Learning Terrain Dynamics: A Gaussian Process Modeling and Optimal Control Adaptation Framework Applied to Robotic Jumping

Alexander H. Chang, *Member, IEEE*, Christian M. Hubicki, *Member, IEEE*, Jeffrey J. Aguilar, Daniel I. Goldman, Aaron D. Ames, *Senior Member, IEEE*, and Patricio A. Vela, *Member, IEEE*

***Abstract*— The complex dynamics characterizing deformable terrain presents significant impediments toward the real-world viability of locomotive robotics, particularly for legged machines. We explore vertical, robotic jumping as a model task for legged locomotion on presumed-uncharacterized, nonrigid terrain. By integrating Gaussian process (GP)-based regression and evaluation to estimate ground reaction forces as a function of the state, a 1-D jumper acquires the capability to learn forcing profiles exerted by its environment in tandem with achieving its control objective. The GP-based dynamical model initially assumes a baseline rigid, noncompliant surface. As part of an iterative procedure, the optimizer employing this model generates an optimal control strategy to achieve a target jump height. Experiential data recovered from execution on the true surface model are applied to train the GP, in turn, providing the optimizer a more richly informed dynamical model of the environment. The iterative control-learning procedure was rigorously evaluated in experiment, over different surface types, whereby a robotic hopper was challenged to jump to several different target heights. Each task was achieved within ten attempts, over which the terrain's dynamics were learned. With each iteration, GP predictions of ground forcing became incrementally refined, rapidly matching experimental force measurements. The few-iteration convergence demonstrates a fundamental capacity to both estimate and adapt to unknown terrain dynamics in application-realistic time scales, all with control tools amenable to robotic legged locomotion.**

***Index Terms*— Gaussian process (GP), learning, optimal control, robotic jumping, terrain dynamics.**

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, employing optimal control methods leads to degraded performance. A mismatch between the presumed model dynamics and the actual dynamics may prevent the satisfaction of 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. Such robotic applications prone to 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]. This particular work focuses on vertical, robotic hopping, as a preliminary model task for legged locomotion, where terrain dynamics that drive locomotion are presumed to be poorly modeled or unknown.

A. Review of Related Work

1) Locomotion on Soft Terrain: Control approaches addressing legged locomotion over soft terrain often relegate surface deformation effects to uncertainty terms in the controller or presume simplifying, closed-form approximations of the terrain model (e.g., spring damper) [11]–[14]. Strategies accommodating terrain variation may then apply sensor-driven switching between terrain-tuned controllers to achieve task objectives [15]. These approaches, however, are not well suited to achieving high-performance control objectives when the underlying terrain is arbitrarily soft or exhibits complex nonlinear behavior.

Prior work regarding robotic jumping on soft ground granular media (GM) alternatively employed optimal trajectory synthesis, in conjunction with an empirically tuned model of

Manuscript received November 30, 2019; revised March 21, 2020; accepted July 1, 2020. Manuscript received in final form July 13, 2020. This work was supported by NSF under Grant CPS#1544857. Recommended by Associate Editor A. Chakrabortty. *(Corresponding author: Alexander H. Chang.)*

Alexander H. Chang and Patricio A. Vela are with the Institute for Robotics and Intelligent Machines, Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: alexander.h.chang@gatech.edu; pvela@ece.gatech.edu).

Christian M. Hubicki is with the Department of Mechanical Engineering, Florida A&M University–Florida State University, Tallahassee, FL 32310 USA (e-mail: hubicki@eng.famu.fsu.edu).

Jeffrey J. Aguilar and Daniel I. Goldman are with the School of Physics, Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: jjaguilar1@gmail.com; daniel.goldman@physics.gatech.edu).

Aaron D. Ames is with the Department of Mechanical and Civil Engineering, California Institute of Technology, Pasadena, CA 91125 USA (e-mail: ames@caltech.edu).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCST.2020.3009636

terrain forcing, to achieve precise control objectives [4]. As generalized, terradynamic models do not exist (in contrast to fluid systems, for example, which can be modeled generally via Navier–Stokes equations), we relied on meticulously derived and experimentally validated models for the GM [16], [17] and applied system identification to the actuator. We demonstrated that distinct ground substrates might lead to diverging optimal control signals for identical control objectives [4]; variation of even the GM ground model parameters will influence different control solutions to accomplish the same task [16]. Although control strategies may be computed in cases where terrain parameters are known *a priori*, in practice, a robot likely lacks the ability and opportunity to methodically characterize terrain properties prior to task execution. Manual effort to do so precludes adaptive use by autonomously operating systems, such as bipedal robots.

Given that the control synthesis must conform to the actual dynamics experienced by the mechanical system, a means to rapidly estimate (or learn) online the unknown forcing function due to foot–ground interaction is essential to high-performance operation.

2) Kernel Machines in Robotics: Kernel machines and radial basis function networks for function approximation are often applied to control 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 [18]–[20]. Gaussian processes (GPs), in particular, have received attention in the area of reinforcement learning (RL) [21], [22], 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 learn the model dynamics and identify the optimal control policy to apply [23]. A benefit of GPs, in this context, is the ability to generate random draws from a function space [24] within the context of stochastic optimal control. The combination of efficient GP learning and sample synthesis methods with stochastic optimal control leads to compelling RL strategies [25], [26]. Importantly, these approaches are experiential, data-driven learning methods. Due to the lack of model information, however, they involve offline training prior to deployment. If deployed online with no prior knowledge, learning typically involves on the order of hundreds or more examples due to the tension between exploration and exploitation.

3) Kernel Machines and Adaptive Control: From a more controls oriented perspective, actor-critic networks with experience replay methods have been used to learn optimal policies [27]. These offline-learned-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 [28], [29]. 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 machine methods that started with nothing and built the model from scratch [30], [31] much like the aforementioned RL methods. To achieve online implementation, data curation and sparsification methods were applied [32], where the data curation was informed by the control task and approximation needs. The method requires a preexisting baseline controller to preserve stability and learning guarantees.

4) Gaussian Process Regression: GP-based regression is one strategy applied to learn uncertainty online. Its utility in modeling uncertain dynamical elements, stemming from complex robot-terrain interactions, has found application in different classes of mobile platforms [33], [34].

GPs applied to learn substrate forcing, in the context of jumping on a particular simulated GM model, previously suggested that online learning would occur rapidly [35]. As opposed to learning the dynamics of the entire system from scratch, online learning targeted the unknown external forcing component. This data-driven approach facilitated the incremental accumulation of knowledge, ultimately permitting estimation of the uncertain element driving the system's dynamics and follow-on model-based optimal control. The underlying strategy of targeted learning of unknown dynamical quantities was demonstrably practical while being more dimensionally tractable than learning the system dynamics in their entirety. The nonparametric nature of a GP frees it from any *a priori* structural presumptions; we show its application as a regression element, in this particular work, provides the flexibility to learn complex dynamical profiles associated with various categorically distinct terrain.

B. Target Problem and Contribution

1) Robotic Jumping, a Model Task: Jumping frequently functions as a model task for legged locomotion as it encompasses similar mathematical (and physical) challenges. Akin to bipedal running, it requires changes in contact modes (hybrid system dynamics) and control through periods of underactuation (flight phases). The control-learning approach [35] presented here uses optimal control methods for hybrid systems, similar to those at the core of many model-based legged control techniques [2], [36]. The particular application we explore entails learning of presumed-unknown terrain dynamics that influence vertical, robotic jumping outcomes. The strategy, however, potentially extends to more complex, higher degree of freedom (DoF) legged systems as well [37], where control feasibility continues to be impacted by modeling uncertainty associated with complex and unknown robot-terrain interactions. Ground forcing, in this context, must be learned quickly with respect to several DoFs describing robot-terrain motion, in lieu of just vertical motion. Safety measures guarding against mechanical instability may guide exploratory control and learning of unmodeled quantities [38], [39]. Evaluation in more deployment-realistic scenarios, where terrain dynamics may transition drastically, warrants adoption of supplemental strategies to manage novel encounters [40], [41].

2) Contribution: We extend a previously proposed GP-based dynamical model of 1-D jumping, accompanied by a deterministic process to iteratively learn an unknown ground

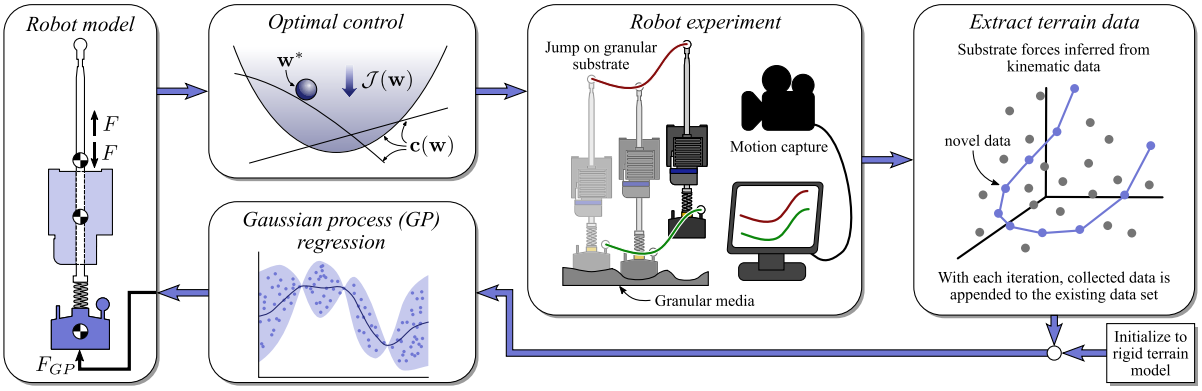


Fig. 1. Flowchart illustrating an iterative optimal control-learning framework. With each jump, the algorithm regresses an updated terrain model and then uses the learned model to generate and execute a new optimal control signal.

forcing profile, all while optimizing control to achieve the desired objective [35]. Whereas the original work involved simulation, the extensions involve experimental validation that the strategy will work in practice by showing reproducible outcomes on three different classes of terrain.

Well-understood equations of motion are rearranged to highlight the presumed-unknown terrain forcing and introduce a GP-based component that enables learning. Without prior experiential data, the GP-based model assumes a baseline rigid ground dynamical representation of the terrain. Optimal control generation is driven by this GP-based model, which initially conveys an inaccurate dynamical understanding. In an iterative process, as shown in Fig. 1, optimal control signals are generated and then evaluated on the true ground model. The forcing profile exerted by the terrain substrate is recovered from measured trajectory data; defects with respect to 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. Experimental validation of this approach was performed on a 1-D, vertical robotic jumper, operating over a variety of categorically distinct surfaces: leveled solid ground, a trampoline surface, and a bed of poppy seeds (a model GM [17]). The robot quickly accomplished specified control tasks, each within 10–15 control-learning iterations. Using this model-based data-driven learning and control approach, each terrain model was learned, and specified tasks were completed precisely, while only requiring a small set of experiential data.

II. System Dynamics

This section reviews the equations governing 1-D jumping, as applicable to any surface type, rigid or deformable. The equations are arranged to highlight terrain substrate forcing, F_{sub}, the unknown quantity in this system.

A. Equations of Motion

The 1-D jumping model, shown in Fig. 2 and presented in [16], 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

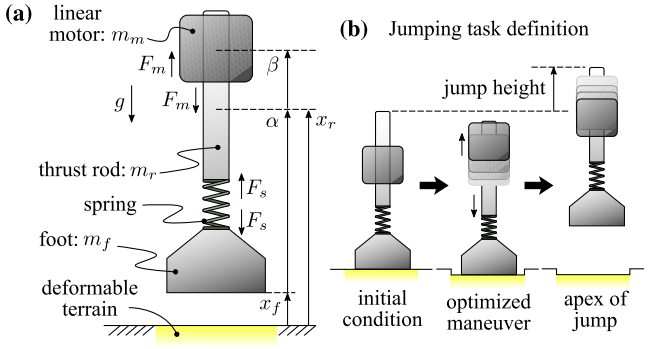


Fig. 2. (a) Visualization and labeling of the 1-D (vertical-only) jumping robot model. (b) Illustration of the jumping task; the robot must thrust its motor mass to jump off the ground and achieve a specified jump height at its apex.

