Active Learning of Dynamics for Data-Driven Control Using Koopman Operators

Ian Abraham and Todd D. Murphey

***Abstract*—This paper presents an active learning strategy for robotic systems that takes into account task information, enables fast learning, and allows control to be readily synthesized by taking advantage of the Koopman operator representation. We first motivate the use of representing nonlinear systems as linear Koopman operator systems by illustrating the improved model-based control performance with an actuated Van der Pol system. Information-theoretic methods are then applied to the Koopman operator formulation of dynamical systems where we derive a controller for active learning of robot dynamics. The active learning controller is shown to increase the rate of information about the Koopman operator. In addition, our active learning controller can readily incorporate policies built on the Koopman dynamics, enabling the benefits of fast active learning and improved control. Results using a quadcopter illustrate single-execution active learning and stabilization capabilities during free fall. The results for active learning are extended for automating Koopman observables and we implement our method on real robotic systems.**

***Index Terms*—Active learning, information theoretic control, Koopman operators, single execution learning.**

I. Introduction

IN ORDER to enable active learning for robots, we need a control algorithm that readily incorporates task information, learns dynamic model representation, and is capable of incorporating policies for solving additional tasks during the learning process. In this paper, we develop an active learning controller that enables a robot to learn an expressive representation of its dynamics using Koopman operators [1]–[4]. Koopman operators represent a nonlinear dynamical system as a linear, infinite dimensional, system by evolving functions of the state (also known as function observables) in time [1]–[4]. Often, these linear representations can capture the behavior of the dynamics globally, while enabling the use of known linear quadratic (LQ) control methods. As a result, the Koopman operator representation changes how we represent the dynamic constraints of the robotic systems, carrying more nonlinear dynamic information, and often improving control authority.

Koopman operator dynamics are typically found through data-driven methods that generate an approximation to the theoretical infinite-dimensional Koopman operator [2], [4], [5]. These data-driven methods require robotic systems to be actuated in order to collect data. The process for data collection in robotics is an active process that relies on control; therefore, learning the Koopman operator formulation, for robotics, is an active learning process.

In this paper, we use the Koopman operator representation for improving control authority of nonlinear robotic systems. Moreover, we address the problem of calculating the linear representation of the Koopman operator by exploiting an information theoretic active learning strategy based on the structure of Koopman operators. As a result, we are able to demonstrate active learning through data-driven control in real-time settings, where only a single execution of the robotic system is possible. Thus, the contribution of this paper is a method for active learning of Koopman operator representations of nonlinear dynamical systems, which exploits both information-theoretic measures and improved control authority based on Koopman operators.

A. History and Related Work

Active learning in robotics has recently been a topic of interest [6]–[10]. Much work has been done in active learning for parameter identification [11]–[14] as well as active learning for state-control mappings in reinforcement learning [9], [15]–[18] and adaptive control [19]–[21]. In particular, much of the mentioned work refers to exciting a robot's dynamics —using information theoretic measures [10], [12], [13], reward functions [9], [10], [15], [17] in reinforcement learning, and other methods [22], [23]—in order to obtain the "best" set of measurements that resolve a parameter or the "best-case" mapping (either of the state-control map or of the dynamics). This paper uses active learning to enable robots to learn Koopman operator representations of a robot's own dynamic process.

Koopman operators were first proposed in 1931 in work by Koopman [1]. At the time, approximating the Koopman operator was computationally infeasible; the onset of computers enabled data-driven methods that approximate the Koopman operator [2], [4], [24]. Other research involves computation of Koopman eigenfunctions and Koopman-invariant subspaces that determine the size of the Koopman operator [25]–[27]. This allows for finite dimensional Koopman operators that captures nonlinear dynamics, while compressing the overall state dimension used to represent the dynamical system.

Recent works, on combining model-based control methods and Koopman operators have suggested that control based on

Manuscript received September 15, 2018; accepted May 25, 2019. Date of publication July 10, 2019; date of current version October 1, 2019. This paper was recommended for publication by Associate Editor D. Kulic and Editor A. Billard upon evaluation of the reviewers' comments. This work was supported by the National Science Foundation under Award NSF CPS 1837515. *(Corresponding author: Ian Abraham.)*

The authors are with the Neuroscience and Robotics Lab (NxR) at the Department of Mechanical Engineering, Northwestern University, Evanston, IL 60208 USA (e-mail: i-abr@u.northwestern.edu; t-murphey@northwestern.edu).

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

Digital Object Identifier 10.1109/TRO.2019.2923880

Koopman operators is a promising avenue for many fields including robotics [3], [5], [26]–[33]. In particular, recent work from the authors implemented a controller using a Koopman operator representation of a robotic system in an experimental setting of a robot in sand [32]. Koopman operators are closely related to latent variable (embedded) dynamic models [34]. In embedded dynamic models, an autoencoder [34], [35] is used to compress the original state space into a lower-dimensional representation. The embedded dynamics model then only evolves the states that are useful for predicting the overall dynamical systems behavior. Koopman operators represent the state of some dynamical system in a higher or lower dimensional representation, where the evolution of the embedding is a linear dynamical systems. Thus, Koopman operators are a special case of an embedded dynamic model where the latent variable describes the nonlinearities of a dynamical system and are represented as a linear differential equation.

B. *Relation to Previous Work*

We extend previous work in [32] with new examples of control with Koopman operator representations of robotic systems. In addition, we provide an example in Section III, which gives further intuition for the use of Koopman operator dynamics. Moreover, we address design choices when generating a Koopman operator dynamic representation of a robotic systems and provide a methodology toward automating these design choices. Last, we introduce a method for enabling the robot to actively learn Koopman operator dynamics, while taking advantage of LQ approaches for control. We note that there is no overlap with the results and the theoretical content that is presented in this paper with [32].

C. *Outline*

The paper outline is as follows. Section II introduces the Koopman operator and data-driven methods to approximate the Koopman operator from data, including a recursively defined online approach for approximating the Koopman operator. Section III motivates using Koopman operator representations of dynamical systems for control. Section IV introduces a controller that enables robots to learn the Koopman operator dynamics. Simulated results for active learning using our method is provided with comparisons in Section V. Section VI discusses methods for automating the design specifications of the Koopman operator. Last, robot experiments are provided in Section VII, and Section VIII concludes this paper.

II. Koopman Operators

This section introduces the Koopman operator and formulates the Koopman operator for control of robotic systems.

A. *Infinite Dimensional Koopman Operator*

Let us first define the continuous dynamical system, whose state evolution is defined by

$$x(t_i + t_s) = F(x(t_i), u(t_i), t_s) \\ = x(t_i) + \int_{t_i}^{t_i+t_s} f(x(s), u(s))ds \tag{1}$$

where t_i is the ith sampling time and t_s is the sampling interval, $x(t) : \mathbb{R} \to \mathbb{R}^n$ is the state of the robot at time t, $u(t) : \mathbb{R} \to \mathbb{R}^m$ is the applied actuation to the robot at time t, $f(x,u) : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is the unknown dynamics of the robot, and $F(x(t_i), u(t_i), t_s)$ is the mapping, which advances the state $x(t_i)$ to $x(t_i + t_s)$. In addition, let us define an observation function $g(x(t)) : \mathbb{R}^n \to \mathbb{R}^c \in \mathbb{C}$ where $\mathbb{C}$ is the space of all observation functions. The Koopman operator $\mathcal{K}$ is an infinite dimensional operator that directly acts on the elements of $\mathbb{C}$

$$[\mathcal{K}g]\,(x(t_i)) = g(F(x(t_i), u(t_i), t_s)) \tag{2}$$

where $u(t_i), t_s$ are implicitly defined in F such that

$$\mathcal{K}g(x(t_i)) = g(F(x(t_i), u(t_i), t_s)) = g(x(t_{i+1})). \tag{3}$$

In words, the Koopman operator $\mathcal{K}$ takes *any* observation of state $g(x(t_i))$ at time t_i and time shifts the observations, subject to the control $u(t_i)$, to the next observable time t_{i+1}. This formulation assumes equal time spacing $t_s = t_{i+1} - t_i = t_i - t_{i-1}$.

B. *Approximating the Data-Driven Koopman Operator*

The Koopman operator $\mathcal{K}$ is infeasible to compute in the infinite dimensional space. A finite subspace approximation to the operator $\mathfrak{K} \in \mathbb{R}^c \times \mathbb{R}^c$ acting on $\mathcal{C} \subset \mathbb{C}$ is used where we define a subset of function observables (or observations of state) $z(x) = [\psi_1(x), \psi_2(x), \ldots, \psi_c(x)]^\top \in \mathbb{R}^c \subset \mathcal{C}$. Each scalar valued $\psi_i \in \mathbb{C}$ and the span of all ψ_i is the finite subspace $\mathcal{C} \subset \mathbb{C}$. The operator $\mathfrak{K}$ acting on $z(x(t_i))$ is then represented in discrete time as

$$z(x(t_{i+1})) = \mathfrak{K} z(x(t_i)) + r(x(t_i)) \tag{4}$$

where $r(x) \in \mathbb{C}$ is the residual function error. In principle, as $c \to \infty$, the residual error goes to zero [3], [4]; however, it is sometimes possible to find $c < \infty$ such that $r(x) = 0$ [26]. Equation (4) gives us the discrete time transition of observations of state in time. We overload the notation for the Koopman operator and write the differential equation for the observations of state as

$$\dot{z} = \mathfrak{K} z(x(t_i)) + r(x(t_i)) \tag{5}$$

where the continuous time $\mathfrak{K}$ is acquired by taking the matrix logarithm as $t_{i+1} - t_i \to 0$.

