2017 IEEE International Conference on Robotics and Automation (ICRA)
Singapore, May 29 - June 3, 2017

Learning to Jump in Granular Media: Unifying Optimal Control Synthesis with Gaussian Process-Based Regression

Alexander H. Chang[1], Christian M. Hubicki[2], Jeff J. Aguilar[3],
Daniel I. Goldman[3], Aaron D. Ames[2], and Patricio A. Vela[1]

Abstract— **The varied and complex dynamics of deformable terrain are significant impediments toward real-world viability of locomotive robotics, particularly for legged machines. We explore vertical jumping on granular media (GM) as a model task for legged locomotion on uncharacterized deformable terrain. By integrating (Gaussian process) GP-based regression and evaluation to estimate ground forcing as a function of state, a one-dimensional jumper acquires the ability to learn forcing profiles exerted by its environment in tandem to achieving its control objective. The GP-based dynamical model initially assumes a baseline rigid, non-compliant surface. As part of an iterative procedure, the optimizer employing this model generates an optimal control to achieve a target jump height while respecting known hardware limitations of the robot model. Trajectory and forcing data recovered from evaluation on the true GM surface model simulation is applied to train the GP, and in turn, provide the optimizer a more richly informed dynamical model of the environment. After three iterations, predicted optimal control trajectories coincide with execution results, within 1.2% jumping height error, as the GP-based approximation converges to the true GM model.**

I. Introduction

Engineered mechanical systems are well-characterized through a combination of known design, machining to precise tolerances, and the use of system identification tools. Consequently, optimal control formulations for engineered systems have demonstrated the ability to provide excellent performance for even complex systems, such as legged locomotion [1]–[3]. However, when the underlying dynamics of the controlled system rely on external factors that must be approximated based on empirical models or are simply unknown, then employing optimal control methods leads to degraded performance. Mismatch between the presumed model dynamics and the actual dynamics may lead to a failure to meet critical performance outcomes specified within the optimal control formulation. As robotics researchers strive to create more complex robotic systems that exploit hard to model physical properties, we can expect optimal control methods to break down. Example robotic applications with an expectation of mismatch between the theoretical equations and the actual experience of the mechanical system include legged movement over unknown ground substrates [4], flapping flight [5], [6], and aquatic swimming [7]–[10].

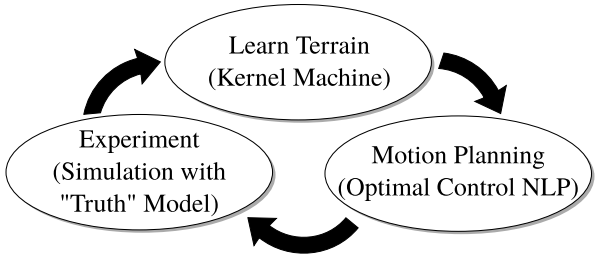

Fig. 1: An overview of the presented approach for robotic jumping on granular media (GM), a task model for locomotion on flowable terrain (e.g. sand and soil). We model the terrain dynamics with a Gaussian process (GP), optimize a robot jumping maneuver with a constrained nonlinear program (NLP), and execute the maneuver on a "true" simulation model of the substrate [11]. The process is repeated until the task is satisfactorily completed; in this case, when a target jump height is achieved.

This paper is motivated by prior work regarding hopping on granular media or soft ground. We demonstrated how the ground substrate forces lead to different optimal control signals for the same control objective [4]. To do so, we relied on meticulously derived and experimentally validated models for the granular media [11] and applied system identification to the actuator. Likewise, related work has involved terrain-tuned controllers to achieve hopping objectives [12]. Both works involve manual effort that preclude adaptive use by autonomously operating bipedal robots. Given that the control synthesis must conform to the actual dynamics experienced by the mechanical system, a means to rapidly estimate online the unknown forcing function due to foot-ground interaction is essential to high performance operation.

Kernel Machines in Robotics. Kernel machines and radial basis function networks for function approximation are often applied to controls and robotics problems due to their universal approximation capabilities. Though computationally complex to implement in the case of online, data-driven operation, modifications to the baseline learning methods provide online, real-time implementable algorithms [13]–[15]. Gaussian processes (GP), in particular, have received attention in the area of reinforcement learning (RL) [16], [17], where their inclusion expedites the learning process by regressing on the unknown model dynamics. RL approaches have been combined with optimal control methods to jointly

1: The authors are with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, USA. Email: alexander.h.chang@gatech.edu, pvela@ece.gatech.edu

2: The authors are with the School of Mechanical Engineering, Georgia Institute of Technology, Atlanta, GA, USA. Email: christian@gatech.edu, ames@me.gatech.edu

3: The authors are with the School of Physics, Georgia Institute of Technology, Atlanta, GA, USA. Email: jeff@gatech.edu, daniel.goldman@physics.gatech.edu

This work was supported by NSF grant CPS#1544857.

learn the model dynamics and identify the optimal control policy to apply [18]. A benefit of GPs in this context is the ability to generate random draws from a function space [19] within the context of stochastic optimal control formulations. The combination of efficient GP learning and sample synthesis methods with stochastic optimal control leads to compelling reinforcement learning strategies [20], [21]. Importantly, these approaches are experiential, data-driven learning methods. Due to the lack of model information, they involve offline training prior to deployment and are often proof-by-demonstration, as they lack associated stability theorems.

