Proceedings of the ASME 2023 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference
IDETC/CIE 2023
August 20-23, 2023, Boston, MA, USA

IDETC2023-116959

DRAFT: GAUSSIAN PROCESS REGRESSION FOR SIM-TO-REAL TRANSFER OF HOPPING GAITS

Jeremy Krause
Mechanical and Industrial Engg.
University of Illinois at Chicago
842 W. Taylor St., Chicago, IL 60607.
Email: jkraus3@uic.edu

Adel Alaeddini
Mechanical Engg.,
The University of Texas at San Antonio,
1 UTSA Circle, San Antonio, TX 78249.
Email: adel.alaeddini@utsa.edu

Pranav A. Bhounsule
Mechanical and Industrial Engg.
University of Illinois at Chicago
842 W. Taylor St., Chicago, IL 60607.
Email: pranav@uic.edu

ABSTRACT

Simulation-based controllers are relatively easy to build and evaluate, but rarely transfer seamlessly to hardware. This is because of the reality-gap which is the discrepancy between simulations and hardware. Narrowing the reality-gap can speed up the deployment of simulated controllers to hardware, also known as sim-to-real. This paper presents sim-to-real transfer of controllers on a single leg hopping robot, a system that cycles between under-actuation during the stance phase to no-actuation during flight phase. Using simulations, we design a controller to achieve speed and height regulation once-per-step, but the controller cannot achieve accurate control on hardware. Using data from hardware, we model the mis-match between simulation and hardware using Gaussian Process Regression (GPR), recompute the controller, and redeploy it in hardware. It takes about 4 iterations to achieve accurate tracking. The results show that when GPR is used to model the step-to-step level model inaccuracy, it can lead to high accuracy sim-to-real transfer while maintaining sample efficiency. A video is here: `tiny.cc/idetc2023`

1 Introduction

To accelerate the development of robotic applications in the real world, it is important to use a simulation-based methodology to develop controllers. Indeed, simulations are computationally cheap (e.g., take a fraction of the time to set up and run), can be fully automated (e.g., running test cases without supervision), and inexpensive (e.g., robots don't break in simulation) as compared to hardware. This allows one to test various controller ideas, refine them, and tune them on simulation before being deployed on hardware. However, even the best simulators have limitations. For example, it is difficult to model friction, mass, inertia, deformations, noise, etc., with high accuracy leading to the reality gap – differences between hardware and simulation. It is important to close this gap to enable seamless deployment of simulation-based controllers to hardware also known as sim-to-real transfer. This work presents a technique for sim-to-real transfer by modeling the discrepancy between simulation and hardware and tuning the controller based on the improved model. In particular, we demonstrate the efficacy of the approach on a single leg hopping robot .

2 Background and Related Work

A single leg robot that can move by hopping is a good system for testing and benchmarking. The system is simple enough to model, yet offers sufficient control challenges. One of the most challenging aspect of the hopping system is that it is under-actuated (less actuators than degrees of freedom) when the foot is in contact with the ground and un-actuated (no control) in the flight phase. Hence to enable stable hopping, the control needs to be clairvoyant (estimate what will happen in future for a given control) and also punctilious (provide accurate control).

The earliest hopping robot was built by Raibert [1] who showed that simple speed and height regulation using foot placement control in the flight phase and push-off by the foot in the stance phase achieves stable hopping. Raibert's method of control may be formalized using Poincaré section and map [2]. The Poincaré section is an instant in the locomotion cycle (e.g., apex for the hopper) and the Poincaré map is a function that maps the state from one Poincaré section to the next. The linearization over the Poincaré map may be used to quantify the stability of the hopping gait. The Poincaré map approach has been widely used for control of hopping robot (for example, [3], [4], [5]).

Sim-to-real transfer has attention in the recent few years. Dynamic randomization is a popular approach to achieve efficient sim-to-real transfer [6, 7]. The control policy (usually a neural network) is trained by randomizing the features of the simulation (mass, inertia, friction, latency). The expectation is that the resulting control policy is robust to variability and hence has a higher chance of succeeding on hardware. Such an approach is computationally demanding and hence not scalable. The scalability issue has been overcome to some extent by seeding the learning with samples from motion capture [8] or learning in lower dimensional space such as in the cartesian space [9]. Another approach is to use real-data to improve sub-system model (e.g., actuator dynamics [10]).

Control improvements may also be achieved directly in hardware by using sample efficient optimization such as Bayesian Optimization [11]. Here, the cost function (e.g., speed, energy) is mapped to parameters of the control policy using a Gaussian Process Regression (GPR). During training on hardware, the control policy parameters are directly optimized using an acquisition function such as probability of improvement, expected improvement, and upper confidence bound. However, GPRs do not work scale well with the input space (here control parameters) and data points. The first issue may be fixed by using latent space representations that map high dimensional input to low dimensional space for GPR [12, 13] and second issue may be addressed using sparse GPR [14].

In this paper, we take a different approach. Instead of learning a control policy in hardware using GPR, we learn the correction to the simulation-based model in hardware using GPR. This approach has been used for bongo board balance [15], control of a blimp [16], control of a quadcopter [17], and control of unmanned ground vehicles [18]. The improved model is then used to develop a controller using existing methods (e.g., model-based control). In these past approaches, the corrected model is estimated over time domain (typically 20 - 200 Hz). This leads scalability issues with GPRs. We exploit the Poincaré map representation, which is a low-dimensional model for control purposes. We use the GPRs to learn a correction to the Poincaré map. The correction is estimated over one or more steps (typically 2 Hz for slow systems like walking and 10 Hz for fast systems like hopping). This ensures scalability of the proposed approach. The *novelty* of the approach is to use Poincaré map based model correction using GPRs to ensure scalability of the approach and its demonstration on a custom-built hopping robot.