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 mass-less, but modeled with linear viscous damping). The closed-form system dynamics are

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

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 represent the spatial position of the foot center of mass (CoM), position of the rod CoM relative to the foot CoM, and the position of the motor CoM relative to the rod CoM, respectively. $\ddot{\beta}$ denotes acceleration of the motor relative to the rod; in the physical platform, this is driven by dynamics related to internal motor feedback tracking [4]. The control input will be a synthesized reference motor trajectory, $u = \beta^*$, against which β is tracked to arrive at $\ddot{\beta}$. For later reference, $x_r = x_f + \alpha$ denotes spatial position of the thrust rod.

$\bar{\alpha}$ represents the position of the rod, relative to the foot, when no external forcing, gravity or otherwise, acts to compress or extend the spring. F_{sub} denotes substrate reaction forces acting on the foot and is always directed positively upward. The system state is defined, $\boldsymbol{x} = [x_f \ \dot{x_f} \ \alpha \ \dot{\alpha}]^{\text{T}} \in \mathbb{R}^4$.

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 from the ground. The model of the substrate force will vary with the substrate type and the relevant phase of the hybrid system. During the *flight phase*, the foot has lifted off the surface and substrate forcing no longer acts on the system, $F_{\text{sub}} = 0$. During the *stance phase*, the foot is in contact with the ground, and x_f is coincident with the surface height. The substrate forcing F_{sub} then becomes a positive quantity, modeled according to the material composition of the jumping surface.

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

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

where superscript $^{\text{SG}}$ denotes the assumption of solid, undeformable ground.

When the ground is no longer rigid and is, instead, modeled as a deformable material, then the assumptions used for solid ground scenarios no longer hold. Consider GM, a material characterizing a particular example class of deformable surfaces. The function $F_{\text{sub}}^{\text{GM}}$, denoting substrate forcing exerted on foot by the GM, becomes a complex nonlinear function of the hopper state and is dependent upon empirically determined parameters modeling the material [16]. It may be decomposed as

$$F_{\text{sub}}^{\text{GM}} = F_p + F_v \tag{3}$$

where F_p denotes a quasi-static forcing component dependent upon foot depth, while F_v varies as a function of depth, velocity, and added mass to the foot during GM intrusion.

In general, deformation of the ground substrate may induce a significant change in the dynamics of the 1-D hopper, in contrast to operation on solid ground. Open-loop signals designed for solid ground scenarios but applied on GM, for example, lead to system trajectories that may diverge greatly from operation on the former. Fig. 3 (left-hand side) shows such a scenario. An open-loop control signal is designed to achieve a peak jump height of 30 mm (dashed) on a simulated 1-D, robotic hopper operating on level solid ground. On simulated GM (solid), however, the ground yields and x_f (red) sinks prior to takeoff. Terrain deformation dissipates energy, robbing it from the spring, ultimately leading to a lower peak jump height, 7.64 mm, which falls short of the target by 74.5%. The nonlinear behavior of this complex, deformable terrain type does not permit the use of simple feedback mechanisms; we have seen that tunable control signal characteristics, such as amplitude, do not necessarily trend linearly, nor monotonically, with achieved peak jump height on this surface [4]. Alternative approaches to legged locomotive control over soft terrain have presumed spring-damper-based models of the robot-ground interactions [11], [13], [14], [42]. These assumptions, however, remain ill-suited for operation over GM, a class of terrain characterized by its complex, hydrodynamically driven nonlinear behavior.

In the context of this study, each terrain type the leg hops on will have its own unique substrate interaction force profile, which will require learning.

III. Online Learning of External Forcing

This section describes the GP-based online learning process. In particular, it describes how the measured state variables of the robot lead to estimates of external ground forcing, through an inverse dynamics procedure.

A. 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 [43]. Given a set of (possibly noisy) evaluations of an unknown function, $\tau(\boldsymbol{x}) : \mathbb{R}^n \to \mathbb{R}$, a GP represents a prior over functions

$$Z(\boldsymbol{x}) \sim \text{GP}(m(\boldsymbol{x}), \kappa) \tag{4}$$

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

$$\kappa(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}')^{\text{T}} \Sigma (\boldsymbol{x} - \boldsymbol{x}')\right) + \sigma_{\text{damp}}\mathbb{I}, \quad \boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^n \tag{5}$$

which provides a measure of similarity between any two elements, $\boldsymbol{x}$ and $\boldsymbol{x}'$, in the domain of $\tau(\boldsymbol{x})$, 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 overfitting in regression outcomes.

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

The GP-based function approximation is more specifically represented by [44]

$$y(\boldsymbol{x}) = \mu(\boldsymbol{x}) + r^{\text{T}}(\boldsymbol{x})R^{-1}(\hat{\boldsymbol{y}} - \mu(\hat{\boldsymbol{x}})) \tag{6}$$

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

B. Learning Ground Forcing as a Function of State

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

$$F_{\text{sub}}(\boldsymbol{x}) = F_{\text{sub}}^{\text{SG}}(\boldsymbol{x}) + \mu(\boldsymbol{x}) + r^{\text{T}}(\boldsymbol{x})R^{-1}(\hat{\boldsymbol{y}} - \mu(\hat{\boldsymbol{x}})) \tag{7}$$

where again $\hat{\boldsymbol{x}}$ and $\hat{\boldsymbol{y}}$ comprise training data; $\hat{\boldsymbol{x}}_i \in \mathbb{R}^4$ 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. For subsequent reference, these tuples compose the reference training data set, $\Gamma = \{(\hat{\boldsymbol{x}}_i, \hat{y}_i) : i \in \mathbb{N}, i \leq p\}$, where $p = |\Gamma|$ designates cardinality of the data set. $F_{\text{sub}}^{\text{SG}}(\boldsymbol{x})$ is defined by (2). When no training data has been collected, the latter two components of (7) vanish, and the model reduces to the baseline solid ground assumption. As training data are collected, the latter two terms of (7) model components of terrain forcing that represent defects from a rigid surface environment.

A global linear trend, $\mu(\boldsymbol{x}) = f(\boldsymbol{x})^{\text{T}}\Theta$, is extracted from the training data, after which residuals from this global trend are modeled by the GP in the final term of (7). To compute $\mu(\boldsymbol{x})$, let $f(\boldsymbol{x}) = [1 \;\; x_f \;\; \dot{x}_f \;\; \alpha \;\; \dot{\alpha}]^{\text{T}}$. Then, the corresponding linear coefficients are computed in closed form from a damped, weighted least-squares fit

$$\Theta = \left(F^{\text{T}}R^{-1}F + \rho_\mu \mathbb{I}\right)^{-1}\left(F^{\text{T}}R^{-1}\hat{\Phi}\right) \tag{8}$$

where $\hat{\Phi} = [(\hat{y}_1 - F_{\text{sub}}^{\text{SG}}(\hat{\boldsymbol{x}}_1), \ldots, (\hat{y}_p - F_{\text{sub}}^{\text{SG}}(\hat{\boldsymbol{x}}_p))]^{\text{T}}$. $F = [f(\hat{\boldsymbol{x}}_1) \;\; f(\hat{\boldsymbol{x}}_2), \ldots, f(\hat{\boldsymbol{x}}_p)]^{\text{T}}$ and p denotes the cardinality of the training set [44]. ρ_μ is a damping parameter influencing the pseudoinverse of F.

We have applied a weighted least squares fit as a low-complexity approach to extract a coarse global trend, $\mu(\boldsymbol{x})$, for the terrain forcing function being approximated. The GP then regresses over the function's residual that remains after this underlying trend has been removed.

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 (7). 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, $\boldsymbol{x} = [x_f \;\; \dot{x}_f \;\; \alpha \;\; \dot{\alpha}]$, is completely observable. In particular, however, acceleration associated with the foot, $\ddot{x}_f$, is not immediately and conveniently accessible. To recover this quantity, measured trajectory data produced by the dynamical system, (1), is subjected to a fixed-lag Kalman smoothing process with trivial integrator dynamics. The output of this process is a smoothed state trajectory, $\boldsymbol{x}^s(t) = [x_f^s(t) \;\; \dot{x}_f^s(t) \;\; \ddot{x}_f^s(t) \;\; \alpha^s(t) \;\; \dot{\alpha}^s(t) \;\; \ddot{\alpha}^s(t)]^{\text{T}}$, providing the missing quantity $\ddot{x}_f^s$. Substrate forcing is then estimated as a function of the smoothed system state

$$F_{\text{sub}}^s(t) = m_f g - \left[k\left(\alpha^s(t) - \bar{\alpha}\right) + c\dot{\alpha}^s(t)\right] + m_f \ddot{x}_f^s(t). \tag{9}$$

This smoothed trajectory and recovered forcing data is appended to the training data set, $\Gamma = \Gamma \cup \{(\hat{\boldsymbol{x}}_i, \hat{y}_i)\}$, where $\hat{\boldsymbol{x}}_i = \boldsymbol{x}_i^s$ and $\hat{y}_i = (F_{\text{sub}}^s)_i$. Time parameter, t, has been replaced by discrete time index, i.

D. Data Curation

GP training incurs matrix inversion costs whose complexity is $O(p^3)$ in the worst case. Evaluation cost of the GP-based regression model, (7), scales linearly with the cardinality of the training set, e.g., it is $O(p)$. When used within the numerical optimization method, the cost of gradient evaluation will naturally scale with the number of terms required to symbolically express the GP-based model of F_{sub}. This scales by the evaluation cost $O(p)$ for each gradient iteration. To speed up the learning and optimization computations, we incorporate a data sparsification procedure that trains a GP model from a subset of the full data set [31], [32], [45].

An initial data set, $\Gamma = \{(\hat{\boldsymbol{x}}_i, \hat{y}_i) : i \in \mathbb{N}, i \leq p\}$, with cardinality $p = |\Gamma|$ is reduced to the subset $\bar{\Gamma} \subseteq \Gamma$ via an iterative procedure. Initially, $\bar{\Gamma} = \emptyset$. Each iteration then selects a random element of the set, $(\hat{\boldsymbol{x}}_{\text{cent}}, \hat{y}_{\text{cent}}) \in \Gamma$, to serve as a *center*; all elements falling within its *shadow* are culled

$$\Gamma_{\text{cull}} = \left\{(\hat{\boldsymbol{x}}_i, \hat{y}_i) \in \Gamma : \|\hat{\boldsymbol{x}}_i - \hat{\boldsymbol{x}}_{\text{cent}}\|_\Sigma < \ell_{\text{cull}}^{-2}\right\}. \tag{10}$$

The threshold ℓ_{cull} determines which elements are to be culled and associated with the center element, $(\hat{\boldsymbol{x}}_{\text{cent}}, \hat{y}_{\text{cent}})$. The scaled Euclidean distance $\|\cdot\|_\Sigma$ is the weighted ℓ^2-norm, with respect to the GP squared exponential kernel's bandwidth parameter, Σ, in (5). The center is appended to the reduced data set, $\bar{\Gamma} = \bar{\Gamma} \cup \{(\hat{\boldsymbol{x}}_{\text{cent}}, \hat{y}_{\text{cent}})\}$, and the reference data set updated, $\Gamma = \Gamma \setminus \Gamma_{\text{cull}}$. This procedure repeats until $\Gamma = \emptyset$.

After every hop, when a new set of batch data is to be assimilated into the GP model, data sparsification is first applied. Training with the reduced set $\bar{\Gamma}$ uses a density-weighted procedure that factors in the removed data [45] when performing the matrix inversion during GP training. Data accumulates into the set $\bar{\Gamma}$ as needed (based on ℓ_{cull}).

