is used to predict the future trajectory gives computational benefit. To include the tracking performance for Σ_i, we modify the objective function to be

$$J^k_{\hat{\theta},\hat{W},\alpha} = \bar{J}^k_{\hat{\theta},\hat{W}} + \rho\|\boldsymbol{\Sigma}_d(k)\|, \tag{23}$$

where $\boldsymbol{\Sigma}_d(k)$ is the covariance of the predictive distribution (14) at the kth step and $\rho > 0$ is a weighting factor. The rationale to include $\boldsymbol{\Sigma}_d$ in the cost function is to incorporate the inverse dynamics model uncertainty in the MPC planning phase. We will demonstrate the results in Section IV.

The optimal control input by the MPC design is

$$\hat{\boldsymbol{W}}^0(k) = \text{argmin}_{\hat{\boldsymbol{W}}(k)}\, J^k_{\hat{\theta},\hat{W},\alpha}. \tag{24}$$

The optimization is formulated as an unconstrained MPC and solved with a gradient decent method. The optimal control input $\hat{\boldsymbol{W}}^0(k)$ is used in the inverse dynamics controller (15). Fig. 2 summarizes the structure of the GP-based planning and control design. In each sampling period, the trajectory planner solves the MPC problem (24) with model (21). It also incorporates the predictive variance $\boldsymbol{\Sigma}_d$ of the inverse dynamics model. The MPC outcome is the planned internal trajectory $\hat{\boldsymbol{W}}^0$. The inverse dynamics controller in (14) and (15) then uses profile $\hat{\boldsymbol{W}}^0$ for input $\boldsymbol{\mu}_d$.

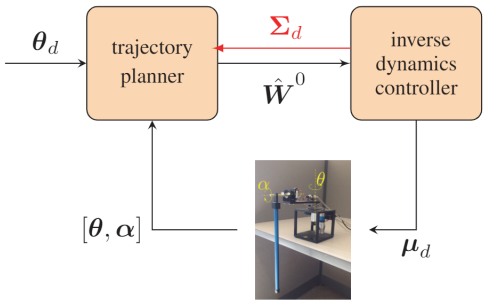


Fig. 2. The structure of the GP-based planning and control design.

IV. Experiments

Fig. 1(a) shows the Furuta pendulum that is used for experiments. The hardware system is provided by Quanser Consulting Inc. [3] The data collection and control systems are implemented through Matlab Simulink. The real-time controller is run on a laptop computer with Intel Core i7-2620M dual-core processor 2.7GHz. The rotary base angle θ is actuated and the pendulum angle α is unactuated. Angle α is defined as zero when the pendulum is vertically upright. The motor voltage V_m is the control input and the control goal is to balance the pendulum while angle θ must follow a given trajectory θ_d.

The Furuta pendulum dynamics are captured by model (4) whose detailed expression is refer to [4]–[6]. It is straightforward to obtain that $\theta_1 = \theta$, $\theta_2 = \dot{\theta}$, $m = 1$ for external dynamics Σ_e, $\alpha_1 = \alpha$, $\alpha_2 = \dot{\alpha}$, $n = 1$ for internal dynamics Σ_i and $u_d = V_m$. To obtain the learned model, we perturb the system and collect the motion data under open-loop control input

$$V_m = \begin{cases} a_1\sin(\omega_1 t) + a_2\sin(\omega_2 t)\ , |\alpha| \le \frac{\pi}{3}, \\ 0\ , |\alpha| > \frac{\pi}{3}, \end{cases} \tag{25}$$

where a_1 and a_2 are chosen to satisfy the voltage limit $|V_m| \le 5$ V, and ω_1 and ω_2 are chosen to excite the system at both low and high frequencies. In experiment, we choose $a_1 = 3$, $a_2 = 1.5$, $\omega_1 = 8$ rad/sec and $\omega_2 = 40$ rad/sec. Under this input, we manually give the pendulum an initial velocity when $|\alpha| \ge \frac{\pi}{2}$. The open-loop controller (25) cannot stabilize the pendulum to stay around upright position. We swing up the pendulum repeatedly to obtain enough data for training the model. Control input V_m and motion data are recorded when $|\alpha| \le \frac{\pi}{3}$. The joint angles (θ, α) are measured with encoders, velocities $(\dot{\theta}, \dot{\alpha})$ and accelerations $(\ddot{\theta}, \ddot{\alpha})$ are obtained by numerically differentiating joint angles with low pass filters. The control and data collection are implemented

[3] http://www.quanser.com.