Provided a dataset $\mathcal{D} = \{x(t_m)\}_{m=0}^{M}$, we can compute the approximate Koopman operator $\mathfrak{K}$ using least-squares minimization over the parameters of $\mathfrak{K}$

$$\min_{\mathfrak{K}} \frac{1}{2} \sum_{m=0}^{M-1} \| z(x(t_{m+1}) - \mathfrak{K} z(x(t_m)) \|^2. \tag{6}$$

Since (6) is convex in $\mathfrak{K}$, the solution is given by

$$\mathfrak{K} = AG^\dagger \tag{7}$$

where † denotes the Moore–Penrose pseudoinverse and

$$A = \frac{1}{M} \sum_{m=0}^{M-1} z(x(t_{m+1}) z(x(t_m))^\top \\ G = \frac{1}{M} \sum_{m=0}^{M-1} z(x(t_m)) z(x(t_m))^\top. \tag{8}$$

The continuous time operator is then given by $\log(\mathfrak{K})/t_s$. Note that we can solve (6) using gradient descent methods [36] or other optimization methods. We write a recursive least-squares update [20], [37] which adaptively updates $\mathfrak{K}$ as more data are acquired.

C. Koopman Operator for Control

The Koopman operator can include a predefined input u that contributes to the evolution of $z(x(t))$. Consider the observable functions that include the control input, $v(x, u) : \mathbb{R}^x \times \mathbb{R}^m \rightarrow \mathbb{R}^{c_u}$ where $c = c_x + c_u$. The resulting computed Koopman operator can be divided into submatrices

$$\mathfrak{K} = \begin{bmatrix} \mathfrak{K}_x & \mathfrak{K}_u \\ \cdot & \cdot \end{bmatrix} \tag{9}$$

where $\mathfrak{K}_x \in \mathbb{R}^{c_x \times c_x}$ and $\mathfrak{K}_u \in \mathbb{R}^{c_x \times c_u}$. Note that the term $(\cdot)$ in (9) refers to terms that evolve the observations on control z_u, which are ignored as there is no ambiguity in their evolution (they are determined by the controller). The Koopman operator dynamical system with control is then

$$\dot{z} = f(z, u) = \mathfrak{K}_x z(x(t_i)) + \mathfrak{K}_u v(x(t_i), u(t_i)). \tag{10}$$

Note that the dataset $\mathcal{D}$ must now store $u(t_i), u(t_{i+1})$ in order to compute the Koopman operator matrix $\mathfrak{K}_u$.

III. Enhancing Control Authority With Koopman Operators

Koopman operators map dynamic constraints into a linear dynamical system in a modified state space. The Koopman operator structure allows one to use LQ control methods to compute optimal controllers for nonlinear systems that can often outperform locally optimal LQ controllers obtained through linearizing the nonlinear dynamics model.

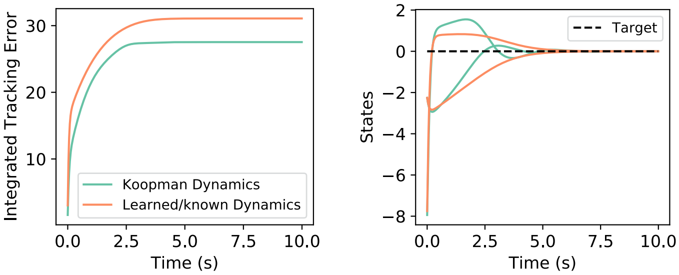
Let us consider control of the nonlinear forced Van der Pol oscillator, the dynamics of which are defined in Appendix A-A, as an example. We specify the control task as minimizing the following LQ objective:

$$J = \int_{t_i}^{t_i+T} x(t)^\top \mathbf{Q} x(t) + u(t)^\top \mathbf{R} u(t) dt + x(t_i+T)^\top \mathbf{Q}_f x(t_i+T) \tag{11}$$

where $\mathbf{Q} \in \mathbb{R}^{n \times n}$, $\mathbf{R} \in \mathbb{R}^{m \times m}$, and $\mathbf{Q}_f \in \mathbb{R}^{n \times n}$. Choosing the set of function observable (see Appendix A-A), we can compute a Koopman operator $\mathfrak{K}$ by repeated simulation of the Van der Pol oscillator subject to uniformly random control inputs for 5000 randomly sampled initial conditions.

Since the Van der Pol oscillator dynamics are nonlinear, a solution to the LQ control problem is to linearize the dynamics about the equilibrium state $x_t = [0, 0]^\top$ and form an LQ control regulator (LQR). Using the Kooman operator formulation of the Van der Pol dynamics, we can compute a controller in a similar manner using the following objective:

$$J = \int_{t_i}^{t_i+T} z(t)^\top \tilde{\mathbf{Q}} z(t) + u(t)^\top \mathbf{R} u(t) dt + z(t_i+T)^\top \tilde{\mathbf{Q}}_f z(t_i+T) \tag{12}$$



(a) (b)

Fig. 1. Control performance of a forced van der pol oscillator with an LQR control using the learned Koopman operator, the linearization of the known system dynamics, and the linearizion of a learned state-space model using the same data and basis functions as the Koopman operator. The control performance using the Koopman operator dynamics is shown to outperform the LQR control with known dynamics. The learned dynamics model performs equally to the known dynamics model and is overlayed on top of the known dynamics results. (a) Int. tracking error. (b) Trajectory.

where

$$\tilde{\mathbf{Q}} = \begin{bmatrix} \mathbf{Q} & \mathbf{0} \\ \mathbf{0} & 0 \end{bmatrix} \in \mathbb{R}^{c_x \times c_x} \text{ and } \tilde{\mathbf{Q}}_f = \begin{bmatrix} \mathbf{Q}_f & \mathbf{0} \\ \mathbf{0} & 0 \end{bmatrix} \in \mathbb{R}^{c_x \times c_x}. \tag{13}$$

Setting $\tilde{\mathbf{Q}}$ and $\tilde{\mathbf{Q}}_f$ to only include the state observables allows us to compare the same control objective using the linearized dynamics against the Koopman operator dynamics where the first terms in the function observable $z(x(t))$ is the state of the Van der Pol system itself.

Fig. 1 shows the improvement in control performance when using the the Koopman operator dynamics for LQ control instead of linearizing the dynamics around a local region. We compare the control authority using a learned dynamics model in the original state space using Bayesian optimization with the same functions used for the Koopman operator. This illustrates that the data used to compute the Koopman operator can learn a nonlinear model of the Van der Pol dynamics in the original state space. The Koopman operator formulation of the Van der Pol approximates the dynamic constraints as a linear dynamical systems in a higher dimensional space that captures nonlinear dynamical behavior. As a result, the Koopman operator formulation coupled with LQ methods can be used to enhance the control the Van der Pol system as shown in Fig. 1(b). Computing the resulting trajectory error [see Fig. 1(a)] shows that the trajectory taken from the Koopman operator controller results in less overall integrated error. This is due to formulating the LQ controller with additional information in the form of a dynamical systems that evolves functions of state.

While this example illustrates the possible benefits of utilizing the Koopman operator formulation, we ignored how the data were collected for the Van der Pol dynamical system. In fact, computing the Koopman operator used random inputs. For this example, such an approach works reasonably, but requires a significant amount of data to fully cover the state space of the Van der Pol system. The following sections introduce a method that enables a robot to actively learn the Koopman operator.

IV. Control Synthesis for Active Learning of Koopman Operator Dynamics

Active learning controllers need to consider existing polices that solve a task while generalizing to learning objectives. In this section, we formulate a controller for active learning that takes into account the Koopman operator dynamics as well as polices generated for solving tasks using the Koopman operator linear dynamics. We generate an active learning controller that takes into account existing policies by first deriving the mode insertion gradient [38], [39]. The mode insertion gradient calculates how an objective changes when switching from one control strategy to another. We then formulate an active learning controller by minimizing the mode insertion gradient, while including policies that solve a specified task.[1] The derived controller is then shown to increase the rate of change of the information measure, which guides the robot toward important regions of state space, improving the data collection and the quality of the learned Koopman operator dynamics model.

A. Control Formulation

Active learning allows a robotic agent to self-excite the dynamical states in order to collect data that result in a Koopman operator $\mathfrak{K}$ that can be used describe system evolution. We formulate the active learning problem as a hybrid switching problem [41], where the goal is to switch between a policy for a task to an information maximizing controller that assists the dynamical system in collecting informative data.