IV. Iterative Optimal Control-Learning Framework

The dynamics model, force recovery procedure, and learning process contribute to an online, iterative optimal control synthesis and learning framework [35]. In a prior implementation of the framework, a cyclic procedure (Fig. 1) to generate an optimal motion plan, apply it to a robotic hopper, then collect trajectory data and retrain a GP-based model of ground forcing, resulted in optimal control solutions whose predicted hopper trajectories coincided with simulated reality after a few control-learning iterations. This section delineates the procedure and supporting components.

A. Optimal Jumping Control

This work leverages trajectory optimization methods to generate an optimal control solution, $u = \beta^*$. Using a direct collocation approach, a trajectory optimization problem is transcribed into a large-scale nonlinear program (NLP) [46], [47]. An NLP formulation allows casting of hard constraints, such as physical and task constraints, as inequality/equality constraints

rather than as weighting factors within the objective function. Furthermore, direct collocation approaches admit computation of all objectives, constraints, and derivatives thereof in closed-form.

In this work, objectives, constraints, and their respective derivatives are all exported as closed-form expressions by a MATLAB-based parser [48]; the resulting NLP is solved using Interior Point OPTimizer (IPOPT) [49], a large-scale interior point solver. We approximate the trajectory using $N = 31$ trapezoidally integrated collocation points for each of the stance and flight phases of this hybrid system (62 total).

The optimal control specification and problem setup follows that of [4]. We begin by rearranging our system dynamics into a typical set of first-order ordinary differential equations (ODEs), $\dot{x} = f(t, \boldsymbol{x}, u(t))$, where t, $\boldsymbol{x}$, and $u(t)$ are time, state, and time-varying control inputs, respectively. Collocation methods discretize the optimal control problem and then apply a numerical optimizer on the discrete set of variables defining the piecewise state and control trajectories. This includes a discrete time tape, t_i^d, where $0 = t_0^d < t_1^d < t_2^d < \ldots < t_N^d = T^d$, and state variables, $\boldsymbol{x}_i^d$, and control inputs, u_i^d, where $i \in \{1, 2, 3, \ldots, N-1\}$. Variables are duplicated for each dynamical domain, $d \in D = \{\text{stance}, \text{flight}\}$. The variables, $\boldsymbol{x}_i^d$, $u_i^d = \beta_i^{*,d}$, and t_i^d, are free variables for the optimization process to determine. In this manner, durations of each dynamical domain, T^d, are free to be designed as well. A vector of all optimization variables, $\mathbf{w} = \{\boldsymbol{x}_i^d, \dot{\boldsymbol{x}}_i^d, u_i^d, \dot{\beta}_i^{*,d}, t_i^d\}$, is built from states, first-order time derivatives, and control inputs, over all i and d defined. Depending on the motor's internal control loop, definition of additional design variables was required to encode the first-order dynamics of integral control terms.

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

$$\left(\boldsymbol{x}_{i+1}^d - \boldsymbol{x}_i^d\right) - \frac{1}{2}\left(t_{i+1}^d - t_i^d\right)\left(\dot{\boldsymbol{x}}_{i+1}^d + \dot{\boldsymbol{x}}_i^d\right) = 0 \tag{11}$$

$$\dot{\boldsymbol{x}}_i^d - f\left(t_i^d, \boldsymbol{x}_i^d, u_i^d\right) = 0 \tag{12}$$

for all i, derived from a trapezoidal integration scheme. The equations of motion f include the GP-based forcing model F_{sub} in symbolic form.

The control task is to achieve a specified "jump height" defined as the difference between the initial rod height, x_r^0, and the rod's highest point during the jump, x_r^f, as shown in Fig. 2(b). Consequently, targeted jumping requires adding an equality constraint $x_r^f - x_r^0 = h^*$ with h^* equal to the target jump height.

In hardware implementation, the true actuation signal, $\ddot{\beta}$, arises from a low-level trajectory tracking control loop applied to the linear motor of Fig. 2(a). The closed-loop motor dynamics introduce a dynamic response that impacts jumping performance. We explicitly account for these tracking dynamics in the optimization by modeling the control loop as part of the system dynamics [4]. In brief, the solution, u, is computed with anticipation, both of how the tracking will perform given the closed-loop dynamics and the saturation limits of the actuator force, F_m.

The objective function is defined

$$\begin{aligned}\mathbf{J}(\mathbf{w}) = \sum_{d \in D}\Bigg[&\frac{t_2^d - t_1^d}{2} J\left(\boldsymbol{x}_1^d, \dot{\boldsymbol{x}}_1^d, u_1^d, \dot{\beta}_1^{*,d}\right) \\ &+ \sum_{i=2}^{N-1} \frac{t_{i+1}^d - t_{i-1}^d}{2} J\left(\boldsymbol{x}_i^d, \dot{\boldsymbol{x}}_i^d, u_i^d, \dot{\beta}_i^{*,d}\right) \\ &+ \frac{t_N^d - t_{N-1}^d}{2} J\left(\boldsymbol{x}_N^d, \dot{\boldsymbol{x}}_N^d, u_N^d, \dot{\beta}_N^{*,d}\right)\Bigg]\end{aligned} \tag{13}$$

where $J(\cdot)$ is the integrand of a continuous-time cost. We defined the objective integrand to be $J = \dot{\beta}_i^{*,d}(t)^2$, which penalized actuator velocity as a proxy for energetic cost and also favored more-easily trackable position trajectories on hardware. One could substitute instantaneous power as the objective; however, such power minimizing solutions tend to rail actuator forces violently between zero and nonzero values. Such "banging" strategies are typically more sensitive to timing errors in hardware implementation.

The resulting NLP is

$$\mathbf{w}^* = \underset{\mathbf{w}}{\text{argmin}}\ \mathbf{J}(\mathbf{w}) \tag{14}$$

$$\text{s.t. } \mathbf{w}_{\min} \leq \mathbf{w} \leq \mathbf{w}_{\max} \tag{15}$$

$$\mathbf{c}_{\min} \leq \mathbf{c}(\mathbf{w}) \leq \mathbf{c}_{\max} \tag{16}$$

where $\mathbf{c}(\mathbf{w})$ represents the concatenation of all constraint functions, including defect constraints (11) and (12) and task constraints, such as jumping height, actuator limits, workspace/acceleration limits, and so on [4]. The optimization routine was tasked to return a locally optimal control trajectory. In addition to the other states in $\mathbf{w}$, the optimization returns a vector $u = \beta^*$ comprising the concatenation of sequences, $\beta_i^{*,\text{stance}}$ and $\beta_i^{*,\text{flight}}$. In implementation, IPOPT solved the NLP to a tolerance of 10^{-9}, for both feasibility and duality. All told, the optimal control computations were performed given **(1)** an *a priori* robot model, **(2)** an *a priori* model of the motor control loop, and **(3)** a GP-based model of the terrain forcing.

B. Iterative Optimal Control and Learning

We apply an iterative control-learning procedure to recover a GP-based dynamical model of the 1-D hopper system, described jointly by (1) and (7). While hopping on an unknown surface, the forcing profile that surface exerts as a function of the system state, $\boldsymbol{x}$, will be simultaneously learned. Prior to any experimental attempts, the reference training data set is empty, $\Gamma = \emptyset$, corresponding to the zero function for any learned components of the unknown forcing model. Each learning iteration begins with the hopper at rest on the jumping surface. An optimizer then employs the GP-based dynamical model, described by (1) and (7), to generate an optimal control signal enabling a target peak jump height to be attained, while minimizing actuation effort. The control signal is subsequently applied to the 1-D hopper. If the target peak jump height is not attained using the generated control signal on the surface, trajectory data are collected, smoothed, and substrate forcing, $F_{\text{sub}}^s(t)$, is recovered per (9). The reference training set, Γ,

Algorithm 1 Iterative Hopper Control and Learning

procedure CONTROL AND LEARN(controlTask)
  reference data set, $\Gamma \leftarrow \emptyset$
  reduced data set, $\bar{\Gamma} \leftarrow \emptyset$
  taskComplete $\leftarrow$ **false**
  initialize GP
  while ¬taskComplete **do**
    generate optimal control, $u = \beta^*$, using GP-based model, (1) and (7)
    evaluate $u = \beta^*$ on ground truth terrain
    extract hopper state trajectory measurements, $\boldsymbol{x}(t)$
    taskComplete $\leftarrow$ TASK COMPLETE(controlTask)
    apply Kalman smoothing to recover $\boldsymbol{x}^s$
    compute F^s_{sub} using (9)
    append Γ with new data: ($\boldsymbol{x}^s$, F^s_{sub})
    extract reduced training set, $\bar{\Gamma} \subseteq \Gamma$ (Sec. III-D)
    $\forall(\hat{\boldsymbol{x}}_i, \hat{y}_i) \in \bar{\Gamma}$, compute $\hat{y}_i - F^{\text{SG}}_{\text{sub}}(\hat{\boldsymbol{x}}_i)$
    update $\mu(\boldsymbol{x})$ using (8)
    train GP on residual forcing data: $\{(\hat{\boldsymbol{x}}_i, \hat{y}_i - F^{\text{SG}}_{\text{sub}}(\hat{\boldsymbol{x}}_i) - \mu(\hat{\boldsymbol{x}}_i)) : i \in \mathbb{N}, i \leq |\bar{\Gamma}|\}$
  end while
end procedure

is augmented with the computed tuples, ($\boldsymbol{x}^s(t)$, $F^s_{\text{sub}}(t)$), for all discretely measured t. This training set is limited to trajectories and corresponding forces occurring during the stance phase of the jump prior to the hopper transitioning to the flight phase.

The reduced training data set, $\bar{\Gamma}$, is extracted from the cumulative reference set, Γ, with cardinality $k \leq p$. It is used to update the GP-based model of substrate forcing. First, the deviation of $F_{\text{sub}}(\hat{\boldsymbol{x}})$ from the equivalent solid ground forcing profile, $F^{\text{SG}}_{\text{sub}}(\hat{\boldsymbol{x}})$, is computed and a global linear trend is extracted, using (8) to compute the linear coefficients. The GP is trained on the remaining residual, $F_{\text{sub}}(\hat{\boldsymbol{x}}) - F^{\text{SG}}_{\text{sub}}(\hat{\boldsymbol{x}}) - \mu(\hat{\boldsymbol{x}})$, 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 terrain forcing. Algorithm 1 sketches the iterative optimal control and learning process.

V. EXPERIMENTAL VALIDATION

Preliminary validation of the simultaneous control and learning approach, detailed in Algorithm 1, was demonstrated in simulation [35]. When challenging the hopper to learn the forcing profile associated with an unknown, simulated GM surface, the target hop height was achieved by the third control-learning iteration. Fig. 3 shows outcomes produced in a similar simulated environment, where the hopper model and learning parameters have been updated to reflect those applied in follow-on physical experiments (Sections V-A and V-B). Results of each control-learning iteration, in Algorithm 1, are depicted. The target jump height is achieved, within approximately 3% error, with the final jump. Hopping with the learned model was a significant improvement over the initial model, which yielded a 22.36 mm (or 74.5%) error. It showed that significant practical improvements can occur over relatively few iterations when the problem and observations are formulated to emphasize the uncertainty in the system (here, the unknown terrain forcing model) as opposed to attempting to learn the entire set of dynamics. A comparison of the underlying ground reaction force models, seen in Fig. 3 (right-hand side), shows that the GP-based estimate of ground forcing converges to the true model. This behavior underscores how a learned model approach can yield quickly adaptive behaviors in robotic applications for a targeted task. To confirm whether the situation would be reproduced in reality, we applied the same iterative control-learning procedure to a physical robotic hopper. This section describes the experiments and their outcomes.