Kernel Machines and Adaptive Control. Coming from a more controls oriented perspective, actor-critic networks with experience replay methods have been used to learn optimal policies [22]. These offline-learning-then-online-implemented methods have proven stability and learning properties. Likewise, a method arising from the model reference neuro-adaptive control literature, known as concurrent learning can also learn online and has proven stability and learning properties [23], [24]. Here, proven learning properties means that persistent excitation is shown to hold for the kernel machine regressor dynamics, so that the regression variables of the kernel machines are known to converge exponentially fast. Subsequent efforts created online learning methods using data-driven kernel machines methods that started with nothing and built the model from scratch [25], [26] much like the aforementioned RL methods. To achieve online implementation, data curation and sparsification methods were applied [27], where the data curation is informed by the control task and approximation needs. To preserve the stability and learning guarantees, the method requires a pre-existing baseline controller.

Contribution. We propose a Gaussian process-based dynamical model of one dimensional hopping, accompanied by a deterministic process to simultaneously, iteratively learn an unknown ground substrate forcing profile while optimizing control to achieve a desired objective. Well-understood equations of motion are re-arranged to highlight the unknown substrate forcing and introduce a GP-based component that enables learning. Without prior experiential data, the GP-based model assumes a baseline dynamic of rigid ground hopping. Optimal control generation is restricted to this GP-based model, conveying an initially inaccurate dynamical understanding of the world. In an iterative process, optimal control signals are generated then evaluated against a 'ground truth' granular media (GM) substrate model. The forcing profile exerted by the GM is recovered from resulting trajectory data; defects from the baseline rigid ground dynamics are applied to train the GP which, in turn, conveys an incrementally more accurate understanding of the environment to the optimizer. After a relatively small set of experiential data, the GP-based forcing model converges to the 'ground truth' GM model. Equivalently, the optimizer effectively adapts to modeling uncertainty by iteratively learning the defect in its baseline understanding, incrementally rectifying errors in its performance.

II. Optimal Jumping Controller

Optimal control methods have been central to many successful implementations of legged robotic control [1]–[3]. This is particularly true of dynamic locomotion, where the joint velocities have significant dynamic effects on locomotion success (as opposed to quasi-static approaches). Jumping is an apt example of dynamic locomotion since the velocity at the instant of takeoff is a major requirement of task completion. Further, many jumping robots have unactuated dynamics (e.g. a mechanical spring) around which control of the jumping maneuver needs to be planned. Modern bipeds are also being designed with passive springs [28]. These dynamic features motivate our choice of optimal control methods to achieve robotic jumping. Specifically, we will employ model-based optimizations, which require mathematically modeling the mechanical system.

A. Equations of Motion

The one dimensional jumping model, illustrated in Figure 2 and presented in [11], is modeled by three massed bodies: a linear motor, a thrust rod, and a foot. This robot model jumps by applying force between the motor and thrust rod, which can be leveraged to pump energy into the spring-loaded system. Vertical motor actuation drives rod displacement which, in turn, transmits forcing to the foot through the connecting spring (which is massless, but modeled with linear viscous damping). The closed form system dynamics are

$$\begin{aligned}\ddot{x}_f =& -g + \frac{1}{m_f}\left[k(\alpha-\bar{\alpha})+c\dot{\alpha}\right] + \frac{1}{m_f}F_{sub}\\ \ddot{\alpha} =& -\left[\frac{m_f+m_r+m_m}{m_f(m_r+m_m)}\left(k(\alpha-\bar{\alpha})+c\dot{\alpha}\right)+\frac{m_m}{m_r+m_m}u\right]\\ &-\frac{1}{m_f}F_{sub}\end{aligned} \tag{1}$$

where body masses of the motor, rod, and foot are designated m_m, m_r, and m_f, respectively. The signals x_f, α, and β

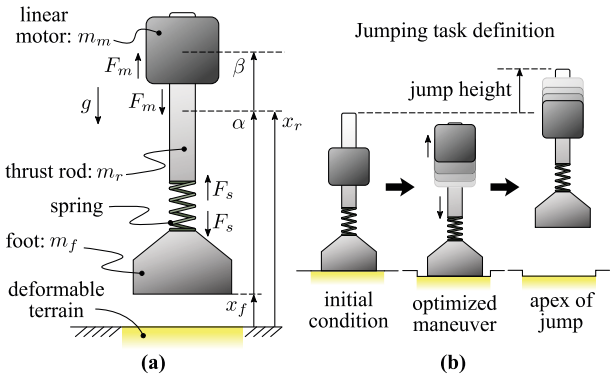

Fig. 2: **(a)** A visualization and labeling of the one-dimensional (linear vertical-only) jumping robot model. **(b)** An illustration of the desired jumping task; the robot must thrust its motor mass in such a manner that it jumps off the ground and achieves a precise desired jump height at its apex.

represent the spatial position of the foot center of mass (CoM), position of the rod CoM relative to the foot CoM, and position of the motor CoM relative to the rod CoM, respectively. Here, motor acceleration $\ddot{\beta}$ serves as the input control signal and has been replaced with u. $\bar{\alpha}$ represents the position of the rod, relative to the foot, when no external forcing, gravity or otherwise, act to compress or extend the spring. F_{sub} denotes substrate reaction forces acting on the foot and is always directed positively upwards.

The system state is defined by, $x = [x_f \ \dot{x_f} \ \alpha \ \dot{\alpha}]^T \in \mathbb{R}^4$.