Consider a general objective function of the form

$$J = \int_{t_i}^{t_i+T} \ell(z(s), \mu(z(s)))ds + m(z(t_i+T)) \tag{14}$$

where $z(t) : \mathbb{R} \to \mathbb{R}^{c_x}$ is the value of the function observables at time t subject to the Koopman dynamics in (10) starting from initial condition $z(x(t_i))$, $\ell(z,u) : \mathbb{R}^{c_x} \times \mathbb{R}^m \to \mathbb{R}$ is the running cost, $m(z) : \mathbb{R}^{c_x} \to \mathbb{R}$ is the terminal cost, and $\mu(z) : \mathbb{R}^{c_x} \to \mathbb{R}^{c_u}$ is a $\mathcal{C}^1$ differentiable policy. In this paper, the running cost is split into two parts

$$\ell(z,u) = \ell_{\text{learn}}(z,u) + \ell_{\text{task}}(z,u)$$

where ℓ_{learn} is the information maximizing objective (learning task) and $\ell_{\text{task}}(z,u)$ is the task objective for which the policy $\mu(z)$ is a solution to (14) when $\ell_{\text{learn}} = 0$.

Given (14), we want to synthesize a controller that is bounded to the policy $\mu(z)$, but also allows for improvement of an information measure for active learning. To do so, we examine in Proposition (1) how sensitive (14) is to switching between the policy $\mu(z)$ to an arbitrary control vector $\mu_\star(t)$ at time τ for a time duration λ.

Proposition 1: The sensitivity of switching from μ to $\mu_\star$ for all $\tau \in [t_i, t_i+T]$ for an infinitesimally small λ, (also known as the mode insertion gradient [38], [39]) is given by

$$\frac{\partial J}{\partial \lambda}\Big|_{\tau,\lambda=0} = \rho(\tau)^\top (f_2 - f_1) \tag{15}$$

where $z(t)$ is a solution to 10 with $u(t) = \mu(z(t))$ and $z(t_i) = z(x(t_i))$, $f_2 = f(z(\tau), \mu_\star(\tau))$, $f_1 = f(z(\tau), \mu(z(\tau)))$, and

$$\dot{\rho} = -\left(\frac{\partial \ell}{\partial z} + \frac{\partial \mu}{\partial z}^\top \frac{\partial \ell}{\partial u}\right) - \left(\frac{\partial f}{\partial z} + \frac{\partial f}{\partial u}\frac{\partial \mu}{\partial z}\right)^\top \rho \tag{16}$$

subject to the terminal condition $\rho(t_i+T) = \frac{\partial}{\partial z} m(z(t_i+T))$.

Proof: See Appendix B-A. ■

We can write an unconstrained optimization problem for calculating $\mu_\star(\tau)$ over the interval $\tau \in [t_i, t_i+T]$ that will minimize the mode insertion gradient. We can write this optimization problem using a secondary objective function

$$J_2 = \int_{t_i}^{t_i+T} \frac{\partial J}{\partial \lambda}\Big|_{\tau=t,\lambda=0} + \frac{1}{2}\|\mu_\star(t) - \mu(z(t))\|^2_{\tilde{\mathbf{R}}} dt \tag{17}$$

where $\tilde{\mathbf{R}} \in \mathbb{R}^{m\times m}$ bounds the change of $\mu_\star$ to $\mu(z)$, and $\frac{\partial J}{\partial \lambda}\Big|_{\tau=t,\lambda=0}$ is evaluated at $\tau = t$. Solving (17) with respect to $\mu_\star(t)$ can be viewed as a functional optimization over $\mu_\star(t) \forall t \in [t_i, t_i+T]$. Since (17) is quadratic in $\mu_\star$, we can compute a closed-form solution for any application time $\tau \in [t_i, t_i+T]$.

Proposition 2: Assuming that $v(x,u)$ is differentiable, the control solution that minimizes (17) is

$$\mu_\star(t) = -\tilde{\mathbf{R}}^{-1}\left(\mathfrak{K}_u \frac{\partial v}{\partial u}\right)^\top \rho(t) + \mu(z(t)). \tag{18}$$

Proof: Since (17) is separable in time, we take the derivative of (17) with respect to $\mu_\star(t)$ at each point in t, which gives the following expression:

$$\begin{aligned}\frac{\partial}{\partial \mu_\star} J_2 &= \int_{t_i}^{t_i+T} \frac{\partial}{\partial \mu_\star}(\rho(t)^\top (f_2 - f_1)) + \tilde{\mathbf{R}}(\mu_\star(t) - \mu(z(t))dt \\ &= \int_{t_i}^{t_i+T} \left(\mathfrak{K}_u \frac{\partial v}{\partial u}\right)^\top \rho(t) + \tilde{\mathbf{R}}(\mu_\star(t) - \mu(z(t)))dt.\end{aligned} \tag{19}$$

Solving for $\mu_\star(t)$ in (19) gives the control solution

$$\mu_\star(t) = -\tilde{\mathbf{R}}^{-1}\left(\mathfrak{K}_u \frac{\partial v}{\partial u}\right)^\top \rho(t) + \mu(z(t)).$$

■

Proposition (2) gives a formula for switching from $\mu_\star(t)$ to improve the objective (14). We can use (18) with Appendix B-A to show that our approach improves the active learning objective subject to bounds placed on arbitrary tasks included in (14).

Corollary 1: Assume that the Koopman operator dynamics for a system are defined by the following control affine structure:

$$\dot{z} = \mathfrak{K}_x z(x(t)) + \mathfrak{K}_u v(x(t))u(t) \tag{20}$$

where $v(x) : \mathbb{R}^n \to \mathbb{R}^{c_u \times m}$.[2] Moreover, assume that $\frac{\partial}{\partial \mu}\mathcal{H} \neq 0$ where $\mathcal{H}$ is the control Hamiltonian for (14). Then

$$\frac{\partial}{\partial \lambda} J = -\|\left(\mathfrak{K}_u v(x)\right)^\top \rho\|^2_{\tilde{\mathbf{R}}^{-1}} < 0 \tag{21}$$

for $\mu_\star(t) \in \mathcal{U}\ \forall t \in [t_i, t_i+T]$ where $\mathcal{U}$ is the control space.

[1] During training, the policies derived from the Koopman operator dynamics will be inaccurate; however, over time and gathered experience, both the model and policy will converge. This is a common approach in most model based reinforcement learning techniques [40].

[2] This formulation assumes that we can recover $x(t)$ from $z(t)$ for computing $v(x)$.

Proof: Inserting (18) into Appendix B-A gives

$$\frac{\partial}{\partial \lambda} J = \rho(t)^\top \left(\mathfrak{K}_u v(x(t))\right) \left(-\tilde{\mathbf{R}}^{-1} \left(\mathfrak{K}_u v(x(t))\right)^\top \rho(t)\right)$$

which can be written as the norm

$$\frac{\partial}{\partial \lambda} J = -\| \left(\mathfrak{K}_u v(x)\right)^\top \rho\|^2_{\tilde{\mathbf{R}}^{-1}} < 0.$$

■

Because we define our objective to be reasonably general, we can add both stabilization terms as well as information measures that allow a robot to actively identify its own dynamics. Section IV-B provides an overview of the Fisher information measure and information bounds based on our controller. We first describe the Fisher information matrix for the Koopman operator parameters and then generate an information measure. We then show that using (18) and Corollary 1, that we can approximately calculate to first order the gain in information.

B. Information Maximization

Using the controller defined in (18), we investigate information measures that we can use in (14) to enable the robot to actively learn the Koopman operator dynamics. In this paper, we use the Fisher information [42], [43] to generate an information measure for active learning. The Fisher information is a way of measuring how much information a random variable has about a set of parameters. If we treat calculating the Koopman operator dynamics as a maximum likelihood estimation problem where the likelihood is given by $\pi(z \mid \mathfrak{K}) : \mathbb{R}^{c_x} \to \mathbb{R}^+$, we can compute the Fisher information matrix over the parameters that compose of the Koopman operator $\mathfrak{K}$. The Fisher information matrix is computed as

$$\mathbf{I}\left[z \mid \mathfrak{K}\right] = \mathbb{E}\left[\frac{\partial}{\partial \kappa} \log \pi(z \mid \mathfrak{K})^\top \frac{\partial}{\partial \kappa} \log \pi(z \mid \mathfrak{K})\right] \in \mathbb{R}^{|\kappa|^2} \tag{22}$$

where $\mathbb{E}$ is the expectation operator $\kappa = \{\mathfrak{K}_{i,j} \mid \mathfrak{K}_{i,j} \in \mathfrak{K}\}$, and $|\kappa|$ is the cardinality of the vector κ. Assuming that π is a Gaussian distribution, (22) becomes

$$\mathbf{I}\left[z \mid \mathfrak{K}\right] = \frac{\partial f}{\partial \kappa}^\top \Sigma^{-1} \frac{\partial f}{\partial \kappa} \tag{23}$$

where $\Sigma \in \mathbb{R}^{c_x \times c_x}$ is the noise covariance matrix. Because the Fisher information defined here is positive semidefinite, we use the trace of the Fisher information matrix [44] in $\ell(z, u)$. This measure allows us to synthesize control actions that maximize the T-optimality measure of the Fisher information matrix [44].