A. Experimental Apparatus

The experimental setup of [4] and [16], with minor modifications, was utilized to generate all experimental results presented in this section. It is shown in Fig. 4(a) on the left together with an annotated model on the right describing the motor, foot, and rod properties associated with (1). A coil spring couples the foot to the bottom of the thrust rod. The motor actuates along the rod and is additionally mounted to a low friction air-bearing, vertical linear slide. This restricts the motion of the entire hopper apparatus to a single dimension. The foot of the robot is the only component to interact with any jumping surface and is composed of a lightweight, 3-D printed cylinder with base radius, 38.1 mm, and height, 35 mm. Fig. 4(b)–(d) shows the three categorically distinct surface types employed during the experiments: solid ground, a trampoline surface, and a GM bed.

Performance of Algorithm 1 was evaluated for control tasks requiring the hopper to achieve a range of targeted peak jump heights. Jump heights were defined to be the maximum tracked height attained by the rod marker, relative to its resting position at the start of the jump, shown in Fig. 2(b). To facilitate the measurement of state variables x_f and x_r, spherical, white markers were mounted to both the foot and rod, respectively. A gray-scale PointGrey Grasshopper camera with a high-frame rate (200 Hz) was positioned to simultaneously capture both markers during the course of each experiment. Marker tracking was performed frame-to-frame.

B. Experimental Procedure

In the interest of repeatability and consistency across experimental trials, we defined and followed an experimental protocol, with minor adjustments for each terrain tested. To reduce the impact of transient effects after setting the hopper in its initial resting configuration, a waiting period was observed after lowering the robot onto the experimental terrain and commanding initial motor position, β^*_0. For each surface, this pause allowed spring-induced oscillations to dampen out (from the hopper spring or the trampoline). On GM, additional procedures were in place to ensure reproducible outcomes. Preceding execution of any robotic jump, the hopper was elevated off the GM surface. The bed of poppy seeds was then fluidized using a 5-hp air blower through a flow diffuser at the bed's base. Cessation of airflow then led to a consistent,

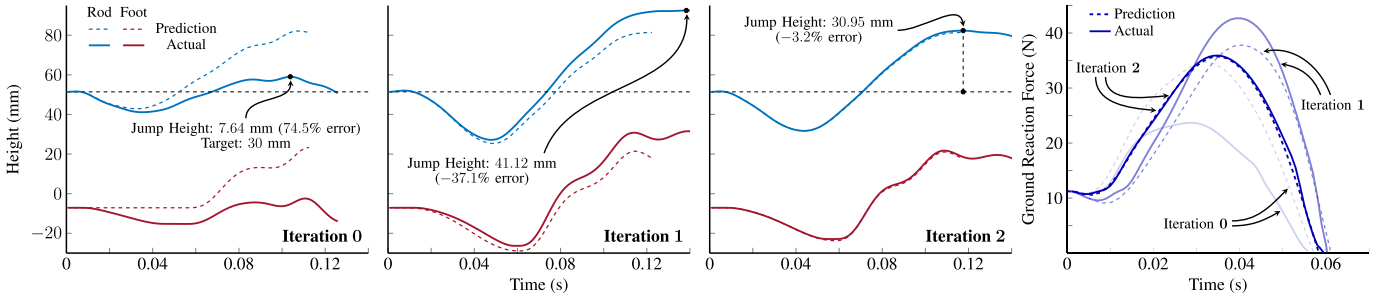


Fig. 3. Simulation of Algorithm 1 jumping experiments on GM, with the goal of achieving a 30-mm jump height (as measured by the rod position), is illustrated in the three ***leftmost plots***. In three iterations of learning → planning → jumping, the simulated jumping task improved in accuracy from 74.5% error to −3.2% error. The evolution of the ground reaction force prediction, across three iterations, is shown in the ***rightmost plot***. Each iteration updated the GP-based model, from which a new strategy was computed by the motion planning optimizer. By the third iteration, the actual experienced ground force sufficiently matched the GP prediction to achieve a jump apex with −3.2% error.

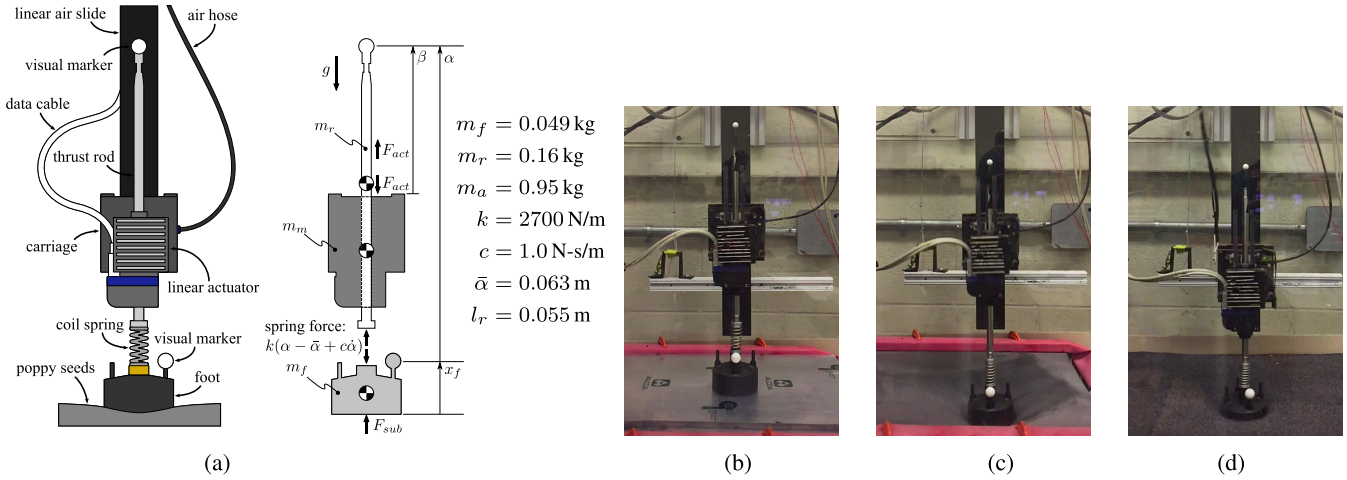


Fig. 4. (a) Experimental Hopper: one-dimensional, vertical robotic hopper resting on a bed of poppy seeds (left-hand side) and a free-body diagram representation of the hopper model (right-hand side), illustrating motor (dark gray), foot (light gray), and rod (white) and model parameters. (b)–(d) Hopping experiments applied the iterative control-learning framework to 3 surfaces types with categorically distinct robot-terrain dynamics (b) solid ground, (c) trampoline, and (d) GM.

loose-packed terrain state [4], $\phi = 0.57$, after which the hopper was lowered to rest on the GM bed, in preparation for jumping.

The experimental apparatus had a specialized terrain bed constructed for the experiments (Fig. 4). The most important consideration was to ensure that the GM terrain type was properly configured. In these experiments the GM surface was realized by a bed of poppy seeds (∼1-mm diameter) [16]. A generously sized 56 cm × 56 cm glass container was filled with poppy seeds, to a depth of 15 cm, to avoid boundary effects associated with walls of the container [Fig. 4(d)]. The solid ground and trampoline terrain types involved placing a custom object over the GM bed. For solid ground, the object was a 12.7-mm-thick aluminum plate, as shown in Fig. 4(b). The plate was mounted and leveled, such that the surface plane normal aligned with the gravitational force vector. For the trampoline, the object was an elevated, hollow square frame (46 cm × 46 cm) outfitted with a thin, elastic rubber sheet secured to the frame's borders [see Fig. 4(c)]. The setup was configured such that the hopper foot maintained its position at the center of the terrain bed during the course of all experiments.

The three terrain types were chosen because of the distinct behaviors they each exhibit. The GM bed behaves as a soft dissipative surface; energy expended by the hopper is absorbed by the GM and never restored. The elasticity of the trampoline's rubber sheet confers energy-restorative (as opposed to dissipative) properties to the soft surface. Lastly, solid ground represented the baseline condition, where no learning should be required (outside of minor modifications to correct for incorrectly identified system parameters). For each terrain type and hop height, we performed ten runs of an optimal control trajectory that would meet the hop height. The standard deviation (SD) of hop height was then measured to serve as an indication of the natural variance associated with the task. Repeatability of task convergence, for the hopper, cannot be better than this natural variance.

Additional parameters influencing performance of the system and Algorithm 1 are detailed in Table I (i.e., GP and data curation parameters). A single set of GP hyperparameter values were used for all experimental trials. These hyperparameters were held static across all experimental iterations of Algorithm 1, irrespective of the targeted jump height or terrain class. Variation of the data curation parameter, ℓ_{cull},

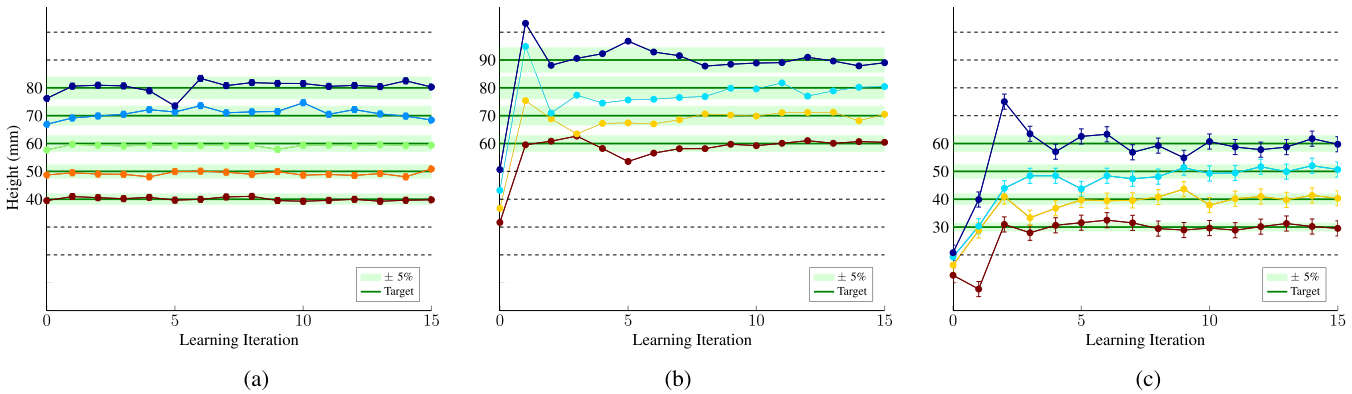


Fig. 5. Hopper was tasked to jump to four different peak heights (dark green line) on each surface type, in accordance with Table II (a) solid ground, (b) trampoline surface, and (c) GM. Peak jump heights, over the course of 15 control-learning iterations, were measured. Task completion was declared when ±5% of the target jump height was attained (shaded green). Error bars show 1 SD associated with jump height measurements on a particular surface. If not visible, it is because the variability is low.

TABLE I
GP LEARNING PARAMETERS

Parameter	Value
Σ	$\text{diag}(\sigma_{x_f}^2, \sigma_{\dot{x}_f}^2, \sigma_{\alpha}^2, \sigma_{\dot{\alpha}}^2)$
σ_{x_f}	$0.8e^{-3}$
$\sigma_{\dot{x}_f}$	$40e^{-3}$
σ_{α}	$5e^{-3}$
$\sigma_{\dot{\alpha}}$	$150e^{-3}$
σ_{damp}	5
ℓ_{cull}	0.20 (solid ground) 0.02 (trampoline) 0.06 (granular media)

TABLE II
EXPERIMENTAL TRIALS

Surface	Target Jump Heights (mm)
Solid Ground	40, 50, 60, 70, 80
Trampoline	60, 70, 80, 90
Granular Media	30, 40, 50, 60

occurred by surface type and was primarily motivated by a desire to achieve roughly equivalent reduced data set cardinalities, $|\bar{\Gamma}|$, across the experiments.