1) Substrate Force, F_{sub}: Substrate forcing, F_{sub}, resists negative displacement of the foot and is one mechanism by which energy is stored in the spring, ultimately enabling the system to lift off the ground. The model of substrate forcing will vary with substrate type.

a) Flight Phase: During flight, the foot has lifted off the surface and substrate forcing no longer acts upon the system, $F_{sub} = 0$.

b) Contact Phase: When the foot is in contact with the ground, x_f is coincident with the surface height.

In a rigid ground scenario, then, $x_f = 0$. The resultant force due to gravity, spring and damping is directed in the negative, downward direction. Substrate forcing counteracts all other forces acting on the foot. F_{sub} can then be inferred from the first equation in (1), and represented in closed-form as a function of state,

$$F_{sub}^{sg} = m_f g - k(\alpha - \bar{\alpha}) - c\dot{\alpha}, \tag{2}$$

where superscript ^{sg} denotes the assumption of solid, undeformable ground.

When the ground is no longer rigid and is, instead, modeled as deformable granular media, the assumptions used for solid ground scenarios no longer hold. F_{sub}^{gm}, denoting substrate forcing exerted on the foot by granular media, becomes a complex nonlinear function of the hopper state and is dependent upon empirically determined parameters modeling the granular material [11]. It may be decomposed into multiple components,

$$F_{sub}^{gm} = \frac{m_f}{m_f + M_{added}}(F_p + F_v) - \frac{M_{added}}{m_f + M_{added}} F_{sub}^{sg}, \tag{3}$$

where F_p represents a quasi-static forcing component dependent on foot depth, F_v represents a depth- and velocity-dependent component and M_{added} denotes velocity-dependent added mass to the foot as it intrudes into the granular media.

B. Solid Ground and Granular Media: A Comparison

Deformation of the ground substrate induces a significant change in the dynamics of the one-dimensional hopper. Figure 3 illustrates, in simulation, the distinct hopper trajectories that evolve assuming a solid ground (solid) versus granular media (dashed) surface. The forcing profile exerted by each surface type is illustrated in the bottom plot of the figure. Identical open-loop sinusoidal control signals have been applied to both models. Prior to leaving solid ground, the foot position, x_f, does not change. On granular media, however, x_f sinks deeper into the granular material prior to take-off. Surface deformation dissipates energy, robbing it from the spring and ultimately leading to a lower peak hop height. Forcing profiles associated with each surface additionally reveal distinctions in both shape as well as peak value.

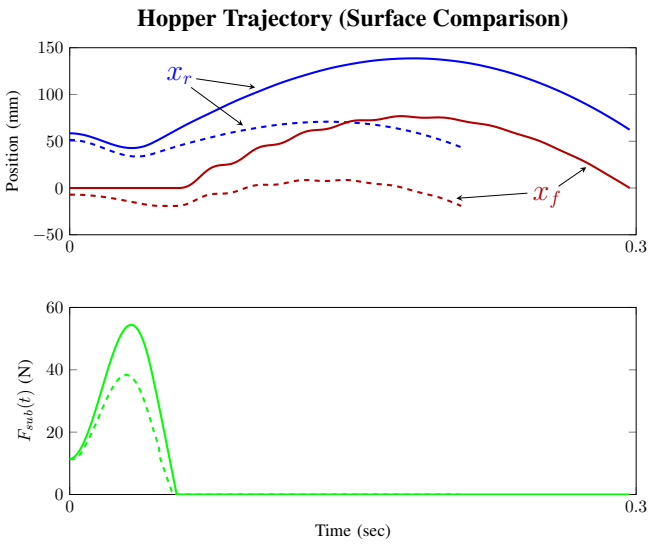

Fig. 3: Trajectory (**top**) and substrate forcing (**bottom**) associated with a single hop of a one-dimensional hopper, beginning from rest. **Solid Ground (solid):** Initial state of the hopper, $x_0 = [0 \text{ mm } 0 \text{ mm/s } 0 \text{ mm } 0 \text{ mm/s}]^T$. Due to the rigid nature of the surface, F_{sub}^{sg} will always negate gravity, spring and damping forces when the hopper is at rest and $x_f = 0$. **Granular Media (dashed):** Initial state of the hopper, $x_0 = [-7.086 \text{ mm } 0 \text{ mm/s } 0 \text{ mm } 0 \text{ mm/s}]^T$. $x_f(0)$ is the equilibrium depth at which quasi-static granular media, gravity, spring and damping forces cancel.

C. Optimal Control

The presented method takes a trajectory optimization approach to optimal control [29]. Specifically, we use direct collocation to transcribe the optimal control problem into a large-scale NLP, a method which improves the reliability of algorithm convergence [30]. Further, direct collocation approaches allow us to compute all objectives, constraints, and derivatives thereof in closed-form. A closed-form representation allows for faster computation, which is an important requirement for applications where we aim to adjust behaviors on-the-fly. For those unfamiliar with trajectory optimization methods, here we present an abbreviated explanation of the methods in [4] along with the appropriate modifications for this study.

We begin by rearranging our system dynamics into a typical set of first-order ODE's, $\dot{x} = f(t, x, u(t))$, where t, x, and $u(t)$ are time, state, and time-varying control inputs respectively. In collocation methods, we then discretize the optimal control problem, allowing the optimizer to operate on a discrete set of variables defining a piecewise trajectory. This includes a discrete time tape, t_i, where $0 = t_0 < t_1 < t_2 < \cdots < t_N = T$, state variables, x_i, and control inputs,

u_i, where $i \in \{1, 2, 3, \ldots, N-1\}$, and includes duplicate variables for each dynamical domain (e.g. contact and flight phases). The variables t_i, x_i, u_i, will all be relegated to free variables for the optimization to design. T is the duration of each domain and also a free design variable. We used $N = 25$ uniformly spaced discrete points, and finer grids did not significantly change the optimal solution.