Definition 1: The T-optimality measure is given by the trace of the Fisher information matrix (22) and is defined as

$$\mathfrak{I}(\mathfrak{K}) = \text{tr}\, \mathbf{I}\left[z \mid \mathfrak{K}\right] \geq 0. \tag{24}$$

In this paper, we incorporate (24) into (14) additively using $1/(\mathfrak{I} + \epsilon)$, that is

$$\ell_{\text{learn}}(z, u) = 1/(\mathfrak{I}(\mathfrak{K}) + \epsilon)$$

where $\epsilon \ll 1$ is a small number to prevent singular solutions due to the positive semidefinite Fisher information matrix [45]–[47], and $\mathfrak{I}$ is computed using the evaluation of $\mathfrak{K}$ at time t_i. By minimizing (14) we also minimize the inverse of the T-optimality (which maximizes the T-optimality).

Algorithm 1: Active Learning Control.

```
initialize: Objective $\ell(z, u)$, policy $\mu(z)$, normally distributed random $\mathfrak{K} \sim \mathcal{N}(0, \mathbf{1})$.
sample state measurement $x(t_i)$
add $x(t_i)$ to dataset $\mathcal{D}$, update $\mathfrak{K}$ and $\mu(z)$
simulate $z(t), \rho(t)$ for $t \in [t_i, t_i + T]$ with conditions $z(t_i) = z(x(t_i))$ and $\rho(t_i + T) = \frac{\partial}{\partial z} m(z(t_i + T))$ with $\mu(z)$
compute $\mu_\star(t) = -\tilde{\mathbf{R}}^{-1} \left(\mathfrak{K}_u \frac{\partial v}{\partial u}\right)^\top \rho(t) + \mu(z(t))$
return $\mu_\star(t_i)$
update timer $t_i \to t_{i+1}$
```

Assumption 1: Assume that $\mathfrak{I}(\tilde{\mathfrak{K}}) > 0$ implies $\mathfrak{I}(\mathfrak{K}) > 0$ where $\tilde{\mathfrak{K}}$ is an approximation to the Koopman operator $\mathfrak{K}$ computed from the dataset $\mathcal{D} = \{x(t_m), u(t_m)\}_{m=0}^{i}$ that contains data up until the current sampling time t_i.

Theorem 1: Given Assumption 1 and dynamics (20), then the change in information[3] $\Delta \mathbf{I}$ subject to (18) is given to first order

$$\begin{aligned}\Delta \mathbf{I} \approx (\|(\mathfrak{K}_u v(x))^\top \rho\|^2_{\tilde{\mathbf{R}}^{-1}} + \ell_{\text{task}}(z, \mu_\star) \\ - \ell_{\text{task}}(z, \mu))\mathfrak{I}_{\mu_\star} \mathfrak{I}_\mu + \mathcal{O}(\Delta t)\end{aligned} \tag{25}$$

where $\mathfrak{I}_{\mu_\star}$, $\mathfrak{I}_\mu$ is the T-optimality measure (24) from applying the control $\mu_\star$ and μ.

Proof: See Appendix B-B. ■

Theorem 1 shows that our controller increases the rate of information that a robot would have normally acquired if it had only used the control policy $\mu(z)$. Weighing the information measure against the task objective allows us to ensure that the relative information gain is positive when using the active learning controller. That is, the difference between the information from using the policy $\mu(x)$ and the control $\mu_\star(t)$ will be positive. Other heuristics can be used, such as a decaying weight on the information gain or setting the weight to 0 at a specific time so that the robot attempts the task. We provide a basic overview of the control procedure in Algorithm 1. Videos of the experiments and example code can be found online.[4]

The following sections use our derived controller to enable active learning of Koopman operator dynamics.

V. Single Execution Active Learning of Free-Falling Quadcopters

In this example, we illustrate the capabilities of combining the Koopman operator representation of a dynamical systems and active learning for single execution model learning of a free-falling quadcopter for stabilization. Additionally, we compare our approach to other common learning strategies, such as active learning with Gaussian processes [48]–[50], online model adaptation through direct attempts at the tasks of stabilization

[3] With respect to the information acquired from applying only $\mu(z)$.

[4] [Online]. Available: https://sites.google.com/view/active-learning-koopman-op.

(common online reinforcement learning and adaptive control approach [19], [20], [37], [51], [52]), and a two-stage noisy motor input (often referred to as "motor babble" [53]–[55]).

A. Problem Statement

The task is as follows. The quadcopter, with dynamics described in Appendix A-B and [56], must learn a model within the first second of free falling and then use the model to generate a stabilizing controller, preventing itself from falling any further. We define success of the quadcopter in the task when $\|x - x_d\|^2 < 0.01$ where x_d is the desired target state defined by zero linear and angular velocity. The controllers are designed as LQ regulators using the model that was learned and the LQ objectives provided in Section III. The parameters used for this example are defined in Appendix A-B and follows the same parameter choices as in Section III for fairness in terms of the learning methods against which we are comparing.

We compare the information gained (based on the T-optimality condition) and the stabilization error in time against various learning strategies. Each learning strategy is tested with the same 20 uniformly sampled initial velocities (and angular velocities) between -2 and 2 rad/m/s. After each trial, the learned dynamics model is reset so that no information from the previous trials are used.

B. Other Active Learning Strategies

We compare our method for active learning against common dynamic model learning strategies. Specifically, we compare three model learning approaches against our method, a two-stage noisy control input approach [53], a direct stabilization with adaptive model using least squares [19], [37], and an active learning strategy using a Gaussian process [57], [58]. Each of these strategies are generating a Koopman operator using the functions of state defined in Appendix A-B to generate a dynamic model of the quadcopter. The Gaussian process formulation is the only model where the functions map to the original state space resulting in a nonlinear dynamics model.

1) Least Squares Adaptive Stabilization: The first strategy we compare to is to do the task of stabilization at the while updating the model of the dynamics recursively [19], [37]. This is often a strategy used in model-based reinforcement learning [54] and adaptive control [37].

2) Two-Stage Motor Babble: The second strategy is a two-stage approach using noisy motor input (motor babble) for the first second and then pure stabilization [53]. Rather than directly attempting to stabilize the dynamics, the priority is to simply try all possible motor inputs regardless of the model of the dynamics that is being constructed. The motor babble strategy allows us to bound the motor excitation which prevents the rotor from destabilizing once the learning stage is complete. As with the direct stabilization method, we use a recursive least squares to update the model of the Koopman operator.

3) Active Learning With Gaussian Process: The last strategy is an active Gaussian process strategy [57], [58]. In this active learning strategy, we build a model of the dynamics of the quadcopter by generating a Gaussian process dynamics model [50], [57]. Using the variance estimate [58], we uniformly sample points around the current state bounded by some ϵ constant and find the state which maximizes the variance. The sampled state with the largest variance is then used to generate a local LQ controller to guide the quadcopter dynamics to that state to collect the data. After the first second, the Gaussian process model is used to generate a stabilizing controller by linearizing the model about the final desired stabilization state. The kernel function used is computed using the functions of state provided in Appendix A-B for a fair comparison.

Note that for the two-stage, least-squared adaptive, and our approach, we learn a Koopman operator dynamics model which we use to compute an LQ controller. The Gaussian process model is in in the original state space as described in [50].

C. Results

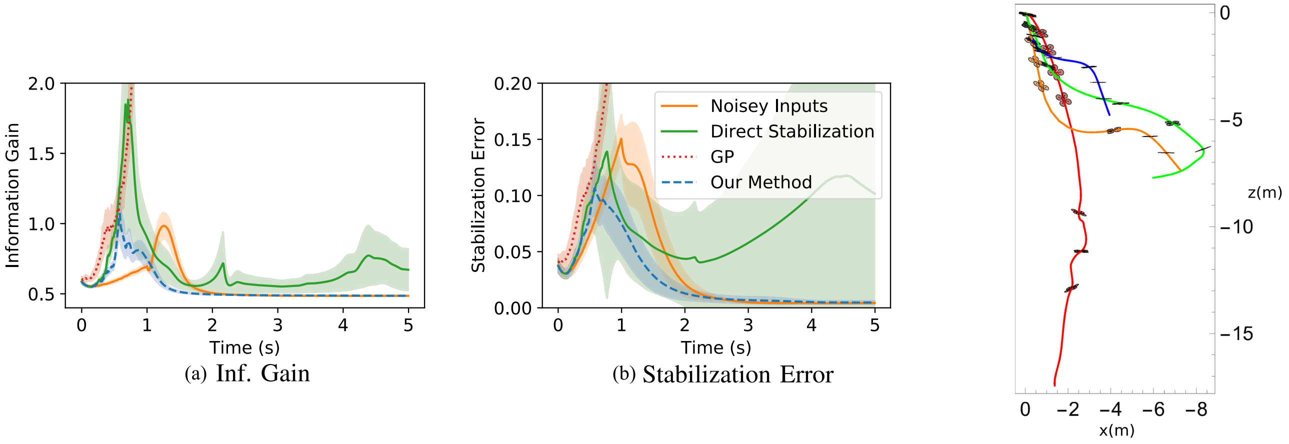
Fig. 2(a) shows the information (T-optimality of the Fisher information matrix) for each method. Our approach to active learning is shown to improve upon the information when compared to motor babble (the most basic method for active learning). The other methods outperform our approach in terms of the overall information gain by overly exciting the dynamics. The direct adaptive stabilization method utilizes the incorrect dynamics model to self-adjust and eventually stabilize the quadcopter (as shown in the variance). The active Gaussian process approach uses the covariance estimate to actuate the quadcopter toward uncertain regions. Collecting data in uncertain regions allows the active Gaussian process approach to actively select where the quadcopter should collect data next.