The carefully engineered experimental environment, in conjunction with the established experimental procedure, was necessary to evaluate the efficacy of the iterative control-learning procedure, Algorithm 1. This experimental structure ensured that all relevant variables were controlled in order to quantify the effectiveness of Algorithm 1, in a consistent and highly repeatable manner. Future work will extend the evaluation of this control-learning approach to additional challenging terrain profiles in less controlled experimental settings.

C. Task: Jump to Target Height

On each terrain type, the robotic hopper was tasked to achieve a set of different peak jump heights (Table II), all realizable for the terrain in question. For each experimental trial associated with a particular target jump height, 15 control-learning iterations from Algorithm 1 were executed. At the start of each trial, both the reference and reduced training data sets were initialized, $\Gamma = \emptyset$ and $\bar{\Gamma} = \emptyset$, prompting the GP-based model, (7), to presume rigid ground dynamics. Peak hop heights attained during each iteration were recorded to capture the progress of the system over the course of a single trial. Fig. 5 shows the outcomes, per terrain type and jump height, over the trials.

D. Results: Achieving the Target Jump Height

As Fig. 5 demonstrates, through successive optimal control-learning iterations, the robotic hopper attained each targeted jump height regardless of its initial ignorance of the particular ground dynamics influencing the system. The shaded green regions show ±5% bounds with respect to each targeted jump height (solid dark green lines); we equate these shaded regions to task completion, for the purposes of this study. The plots are broken down by terrain type: solid ground [Fig. 5(a)], trampoline [Fig. 5(b)], and GM [Fig. 5(c)]. Each colored line shows the peak jump height measured over the course of successive optimal control-learning iterations for a particular targeted jump height. Error bars illustrate ±1 SD, characterizing variance in measured experimental jump heights on a particular surface type. If not visible, then the error bar is smaller than the plotted dot's radius, meaning that the experimental variance is small. From the plots, only the GM terrain has an experimental variance on par with the plotted ±5% error region. Convergence to within 5% error typically occurred within 10 to 11 iterations, as seen by the experimental traces lying within the shaded green bands. The majority of progress, to achieve each desired jump height, was made within the first six iterations (error less than 10%). The remaining iterations saw the convergence of the measured peak jump heights toward the targeted height.

During solid ground experiments [Fig. 5(a)], the initial predicted model was sufficiently close to reality such that measured jump heights were within the 5% window on the first hop (i.e., Iteration 0), but consistently below the target height. Higher target jump heights led to a greater initial error, which was corrected within two jumps as the system refined its parametric model of the ground interaction forces and instantiated a GP model for additional corrections. Across

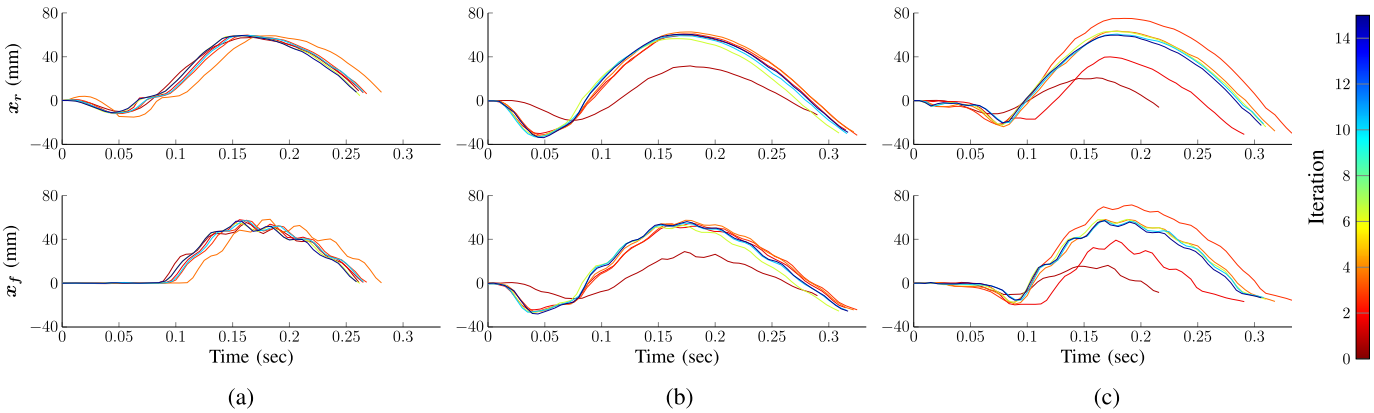


Fig. 6. Spatial **rod** (top), x_r, and **foot** (bottom), x_f, trajectories progress toward task completion as the GP-based model becomes incrementally better informed through accumulated experiential data. Trajectories measured for control-learning Iterations 0–3, 6, 10, and 15 are illustrated, where a jump height of 60 mm was targeted on each surface type (a) solid ground, (b) trampoline surface, and (c) GM.

the learning iterations, the median error for each target height was close to the initial error at Iteration 0.

The trampoline peak height graphs in Fig. 5(b), indicate consistently incorrect hop heights were achieved during Iteration 0; they all fell short of targeted hop heights by more than 40%. After accumulating ground reaction force data and learning a revised model from the data recovered during Iteration 0, the subsequently synthesized control then drove the hopper to overshoot the target height, in Iteration 1 (with the exception being the target hop height of 60 mm). By Iteration 6 and onward, measured peak jump heights converged toward the target height, remaining within a $\pm 5\%$ vicinity. Importantly, from Iteration 3 to Iteration 6, the peak height outcomes lay within 10% of the target height.

Similar outcomes were seen for the experiments on GM, in Fig. 5(c); only, the initial errors were greater and in the range of $55\% - 65\%$. As GM is energy dissipative in nature, the energy expended by the robotic hopper to deform the terrain was dissipated and never restored. By Iteration 3, peak heights were within 10% of the target height and remained there. Iterations 6–10 consisted of learning trials whereby recovered data better informed the dynamics, and subsequently the optimizer, in order to synthesize an appropriate control signal. By Iteration 10, the system was capable of synthesizing control signals whose outcomes were within 5% of the target hop height. Considering that the natural variability of hopping on GM (for repeated runs of the exact same signal) was as large as or larger than the 5% bounds, one could conclude that learning occurred within seven hops for almost all of the cases.

When considering the simulated outcomes of Fig. 3, the experimental outcomes are quite close. In the case of GM, experimental variation provides additional sources of discrepancy. These, however, do not prevent the learning rate from being roughly equivalent; experimental learning convergence did not differ by an order of magnitude versus simulation. Overall, the outcomes indicate that the preliminary findings from [35] are experimentally reproducible for GM and other terrain types (here, the trivial solid ground and trampoline surfaces).

E. Results: State Space Exploration

To explore the experimental results further, this section reviews the evolution of the hopper state–space trajectories over successive learning iterations, as shown in Fig. 6, for the case of a task hop height of 60 mm. Early iterations are colored red with later iterations trending to blue. In the case of solid ground [Fig. 6(a)], all measured trajectories closely overlapped with that of Iteration 0; accumulated experiential data introduced little to be learned since the GP-based model's baseline presumption of rigid ground dynamics already served as a sufficient model of the terrain. For the other surfaces, the measured trajectories undershot the desired target height in Iteration 0. The undershoot was corrected during subsequent iterations, with trajectories clustering as less novel experiential data was accumulated by the GP, and the targeted peak jump height was effectively attained.

Fig. 7 shows the iteration-to-iteration progression of the hopper state trajectory for a target jump height of 60 mm, on each terrain class. Although the hopper state space is 4-D, the state coordinate $\dot{\alpha}$ is absent from this plot since we may visualize up to three dimensions. We found $\dot{\alpha}$ to be the least influential coordinate for the recovered ground forcing, F_{sub}, as computed from the inverse dynamics of (1). The coordinate $\dot{\alpha}$ informs (1) solely through a damping term, $c\dot{\alpha}$; the small damping coefficient, c, makes this term negligible, especially compared with its neighboring term that captures more dominant spring stiffness effects. Through Fig. 7, a more complete illustration is provided of the trajectories explored by the robotic hopper while exercising Algorithm 1 over the course of successive control-learning attempts. Because little is learned during the course of a solid ground trial, the initial trajectory traveled, in Fig. 7(a), clustered closely with trajectories measured during subsequent iterations. On the GM and trampoline surfaces [Fig. 7(b) and (c), respectively], however, a greater separation was present between trajectories during early iterations, as new experiential data more drastically refined the GP-based model of ground reaction forcing. As the model began to coincide with reality and the task objective was met, the trajectories produced in later iterations also aligned with optimizer predictions. Trajectories measured during late

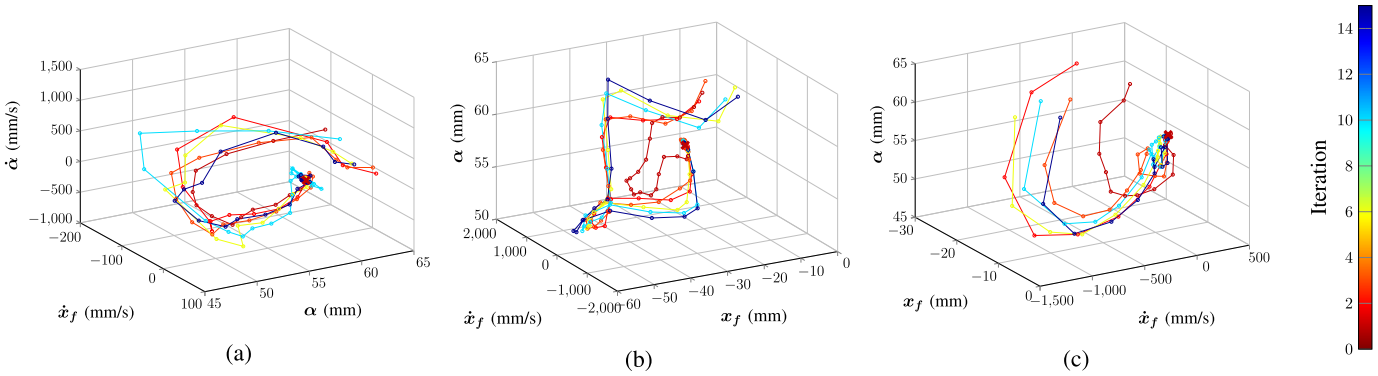


Fig. 7. Stance phase hopper state trajectories explored over successive iterations for a target jump height of 60 mm (a) solid ground, (b) trampoline surface, and (c) GM.