The paper is organized as follows. In Sec. 3 we provide details of the physics-based model. In Sec. 4 we present the low-level and the step-level controller. In Sec. 5, we present details of the custom-built hardware. This is followed by Results in Sec. 6, Discussion in Sec 7, and Conclusion and Future work in Sec. 8.

3 Model

Figure 1 shows the 2D model based on the custom-built hopping robot. The hopper is confined to the sagittal plane using a boom. The coordinates of the boom connection to the hopper are (x,y). The hopper is a closed chain. The top two and bottom two links are symmetrical except for the protruding second link on one side as shown. Only two angles are needed to define the kinematics of the chain. We use α_1 as the angle between the vertical and the line joining the boom axis to the joint between bottom two links and α_2 as half the angle between the top two links.

The mass of the boom and its inertia are $M = 1.466$ kg and $I = diag\{I_x, I_y, I_z\}$ where $I_x = I_z = 5$ kg/m^2 and $I_y = 0.0015$ kg/m^2. The two links are connected to each other and to the boom through pin joints. The robot top links have length $\ell_1 = 0.11$ m and bottom links have length $\ell_2 = 0.2$ m. The top links have mass $m_1 = 0.043$ kg and inertia $I_1 = 0.0001$ kg/m^2 while the bottom link have mass $m_2 = 0.047$ kg and inertia $I_2 = 0.00025$ kg/m^2. The center of mass of the top link is at a distance of $c_1 = 0.0485$ m from the boom axis and the center of mass of the bottom link is at a distance of $c_2 = 0.1054$ m from the pin joint connecting the top and bottom link. The protrusion of the second link is $\ell_3 = 0.045$ m. Gravity is $g = 9.81$ m/s^2 and points vertically downward.

We use Euler-Lagrange equations to describe the movement of the hopper. The motion consists of a stance phase where we assume that the foot is in contact with the ground and a flight phase where the hopper moves freely under gravity. There are two transitions, the collision phase where the hopper transitions from flight to stance phase and the take-off phase where the hopper transitions from stance phase to the flight phase. These are described in detail next.

3.1 Stance phase equations

The state variables for derivation are defined as $\mathbf{q} = \begin{bmatrix} x \; y \; \alpha_1 \; \alpha_2 \end{bmatrix}^T$. The Lagrangian $\mathscr{L} = \mathscr{T} - \mathscr{V} = 0.5(Mv^T v + I_x \dot{y}^2/R + I_z \dot{x}^2/R + I_y \omega^2) + \sum \left(m_i v_i^T v_i + I_i \omega_i^2 \right) - Mgy - \sum \left(m_i g y_i \right)$ where v_i, ω_i, y_i are the linear velocity, angular velocity, and y-position center of mass of link i respectively and boom length is $R = 1.854$ m. We take the summation over the 4

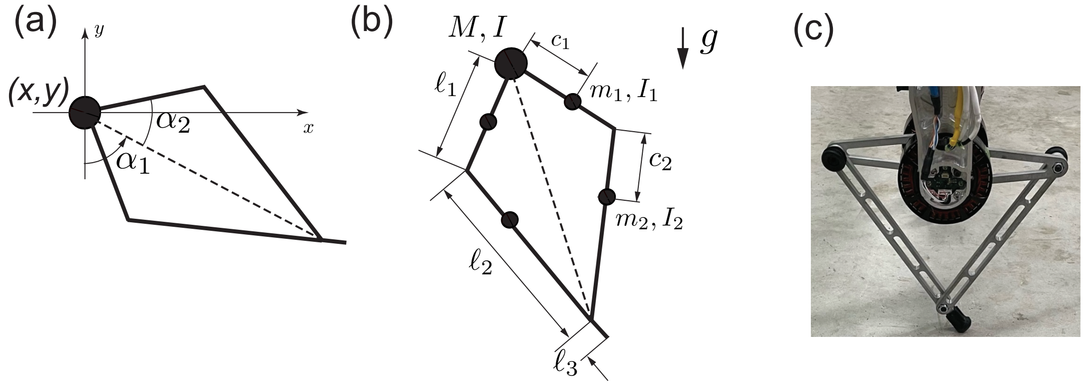

FIGURE 1. 2D model: (a) configuration variables describing the degrees of freedom, (b) mass, center of mass, inertia about center of mass, and length parameters, (c) hardware

links. Using the Euler-Lagrange equations gives 4 equations

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{N}(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{B}\mathbf{u} + \mathbf{J}^T\mathbf{P} \tag{1}$$

where $\mathbf{M}$, $\mathbf{N}$, $\mathbf{B}$ are the mass matrix, accelerations due to Coriolis, centrifugal acceleration and gravity, and torque selection matrices. The control torques are $\mathbf{u} = \begin{bmatrix}\tau_1 & \tau_2\end{bmatrix}^T$ where τ_i is the torque for the two top links, $\mathbf{J}$ is the Jacobian of the contact point with respect to the degrees of freedom and $\mathbf{P}$ is the ground reaction force on the stance leg.