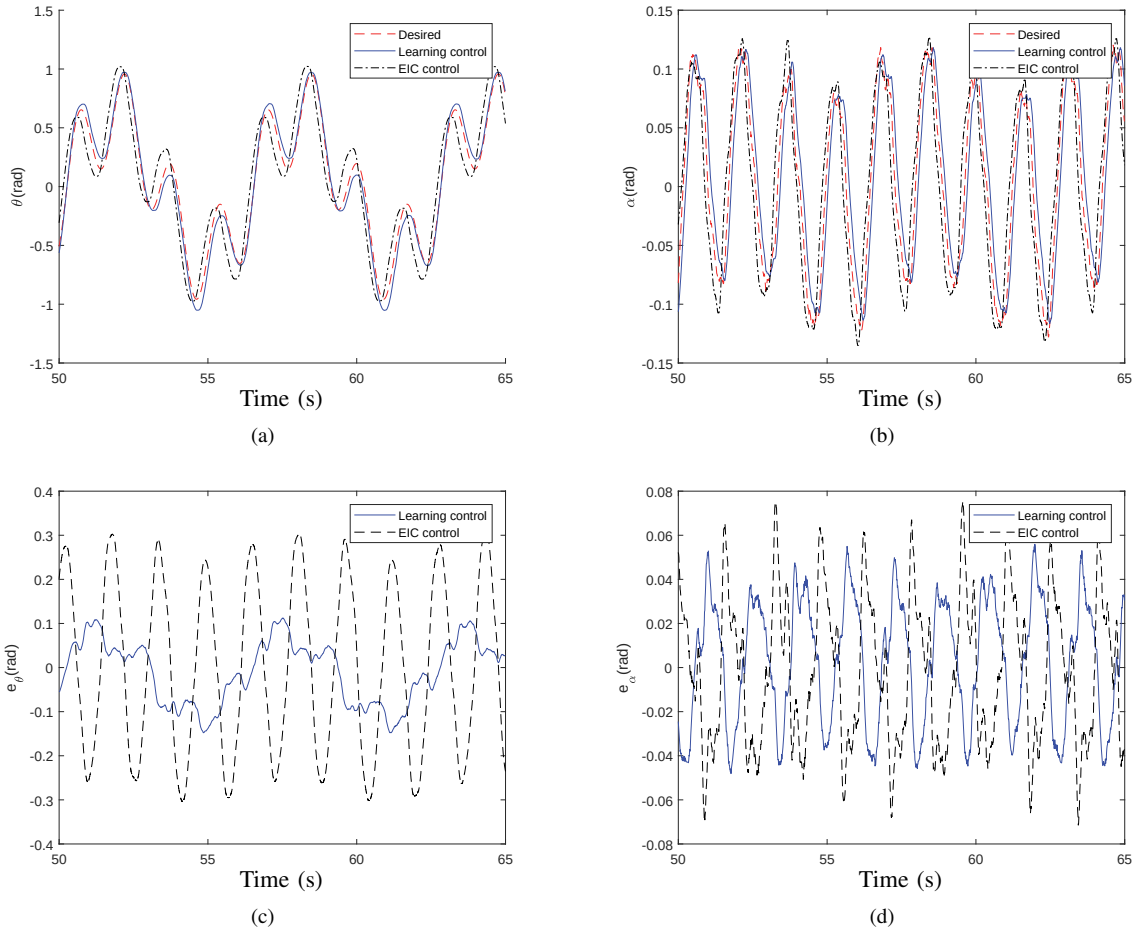


Fig. 3. Control performance comparison between the proposed learned model-based design and the EIC-based controller. (a) External angle θ tracking profiles. (b) Internal angle α tracking profiles. (c) External angle tracking errors e_θ. (d) Internal angle tracking errors e_α.