Each discrete point, i, is rendered dynamically consistent with the next point, $i+1$, via "defect" constraints in the NLP, which approximate implicit integration:

$$(x_{i+1} - x_i) - \frac{1}{2}(t_{i+1} - t_i)(\dot{x}_{i+1} + \dot{x}_i) = 0, \quad (4)$$

$$\dot{x}_i - f(t_i, x_i, u_i) = 0, \quad (5)$$

$\forall i$, with the above constraint encoding a trapezoidal integration scheme. While Hermite-Simpson methods are also common [31], for the hopping application, trapezoidal methods have been sufficiently accurate to achieve satisfactory matches with experimental hopping results [4]. We now build a vector $\mathbf{w}$ of all optimization variables, $\mathbf{w} = \{x_i, \dot{x}_i, u_i\}$, and define the objective,

$$\mathbf{J}(\mathbf{w}) = \frac{t_2 - t_1}{2} J(x_1, \dot{x}_1, u_1) + \quad (6)$$

$$\sum_{i=2}^{N-1} \frac{t_{i+1} - t_{i-1}}{2} J(x_i, \dot{x}_i, u_i) + \frac{t_N - t_{N-1}}{2} J(x_N, \dot{x}_N, u_N)$$

where $J(x, \dot{x}, u(t))$ is the integrand of a continuous-time cost. The resulting NLP becomes simply:

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}} \ \mathbf{J}(\mathbf{w}) \quad (7)$$

$$\text{s.t} \quad \mathbf{w}_{\min} \leq \mathbf{w} \leq \mathbf{w}_{\max}, \quad (8)$$

$$\mathbf{c}_{\min} \leq \mathbf{c}(\mathbf{w}) \leq \mathbf{c}_{\max}, \quad (9)$$

in which $\mathbf{c}(\mathbf{w})$ represents the concatenation of all constraint functions, including defect constraints (4-5) and additional task constraints[1] (see [4] for additional constraint details). To facilitate faster solving, we export symbolic representations of $\mathbf{J}(\mathbf{w})$, $\delta\mathbf{J}(\mathbf{w})/\delta\mathbf{w}$, $\mathbf{c}(\mathbf{w})$, and $\delta\mathbf{c}(\mathbf{w})/\delta\mathbf{w}$ to the large-scale interior-point solver, IPOPT [32], using a MATLAB-based NLP parsing framework [33]. Note that this trick is possible because $f(t, x, u(t))$ and $J(x, \dot{x}, u(t))$ can be written symbolically.

In a previous study, the dynamics governing granular media were given *a priori* from physics-based models and measurement [4]. Here, we replace this measured model with a Gaussian process model. Since Gaussian processes can similarly be written as closed form expressions, we hypothesize that optimal control formulations with GP's will be similarly tractable.

III. Online Learning of External Forcing

This section describes the GP-based online learning process. For simplicity, we will use the full GP equations. The actual online implementation should follow the data curation and sparsification approaches described in [25]–[27].

[1]Such as jumping to the correct height, not exceeding actuator force limits, enforcing workspace/acceleration limits, etc.

A. Gaussian Processes

Gaussian processes (GPs) present a Bayesian approach to function regression and modeling over potentially high-dimensional domains, and in the presence of a limited set of function evaluations or training data [34]. Given a set of (possibly noisy) evaluations of an unknown function, $\tau(x) : \mathbb{R}^n \to \mathbb{R}$, a GP represents a prior over functions,

$$Z(x) \sim \text{GP}(m(x), \kappa), \quad (10)$$

where $Z(x) : \mathbb{R}^n \to \mathbb{R}$ represents the function regression, $m(x)$ is the mean function and $\kappa(x, x')$ is a kernel function describing covariance. We choose to employ a squared exponential kernel,

$$\kappa(x, x') = \exp\left(-\frac{1}{2}(x - x')^T \Sigma (x - x')\right) + \sigma_{damp}\mathbb{I}, \quad x, x' \in \mathbb{R}^n, \quad (11)$$

which provides a measure of similarity between any two elements, x and x', in the domain of $\tau(x)$ based on L_2 proximity. Σ is a bandwidth parameter influencing the scale, over the domain, with which the function regression varies, while σ_{damp} represents variance associated with noise in the training data. The latter may be interpreted as a damping factor to tune for over-fitting in regression.

In particular, we approximate an unknown function, $\tau(x)$, by the model, $y(x) = \mu(x) + Z(x)$, where $\mu(x)$ captures a global trend from the training data. The global trend is represented as a linear combination of a set of basis functions, $f(x) = \{1 \ f_1(x) \ f_2(x) \ \ldots \ f_m(x)\}^T$, such that $\mu(x) = f(x)^T\Theta$, where $\Theta \in \mathbb{R}^m$ are linear coefficients. $Z(x)$ depicts a non-stationary GP representing regression over any residual deviations from $\mu(x)$.