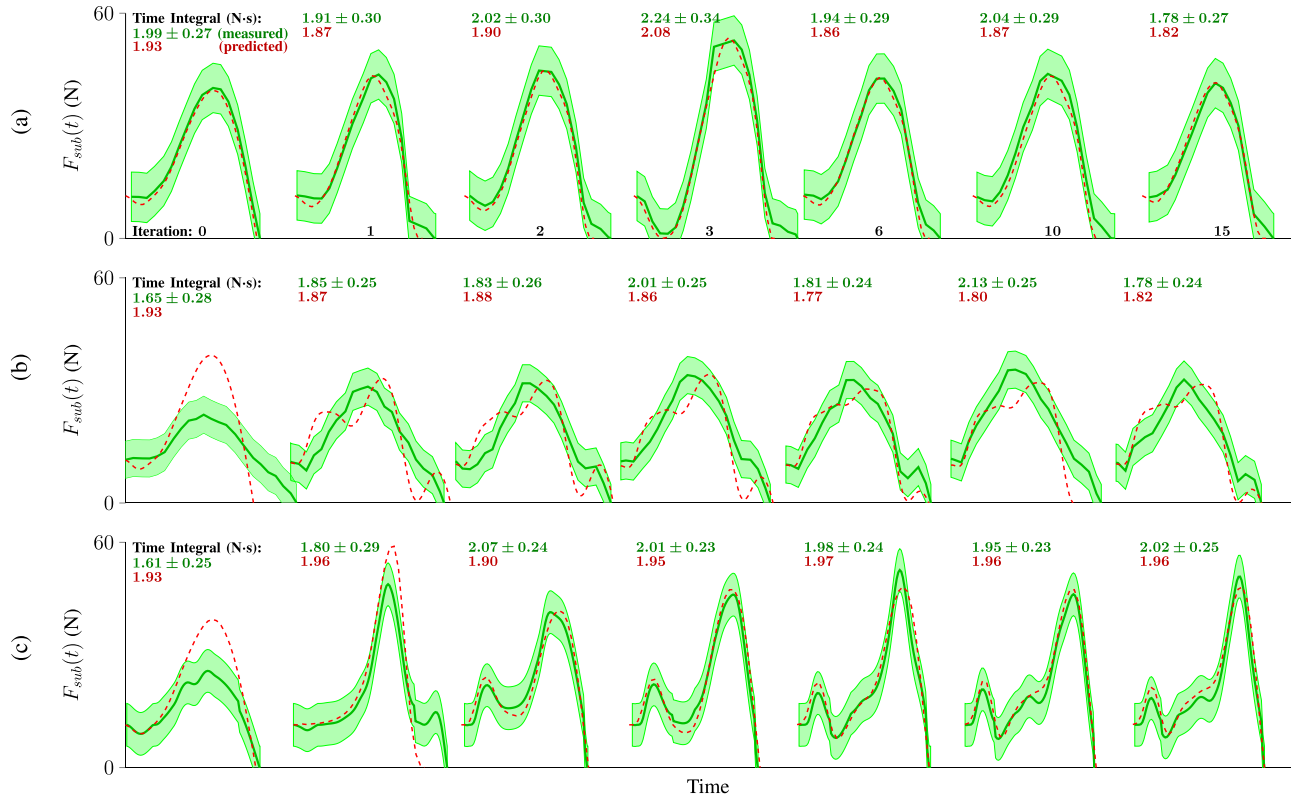


Fig. 8. Predicted (dashed red) versus measured (dark green) ground forcing (a) solid ground, (b) trampoline surface, and (c) GM.

iterations incurred only minor changes, having already accomplished the desired task.

F. Results: Ground Reaction Force Prediction

Success at the task objective relies on accurate force prediction for the synthesis of a task-achieving control signal. Using force recovery via inverse dynamics of (1), as employed for learning, we compare actual ground reaction forcing, recovered from measured jump trajectories, with the force predictions used to craft the corresponding control.

The GP-based model of F_{sub} entails a baseline assumption of rigid ground dynamics, in the absence of any training data. Then, we expect the predicted ground forces to coincide with the recovered ground forcing in the *solid ground* experiments. Because the true ground forcing model should already closely coincide with the baseline assumption of (7), in this context, Iterations 1 and onward should introduce few if any, drastic defects with respect to the baseline solid ground dynamics. Fig. 8(a) shows a comparison of the predicted and actual recovered ground reaction forcing, applied by solid ground terrain, over the course of 15 control-learning iterations where a target peak jump height of 60 mm was commanded. Ground reaction force predictions align very closely with actual recovered measurements of F_{sub} in most iterations. Shaded green shows a 2 SD region associated with force measurements. Associated measurements of the area under the plots are found above and to the left of each plot.

The trampoline and GM cases are a bit more interesting. The underestimated trampoline forcing in Iteration 0 of

Fig. 8(b) illustrates the mismatch between reality and the GP-based model's prediction (initially solid ground). The areas under the predicted and actual recovered ground forcing plots (i.e., effective work exerted on the hopper) are indicated in red and green texts, respectively, above each forcing plot in Fig. 8(b). As iterations proceeded, these values began to coincide, along with the overall shapes. The actual measured forces do not precisely match, with the trampoline error being the largest. The mismatch for both scenarios is offset by higher than actual force predictions prior to- and after- the actual recovered force profile's peak time. Despite this, we observed that the total area under the actual and predicted forcing profiles (i.e., total work) still proceeded to align over the course of 15 control-learning iterations.

For the GM case in Fig. 8(c), the force predictions better matched the actual trend experienced in reality. As the learned ground forcing model permitted improved predictions of the achieved hop height, the synthesized control signals exhibited less deviation from one iteration to the next.

The three sets of forcing plots for Iteration 15 have visually different profiles. The trampoline force plot has a lower peak value and a larger spread than that of the solid ground model. Furthermore, there is a small force increase at the tail end of the jump maneuver relative to the solid ground model. This may be the optimizer attempting to gain extra forcing from the elastic surface prior to losing contact. The actual hopper motor trajectories to be discussed in Section V-G provide further evidence for this assertion. In contrast, the GM force profile is more peaked, in the sense of having a higher peak value and a lower spread relative to solid ground. Furthermore, there is a clear transition from a single "hump" to a double "hump" forcing profile, between Iterations 1 and 2 in Fig. 8(c). This suggests a shift in the control strategy generated by the optimizer. This strategy is accentuated with successive iterations, leading to a small initial peak in the force profile, following by a later, larger peak, consistent with a double-pump motor action. Prior research uncovered a similar motion strategy, whereby the "delayed stutter jump" exploited GM jamming phenomena to improve jump height [16]. Here, the optimal hopping strategy appears to be exploiting similar phenomena to induce higher forces than would be generated by a single-pump control trajectory. More evidence for the double-pump will be seen when reviewing the experimental motor control trajectories. Ultimately, as the system learned the unique forcing profile of each terrain type, customized control strategies arose from the exploitation of these forces to achieve the specified task objective. These customized control actions formed quickly (within two hops), and converged to an optimal response shortly thereafter (usually within four more hops), becoming more refined as more forcing data were collected.

Discussion (Proposed Approach Versus RL): RL has emerged as a popular approach to solving for the optimal policy in a model-free manner with some generalization capacity. As noted in the literature review of Section I-A, RL approaches typically fall into one of two categories. In the first, they require offline, *a priori* training to provide a strong prior during deployment that can be refined through additional experience and exploration. In the second, where only online learning is involved, they require a large number of iterations to establish an ideal policy for a given task. The generalization to variations in the task will require more iterations, usually in direct proportion to the variation. These properties hold for traditional RL, and for deep RL methods. The training data requirements are a function of the model-free or model-minimizing nature of RL approaches. Methods reflecting a mixture of these two categories reflect some mix of the two properties. Here, instead of removing the role of the model, the described iterative-learning method aims to overcome the model's limitations through online learning. An idealized, but inaccurate model, still provides a sufficiently good prior from which to hypothesize optimal control strategies for achieving a specified task. The current results show that learning the mismatch during the course of execution, plus using it to revise the model accordingly, leads to task achievement within a few iterations. It suggests that any engineering knowledge used to design a control system has value when deployed toward a task, even when there is substantial uncertainty in its fidelity. The inclusion of adaptive or learning components will permit the rapid realization of the task without requiring *a priori* knowledge of the scenario's properties. The generalization to parametric task variation is handled by the optimal control synthesis step.

G. Results: Control Trajectories

Observations made when examining the evolution of the forcing profile are further supported by the evolution of the control signal, β^*, the output from the optimal control synthesis described in Section IV-A (Fig. 9). Reference motor trajectories, β^*, are shown in dashed green; actual motor trajectories, β, are shown in red. Just as the learned forcing profiles for the three terrains differed, therefore, too did the control strategies. The solid ground trajectory [Fig. 9(a)] converged to a policy that pushed, paused slightly, then pushed further before quickly retracting (higher downward slope than there was an upward slope initially). The forcing profiles in Fig. 8(a), however, are absent of any similar "kinks." We find the initial pump injected sufficient energy to continue robot spring compression during the brief actuation pause. In this manner, a smooth forcing profile still resulted, per (2). The subsequent pump then continued the compression. Influenced by the selected objective function, (13), the optimization effectively exploited robot spring dynamics to minimize $(\dot{\beta}^*(t))^2$ terms. In contrast, for the trampoline surface, the pause was extended in time and occurred during the period of peak rod contraction, where the trampoline's restorative elastic forces were greatest. The slope of the motor retraction phase was also smaller, indicating less of a need to retract from the surface, compared with solid ground scenarios, and instead exploit the trampoline's elastic response. Lastly, for the GM case, the inflection point in the motor trajectory, which developed by Iteration 2, was further exaggerated during subsequent optimization iterations. A pause, similar to that observed for the solid ground scenario, was effectively advanced in time and made more pronounced, along with a

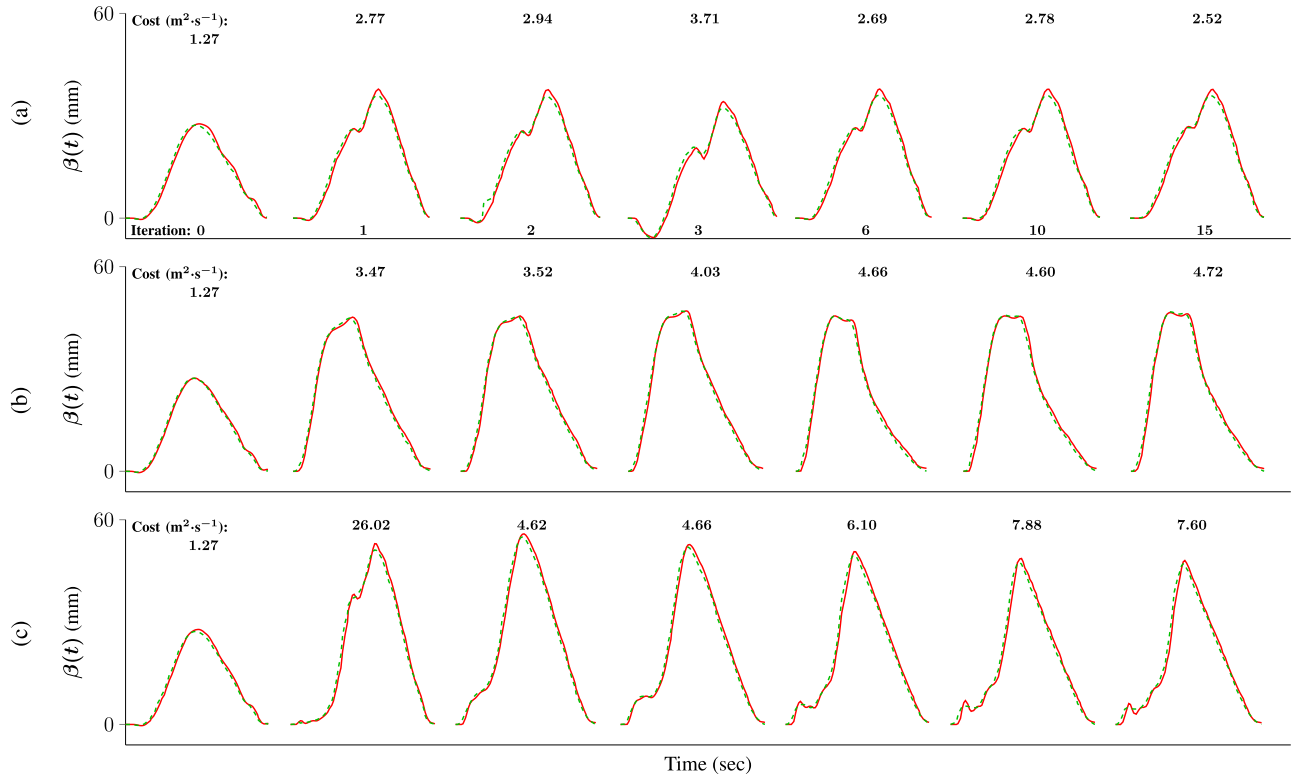


Fig. 9. Synthesized control inputs (a) solid ground, (b) trampoline surface, and (c) GM.