at 100 Hz. The proposed controller is implemented with sampling period $\Delta t = 20$ ms. In (22), the parameters values are chosen as $H = 27$, weight matrices $\boldsymbol{Q}_1 = \boldsymbol{Q}_e = \text{diag}(1000, 100)$, $\boldsymbol{Q}_2 = \text{diag}(100, 100)$ and $R = 10$ and $\rho = 1$ are chosen for (23).

Fig. 3 shows the control performance under the proposed design. We collect 800 training data points (8 s experiment run) in the training data set. In the testing experiments, the desired external trajectory was designed as $\theta_d = 0.6\sin(t) + 0.4\sin(4t)$ rad. The training data is collected from unsuccessful balance experiments. Fig. 3(a) shows the tracking performance of external subsystem angle θ and Fig. 3(b) for internal subsystem angle α. For comparison purpose, we also include the physical model-based EIC control performance in [2]. Figs. 3(c) and 3(d) further compare the tracking errors $\boldsymbol{e}_\theta$ and $\boldsymbol{e}_\alpha$ under these two controllers. It is clear from these results that the learning-based control design effectively captures the underactuated balance robotic dynamics and both the external tracking and internal balancing tasks are satisfactory. The performance under the learning-based design outperforms that with the physical model-based controller.

To understand the influence of training data set on the control performance, we vary the number of the training data set from 200 to 800 points to obtain different learned models. These models are used to track the same trajectory $\theta_d(t)$ as mentioned above. Fig. 4(a) shows the error distribution contours under different numbers of training data set for learning control and the EIC-based control design. For each learned model, the figure includes the tracking errors of a 90 s motion duration. The results clearly imply that with only 200 training data, the controller achieves the balancing and tracking tasks with large errors. With the increasing training data sets, the magnitudes of both the balancing and tracking errors decrease. With a set of 800 training data points, the learned model-based controller achieves superior performance than that under the analytical model-based controller.

The trade-off between the tracking and balancing performance is tuned by the choice of ρ value in the MPC objective function (23). Experiments are conducted to show the performance with the same learned model (obtained by using 200 training data points) under different values of ρ. Fig. 4(b) shows the performance of the tracking and balancing errors with different ρ values. We intentionally chose an inaccurate learned model and therefore, the value of $\|\boldsymbol{\Sigma}_d\|$ in (23) is