The GP-based function approximation is more specifically represented by [35],

$$y(x) = \mu(x) + r^T(x)R^{-1}(\hat{y} - \mu(\hat{x})). \quad (12)$$

$\hat{x} = [\hat{x}_1 \ \hat{x}_2 \ \ldots \ \hat{x}_p]^T$, where $\hat{x}_i \in \mathbb{R}^{\text{n}}$ and $\hat{y} \in \mathbb{R}^{\text{p}}$, comprises a set of p training samples; evaluation of each element of $\hat{x}$ by the unknown function has yielded $\hat{y}_i = \tau(\hat{x}_i)$, possibly with noise. R is the covariance matrix such that $R_{i,j} = \kappa(\hat{x}_i, \hat{x}_j)$ and $r^T(x) = [\kappa(x, \hat{x}_1) \ \kappa(x, \hat{x}_2) \ \ldots \ \kappa(x, \hat{x}_p)]^T$.

B. Learning Ground Forcing as a Function of State

To construct a learning-enabled dynamical model of one dimensional hopping, we begin with (1). However, ground forcing, F_{sub}, will instead be formulated as the summation of multiple components, including contribution from a GP,

$$F_{sub}(x) = F_{sub}^{sg}(x) + \mu(x) + r^T(x)R^{-1}(\hat{y} - \mu(\hat{x})), \quad (13)$$

where again $\hat{x}$ and $\hat{y}$ comprise training data; $\hat{x}_i \in \mathbb{R}^4$ and $\hat{y}_i \in \mathbb{R}$ together comprise a training tuple, identifying a point in the state space and the corresponding ground force experienced, respectively.

$F_{sub}^{sg}(x)$ is defined by (2). When no training data has been collected, the latter two components of (13) vanish and the model reduces to the baseline solid ground assumption. As

training data is collected, the latter two terms of (13) model forcing components of the terrain that represent defects from a rigid surface environment.

A global linear trend, $\mu(x)$, is extracted from the training data, with residuals from this global trend modeled by the GP in the final term of (13). To compute, $\mu(x)$, let $f(x) = [1\ x_f\ \dot{x}_f\ \alpha\ \dot{\alpha}]^T$. Then the corresponding linear coefficients are computed in closed form from a weighted least-squares fit,

$$\Theta = (F^T R^{-1} F)^{-1}(F^T R^{-1}\hat{\Phi}), \tag{14}$$

where, $\hat{\Phi} = [(\hat{y}_1 - F^{sg}_{sub}(\hat{x}_1)\ \ldots\ (\hat{y}_p - F^{sg}_{sub}(\hat{x}_p))]$. $F = [f(\hat{x}_1)\ f(\hat{x}_2)\ \ldots\ f(\hat{x}_p)]^T$ and p denotes cardinality of the training set [35].

C. State Smoothing

Ground forcing, F_{sub}, applied during the course of the hopper trajectory must be recovered to train the GP-based model of (13). Any direct sensing of this quantity is assumed absent. Through recovery of the relevant quantities from the first equation of (1), however, F_{sub} can be computed. We assume the state of the system, $x = [x_f\ \dot{x}_f\ \alpha\ \dot{\alpha}]$, is completely observable. In particular, however, acceleration associated with the foot, $\ddot{x}_f$, is not immediately and conveniently accessible. To recover this quantity, trajectory data measured from the dynamical system (1) is subjected to a fixed-lag Kalman smoothing process, which will incur some time delay. The output of this process is a smoothed state trajectory, $x^s(t) = \left[x^s_f(t)\ \dot{x}^s_f(t)\ \ddot{x}^s_f(t)\alpha^s(t)\ \dot{\alpha}^s(t)\ \ddot{\alpha}^s(t)\right]^T$, which yields the missing quantity $\ddot{x}^s_f$. Substrate forcing is then estimated as a function of the smoothed system state,

$$F_{sub} = m_f g - [k(\alpha^s(t) - \bar{\alpha}) + c\dot{\alpha}^s(t)] + m_f \ddot{x}^s_f(t). \tag{15}$$

D. Iterative Optimal Control and Learning

We apply an iterative procedure to recover a GP-based dynamical model of the one dimensional hopper system, described jointly by (1) and (13), simultaneously hopping on an unknown surface while learning the forcing profile that surface exerts as a function of the system state, x. From a validation perspective, the unknown surface is modeled by granular media with known environmental parameters; this granular media model serves as our ground truth against which to compare the evolution and accuracy of the GP-based dynamical model that is iteratively trained.

Prior to any experimental attempts, the set of training data is empty, $\Gamma = \emptyset$. Each learning iteration begins with the hopper in an initial state, at rest on the hopping surface. An optimizer then employs the GP-based dynamical model, described by (1) and (13), to generate an optimal control signal enabling a target peak hop height to be attained while minimizing actuation effort. The control signal is subsequently evaluated on a one dimensional hopper operating on the ground truth surface, whose complex, nonlinear forcing profile is described by a granular media model with known parameters. If target peak hop height is not attained using the generated control signal on the ground truth GM surface, trajectory data is collected, smoothed and substrate forcing, $F_{sub}(x^s(t))$, is recovered. The training set, Γ, is augmented with the tuples, $(x^s(t), F_{sub}(x^s(t)))$ for all t discretely measured and computed. This training set is limited to trajectories, and corresponding forces, occurring during contact with the ground, prior to the hopper attaining flight.

$\Gamma = (\hat{x}_i, \hat{y}_i) : \forall i = 1 \ldots p$, with cardinality p, is subsequently applied to update the GP-based model of substrate forcing. First, the deviation of $F_{sub}(\hat{x})$ from the equivalent solid ground forcing profile, $F^{sg}_{sub}(\hat{x})$, is computed and a global linear trend is extracted, using (14) to compute the linear coefficients. The GP is trained on the remaining residual, $F_{sub}(\hat{x}) - F^{sg}_{sub}(\hat{x}) - \mu(\hat{x})$, and represents a regression estimate of this quantity over the state space. This process then repeats, with the optimizer informed by a newly trained GP-based approximation of substrate forcing. Algorithm 1 sketches the iterative optimal control and learning process.