The foot does not move during stance phase. This can be computed by differentiating the position of the contact points twice, $\ddot{x}_{\text{foot}} = \ddot{y}_{\text{foot}} = 0$ to give

$$\mathbf{J}\ddot{\mathbf{q}} = -\dot{\mathbf{J}}\dot{\mathbf{q}} \tag{2}$$

We can combine Eqns. 1 and 2 to get

$$\begin{bmatrix}\mathbf{M}(\mathbf{q}) & -\mathbf{J}^T \\ \mathbf{J} & \mathbf{0}\end{bmatrix}\begin{bmatrix}\ddot{\mathbf{q}} \\ \mathbf{P}\end{bmatrix} = \begin{bmatrix}-\mathbf{N}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{B}\mathbf{u} \\ -\dot{\mathbf{J}}\dot{\mathbf{q}}\end{bmatrix} \tag{3}$$

3.2 Flight phase equations

The swing phase equations are derived from the stance phase equations by setting $\mathbf{P} = 0$ in Eqn. 1 to get

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{N}(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{B}\mathbf{u} \tag{4}$$

3.3 Flight to stance phase transition

The transition from flight to stance phase occurs when the foot makes contact with the ground. This is given by $y_{\text{foot}} = 0$.

During the transition, the position variables are assumed to be the same while the velocity variables are assumed to change based on the conservation of angular momentum about the stance foot. We obtain the velocity during transition by integrating Eqn 1 and taking the limit as time goes to 0 to yield

$$\mathbf{M}(\mathbf{q}^-)(\dot{\mathbf{q}}^+ - \dot{\mathbf{q}}^-) = \mathbf{J}^T\mathbf{P}_t \tag{5}$$

where the superscripts $^-$ and $^+$ indicate instants before and after collision and $\mathbf{P}_t$ is the impulse on the stance foot. The foot comes to rest after the collision. This is give by $\dot{y}^+_{\text{foot}} = 0$ to get

$$\mathbf{J}\dot{\mathbf{q}}^+ = \mathbf{0} \tag{6}$$

We can combine Eqns. 5 and 6 to get

$$\begin{bmatrix}\mathbf{M}(\mathbf{q}^-) & -\mathbf{J}^T \\ \mathbf{J} & \mathbf{0}\end{bmatrix}\begin{bmatrix}\dot{\mathbf{q}}^+ \\ \mathbf{P}_t\end{bmatrix} = \begin{bmatrix}\mathbf{M}(\mathbf{q}^-)\dot{\mathbf{q}}^- \\ \mathbf{0}\end{bmatrix} \tag{7}$$

3.4 Stance to flight phase transition

The stance to flight phase transition occurs when the vertical ground reaction force $\mathbf{P}_y = 0$. This reaction force is obtained from $\mathbf{P}$ in Eqn. 3. During the transition from stance to flight phase the positions and velocities are assumed to remain the same.

4 Controller

4.1 Low-level controller

The low-level controller runs at 1 kHz. The controller takes in sensor data, the angles and rates, and outputs joint torque. There is a controller for the stance and the flight phase.

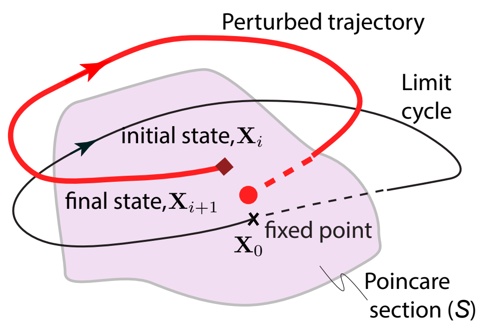

FIGURE 2. Poincaré map and section

Flight Phase controller: The flight phase controller uses a position-derivative controller to set the joint angles. In the flight phase we input a desired leg length ℓ_{des} and leg orientation θ_{des}. There is a kinematic mapping $[\ell, \theta] = \mathbf{f}(\alpha_1, \alpha_2)$. We are able to analytically invert $\mathbf{f}$ to compute the desired angles $(\alpha_1, \alpha_2)_{\text{des}} = \mathbf{f}^{-1}(\ell_{\text{des}}, \theta_{\text{des}})$. The feedback controller can then be written as $\tau = [\tau_1 \ \tau_2] = -\mathbf{K}_p(\alpha - \alpha_{\text{des}}) - \mathbf{K}_d(\dot{\alpha} - \dot{\alpha}_{\text{des}})$ where $\mathbf{K}_p$ and $\mathbf{K}_d$ are the proportional and damping gain respectively. There is a feedback control on the desired landing angle $\theta_{\text{des}} = K_1(\dot{x} - \dot{x}_{\text{des}})$ where K_1 and $\dot{x}_{\text{des}}$ are the proportionality gain and the desired speed respectively.

Stance Phase controller: The stance phase controller regulates the force in the leg. The force is simulated to be a virtual spring. The force $F = -K_2(\ell - \ell_0)$ where K_2 and ℓ_0 are the user specified stiffness and the rest length of the virtual spring, respectively. The force is then resolved in the x- and y- direction as follows $F_x = F\cos(\alpha_1)$ and $F_y = F\sin(\alpha_1)$. Using the Jacobian $\mathbf{J}$ from the toe to the joints, we compute the desired torques, $\tau = [\tau_1 \ \tau_2] = \mathbf{J}^T [F_x \ F_y]$.

4.2 High-level (Step-level) controller

Our goal is to track a desired apex horizontal speed $\dot{x}_{\text{des}}$ and desired apex height y_{des}. This is achieved using a step-level control using a Poincaré map. The step-level control attempts to regulate the speed and height once per step and hence the name. We first describe the Poincaré map/section that is used as a model for the step-level controller

Poincaré section and map: Figure 2 shows pictorial depiction of the Poincaré section and map [2]. Consider an instant in the gait cycle (e.g., instant when the hopper is the apex). This is the Poincaré section. Let the reduced state (which is a subset of the complete state) at the Poincaré section at the current step be $\mathbf{X}_i$. This is shown with the red diamond. Our low-level control parameterization choses the control $\mathbf{U}_i$ (e.g., amplitude, gain, set-point) which takes the reduced state to $\mathbf{X}_{i+1}$ at the Poincaré section at the next step. There is a function $\mathbf{P}$, known as the Poincaré map, that maps the reduced state from one step to the next. This is given by

$$\mathbf{X}_{i+1} = \mathbf{P}(\mathbf{X}_i, \mathbf{U}_i) \tag{8}$$

Note that the control $\mathbf{U}^i$ is set once per step and kept constant during the step. We can also find a state, control pair $(\mathbf{X}_0, \mathbf{U}_0)$ that would repeat itself at the next step, $\mathbf{X}_0 = \mathbf{P}(\mathbf{X}_0, \mathbf{U}_0)$. This state, $\mathbf{X}_0$, is known as the fixed point and it gives rise to a periodic gait known as the limit cycle.