It is worth noting that these approaches will often lead the quadcopter toward unstable regions, making it difficult to stabilize the dynamics in time. Our approach actively synthesizes when it is best to learn and stabilize which assists in quickly stabilizing the quadcopter dynamics [see Fig. 2(b)]. The addition of the Koopman operator dynamics further enhances the control authority of the quadcopter as shown with the direct adaptive stabilization, motor babble, and our approach to active learning. While the active Gaussian process model does at times succeed, the method relies on both the quality of data acquired and the local linear approximation to the dynamics. This results in a deficit of nonlinear information that is needed to successfully achieve the learning task in a single execution.

D. Sensitivity to Initialization and Parameters

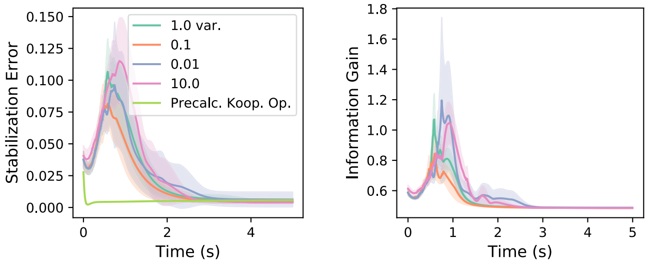
We further test our algorithm against sensitivities to initialization of the Koopman operator. Our algorithm requires an initial guess at the Koopman operator in order to boot strap the active learning process. We accomplish this using the same experiment described in the previous section which used a zero mean, variance of 1 normally distributed initialization of the Koopman operator. We vary the variance that initializes the Koopman operator parameters using a normal distribution with zero mean and a variance experiment set of $\{0.01, 0.1, 1.0, 10.0\}$.

In Fig. 3, we find that so long as the initialization of the Koopman operator is within a reasonable initialization (nonzero and within an order of magnitude), the performance is comparable to active learning described in Fig. 2. However, this may not be true for all autonomous systems and results may vary depending on the sampling frequency and the behavior of the underlying



(a) Inf. Gain (b) Stabilization Error (c) Sample Trajectories

Fig. 2. Monte Carlo simulation comparing various learning strategies to stabilize a quadcopter falling for 20 trials with uniformly sampled initial linear and angular velocities. (a) Information gain (trace of the Fisher information matrix) is shown for the various learning strategies. (b) Stabilization error and standard deviation is shown over time for each learning strategy over 20 trajectories. (c) Representative time series snapshots are shown depicting the various learning strategies. With our approach, maximization of the information measure, coupled with the Koopman operator formulation of the dynamics, enables quick stabilization of the quadcopter.



(a) (b)

Fig. 3. Resulting sensitivities in stabilization error and information gain with respect to variance levels in Koopman operator initialization. Benchmark stabilization performance is provided for known/precalculated Koopman operator. (a) Stab. err. (b) Inf. gain.

system. A benchmark is provided for stabilizing the quadcopter when the Koopman operator is precomputed in Fig. 3 illustrating the performance of the control authority when using the Koopman operator based controller.

The choices in the parameters of our algorithm can also effect its performance. Specifically, setting the value of the regularization term $\tilde{\mathbf{R}}$ too large will prevent the robot from significantly exploring the states of the robot. In contrast, if the regularization term is set too low, the robot will widen its breath of exploration which can be harmful to the robot if the states are not bounded. A similar effect is achieved by adding a weight on the active learning objective.

Changes in the time horizon T will also affect the performance of the algorithm. Generally, smaller T will result in more reactive behaviors where larger T tends to have more intent driven control responses. Choosing these values appropriately will be problem specific; however, the limited number of tunable parameters (not including choosing a task objective) provides the advantage of ease of implementation.

E. Discussion

While the single execution capabilities of the Koopman operator with active learning is appealing, not all robotic systems will be capable of such drastic performance. In particular, this example relies on some prior knowledge of the underlying robotic system and the dynamics that govern the system. The functions of state are chosen such that they include nonlinear elements (e.g., cross product terms that we expect will help in stabilization). Thus, the approximate Koopman operator is predicting the evolution of nonlinear elements found in the original nonlinear dynamics. Often these underlying structures that we can exploit are not known or easily found in robotics. Choosing random polynomial or Fourier expansions as function observables can sometimes work (see Section VII), but often can lead to unstable eigenvalues in the Koopman operator dynamics, which can make model-based control difficult to synthesize [26].

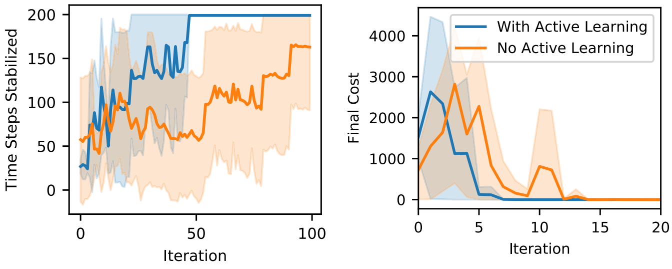
Recent work has attempted to address these issues using sparse optimization [59] or discovering invariances in the state space [26]. A promising method is automating the discovery of the function observables by learning the functions from data [60]. By using current advances in neural networks and function representation, it is possible to automate the discovery of function observables. Section VI further develops the work in automating the discovery of function observables for Koopman operators through the use of our approach for active learning.

VI. Automating Discovery of Koopman operator Function Observables

As a solution to automating the choice of function observables, the use of deep neural networks [60] have been used to automatically discover the function observables. In this section, we illustrate that we can use these neural networks coupled with our approach for active learning to automatically discover the Koopman operator and the associated functions of state.

A. Including Automatic Function Discovery

Revisiting (10), we can parameterize $z(x)$ and $v(x,u)$ using a multilayer neural network with parameters $\theta \in \mathbb{R}^d$. We denote the parameterization of z, v as $z_\theta(x)$ and $v_\theta(x,u)$ where



(a) Cart Pendulum (b) 2-Link Robot

Fig. 4. (a) Resulting stabilization time of a cart pendulum using Koopman operators with automatic function discovery. (b) Control response of a two-link robot using Koopman operators with automatic function discovery. Active learning improves the rate of success of each task.

the subscript θ denotes the function observables are parameterized by the same set of parameters θ. Given the same dataset that was defined previously, $\mathcal{D} = \{x(t_m), u(t_m)\}_{m=0}^{M}$, the new optimization problem that is to be solved is

$$\min_{\mathfrak{K},\theta} \frac{1}{2} \sum_{m=0}^{M-1} \|\tilde{z}_\theta(x(t_{m+1}), u(t_{m+1})) - \mathfrak{K}\tilde{z}_\theta(x(t_m), u(t_m))\|^2 \tag{26}$$

where $\tilde{z}_\theta(x, u) = [z_\theta(x)^\top, v_\theta(x, u)^\top]^\top$. Equation (26) can be solved using any of the current techniques for gradient descent (Adams method [61] is used in this paper). The continuous time Koopman operator is obtained similarly using the matrix log of $\mathfrak{K}$, resulting in the differential equation

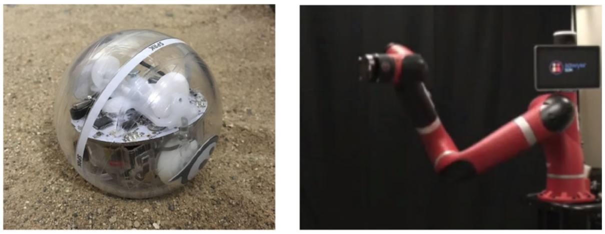
$$\dot{z}_\theta = \mathfrak{K}_x z_\theta(x(t)) + \mathfrak{K}_u v_\theta(x(t), u(t)). \tag{27}$$

Because we are now optimizing over θ, we lose the sample efficiency of single execution learning that was illustrated in the example in Section V. Active learning can be used; however, adding the additional parameters θ to the information measure significantly increases the computational cost of calculating the Fisher information measure (22). As a result, we only compute the information measure with respect to $\mathfrak{K}$ in order to avoid the computational overhead of maximizing information with respect to θ.

B. Examples

We illustrate the use of deep networks for automating the function observables for the Koopman operator for stabilizing a cart pendulum and controlling a two-link robot arm to a target. A neural network is first initialized (see Appendix A-C for details) for the Koopman operator functions z_θ, v_θ as well as an LQ controller for the task at hand. At each iteration, the robot attempts the task and learns the Koopman operator dynamics by minimizing (26). We compare against decaying additive control noise as well as our method for active learning where a weight on information measure is used which decays at each iteration according to γ^{i+1} where $0 < \gamma < 1$ and i is the iteration number. The data collected are then used to update the parameters θ and $\mathfrak{K}$ using (26) and the LQ controller is updated with the new $\mathfrak{K}_x, \mathfrak{K}_u$ parameters.