Algorithm 1 Iterative Hopper Control and Learning

1: **procedure** CONTROL AND LEARN(targetHopHeight)
2: training set, $\Gamma \leftarrow \emptyset$
3: currentHopHeight $\leftarrow 0$
4: initialize GP
5: **while** currentHopHeight $\neq$ targetHopHeight **do**
6: generate optimal control, u_{opt}, using GP-based model, (1) and (13)
7: evaluate u_{opt} on ground truth GM surface
8: extract hopper state trajectory, $x(t)$
9: currentHopHeight $\leftarrow \max(x_f(t) + \alpha(t))$
10: apply Kalman smoothing to recover x^s
11: compute F_{sub} using (15)
12: append Γ with new data: (x^s, F_{sub})
13: $\forall(\hat{x}_i, \hat{y}_i) \in \Gamma$, compute $\hat{y}_i - F^{sg}_{sub}(\hat{x}_i)$
14: update $\mu(x)$ using (14)
15: train GP with data: ($\hat{x}_i$, $F_{sub}(\hat{x}_i) - F^{sg}_{sub}(\hat{x}_i) - \mu(\hat{x}_i)$)
16: **end while**
17: **end procedure**

IV. RESULTS

Validation of the simultaneous control and learning approach, detailed in Algorithm 1, is demonstrated in simulation. We challenge the hopper to learn the forcing profile associated with an unknown granular media surface. In an iterative process, it generates optimal control signals to hop to an arbitrarily selected target hop height using its current GP-based model of the world. The results from exercising that control signal on a ground truth model of the GM environment then informs the GP-based approximation of state-dependent substrate forcing. Line 7 of Algorithm 1 is accomplished in simulation using an *ode45* numerical integrator. `MATLAB` is employed for all implementation and computation of the process delineated in Algorithm 1.

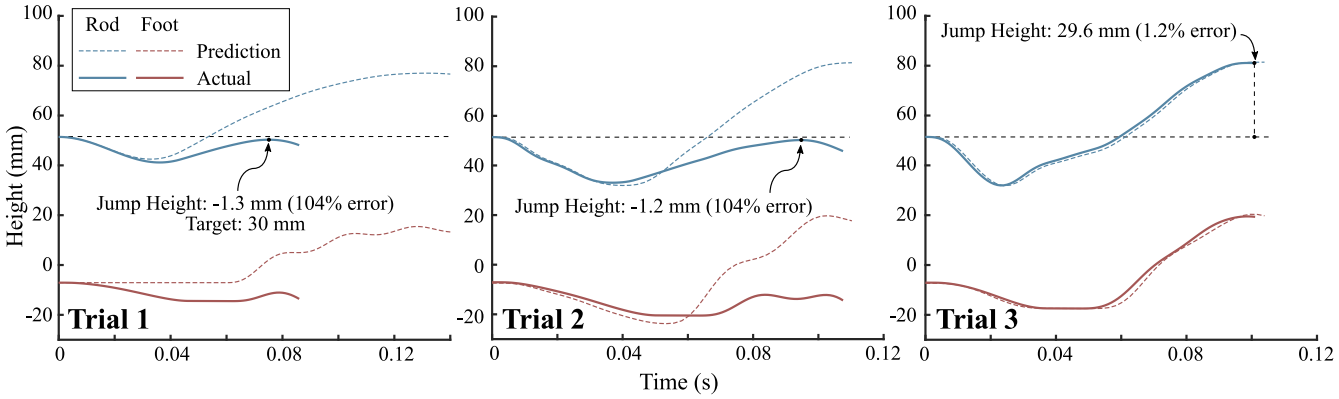

Fig. 4: Execution of the GP/motion planning jumping experiment with the goal of achieving a 30mm hop height (as measured by the rod position). In three iterations of learning → planning → jumping, the simulated jumping task improves in accuracy from 104% error to 1.2% error.

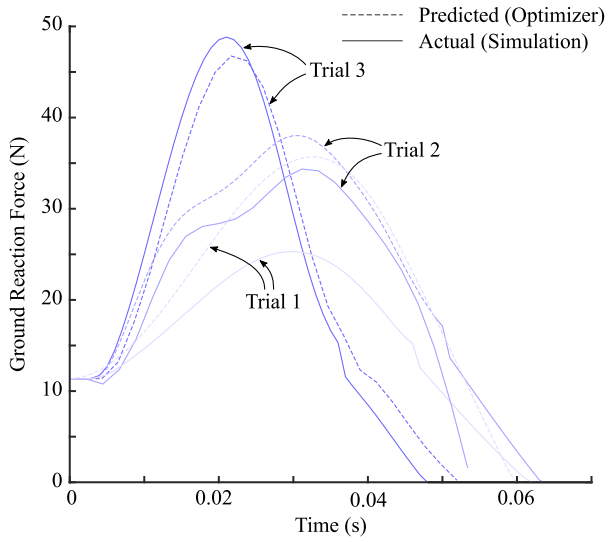

Fig. 5: Evolution of the ground-reaction force predition across three trials. Each trial updates the Gaussian process model and a new strategy is computed by the motion planning optimizer. By the third trial, the actual experienced force sufficiently matches the GP prediction to achieve a jump apex with 1.2% error.