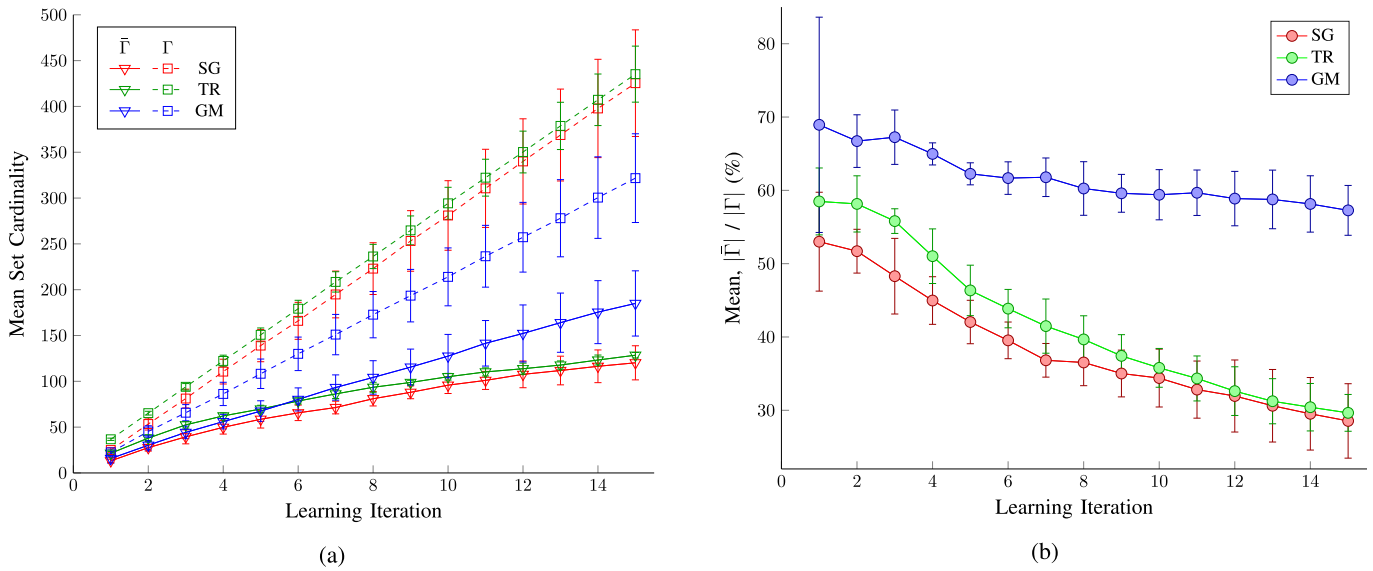


Fig. 10. (a) Mean, over all experimental runs, of each of the reference and reduced training set cardinalities ($|\Gamma|$ and $|\bar{\Gamma}|$, respectively) are compared across 15 control-learning iterations. (b) Mean of $|\bar{\Gamma}|$, as a percentage of $|\Gamma|$, is additionally depicted across successive iterations, demonstrating a linearly decreasing trend; while $|\Gamma|$ grows linearly, $|\bar{\Gamma}|$ grows sublinearly.

more gradual transition from the first short peak to the next larger peak. This feature signified the switch from a single push to a double push strategy as the optimization's means of exploiting the learned terrain dynamics. The converged optimal control solutions inherently depend on the GP-based model of the terrain reaction forces. This computed control input evolves as the GP's understanding of the terrain is incrementally refined.

H. Results: Training Set Reduction

The applied data curation procedure (Section III-D) is designed to target system bottlenecks with respect to computational complexity and further facilitate online application. Computational costs are primarily driven by the cardinality of the training set used to train the GP-based model, (7). Fig. 10(d) shows the accumulated cardinalities for the full reference data set, Γ, and the reduced set, $\bar{\Gamma}$, over the

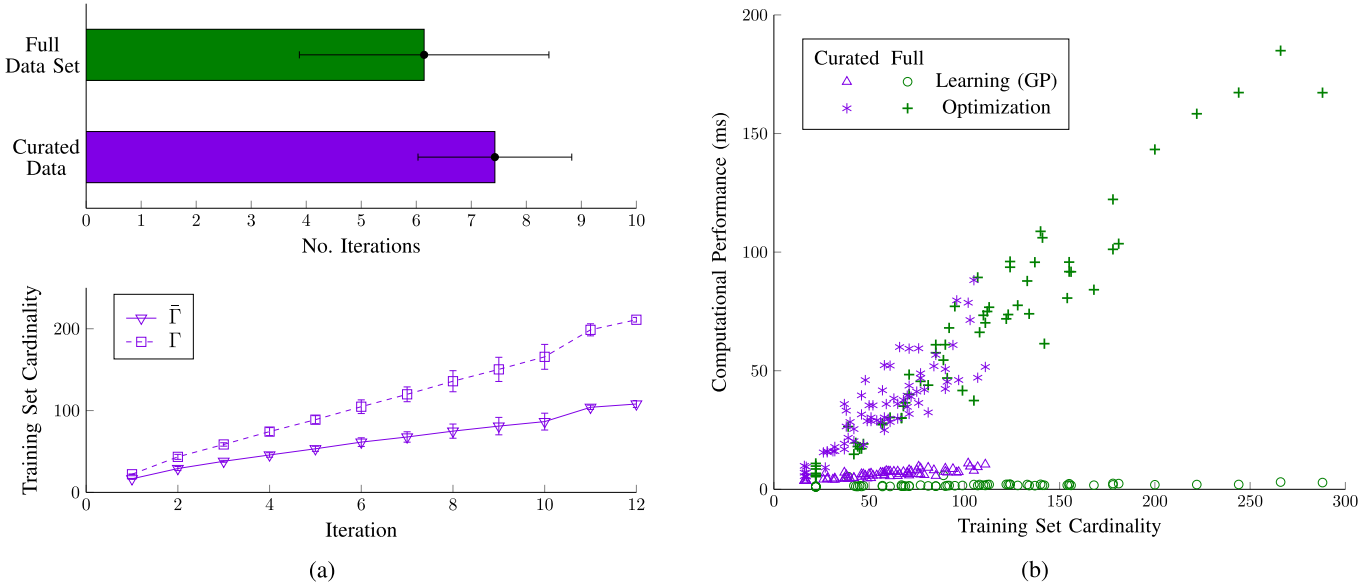


Fig. 11. Simulation analysis of data curation performance impact. (a) *Algorithm Performance:* Comparative experiments were run, in simulation, of Algorithm 1 with and without data curation. For a target jump height of 60 mm on GM, an average of 7.43 and 6.14 iterations were run prior to achieving the target height and learning the terrain model, with and without data curation, respectively (top). Mean iteration-to-iteration training set cardinalities, $|\Gamma|$ and $|\bar{\Gamma}|$, were collected as well (bottom). (b) *Computational Performance:* GP training durations and average CPU time (per step) utilized during gradient descent-based search for optimal trajectory solutions. Measurements corresponding to simulated experiments with and without data curation are indicated by different colored symbols. Time spent to curate the data was recovered by improved optimization iterations.

course of successive control-learning iterations. Mean training set cardinalities across all experiments for each terrain type, versus the learning iteration, are plotted. Data growth of Γ is linear, whereas the reduced set growth is sublinear, as can be seen by the diminishing percentage retained with respect to the full reference set, $|\bar{\Gamma}|/|\Gamma|$, in Fig. 10(b).

Comparative analyses of computational performance associated with data curation, in Algorithm 1, was further studied in simulation. The robotic hopper and experimental GM terrain were replaced by a simulated model, in which measurement noise was injected to mimic real-world uncertainty in observed hopper state trajectories (and indirectly in subsequently recovered terrain forcing profiles). Fig. 11(a) (top) shows the mean number of control-learning iterations, in Algorithm 1, simulated on the GM terrain prior to attaining the targeted jump height of 60 mm. With data curation in place, an average of 7.43 control-learning iterations was needed. Without data curation, an average of 6.14 iterations was needed. Mean cardinalities describing the full versus reduced data sets, over the course of several iterations, are shown in Fig. 11(a) (bottom). Computational performance is captured in Fig. 11(b). Here, we illustrate measurements of CPU time utilized during both learning and trajectory optimization, with respect to training set cardinality. Measurements collected during simulated experiments that employed data curation are shown in purple; those associated with experiments absent of any data curation are shown in green. Computational timing associated with learning (triangles and circles) does not exceed 11 ms in these experiments; for large data set cardinalities, it remains orders of magnitude less than time spent during even a single step of the gradient descent-based trajectory optimization (asterisks and crosses). Computation associated with both learning and optimization was performed in MATLAB (for both physical and simulated experiments). These results demonstrate, especially at later control-learning iterations where $|\bar{\Gamma}|$ is a fraction of $|\Gamma|$, data curation greatly ameliorates the computational penalty associated with large reference data sets, Γ, without severely increasing the number of control-learning iterations needed to accomplish the specified task.

VI. Conclusion

This work proposed and presented experimental validation of an iterative adaptation framework combining GP modeling with optimal control methods. The approach is especially promising in the context of terradynamic applications, where accurate environmental models are scarce or entail empirical characterization efforts ill-suited for practical, rapid robot deployment. As an example application for learning presumed-unknown dynamics in the context of dynamic robotic maneuvers, we addressed the problem of robotic jumping on dissipative poppy seed GM, energy-restorative trampoline material, and solid ground. With every jump, the robot incrementally learned the terrain dynamics and reoptimized its control strategy to accommodate new understanding. After five jump attempts, the robot reached to within 10% of the targeted jump height. After ten jump attempts, it developed a strategy to leap within 5% of the targeted jump height.

While jumping is a lower dimensional task model for legged locomotion, this work demonstrates that the key tools of model-based legged locomotion control are tractable even when the terrain dynamics, a key component of the model, are initially unknown and learned on-the-fly. Otherwise, the necessity to manually characterize complex deformable terrain would hinder locomotion in natural environments where the class of terrain (or even terrain parameters) can change drastically. This work experimentally demonstrated that dynamic

locomotion tasks can be achieved precisely even when key system quantities begin poorly modeled or entirely unknown. Specifically, incremental learning of the unknown external forcing, during the course of iterative control synthesis and execution, can be accomplished in only a handful of repetitions. Though promising, one aspect not considered was safety or stability during the learning iterations. We aim to consider this in future efforts.

REFERENCES

[1] E. R. Westervelt, J. W. Grizzle, and D. E. Koditschek, "Hybrid zero dynamics of planar biped walkers," *IEEE Trans. Autom. 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," *Int. J. Robot. Res.*, vol. 30, no. 3, pp. 265–279, Mar. 2011.

[3] A. D. Ames, "Human-inspired control of bipedal walking robots," *IEEE Trans. Autom. 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 *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2016, pp. 3887–3892.

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

[6] A. Ramezani, X. Shi, S.-J. Chung, and S. Hutchinson, "Lagrangian modeling and flight control of articulated-winged bat robot," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 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. New York, NY, USA: 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. Robot.*, vol. 23, no. 6, pp. 1184–1199, Dec. 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," *Int. J. Robot. Res.*, vol. 33, no. 10, pp. 1322–1341, Sep. 2014.

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

[11] V. Vasilopoulos, I. S. Paraskevas, and E. G. Papadopoulos, "Compliant terrain legged locomotion using a viscoplastic approach," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2014, pp. 4849–4854.

[12] F. B. Mathis and R. Mukherjee, "Two-mass robot hopping on an elastic foundation: Apex height control," in *Proc. IEEE 1st Int. Conf. Control, Meas. Instrum. (CMI)*, Jan. 2016, pp. 167–171.

[13] D. Koepl and J. Hurst, "Impulse control for planar spring-mass running," *J. Intell. Robotic Syst.*, vol. 74, nos. 3–4, pp. 589–603, Jun. 2014.