Fig. 4 shows that we can automate the process of learning the function observables as well as the Koopman operator. With the addition of active learning, the process of learning the Koopman operator and the function observables is improved. In particular, stabilization of the cart pendulum is achieved in only 50 iterations in comparison to additive noise which takes over 100 iterations. Similarly, the two-link robot can be controlled to the target configuration within five iterations with our active learning approach.



(a) (b)

Fig. 5. Depiction of robots used for experimentation. (a) Sphero SPRK. (b) Sawyer robot.

C. Discussion

While this method is promising, there still exist significant issues that merit more investigation in future work. One of which is the trivial solution, where $z_\theta, v_\theta = 0$. This issue often occurs with how the parameters θ were initialized. This trivial solution has been addressed in [62]; however, their approach requires significantly complicating how the regression (26) is formulated. We found that adding the state x as part of the neural network output of z_θ was enough to overcome the trivial solution.

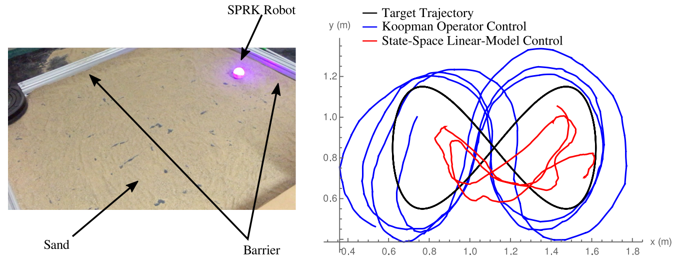
VII. Robot Experiments

Our last set of examples test our active learning strategy with robot experiments. We use the robots shown in Fig. 5 to illustrate control and active learning with Koopman operators. The sphero SPRK robot [see Fig. 5(a)] is a differential drive robot inside of a clear outer ball. We test trajectory tracking of the SPRK robot in a sand terrain where the challenge is that the SPRK must be able to learn how to maneuver in sand. The Sawyer robot [see Fig. 5(b)] is a seven-link robot arm whose task is to track a trajectory defined at the end effector where the challenge is the high dimensionality of the robot. We refer the reader to the attached multimedia which has clips of the experiments.

A. Experiments: Granular Media and Sphero SPRK

Active learning is applied in an experimental setting using the Sphero SPRK robot [see Fig. 5(a)] in sand. The interaction between sand and the SPRK robot makes physics-based models challenging.

The parameters for the experiment are defined in Appendix A-D. The experiment starts with 20 s of active learning. After actively identifying the Koopman operator, the weight on information maximizing is set to zero at $t = 20$ and the objective is switched to track the trajectory shown in Fig. 6(b). In Fig. 6(c), we show the average root mean squared error (RMSE) of the $x - y$ trajectory tracking, the average $x - y$ Pearson's correlation using a two-sided hypothesis testing (values close to 1 indicate responsive controllers), and the phase lag of the experimental results. Note that in contrast to previous work by the



(a) Experimental Setup (b) SPRK Trajectories

Method	RMSE	Correlation	Phase Lag (rad)
Koopman-based Control	0.3010	0.4028	1.1262
Controller in [32]	0.3535	0.1034	1.4667

(c) Controller performance

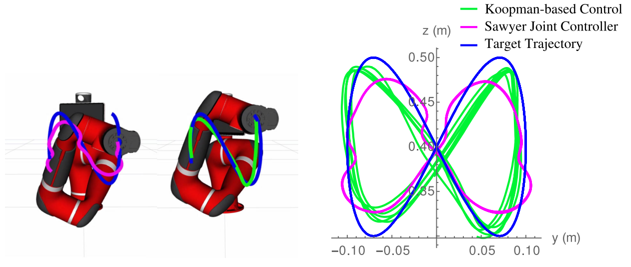
Fig. 6. Experiment using the Sphero SPRK robot in sand. (a) Experimental setup is depicted with the SPRK robot inside the sand pit. Position information is calculated with an overhanging Xbox Kinect using OpenCV [63] for tracking. (b) Performance of the SPRK robot using the Koopman operator-based controller after active learning. Performance is compared with results from [32]. (c) Performance measures showing active learning significantly outperforms nonactive learning in robot experiment. The attached multimedia shows the experiment executed.

authors [32], the method of actively learning the Koopman operator improves the performance of the model-based controller. In particular, we find that the overall responsiveness and phase lag of the Koopman-based controller improved after active learning in sand.

B. Experiments: Trajectory Tracking of Rethink Sawyer Robot

In this experiment, we use active learning with the Koopman operator to model a 7 DoF Sawyer robot arm from Rethink Robotics. The 7-DoF system is of interest because it is both high dimensional and inertial effects tend to dominate the dynamics of the system. We define the parameters used for this experiment in Appendix A-E.

Fig. 7 shows a comparison of the embedded controller in the Sawyer robot and the data-driven Koopman operator controller. Here, we show the average RMSE of the tracking position, the Pearson's correlation using a two-sided hypothesis testing (values close to 1 indicate responsive controllers), and the phase lag of the trajectory tracking. The resulting controller using the Koopman operator is shown to be comparable to the built-in controller with the inclusion of the evolution of the nonlinearities on the Sawyer robot which improve overall trajectory tracking performance. The trajectories of the two methods are overlaid, which illustrates the improvement in control from the Koopman operator after active learning has occurred. Since data are always being acquired online, the Koopman operator is continuously being updated as the robot is tracking the trajectory. The Koopman operator based controller is able to capture dynamic effects of the individual joints from data. This is further reinforced by the improved results found. Note that one can build a model to solve for similar, if not, better, inverse dynamics of the Sawyer robot that can be computed for control. In particular, the Sawyer robot provides an implementation of inverse dynamics in the robot's embedded controller. However, our approach provides high accuracy without needing such a model ahead of time and without linearizing the nonlinear dynamics.



(a) Experimental Visualization (b) Sawyer Trajectories

Method	RMSE	Correlation	Phase Lag (rad)
Koopman-based Control	0.0228	0.9777	0.2826
Sawyer Joint Controller	0.0443	0.6026	0.7041

(c) Controller performance

Fig. 7. Experiment using Sawyer. Experimental data visualized using RViz [64]. (a) End-effector trajectory paths using the embedded Rethink Joint controller and Koopman operator controller. Both controllers are running at 100 Hz. (b) Trajectory overlaid from both controller responses. (c) Controller performance shows that active learning for Koopman operator-based controllers performs comparably. We refer the reader to the attached multimedia to view clips of this experiment.

VIII. Conclusion

In this paper, we use Koopman operators as a method for enhancing control of robotic systems. In addition, we contribute a method for active learning of Koopman operator dynamics for robotic systems. The active learning controller enables the robots to learn their own dynamics quickly, while taking into account the linear structure of the Koopman operator to enhance LQ control. We illustrate various examples of robot control with Koopman operators and provide examples for automating design choices for Koopman operators. Last, we show that our method is applicable to actual robotic systems.

Appendix A
Parameters for Various Examples

A. Control of Forced Van der Pol Oscillator

The nonlinear dynamics that govern the Van der Pol oscillator are given by the differential equations

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} x_2 \\ -x_1 + \epsilon(1 - x_1^2)x_2 + u \end{bmatrix}$$

where $\epsilon = 1$ and u is the control input.

The Koopman operator functions used are defined as

$$z(x) = \left[x_1, x_2, x_1^2, x_2 x_1^2\right]^\top$$

and $v(u) = u$. The same functions are used to compute a regression problem where the final equation is given by

$$\frac{d}{dt}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \mathbf{A}z(x) + \mathbf{B}v(u)$$

where $\mathbf{A} \in \mathbb{R}^{n \times c_x}$ and $\mathbf{B} \in \mathbb{R}^{n \times c_u}$ are both generated using linear regression.

The weight parameters for LQ control are

$$\mathbf{Q} = \text{diag}\left([1, 1]\right) \text{ and } \mathbf{R} = 0.1$$

where

$$\tilde{\mathbf{Q}} = \begin{bmatrix} \mathbf{Q} & \mathbf{0} \\ \mathbf{0} & 0 \end{bmatrix} \in \mathbb{R}^{c_x \times c_x}. \tag{28}$$

B. Quadcopter Free Falling

The quadcopter system dynamics are defined as

$$\begin{aligned} \dot{h} &= h \begin{bmatrix} \hat{\omega} & v \\ \mathbf{0} & 0 \end{bmatrix} \\ J\dot{\omega} &= M + J\omega \times \omega \\ \dot{v} &= \frac{1}{m} F e_3 - \omega \times v - g R^T e_3 \end{aligned}$$

where $h = (R, p) \in \text{SE}(3)$, the inputs to the system are $u = [u_1, u_2, u_3, u_4]$, and

$$F = k_t (u_1 + u_2 + u_3 + u_4)$$

$$M = \begin{bmatrix} k_t l (u_2 - u_4) \\ k_t l (u_3 - u_1) \\ k_m (u_1 - u_2 + u_3 - u_4) \end{bmatrix}$$

(see [56] for more details on the dynamics and parameters used). Note that in this formulation of the quadcopter, the control vector u has bidirectional thrust.