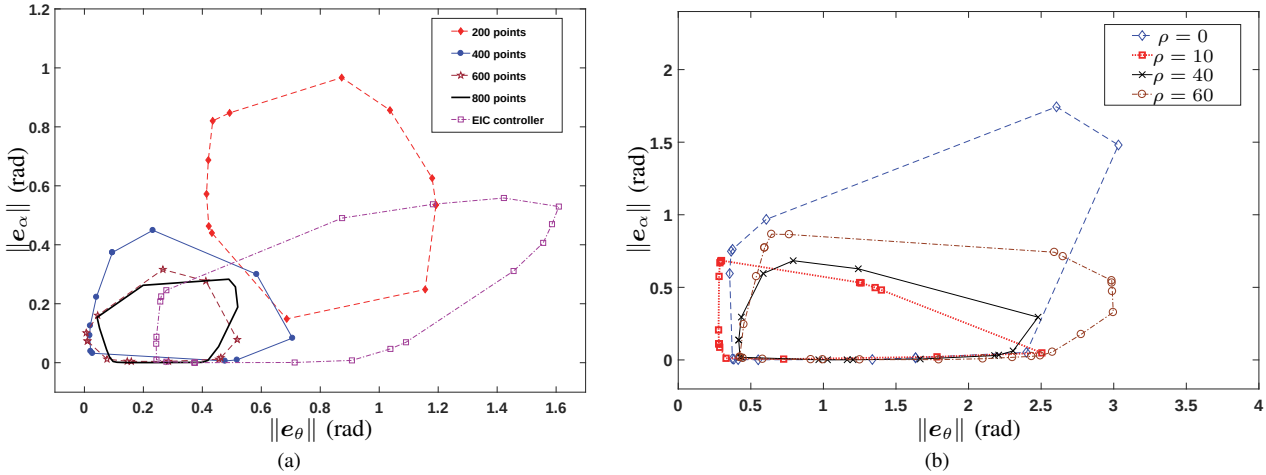


Fig. 4. (a) Comparison results of the internal balance error $\|\boldsymbol{e}_\alpha\|$ and external tracking error $\|\boldsymbol{e}_\theta\|$ under the learning model-based control by various training data points and the EIC-based control (with physical model). (b) Comparison of the balance and tracking errors $\|\boldsymbol{e}_\theta\|$ and $\|\boldsymbol{e}_\alpha\|$ under different values of the weight factor ρ.

relatively large. When $\rho = 0$, the balancing performance is not robust due to the poor inverse dynamics model. With $\rho = 10$, the system achieves a balanced trade-off between balancing and tracking tasks. With a further increased ρ value (i.e., $\rho = 40, 60$), the tracking performance becomes worse, and when $\rho > 80$ the controller fails to balance the pendulum. The averages of the variance of the inverse dynamics model for 60 s trials are 0.255, 0.174, 0.108 and 0.108 for $\rho = 0, 10, 40, 60$, respectively. The results clearly show that with increased ρ values, the magnitude of $\boldsymbol{\Sigma}_d$ decreases. This confirms that the integration of $\|\boldsymbol{\Sigma}_d\|$ in the objective function helps the performance improvements.

V. Conclusion and Future Work

We presented a learning model-based control design for underactuated balance robots. Besides non-minimum phase and unstable internal dynamics, one control challenge of underactuated balance robots is dependency of the equilibria on output trajectory. The proposed control system integrated a trajectory planner and feedback stabilization design. The trajectory planner used an MPC design to optimize the desired internal state profile and an inverse dynamics controller then stabilized the system around the planned profile. The control system was built on a GP learning model with no need of a priori knowledge about robot dynamics or successful balance training demonstration. The controller also incorporated the predictive model uncertainties into the online optimization process to enhance the performance. The design and analysis were demonstrated by Furuta pendulum experiments.

The proposed control system is readily applicable to a broad class of underactuated balance robots. Currently, we are implementing the control design to a bikebot system [28]. Performance improvement is also among the future research directions.

Acknowledgments

The authors would like to thank Dr. Pengcheng Wang of Rutgers University for his help and support for various constructive discussions. They are also grateful for anonymous reviewers for their helpful comments and suggestions to improve the quality of the paper.

Appendix

Suppose that the training data set contains n input-output data pairs $\mathcal{D} = \{(\boldsymbol{x}_i, y_i) | i = 1, \ldots, n\}$. The observation y_i is a noisy underlying output with zero mean Gaussian noise ε, i.e., $y_i = f(\boldsymbol{x}_i) + \varepsilon$. The variance of ε is σ_n^2.

The testing data set contains m input $\{(\boldsymbol{x}_i^*) | i = 1, \ldots, m\}$ for which the output f_i^* needs to be predicted. The joint distribution of the training outputs $\boldsymbol{y}$ and the test outputs $\boldsymbol{f}^*$ is