A. Optimization

The optimization routine is tasked with returning a locally optimal control trajectory given a commanded hop height (30 mm) and a GP external forcing model. In this case, the optimization returns a vector of β^i (a motor trajectory β corresponding to each discrete time step, where $N = 25$ each in the flight and contact phases). We define our objective integrand to be $J(x^i, \dot{x}^i, u^i) = (\dot{\beta}^i)^2$, which penalizes actuator velocity as both a proxy for energy cost and favors a more-easily trackable position trajectory on hardware. One could substitute instanteous power as the objective, however, such power minimizing solutions tend to rail actuator forces suddenly between zero and non-zero values. Such "banging" strategies are typically more sensitive to timing errors in hardware implementation.

The GP forcing model F_{sub} is converted to a symbolic expression, substituted into the system dynamics for symbolic differentiation, and converted into a set of constraints for the NLP. IPOPT solves the NLP to a tolerance of 10^{-9}, for both feasibility and duality. Each optimization converged to an optimal solution in less than 10 seconds.

B. Simulation

After each optimization, the output motor trajectory is simulated in a variable-time-step "truth" model and compared to the optimizer's prediction. Figure 4 shows the iterations, from Algorithm 1, until the target jump height is reached (less than 1 mm error). After three iterations, the GP model had been sufficiently trained such that the optimizer generated a successful, accurate jump (with 0.4 mm or 1.2% error). This is a significant improvement over the initial model, which yielded a 31.3 mm error (or 104%), showing that significant practical improvements can occur over relatively few iterations, when the problem and observations are formulated to emphasize the uncertainty.

In terms of task completion, the course of improvement was not gradual or smooth. In fact, iteration 2 showed nearly the same jumping performance as iteration 1 in terms of effective jump height. However, the underlying difference can be seen in Figure 5, where the GP estimate of the forcing has converged closer to the true model. Thereafter, just a third iteration is sufficient to cut the jumping error from 104% to 1.2%. This behavior underscores how a learned model approach can yield quickly adaptive behaviors in robotic applications, for a targeted task.

V. Conclusion

Introducing a GP representation of an uncertain quantity into the dynamical model of a vertical hopper provides a means to iteratively learn characteristics of its environment in tandem with refining optimization processes for achieving its control objectives. Because the uncertain GP model is formulated to target the uncertain components, as is done in neuro-adaptive control, a relatively small number of iterations is required for the GP-based model of substrate forcing

to converge to the true GM model. Simultaneously, defects when executing the generated optimal controls also vanish as the optimizer becomes better informed through a GP trained on incrementally larger and diversified training sets.

The utility of integrating a GP into the modeling and optimization process is demonstrated in §IV. After three iterations, evaluation of the generated optimal control on the GM surface meets the execution expectations of the optimizer. Simultaneously, we find the GP-based approximation of ground forcing has quickly converged to that of the ground truth GM model. This is particularly significant given the model began with only the baseline assumption of a solid ground environment and quickly bridged the gap between these very distinct dynamical models of the environment; a gap that is comparatively illustrated by Figure 3.

In the study of legged robotic locomotion, we find different substrates require diverging control strategies, which are computed in cases where terrain parameters are known a priori. In practice, however, a robot likely lacks the ability and opportunity to methodically characterize properties prior to task execution. An ability to both learn and exploit the terrain dynamics in tandem with the process of locomotion, where every step is an experiment, presents great utility to legged mobility over uncertain and challenging terrain.

REFERENCES

[1] E. Westervelt, J. W. Grizzle, and D. E. Koditschek, "Hybrid Zero Dynamics of Planar Biped Walkers," *IEEE Trans. on Automatic Control*, vol. 48, no. 1, pp. 42–56, Jan 2003.

[2] I. R. Manchester, U. Mettin, F. Iida, and R. Tedrake, "Stable dynamic walking over uneven terrain," *International Journal of Robotics Research*, vol. 30, no. 3, pp. 265–279, 2011.

[3] A. D. Ames, "Human-inspired control of bipedal walking robots," *IEEE Trans. on Automatic Control*, vol. 59, no. 5, pp. 1115–1130, May 2014.

[4] C. M. Hubicki, J. J. Aguilar, D. I. Goldman, and A. D. Ames, "Tractable terrain-aware motion planning on granular media: An impulsive jumping study," in *IEEE/RSJ IROS*, Daejon, Korea, 2016.

[5] H. E. Taha, C. A. Woolsey, and M. R. Hajj, "Geometric control approach to longitudinal stability of flapping flight," *Journal of Guidance, Control, and Dynamics*, vol. 39, no. 2, pp. 214–226, 2016.

[6] A. Ramezani, X. Shi, S.-J. Chung, and S. Hutchinson, "Lagrangian modeling and flight control of articulated-winged bat robot," in *IEEE/RSJ IROS*, Hamburg, Germany, 2015, pp. 2867–2874.

[7] S. D. Kelly, P. Pujari, and H. Xiong, "Geometric mechanics, dynamics, and control of fishlike swimming in a planar ideal fluid," in *Natural Locomotion in Fluids and on Surfaces: Swimming, Flying, and Sliding*, S. Childress, A. Hosoi, W. W. Schultz, and J. Wang, Eds. Springer, 2012, pp. 101–116.

[8] K. A. Morgansen, B. I. Triplett, and D. J. Klein, "Geometric methods for modeling and control of free-swimming fin-actuated underwater vehicles," *IEEE Trans. on Robotics*, vol. 23, no. 6, pp. 1184–1199, 2007.