The measurements of the state of the quadcopter are given by

$$[a_g, \omega, v]^\top \in \mathbb{R}^9 \tag{29}$$

where $a_g \in \mathbb{R}^3$ denotes the body centered gravity vector and ω, v are the body angular and linear velocities, respectively. The sampling rate for this system is 200 Hz.

We define the basis functions for this system as

$$z(x) = [a_g, \omega, v, g(v, \omega)]^T \in \mathbb{R}^{18}$$

where $g(v, \omega) = [v_3\omega_3, v_2\omega_3, v_3\omega_1, v_1\omega_3, v_2\omega_1, \; v_1\omega_2, \omega_2\omega_3, \omega_1\omega_3, \omega_1\omega_2]$ are the chosen basis functions such that ω_i, v_i are elements of the body-centered angular and linear velocity ω, v, respectively. The functions for control are

$$v(u) = u \in \mathbb{R}^4.$$

The LQ control parameters for the stabilization problem are given as

$$\mathbf{Q} = \text{diag}\left([1, 1, 1, 1, 1, 1, 5, 5, 5]\right) \text{ and } \mathbf{R} = \text{diag}\left([1, 1, 1, 1]\right)$$

where the weight on the additional functions $\tilde{\mathbf{Q}}$ are set to zero as in (28) . The time horizon used in 0.1s.

The active learning controller uses a weight on the information measure of 0.1 and a regularization weight $\tilde{\mathbf{R}} = \text{diag}(1000, 1000, 1000, 1000])$. Motor noise used in the two-stage method is given by uniform noise at 33% of the control saturation.

C. Neural Network Automatic Function Discovery Configuration

In this example, we use the Roboschool environments [65] for the robot simulations.

For the cart pendulum example, we use a three layer network with a single hidden layer for z_θ and v_θ with $\{4, 20, 40\}$ and $\{2, 20, 10\}$ nodes, respectively, for each layer making $c_x = 40$ and $c_u = 10$. The exploration noise used on the control is given by additive zero mean noise with a variance of 40% motor saturation decreasing at a rate of 0.9^{i+1}. The decay weight on the information measure is given by 0.2^{i+1}. The LQ weights are given by $\tilde{\mathbf{Q}} = \text{diag}([50.0, 1.0, 10.0, 0.1] + \vec{\mathbf{0}})$ where the first nonzero weights correspond to the states of the cart pendulum. A time horizon of 0.1 s is used with a sampling rate of 50 Hz. The regularization weight $\tilde{\mathbf{R}} = 1 \times 10^6$.

For the two-link robot example, we use a similar three layer network with a single hidden layer for z_θ and v_θ with $\{4, 20, 40\}$ and $\{2, 20, 20\}$ nodes, respectively, for each layer making $c_x = 40$ and $c_u = 10$. The exploration noise used on the control is given by additive zero mean noise with a variance of 40% motor saturation decreasing at a rate of 0.9^{i+1}. The decay weight on the information measure is given by 0.2^{i+1}. The LQ weights are given by $\tilde{\mathbf{Q}} = \text{diag}([10.0, 1.0, 20.0, 1.0] + \vec{0})$ where the first nonzero weights correspond to the states of the cart pendulum. A time horizon of 0.05 s is used with a sampling rate of 100 Hz. The regularization weight $\tilde{\mathbf{R}} = \text{diag}([1 \times 10^6, 1 \times 10^6])$.

D. SPRK Tracking in Sand

The SPRK robot is running a 30 Hz sampling rate for control and state estimation. Control vectors are filtered using a low-pass filter to avoid noisy responses in the robot. The controller weights are defined as

$$\tilde{\mathbf{Q}} = \text{diag}([60, 60, 5, 5, \vec{\mathbf{1}}]) \text{ and } \mathbf{R} = \text{diag}([0.1, 0.1]).$$

The control regularization is $\tilde{\mathbf{R}} = \mathbf{R}$. A weight of 80 is added to the information measure. A time horizon of 0.5 s is used to compute the controller.

We run the active learning controller for 20 s and then set the weight of the information measure to zero and track the end effector trajectory given by

$$\begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = \begin{bmatrix} 0.5\cos(t) + 1.12 \\ 0.3\sin(2t) + 0.85 \end{bmatrix}.$$

In this example, the set of functions are chosen as a polynomial expansion of the velocity states $\mathbf{x} = [\dot{x}, \dot{y}]$ to the third order. The function observables are defined as

$$z(x) = \left[x, y, \dot{x}, \dot{y}, 1, \dot{x}^2, \dot{y}^2, \dot{x}^2\dot{y}, \ldots, \dot{x}^3\dot{y}^3\right]^T \in \mathbb{R}^{18}$$

and

$$v(x, u) = u \in \mathbb{R}^2.$$

E. Sawyer Control

The Sawyer robot was run on a sampling rate of 100 Hz. Control vectors are filtered using a low-pass filter to avoid noisy responses in the robot. The controller weights are defined as

$$\tilde{\mathbf{Q}} = \text{diag}([200 \times \vec{\mathbf{1}} \in \mathbb{R}^{14}, \vec{\mathbf{1}}])$$
$$\text{and } \mathbf{R} = \text{diag}([0.001 \times \vec{\mathbf{1}} \in \mathbb{R}^{7}]).$$

The control regularization is $\tilde{\mathbf{R}} = \mathbf{R}$. A weight of 2000 is added to the information measure. A time horizon of 0.5 s is used to compute the controller.

We run the active learning controller for 20 s and then set the weight of the information measure to zero and track the end effector trajectory given by

$$\begin{bmatrix} x(t) \\ y(t) \\ z(t) \end{bmatrix} = \begin{bmatrix} 0.8 \\ 0.1\cos(2t) \\ 0.1\sin(4t) + 0.4 \end{bmatrix}.$$

The functions of state using to compute the Koopman operator are defined as

$$z(x) = \left[\mathbf{x}^T, 1, \theta_1\theta_2, \theta_2\theta_3, \ldots, \theta_6^3\theta_7^3, \dot{\theta}_1\dot{\theta}_2, \ldots, \dot{\theta}_6^3\dot{\theta}_7^3\right]^T \in \mathbb{R}^{51}$$

with $v(u) = u \in \mathbb{R}^7$ as the torque input control of each individual joint and states $\mathbf{x}$ containing the joint angles and joint velocities.

Appendix B
Proofs

A. *Proof of Proposition 1*

Proposition 1: The sensitivity of switching from μ to $\mu_\star$ at any time $\tau \in [t_i, t_i + T]$ for an infinitesimally small λ, (also known as the mode insertion gradient [38], [39]) is given by

$$\frac{\partial J}{\partial \lambda}\bigg|_{\tau, \lambda=0} = \rho(\tau)^\top (f_2 - f_1)$$

where $z(t)$ is a solution to 10 with $u(t) = \mu(z(t))$ and $z(t_i) = z(x(t_i))$, $f_2 = f(z(\tau), \mu_\star(\tau))$, $f_1 = f(z(\tau), \mu(z(\tau)))$, and

$$\dot{\rho} = -\left(\frac{\partial \ell}{\partial z} + \frac{\partial \mu}{\partial z}^\top \frac{\partial \ell}{\partial u}\right) - \left(\frac{\partial f}{\partial z} + \frac{\partial f}{\partial u}\frac{\partial \mu}{\partial z}\right)^\top \rho$$

subject to the terminal condition $\rho(t_i + T) = \frac{\partial}{\partial z} m(z(t_i + T))$.

Proof: Consider the objective (14) evaluated at a trajectory $z(t) \forall t \in [t_i, t_i + T]$ generated from a dynamical system. Furthermore, assume that $z(t_i + T)$ is generated by a policy $\mu(z(t)) \forall t \notin [\tau, \tau + \lambda]$ and a controller $\mu_\star(t) \forall t \in [\tau, \tau + \lambda]$ where τ is the time of application of control $\mu_\star$ and λ is the duration of the control. Formally, $z(t_i + T)$ can be written as

$$\begin{aligned} z(t_i + T) = z(t_i) &+ \int_{t_i}^{\tau} f(z(t), \mu(z(t)))dt \\ &+ \int_{\tau}^{\tau+\lambda} f(z(t), \mu_\star(t))dt \\ &+ \int_{\tau+\lambda}^{t_i+T} f(z(t), \mu(z(t)))dt \end{aligned} \tag{30}$$

where $f(z, u) : \mathbb{R}^{c_x} \times \mathbb{R}^{c_u} \to \mathbb{R}^{c_x}$ is a mapping which describes the time evolution of the state $z(t)$.

Using (14) and (30), we compute the derivative of (14) with respect to the duration λ of control $\mu_\star$ applied at any time $\tau \in [t_i, t_i + T]$

$$\frac{\partial}{\partial \lambda} J \bigg|_{\tau} = \int_{\tau+\lambda}^{t_i+T} \left(\frac{\partial \ell}{\partial z} + \frac{\partial \mu}{\partial z}^\top \frac{\partial \ell}{\partial u}\right)^\top \frac{\partial z}{\partial \lambda} dt \tag{31}$$

where

$$\frac{\partial z(t)}{\partial \lambda} = f_2 - f_1 + \int_{\tau+\lambda}^{t} \left(\frac{\partial f}{\partial z} + \frac{\partial f}{\partial u}\frac{\partial \mu}{\partial z}\right)^\top \frac{\partial z(s)}{\partial \lambda} ds \tag{32}$$

such that $f_2 = f(z(\tau), \mu_\star(\tau))$, $f_1 = f(z(\tau), \mu(z(\tau)))$ are boundary terms from applying Leibniz's rule.