For the hopper we choose the Poincaré map at the apex where $\dot{y}_i = 0$ and $\mathbf{X}_i = [\dot{x}_i \ y_i]$ and $\mathbf{U}_i = [fx_i \ fy_i]$ where fx_i and fy_i are constant forces added to the stance level controller (described in Sec. 4.1) after the mid-stance phase (when $\dot{y} > 0$).

Thus, we have

$$[\dot{x}_{i+1} \ y_{i+1}] = \mathbf{P}([\dot{x}_i \ y_i \ fx_i \ fy_i]) \tag{9}$$

Unfortunately, there is no analytical solution to the Poincaré map $\mathbf{P}$. We use numerical integration to determine the state $[\dot{x}_{i+1} \ y_{i+1}]$ for the given inputs $[\dot{x}_i \ y_i \ fx_i \ fy_i]$.

4.3 Step-level control problem

The step-level control problem is stated as follows. Given the state at the current step, $[\dot{x}_i \ y_i]$, compute the control at the current step, $[fx_i \ fy_i]$, such that the state at the next step is $[\dot{x}_{i+1} \ y_{i+1}] = [\dot{x}_{\text{des}} \ y_{\text{des}}]$. From Eqn. 9 we can write

$$[\dot{x}_{\text{des}} \ y_{\text{des}}] = \mathbf{P}([\dot{x}_i \ y_i \ fx_i \ fy_i]) \tag{10}$$

These are two nonlinear equations and two control variables fx_i and fy_i. These equations may be solved using nonlinear root finding algorithm.

4.4 Sim-to-real transfer

The controller derived from the simulation-based Poincaré map, Eqn. 10, leads to relatively poor tracking on hardware. This is known as the reality gap [7]; simulation-based controllers do not work as intended on hardware. We use a Gaussian Process Regression to update the Poincaré map as follows [19].

For a given reference horizontal speed and height, we use the simulation-based Poincaré map (Eqn. 10) to compute a control. We store the state obtained on the hardware, $[\dot{x}^{\text{true}}_{i+1} \ y^{\text{true}}_{i+1}]$. We then fit the error in the Poincaré map as follows

$$[\dot{x}^{\text{true}}_{i+1} \ y^{\text{true}}_{i+1}] - \mathbf{P}([\dot{x}_i \ y_i \ fx_i \ fy_i]) = GP([\dot{x}_i \ y_i \ fx_i \ fy_i]) \tag{11}$$

where *GP* is the Gaussian Process Regression model.

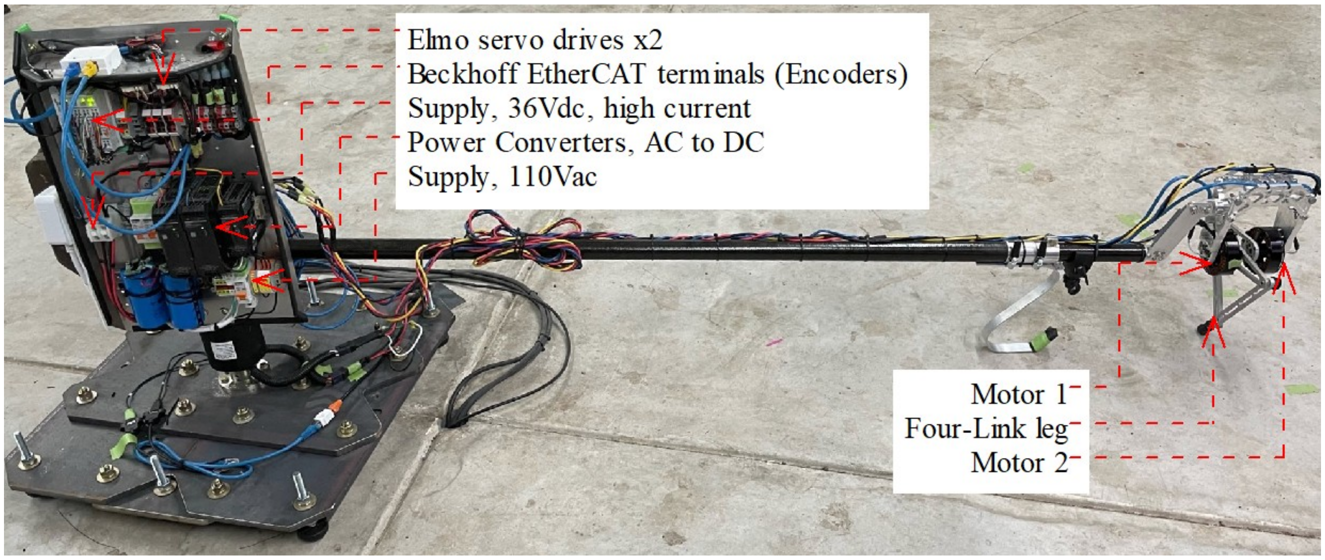
Elmo servo drives x2
Beckhoff EtherCAT terminals (Encoders)
Supply, 36Vdc, high current
Power Converters, AC to DC
Supply, 110Vac
Motor 1
Four-Link leg
Motor 2

FIGURE 3. Hardware setup

Then we re-compute the control using the improved Poincaré map as follows

$$\left[\dot{x}_{\mathrm{des}}\ y_{\mathrm{des}}\right] = \mathbf{P}(\left[\dot{x}_i\ y_i\ fx_i\ fy_i\right]) + GP(\left[\dot{x}_i\ y_i\ fx_i\ fy_i\right]) \qquad (12)$$

This is solved using a root solving algorithm.