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

where $\boldsymbol{K}_{X,X^*}$ is the $n \times m$ matrix composed by the covariance function, that is, $\boldsymbol{K}_{X,X^*}(i,j) = k(\boldsymbol{x}_i, \boldsymbol{x}_j^*)$ and $\boldsymbol{K}_{X,X}$, $\boldsymbol{K}_{X^*,X}$ and $\boldsymbol{K}_{X^*,X^*}$ are defined similarly. The probabilistic prediction of $\boldsymbol{f}^*$ is given by the conditional distribution $\boldsymbol{f}^*|X, \boldsymbol{y}, X^* \sim \mathcal{N}(\bar{\boldsymbol{f}}^*, \boldsymbol{\Sigma}_{f^*})$

$$\begin{aligned} \bar{\boldsymbol{f}}^* &= \mathbb{E}[\boldsymbol{f}^*|X, \boldsymbol{y}, X^*] = \boldsymbol{K}_{X^*,X}[\boldsymbol{K}_{X,X} + \sigma_n^2 \boldsymbol{I}]^{-1} \boldsymbol{y} \\ \boldsymbol{\Sigma}_{f^*} &= \boldsymbol{K}_{X^*,X^*} - \boldsymbol{K}_{X^*,X}[\boldsymbol{K}_{X,X} + \sigma_n^2 \boldsymbol{I}]^{-1} \boldsymbol{K}_{X,X^*}. \end{aligned} \tag{26}$$

GP model captures dynamical systems $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) + \boldsymbol{\epsilon}$, $\boldsymbol{x} \in \mathbb{R}^{n_s}$ and $\boldsymbol{u} \in \mathbb{R}^{n_{in}}$, by modeling each dimension independently. The input for the regression is $[\boldsymbol{x}^T \, \boldsymbol{u}^T]^T$ while the output is $\dot{\boldsymbol{x}}$.

References

[1] A. Choukchou-Braham, B. Cherki, M. Djema, and K. Busawon, *Analysis and Control of Underactuated Mechanical Systems*. New York, NY: Springer, 2014.

[2] N. Getz, "Dynamic inversion of nonlinear maps with applications to nonlinear control and robotics," Ph.D. dissertation, Dept. Electr. Eng. and Comp. Sci., Univ. Calif., Berkeley, CA, 1995.

[3] J. Lee, R. Mukherjee, and H. K. Khalil, "Output feedback stabilization of inverted pendulum on a cart in the presence of uncertainties," *Automatica*, vol. 54, pp. 146–157, 2015.

[4] A. S. Shiriaev, L. B. Freidovich, A. Robertsson, R. Johansson, and A. Sandberg, "Virtual-holonomic-constraints-based design of stable oscillations of Furuta pendulum: Theory and experiments," *IEEE Trans. Robotics*, vol. 23, no. 4, pp. 827–832, 2007.

[5] M.-S. Park and D. Chwa, "Orbital stabilization of inverted-pendulum systems via coupled sliding-mode control method," *IEEE Trans. Ind. Electron.*, vol. 56, no. 9, pp. 3556–3570, 2009.

[6] L. B. Freidovich, A. S. Shiriaev, F. Gordillo, F. Gómez-Estern, and J. Aracil, "Partial-energy-shaping control for orbital stabilization of high-frequency oscillations of the Furuta pendulum," *IEEE Trans. Contr. Syst. Technol.*, vol. 17, no. 4, pp. 853–858, 2009.

[7] J. Yi, D. Song, A. Levandowski, and S. Jayasuriya, "Trajectory tracking and balance stabilization control of autonomous motorcycles," in *Proc. IEEE Int. Conf. Robot. Autom.*, Orlando, FL, 2006, pp. 2583–2589.

[8] E. R. Westervelt, J. W. Grizzle, C. Chevallereau, J. H. Choi, and B. Morris, *Feedback Control of Dynamic Bipedal Robot Locomotion*. Boca Raton, FL: CRC Press, 2007.

[9] K. Chen, M. Trkov, J. Yi, Y. Zhang, T. Liu, and D. Song, "A robotic bipedal model for human walking with slips," in *Proc. IEEE Int. Conf. Robot. Autom.*, Seattle, WA, 2015, pp. 6301–6306.

[10] J. Grizzle, M. Di Benedetto, and F. Lamnabhi-Lagarrigue, "Necessary conditions for asymptotic tracking in nonlinear systems," *IEEE Trans. Automat. Contr.*, vol. 39, no. 9, pp. 1782–1794, 1994.