[14] H.-J. Kang *et al.*, "Biped walking stabilization on soft ground based on gait analysis," in *Proc. 4th IEEE RAS EMBS Int. Conf. Biomed. Robot. Biomechatronics (BioRob)*, Jun. 2012, pp. 669–674.

[15] W. Bosworth, J. Whitney, S. Kim, and N. Hogan, "Robot locomotion on hard and soft ground: Measuring stability and ground properties *in-situ*," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2016, pp. 3582–3589.

[16] J. Aguilar and D. I. Goldman, "Robophysical study of jumping dynamics on granular media," *Nature Phys.*, vol. 12, pp. 278–283, Nov. 2016.

[17] C. Li, T. Zhang, and D. I. Goldman, "A terradynamics of legged locomotion on granular media," *Science*, vol. 339, no. 6126, pp. 1408–1412, Mar. 2013.

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

[19] D. Nguyen-Tuong, B. Scholkopf, and J. Peters, "Sparse online model learning for robot control with support vector regression," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2009, pp. 3121–3126.

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

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

[22] S. Levine and P. Abbeel, "Learning neural network policies with guided policy search under unknown dynamics," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 1071–1079.

[23] J. Boedecker, J. T. Springenberg, J. Wulfing, and M. Riedmiller, "Approximate real-time optimal control based on sparse Gaussian process models," in *Proc. IEEE Symp. Adapt. Dyn. Program. Reinforcement Learn. (ADPRL)*, Dec. 2014, pp. 1–8.

[24] Y. Pan and E. A. Theodorou, "Data-driven differential dynamic programming using Gaussian processes," in *Proc. Amer. Control Conf. (ACC)*, Jul. 2015, pp. 4467–4472.

[25] Y. Pan, X. Yan, E. Theodorou, and B. Boots, "Scalable reinforcement learning via trajectory optimization and approximate Gaussian process regression," in *Proc. NIPS Workshop Adv. Approx. Bayesian Inference*, 2015, pp. 1–5.

[26] Y. Pan, X. Yan, E. Theodorou, and B. Boots, "Adaptive probabilistic trajectory optimization via efficient approximate inference," 2016, *arXiv:1608.06235*. [Online]. Available: http://arxiv.org/abs/1608.06235

[27] 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.

[28] G. Chowdhary, "Concurrent learning for convergence in adaptive control without persistency of excitation," Ph.D. dissertation, School Aerosp. Eng., Georgia Inst. Technol., Atlanta, GA, USA, 2010.

[29] G. Chowdhary, T. Yucelen, M. Mühlegg, and E. N. Johnson, "Concurrent learning adaptive control of linear systems with exponentially convergent bounds," *Int. J. Adapt. Control Signal Process.*, vol. 27, no. 4, pp. 280–301, Apr. 2013.

[30] H. A. Kingravi, G. Chowdhary, P. A. Vela, and E. N. Johnson, "Reproducing kernel Hilbert space approach for the online update of radial bases in neuro-adaptive control," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 7, pp. 1130–1141, Jul. 2012.

[31] G. Chowdhary, H. A. Kingravi, J. P. How, and P. A. Vela, "Bayesian nonparametric adaptive control using Gaussian processes," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 26, no. 3, pp. 537–550, Mar. 2015.

[32] H. Kingravi, "Reduced-set models for improving the training and execution speed of kernel methods," Ph.D. dissertation, School Elect. Comput. Eng., Georgia Inst. Technol., Atlanta, GA, USA, 2014.

[33] J. Hidalgo-Carrio, D. Hennes, J. Schwendner, and F. Kirchner, "Gaussian process estimation of odometry errors for localization and mapping," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2017, pp. 5696–5701.

[34] C. Cunningham, M. Ono, I. Nesnas, J. Yen, and W. L. Whittaker, "Locally-adaptive slip prediction for planetary rovers using Gaussian processes," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2017, pp. 5487–5494.

[35] A. H. Chang, C. M. Hubicki, J. J. Aguilar, D. I. Goldman, A. D. Ames, and P. A. Vela, "Learning to jump in granular media: Unifying optimal control synthesis with Gaussian process-based regression," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2017, pp. 2154–2160.

[36] 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 *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2016, pp. 1–8.

[37] X. Xiong, A. D. Ames, and D. I. Goldman, "A stability region criterion for flat-footed bipedal walking on deformable granular terrain," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Sep. 2017, pp. 4552–4559.

[38] F. Berkenkamp, R. Moriconi, A. P. Schoellig, and A. Krause, "Safe learning of regions of attraction for uncertain, nonlinear systems with Gaussian processes," in *Proc. IEEE 55th Conf. Decis. Control (CDC)*, Dec. 2016, pp. 4661–4666.

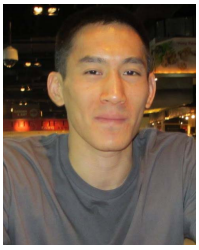
[39] F. Berkenkamp, A. P. Schoellig, and A. Krause, "Safe controller optimization for quadrotors with Gaussian processes," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2016, pp. 493–496.

[40] C. D. McKinnon and A. P. Schoellig, "Learning multimodal models for robot dynamics online with a mixture of Gaussian process experts," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2017, pp. 322–328.

[41] A. H. Chang, C. Hubicki, A. Ames, and P. A. Vela, "Every hop is an opportunity: Quickly classifying and adapting to terrain during targeted hopping," in *Proc. Int. Conf. Robot. Autom. (ICRA)*, May 2019, pp. 3188–3194.

[42] C. Hubicki and J. Hurst, "Running on soft ground: Simple, energy-optimal disturbance rejection," in *Proc. CLAWAR*, 2012, pp. 543–547.

[43] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. Cambridge, MA, USA: Massachusetts Institute of Technology, 2012.
[44] S. Ba and V. R. Joseph, "Composite Gaussian process models for emulating expensive functions," *Ann. Appl. Statist.*, vol. 6, no. 4, pp. 1838–1860, Dec. 2012.
[45] H. A. Kingravi, P. A. Vela, and A. Gray, "Reduced set KPCA for improving the training and execution speed of kernel machines," in *Proc. SIAM Int. Conf. Data Mining*, May 2013, pp. 441–449.
[46] C. R. Hargraves and S. W. Paris, "Direct trajectory optimization using nonlinear programming and collocation," *J. Guid., Control, Dyn.*, vol. 10, no. 4, pp. 338–342, Jul. 1987.
[47] A. V. Rao, "A survey of numerical methods for optimal control," *Adv. Astron. Sci.*, vol. 135, no. 1, pp. 497–528, Aug. 2009.
[48] M. S. Jones, "Optimal control of an underactuated bipedal robot," Ph.D. dissertation, School Mech., Ind., Manuf. Eng., Oregon State Univ., Corvallis, OR, USA, 2014.
[49] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Math. Program.*, vol. 106, no. 1, pp. 25–57, 2006.



Alexander H. Chang (Member, IEEE) received the bachelor's degree in computer and telecommunications engineering and the master's degree in electrical engineering from the University of Pennsylvania, Philadelphia, PA, USA, in 2005 and 2006, respectively. He is currently pursuing the robotics Ph.D. degree with the Intelligent Vision and Automation Laboratory, School of Electrical and Computer Engineering, Institute for Robotics and Intelligent Machines, Georgia Institute of Technology, Atlanta, GA, USA, under the supervision of Professor Patricio A. Vela.

His research interests focus on biologically inspired robotic locomotion, specifically snakelike robotic motion modeling and control.

Christian M. Hubicki (Member, IEEE) received the bachelor's and master's degrees in mechanical engineering, with minor degrees in physics and music, from Bucknell University, Lewisburg, PA, USA, in 2007 and 2011, respectively, and the dual Ph.D. degree in robotics and mechanical engineering from Oregon State University, Corvallis, OR, USA, in 2015.

In 2015, he joined the Georgia Institute of Technology, Atlanta, GA, USA, where he was a Post-Doctoral Fellow with the Department of Mechanical Engineering and the School of Physics. In 2018, he started his position as the Director of the Optimal Robotics Laboratory, Florida State University, Tallahassee, FL, USA, with research specializing in legged robotics and applied optimization. He is currently an Assistant Professor of mechanical engineering with Florida State University and the FAMU-FSU College of Engineering, Tallahassee, FL, USA.

Dr. Hubicki was a recipient of the Gilbreth Lectureship from the National Academy of Engineering in 2020.

Jeffrey J. Aguilar received the B.S., master's, and Ph.D. degrees in mechanical engineering from the Georgia Institute of Technology, Atlanta, GA, USA, in 2010, 2012, and 2016, respectively.

During his graduate program, he worked in the CRAB Laboratory, Georgia Institute of Technology, and utilized a simple robotic system to study the dynamics of jumping on hard substrates and granular media. He is currently Managing Engineering of research and development with Hi Fidelity Genetics, Durham, NC, USA, a startup focused on developing new sensor technologies to collect data for computational crop breeding. His work has involved the creation of the RootTracker, a scalable capacitance-based sensor for mapping in-field root systems.

Daniel I. Goldman received the B.S. degree in physics from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 1994, and the Ph.D. degree with a focus on nonlinear dynamics and granular media from The University of Texas at Austin, Austin, TX, USA, in 2002.

He was a Post-Doctoral Researcher in locomotion biomechanics with the University of California at Berkeley, Berkeley, CA, USA. He became a Faculty Member with the Georgia Institute of Technology, Atlanta, GA, USA, in 2007, where he also became an Adjunct Member with the School of Biology and a member of the Bioengineering Graduate Program. He is currently a Dunn Family Professor with the School of Physics, Georgia Institute of Technology. His research program investigates the interaction of biological and physical systems with complex materials, including granular media.

Prof. Goldman is a Georgia Power Professor of Excellence and a Fellow of the American Physical Society. He received the NSF CAREER/PECASE Award, the DARPA Young Faculty Award, the Sigma Xi Young Faculty Award, and the Burroughs Welcome Fund Career Award at the Scientific Interface.

Aaron D. Ames (Senior Member, IEEE) received the B.S. degree in mechanical engineering and the B.A. degree in mathematics from the University of St. Thomas, St. Paul, MN, USA, in 2001, and the M.A. degree in mathematics and the Ph.D. degree in electrical engineering and computer sciences from the University of California at Berkeley, Berkeley, CA, USA, in 2006.

He was a Post-Doctoral Scholar at the California Institute of Technology, Pasadena, CA, USA, from 2006 to 2008. He began his faculty career at Texas A&M University, College Station, TX, USA, in 2008. He was subsequently an Associate Professor in mechanical engineering and electrical and computer engineering with the Georgia Institute of Technology, Atlanta, GA, USA. He is currently the Bren Professor of mechanical and civil engineering and control and dynamical systems at the California Institute of Technology. His laboratory designs build and test novel bipedal robots, exoskeletons, and prostheses with the goal of achieving human-like robotic locomotion. His research interests span the areas of robotics, safety-critical nonlinear control, and hybrid systems.

Dr. Ames received the NSF CAREER, Eckman, and Ruberti awards, among others.

Patricio A. Vela (Member, IEEE) received the B.Sc. and Ph.D. degrees in control and dynamical systems from the California Institute of Technology, Pasadena, CA, USA, in 1998 and 2003, respectively, where his graduate research focused on geometric nonlinear control and robotics.

In 2004, he was a Post-Doctoral Researcher in computer vision with the School of Electrical and Computer Engineering (ECE), Georgia Institute of Technology, Atlanta, GA, USA, He joined the ECE Faculty, Georgia Institute of Technology, in 2005, where he is currently an Associate Professor with the School of ECE, Institute of Robotics and Intelligent Machines. His research interests lie in the geometric perspectives to control theory, computer vision, the role that computer vision can play for achieving control-theoretic objectives of (semi-)autonomous systems, and the control of nonlinear systems, typically robotic systems.