5 Hardware platform

5.1 Overview

The hardware platform utilized in this work is shown in Fig.3. The setup consists of a custom, two degree-of-freedom, portable rotating boom with a boom-arm-attached monoped leg, linked via EtherCAT network to a real time linux server. The boom-arm attached monoped leg is an, electrically driven, asymmetric four-link mechanism that acts to convert two opposing torques into linear force and torque. The drive section for the monoped leg mechanism consist of opposite mounted T-motor U10plus brushless DC motors with attached custom 14bit magnetic encoder modules, build around the IC Haus MHM encoder IC. The closed torque control loops of the monoped drive section are controlled within dual onboard Elmo Motion Control 80V/80A servo drives. Higher level control is provided by a real time Linux master terminal via an synchronous EtherCAT network operating at a 4kHz.

5.2 Boom

The custom portable rotating boom is a platform that facilitates the controlled motion and observation of an experimental package while constraining unwanted degrees-of-freedom (dof). The rotating boom utilized in this work has two unconstrained dof: about the vertical axis (oriented parallel to gravity) and about a rotating horizontal axis (oriented perpendicular to the vertical axis). The boom rotating axes are equipped with mechanical encoders which offer a resolution of 0.072 and 0.022 degrees in the horizontal and rotating vertical planes, respectively. Boom encoder position computation and tracking are performed by onboard Beckhoff EtherCAT terminals, operating synchronously with the EtherCAT-network-linked Elmo servo drives. The ultra high modulus carbon fiber boom arm, which couples the monoped leg mechanism utilized in this work to the body of the portable rotating boom, is adjustable in length from 1.85m to 2.75m. Onboard EtherCAT communication, high-voltage low-current supply (120Vac), and low-voltage high-current supply (36Vdc) are provided via an electro-mechanical slip ring, allowing for infinite and unobstructed rotation of the boom arm about the vertical axis of the boom.

5.3 Control terminal and EtherCAT communication

High level command and control for both the monoped leg mechanism and the rotating boom platform are performed on a Dell Optiplex 9020 with an Intel i7 processor, operating Ubuntu Linux with a real time kernel. The EtherCAT network interface adapter is 10Gtek network interface card (NIC) with an Intel 82574L chipset. EtherCAT network management and EtherCAT network interfacing between the hardware and main state machine are provided by the Acontis Technologies EtherCAT master stack. The Acontis Technologies EtherCAT master stack is a collection of libraries that provide link layer management be-

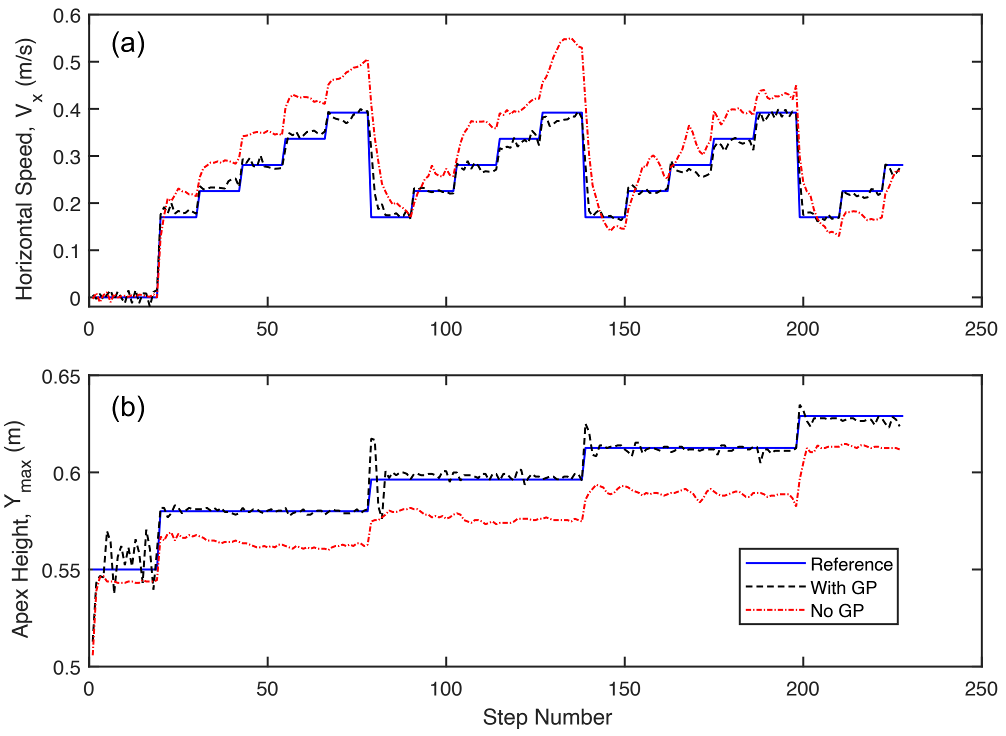

FIGURE 4. Comparing controllers on hardware. Controller based on simulation-based Poincaré map (no GP) versus controller based on learnt Poincaré map (with GP).

tween an NIC and user software, allowing off-the-shelf network and computer hardware to be utilized as an EtherCAT master terminal.