[9] M. Porez, F. Boyer, and A. J. Ijspeert, "Improved lighthill fish swimming model for bio-inspired robots: Modeling, computational aspects and experimental comparisons," *International Journal of Robotics Research*, vol. 33, no. 10, pp. 1322–1341, 2014.

[10] J. Wang and X. Tan, "Averaging tail-actuated robotic fish dynamics through force and moment scaling," *IEEE Trans. on Robotics*, vol. 31, no. 4, pp. 906–917, 2015.

[11] J. Aguilar and D. I. Goldman, "Robophysical study of jumping dynamics on granular media," *Nature Physics*, vol. 12, pp. 278–184, 2016.

[12] W. Boswroth, J. Whitney, S. Kim, and N. Hogan, "Robot locomotion on hard and soft ground: Measuring stability and ground properties in-situ," in *IEEE ICRA*, 2016, pp. 3582–3589.

[13] A. Gijsberts and G. Metta, "Real-time model learning using incremental sparse spectrum gaussian process regression," *Neural Networks*, vol. 41, pp. 59–69, May 2013.

[14] D. Nguyen-Tuong, B. Schölkopf, and J. Peters, "Sparse online model learning for robot control with support vector regression," in *IEEE/RSJ IROS*, Oct. 2009, pp. 3121–3126.

[15] S. Vijayakumar, A. D'Souza, and S. Schaal, "Incremental online learning in high dimensions," *Neural Computation*, vol. 17, no. 12, pp. 2602–2634, Dec 2005.

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

[17] S. Levine and P. Abbeel, "Learning neural network policies with guided policy search under unknown dynamics," in *Advances in Neural Information Processing Systems*, 2014, pp. 1071–1079.

[18] J. Boedecker, J. T. Springenberg, J. Wuelfing, and M. Riedmiller, "Approximate real-time optimal control based on sparse gaussian process models," in *IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, Orlando, FL, 2014, pp. 1–8.

[19] Y. Pan and E. A. Theodorou, "Data-driven differential dynamic programming using gaussian processes," in *American Control Conference*, 2015, pp. 4467–4472.

[20] Y. Pan, X. Yan, E. Theodorou, and B. Boots, "Scalable reinforcement learning via trajectory optimization and approximate Gaussian process regression," in *NIPS Workshop on Advances in Approximate Bayesian Inference*, 2015.

[21] ——, "Adaptive probabilistic trajectory optimization via efficient approximate inference," *CoRR*, vol. abs/1608.06235, 2016.

[22] H. Modares, F. L. Lewis, and M.-B. Naghibi-Sistani, "Integral reinforcement learning and experience replay for adaptive optimal control of partially-unknown constrained-input continuous-time systems," *Automatica*, vol. 50, no. 1, pp. 193–202, Jan 2014.

[23] G. Chowdhary, "Concurrent learning for convergence in adaptive control without persistency of excitation," Ph.D. dissertation, Atlanta, GA, 2010.

[24] G. Chowdhary, T. Yucelen, M. Mhlegg, and E. Johnson, "Concurrent learning adaptive control of linear systems with exponentially convergent bounds," *International Journal of Adaptive Control and Signal Processing*, vol. 27, no. 4, pp. 280–301, 2013.

[25] H. Kingravi, G. Chowdhary, P. Vela, and E. Johnson, "Reproducing kernel Hilbert space approach for the online update of radial bases in neuro-adaptive control," *IEEE Trans. on Neural Networks and Learning Systems*, vol. 23, no. 7, pp. 1130 –1141, 2012.

[26] G. Chowdhary, H. Kingravi, J. How, and P. Vela, "Bayesian nonparametric adaptive control using Gaussian processes," *IEEE Trans. on Neural Networks Learning Systems*, vol. 26, no. 3, pp. 537–550, 2014.

[27] H. Kingravi, "Reduced-set models for improving the training and execution speed of kernel methods," Ph.D. dissertation, Georgia Institute of Technology, 2014.

[28] J. Reher, E. Cousineau, A. Hereid, C. Hubicki, and A. Ames, "Realizing dynamic and efficient bipedal locomotion on the humanoid robot durus," in *IEEE ICRA*, Stockholm, Sweden, 2016.

[29] C. R. Hargraves and S. W. Paris, "Direct trajectory optimization using nonlinear programming and collocation," *Journal of Guidance, Control, and Dynamics*, vol. 10, no. 4, pp. 338–342, 1987.

[30] A. Rao, "A survey of numerical methods for optimal control," *Advances in the Astronautical Sciences*, vol. 135, no. 1, pp. 497–528, 2009.

[31] A. Hereid, E. A. Cousineau, C. M. Hubicki, and A. D. Ames, "3D Dynamic Walking with Underactuated Humanoid Robots: A Direct Collocation Framework for Optimizing Hybrid Zero Dynamics," in *IEEE ICRA*, 2016, pp. 1–8.

[32] A. Wächter, L. Biegler, Y. Lang, and A. Raghunathan, "Ipopt: An interior point algorithm for large-scale nonlinear optimization," 2002.

[33] M. S. Jones, "Optimal Control of an Underactuated Bipedal Robot," Ph.D. dissertation, 2014.

[34] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. Massachusetts Institute of Technology, 2012.

[35] S. Ba and V. R. Joseph, "Composite gaussian process models for emulating expensive functions," *The Annals of Applied Statistics*, vol. 6, no. 4, pp. 1838–1860, 2012.