[11] P. Wang and J. Yi, "Balance equilibrium manifold and control of rider-bikebot systems," in *Proc. Amer. Control Conf.*, Boston, MA, 2016, pp. 2168–2174.

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

[13] D. Nguyen-Tuong, J. Peters, M. Seeger, and B. Schölkopf, "Learning inverse dynamics: a comparison," in *Europ. Symp. Artificial Neural Networks*, 2008.

[14] D. Nguyen-Tuong, M. Seeger, and J. Peters, "Model learning with local gaussian process regression," *Auton. Robots*, vol. 23, no. 15, pp. 2015–2034, 2009.

[15] T. Beckers, J. Umlauft, D. Kulic, and S. Hirche, "Stable gaussian process based tracking control of lagrangian systems," in *Proc. IEEE Conf. Decision Control*, Melbourne, Australia, 2017, pp. 5180–5185.

[16] M. K. Helwa, A. Heins, and A. P. Schoellig, "Provably robust learning-based approach for high-accuracy tracking control of lagrangian systems," 2018, arXiv preprint arXiv:1804.01031.

[17] J. Ko, D. J. Klein, D. Fox, and D. Haehnel, "Gaussian processes and reinforcement learning for identification and control of an autonomous blimp," in *Proc. IEEE Int. Conf. Robot. Autom.*, Roma, Italy, 2007, pp. 742–747.

[18] J. Kocijan, R. Murray-Smith, C. E. Rasmussen, and A. Girard, "Gaussian process model based predictive control," in *Proc. Amer. Control Conf.*, Boston, MA, 2004, pp. 2214–2219.

[19] M. P. Deisenroth, D. Fox, and C. E. Rasmussen, "Gaussian processes for data-efficient learning in robotics and control," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 37, no. 2, pp. 408–423, 2015.

[20] G. Cao, E. M.-K. Lai, and F. Alam, "Gaussian process model predictive control of an unmanned quadrotor," *J. Intelli. Robot. Syst.*, vol. 88, no. 1, pp. 147–162, 2017.

[21] C. J. Ostafew, A. P. Schoellig, and T. D. Barfoot, "Robust constrained learning-based nmpc enabling reliable mobile robot path tracking," *Int. J. Robot. Res.*, vol. 35, no. 13, pp. 1547–1563, 2016.

[22] R. Murray-Smith, D. Sbarbaro, C. E. Rasmussen, and A. Girard, "Adaptive, cautious, predictive control with gaussian process priors," *IFAC Proc. Vol.*, vol. 36, no. 16, pp. 1155–1160, 2003.

[23] K. Chen, J. Yi, and T. Liu, "Learning-based modeling and control of underactuated balance robotic systems," in *Proc. IEEE Conf. Automat. Sci. Eng.*, Xi'an, China, 2017, pp. 1118–1123.

[24] S. Zhou, M. K. Helwa, and A. P. Schoellig, "An inversion-based learning approach for improving impromptu trajectory tracking of robots with non-minimum phase dynamics," *IEEE Robot. Automat. Lett.*, vol. 3, no. 3, pp. 1663–1670, 2018.

[25] Y. Zhang, P. Wang, J. Yi, D. Song, and T. Liu, "Stationary balance control of a bikebot," in *Proc. IEEE Int. Conf. Robot. Autom.*, Hong Kong, China, 2014, pp. 6706–6711.

[26] H. K. Khalil, *Nonlinear Systems*, 3rd ed. Upper Saddle River, NJ: Prentice Hall, 2002.

[27] K. Chen, "Learning-based model reduction and control of underactuated balancing robots," Ph.D. dissertation, Dept. Mech. Aero. Eng., Rutgers Univ., Piscataway, NJ, 2019.

[28] P. Wang, J. Yi, T. Liu, and Y. Zhang, "Trajectory tracking and balance control of an autonomous bikebot," in *Proc. IEEE Int. Conf. Robot. Autom.*, Singapore, 2017, pp. 2414–2419.