Through the use of the EC master stack libraries, the state machine is able to synchronously access the torque, velocity and position of each U10plus drive motor, as well as the angular position of both the horizontal and vertical boom-mounted encoders. As the torque control loop executes directly on the Elmo servo drives, the only data passed to the servo drives, synchronously, by the state machine is the reference torque command for each servo drive. Due to the utilized real-time Linux kernel, the user software (including the state machine) is able to execute significantly faster than on a non-rt system and, due to the use of a multithreaded program architecture, numerous computational operations are able to be performed between each successive EtherCAT network cycle without blocking the operation of the EtherCAT masterstack, providing for the ability to handle substantial computational problems in real time while the hardware is in operation without blocking EtherCAT communication.

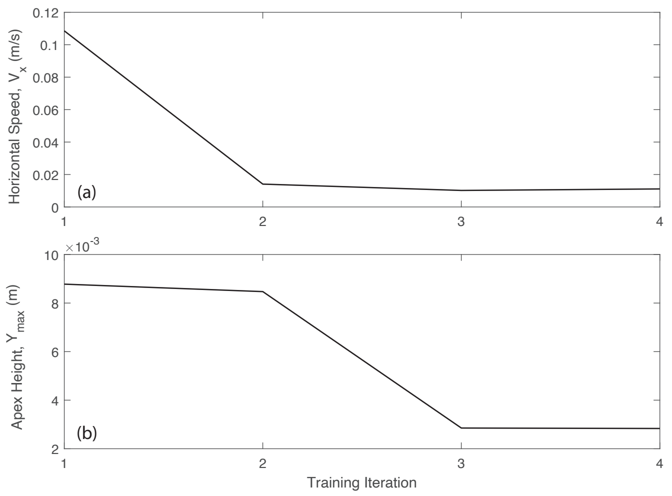

FIGURE 5. Average mean squared error of the Poincaré map correction vs iteration number.

6 Results

The simulation was done using MATLAB. We used the function *ode113* to integrate the equations of motion given by

Eqns 3, 4, and 7. The integrator has an in-built event detection that helps to detect transitions. The Poincaré map (Eqn. 9) is obtained by integrating the equations from the one apex to the next.

We generated a reference profile for $\dot{x}_{\text{des}}$ and y_{des} as shown in Fig. 4 (blue solid line). Using the simulation-based Poincaré map (Eqn. 9) and MATLAB *fsolve*, we generated a controller as a lookup table. This was then executed on the robot. This is shown in Fig. 4 (red dash-dotted line). It can be seen that there is a significant tracking error for both, $\dot{x}$ and y.

Next, we used the experimental data to fit the error in the Poincaré map estimation using a Gaussian Process Regression (Eqn. 11). We used the MATLAB function *fitrgp* with a constant basis function and matern52 kernel. Then we used the corrected Poincaré map to compute the control (see Eqn. 12). This was tested on the hardware and the resulting data was use to improve the estimate of the Gaussian Process. This process was repeated 3 more times, for a total of 4 training iterations and 4 hardware deployment. The 4th iteration led to acceptable results. The error in tracking using the 4th iteration GP is shown in Fig. 4 (black dashed line). It can be seen that the error has substantially reduced compared to the simulation-based controller. Thus the approach is shown to successfully improve the sim-to-real transfer using just 4 training iterations. A video comparing the controllers with and without GPR is here: `tiny.cc/idetc2023`.

Figure 5 shows the progress in the training, the average mean squared error (MSE) vs the iteration number for both the variables. It can be seen that the MSE converges in a mere 1 iteration for velocity and about 3 iterations for apex height. Figure 6 and 7 gives more details about the mean/confidence interval for the converged iteration (top) and the standard deviation over the first and last converged iteration (bottom). From the top plots it can be seen that that confidence interval is narrow for both variables. From bottom figures it can be seen that initially (iteration 1), the standard deviation is the largest when there is a step change in the reference. However, with more training (iteration 4), the standard deviation at the step change decreases. In particular, the maximum deviation iteration 1 is about 0.03 m/s which decreases to 0.02 m/s for velocity and from 0.015 m to under 0.01 m for apex height.

7 Discussion

We presented a technique for sim-to-real transfer tracking control of hopping robot. The technique consists of using a physics-based model to numerically obtain the Poincaré map. Then the map is iteratively improved by designing a controller with the Poincaré map, collecting data from hardware, and fitting a correction to the Poincaré map using Gaussian Process Regression. This process is repeated till the Gaussian Process Regression model converges, i.e., there is no improvement on the mean square error. The converged estimate of the Poincaré map provides accurate tracking compared against simulation obtained Poincaré map.

The differences between the model and simulation were due to boom dynamics (the boom flexed during motion), ground variance, and mismatch in mass, center of mass, and inertia to some extent. Some of these difference are best modeled as a bias (e.g., mass, inertia, center of mass), but others such as boom dynamics and ground variance are best modeled as a bias and an uncertainty. A Gaussian Process Regression (GPR) can effectively model both, bias and uncertainty as a mean and standard deviation respectively. The mis-modeled dynamics shows up as a steady state error (a bias) in the tracking using the simulation-based controller as shown in Fig. 4 (red dash-dotted line). The boom dynamics have the biggest effect when the hopper changes the reference speed and velocity. This can be seen in the relatively large standard deviations during these step changes (see Fig. 6 and 7).

Unlike past approaches which learn the control policy in hardware, we learn the model mismatch. The advantage of learning the model mismatch (as opposed to the control) is that the resulting model can be used in a novel scenario without taking more experimental data. Also, past approaches learn the GPR for model- or control-policy in the time domain. Since the GPR scales as N^3 where N is the number of data points, time domain models are challenging to scale. In contrast, we learn the GPR model at the step-level where N is step number, our GPR model can model more data sets than using time-based parameterization.

There are several limitations of our approach. unlike past approaches that use neural networks for modeling, GPR provide local approximations. However, it is possible to use neural networks or polynomials to approximate the Poincaré map [20]. For the hopping system, there are only 4 independent variables for the GPR (see Eqn. 11) which helps in scaling the approach. However, this may not be true for high-dimensional systems. In such cases, one attractive option is to use a feature space to map the Poincaré variable to a low-dimensional space for GPR (e.g., see [12]). Finally, it is important to choose free parameters from the low-level control that have appreciable effect on the states on Poincaré map. There is so far, no systematic way to find such parameters, but human intuition and trial-and-error might help [21, 22].

8 Conclusions and Future Work

We conclude that Gaussian Process Regression (GPR) can scale for sim-to-real transfer when used to model the correction to the Poincaré map rather than the low-level physics. Our results show that the convergence is fast (about 4 iterations). Thus, GPR is promising method for sim-to-real transfer.

The future work will investigate GPR in situation where there is added uncertainty (e.g., manipulate the ground profile)

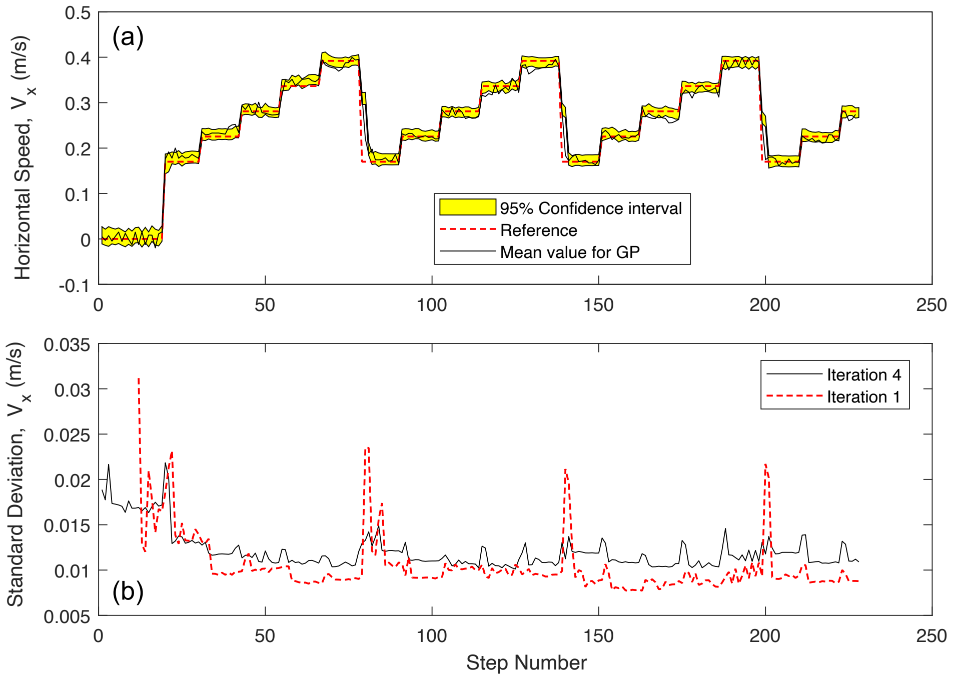

FIGURE 6. Training of Poincaré map for horizontal velocity, (a) Mean and confidence intervals for the training, (b) Standard deviation for the first and fourth (last) iteration.

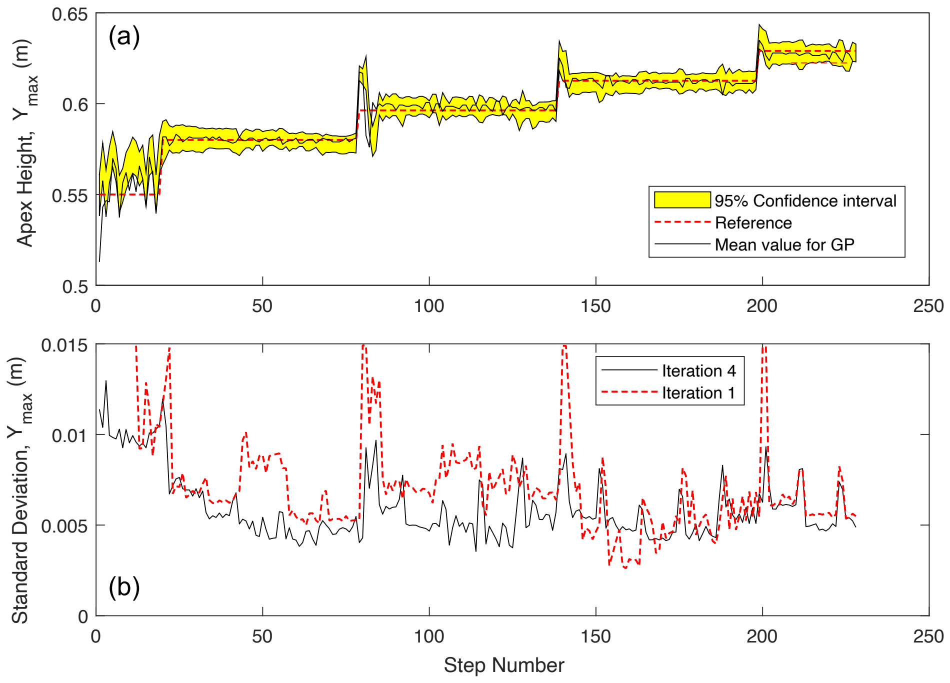

FIGURE 7. Training of Poincaré map for apex height, (a) Mean and confidence intervals for the training, (b) Standard deviation for the first and fourth (last) iteration.

and the use of model-based control framework for tracking of reference signals.

ACKNOWLEDGMENT

The authors would like to thank Acontis Technologies for the use of and assistance with their EtherCAT Master Stack software. The work is funded by US National Science Foundation through grant 2128568.

REFERENCES

[1] Raibert, M., 1986. *Legged robots that balance*. MIT press Cambridge, MA.

[2] Strogatz, S., 1994. *Nonlinear dynamics and chaos*. Addison-Wesley Reading.

[3] Dadashzadeh, B., Mahjoob, M., Nikkhah Bahrami, M., and Macnab, C., 2014. "Stable active running of a planar biped robot using poincare map control". *Advanced Robotics,* ***28***(4), pp. 231–244.

[4] Cheng, M.-Y., and Lin, C.-S., 1996. "Measurement of robustness for biped locomotion using a linearized poincaré map". *Robotica,* ***14***(3), pp. 253–259.

[5] Bhounsule, P. A., Zamani, A., Krause, J., Farra, S., and Pusey, J., 2021. "Control policies for a large region of attraction for dynamically balancing legged robots: a sampling-based approach". *Robotica,* ***39***(1), pp. 107–122.

[6] Mordatch, I., Lowrey, K., and Todorov, E., 2015. "Ensemble-cio: Full-body dynamic motion planning that transfers to physical humanoids". In 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, pp. 5307–5314.

[7] Tan, J., Zhang, T., Coumans, E., Iscen, A., Bai, Y., Hafner, D., Bohez, S., and Vanhoucke, V., 2018. "Sim-to-real: Learning agile locomotion for quadruped robots". *arXiv preprint arXiv:1804.10332*.

[8] Peng, X. B., Coumans, E., Zhang, T., Lee, T.-W., Tan, J., and Levine, S., 2020. "Learning agile robotic locomotion skills by imitating animals". *arXiv preprint arXiv:2004.00784*.

[9] Bellegarda, G., and Nguyen, Q., 2021. "Robust high-speed running for quadruped robots via deep reinforcement learning". *arXiv preprint arXiv:2103.06484*.

[10] Hwangbo, J., Lee, J., Dosovitskiy, A., Bellicoso, D., Tsounis, V., Koltun, V., and Hutter, M., 2019. "Learning agile and dynamic motor skills for legged robots". *Science Robotics,* ***4***(26), p. eaau5872.

[11] Calandra, R., Seyfarth, A., Peters, J., and Deisenroth, M. P., 2014. "An experimental comparison of bayesian optimization for bipedal locomotion". In 2014 IEEE international conference on robotics and automation (ICRA), IEEE, pp. 1951–1958.

[12] Rai, A., Antonova, R., Song, S., Martin, W., Geyer, H., and Atkeson, C., 2018. "Bayesian optimization using domain knowledge on the atrias biped". In 2018 IEEE International Conference on Robotics and Automation (ICRA), IEEE, pp. 1771–1778.

[13] Yu, W., Kumar, V. C., Turk, G., and Liu, C. K., 2019. "Sim-to-real transfer for biped locomotion". In 2019 ieee/rsj international conference on intelligent robots and systems (iros), IEEE, pp. 3503–3510.

[14] Snelson, E., and Ghahramani, Z., 2007. "Local and global sparse gaussian process approximations". In Artificial Intelligence and Statistics, PMLR, pp. 524–531.

[15] Ha, S., and Yamane, K., 2015. "Reducing hardware experiments for model learning and policy optimization". In 2015 IEEE International Conference on Robotics and Automation (ICRA), IEEE, pp. 2620–2626.

[16] Ko, J., Klein, D. J., Fox, D., and Haehnel, D., 2007. "Gaussian processes and reinforcement learning for identification and control of an autonomous blimp". In Proceedings 2007 ieee international conference on robotics and automation, IEEE, pp. 742–747.

[17] Berkenkamp, F., Schoellig, A. P., and Krause, A., 2016. "Safe controller optimization for quadrotors with gaussian processes". In 2016 IEEE international conference on robotics and automation (ICRA), IEEE, pp. 491–496.

[18] Ostafew, C. J., Schoellig, A. P., and Barfoot, T. D., 2014. "Learning-based nonlinear model predictive control to improve vision-based mobile robot path-tracking in challenging outdoor environments". In 2014 IEEE International Conference on Robotics and Automation (ICRA), IEEE, pp. 4029–4036.

[19] Williams, C. K., and Rasmussen, C. E., 2006. *Gaussian processes for machine learning*, Vol. 2. MIT press Cambridge, MA.

[20] Bhounsule, P. A., Kim, M., and Alaeddini, A., 2020. "Approximation of the step-to-step dynamics enables computationally efficient and fast optimal control of legged robots,". In ASME 2020 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers.

[21] Bhounsule, P. A., Ruina, A., and Stiesberg, G., 2015. "Discrete-decision continuous-actuation control: balance of an inverted pendulum and pumping a pendulum swing". *Journal of Dynamic Systems, Measurement, and Control,* ***137***(5), p. 051012.

[22] Bhounsule, P. A., Cortell, J., Grewal, A., Hendriksen, B., Karssen, J. D., Paul, C., and Ruina, A., 2014. "Low-bandwidth reflex-based control for lower power walking: 65 km on a single battery charge". *International Journal of Robotics Research*.