Because (32) is a linear convolution with initial condition, $\frac{\partial z(\tau)}{\partial \lambda} = f_2 - f_1$, we are able to rewrite the solution to $\frac{\partial z(t)}{\partial \lambda}$ using a state-transition matrix $\Phi(t, \tau)$ [66] with initial condition $f_2 - f_1$ as

$$\frac{\partial z(t)}{\partial \lambda} = \Phi(t, \tau)\left(f_2 - f_1\right). \tag{33}$$

Since the term $f_2 - f_1$ is evaluated at time τ, we can write (31) as

$$\frac{\partial}{\partial \lambda} J \bigg|_{\tau} = \int_{\tau+\lambda}^{t_i+T} \left(\frac{\partial \ell}{\partial z} + \frac{\partial \mu}{\partial z}^\top \frac{\partial \ell}{\partial u}\right)^\top \Phi(t, \tau)dt \left(f_2 - f_1\right). \tag{34}$$

Taking the limit of (34) as $\lambda \to 0$ gives us the sensitivity of (14) with respect to switching at any time $\tau \in [t_i, t_i + T]$. We can further define the adjoint (or costate) variable

$$\rho(\tau)^\top = \int_{\tau}^{t_i+T} \left(\frac{\partial \ell}{\partial x} + \frac{\partial \mu}{\partial x}^\top \frac{\partial \ell}{\partial u}\right)^\top \Phi(t, \tau)dt \in \mathbb{R}^{c_x}$$

which allows us to define the mode insertion gradient [39] as

$$\frac{\partial}{\partial \lambda} J \bigg|_{t=\tau} = \rho(\tau)^\top \left(f_2 - f_1\right)$$

where

$$\dot{\rho} = -\left(\frac{\partial \ell}{\partial z} + \frac{\partial \mu}{\partial z}^\top \frac{\partial \ell}{\partial u}\right) - \left(\frac{\partial f}{\partial z} + \frac{\partial f}{\partial u}\frac{\partial \mu}{\partial z}\right)^\top \rho$$

subject to the terminal condition $\rho(t_i + T) = \frac{\partial}{\partial z} m(z(t_i + T))$. ■

B. *Proof of Theorem 1*

Theorem 1: Given Assumption 1 and dynamics (20), then the change in information[5]$\Delta \mathbf{I}$ subject to (18) is given to first order

$$\begin{aligned} \Delta \mathbf{I} \approx (\|(\mathfrak{K}_u v(x))^\top \rho\|^2_{\tilde{\mathbf{R}}^{-1}} &+ \ell_{\text{task}}(z, \mu_\star) \\ &- \ell_{\text{task}}(z, \mu))\mathfrak{I}_{\mu_\star}\mathfrak{I}_\mu + \mathcal{O}(\Delta t) \end{aligned} \tag{35}$$

where $\mathfrak{I}_{\mu_\star}$, $\mathfrak{I}_\mu$ is the T-optimality measure (24) from applying the control $\mu_\star$ and μ.

[5] With respect to the information acquired from applying only $\mu(z)$.

Proof: First define (14) for a controller as

$$J(u(t)) = \int_{t_i}^{t_i+\Delta t} \frac{1}{\mathfrak{I}_u} + \ell_{\text{task}}(z(t), u(t)) dt \tag{36}$$

where $\Delta t < T$ is a time duration, $z(t)$ is subject to the controller $u(t)$, and $\mathfrak{I}_u$ is the measure of information from applying the control u. If we consider the difference between $J(\mu_\star)$ and $J(\mu)$ where μ is a controller that minimizes $\ell_{\text{task}}(z, u)$, then

$$\begin{aligned} J(\mu_\star) - J(\mu) &= \int_{t_i}^{t_i+\Delta t} \frac{1}{\mathfrak{I}_{\mu_\star}} - \frac{1}{\mathfrak{I}_\mu} + \ell_{\text{task}}(z, \mu_\star) - \ell_{\text{task}}(z, \mu) dt \\ &\approx \Delta t \left(\frac{1}{\mathfrak{I}_{\mu_\star}} - \frac{1}{\mathfrak{I}_\mu} + \ell_{\text{task}}(z, \mu_\star) - \ell_{\text{task}}(z, \mu) \right) \\ &\quad + \mathcal{O}(\Delta t). \end{aligned} \tag{37}$$

From Corollary 1 and that

$$\frac{\partial}{\partial \lambda} J \Delta t \approx J(\mu_\star) - J(\mu)$$

we can show that

$$\begin{aligned} \frac{\partial}{\partial \lambda} J \Delta t &\approx J(\mu_\star) - J(\mu) \\ &\approx \Delta t \left(\frac{1}{\mathfrak{I}_{\mu_\star}} - \frac{1}{\mathfrak{I}_\mu} + \ell_{\text{task}}(z, \mu_\star) - \ell_{\text{task}}(z, \mu) \right) \\ &\quad + \mathcal{O}(\Delta t). \end{aligned} \tag{38}$$

which we rearrange (38) and insert (21) to get

$$\begin{aligned} &- \| \left(\mathfrak{K}_u v(x) \right)^\top \rho \|^2_{\tilde{\mathbf{R}}^{-1}} \\ &\approx \left(\frac{1}{\mathfrak{I}_{\mu_\star}} - \frac{1}{\mathfrak{I}_\mu} + \ell_{\text{task}}(z, \mu_\star) - \ell_{\text{task}}(z, \mu) \right) + \mathcal{O}(\Delta t) \\ &\approx \frac{\mathfrak{I}_\mu - \mathfrak{I}_{\mu_\star} + (\ell_{\text{task}}(z, \mu_\star) - \ell_{\text{task}}(z, \mu)) \mathfrak{I}_{\mu_\star} \mathfrak{I}_\mu}{\mathfrak{I}_{\mu_\star} \mathfrak{I}_\mu} + \mathcal{O}(\Delta t). \end{aligned} \tag{39}$$

Setting $\Delta \mathbf{I} = \mathfrak{I}_{\mu_\star} - \mathfrak{I}_\mu$ in (39) and simplifying gives the relative information gain

$$\begin{aligned} \Delta \mathbf{I} \approx (&\| \left(\mathfrak{K}_u v(x) \right)^\top \rho \|^2_{\tilde{\mathbf{R}}^{-1}} + \ell_{\text{task}}(z, \mu_\star) \\ &- \ell_{\text{task}}(z, \mu)) \mathfrak{I}_{\mu_\star} \mathfrak{I}_\mu + \mathcal{O}(\Delta t). \end{aligned}$$

■

Acknowledgment

The authors would like to thank G. Mamakoukas for his insight, discussion, and thorough review of this paper. Any opinions, findings and conclusions, or recommendations expressed in this paper are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

[1] B. O. Koopman, "Hamiltonian systems and transformation in Hilbert space," *Proc. Nat. Academy Sci.*, vol. 17, no. 5, pp. 315–318, 1931.

[2] I. Mezić, "Analysis of fluid flows via spectral properties of the Koopman operator," *Annu. Rev. Fluid Mech.*, vol. 45, pp. 357–378, 2013.

[3] I. Mezić, "On applications of the spectral theory of the Koopman operator in dynamical systems and control theory," in *Proc. IEEE Int. Conf. Decis. Control*, 2015, pp. 7034–7041.

[4] M. Budišić, R. Mohr, and I. Mezić, "Applied Koopmanism," *Chaos:Interdisciplinary J. Nonlinear Sci.*, vol. 22, no. 4, p. 047510, 2012.

[5] M. Korda and I. Mezić, "Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control," 2016, *arXiv:1611.03537*.

[6] N. Roy and A. McCallum, "Toward optimal active learning through monte carlo estimation of error reduction," in *Proc. Int. Conf. Mach. Learn.*, 2001, pp. 441–448.

[7] A. Baranes and P.-Y. Oudeyer, "Active learning of inverse models with intrinsically motivated goal exploration in robots," *Robot. Auton. Syst.*, vol. 61, no. 1, pp. 49–73, 2013.

[8] C. Dima, M. Hebert, and A. Stentz, "Enabling learning from large datasets: Applying active learning to mobile robotics," in *Proc. IEEE Int. Conf. Robot. Autom.*, vol. 1, 2004, pp. 108–114.

[9] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1238–1274, 2013.

[10] G. Williams *et al.*, "Information theoretic MPC for model-based reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2017, pp. 1714–1721.

[11] B. Armstrong, "On finding exciting trajectories for identification experiments involving systems with nonlinear dynamics," *Int. J. Robot. Res.*, vol. 8, no. 6, pp. 28–48, 1989.

[12] A. D. Wilson, J. A. Schultz, A. R. Ansari, and T. D. Murphey, "Dynamic task execution using active parameter identification with the Baxter research robot," *IEEE Trans. Autom. Sci. Eng.*, vol. 14, no. 1, pp. 391–397, Jan. 2017.

[13] A. D. Wilson, J. A. Schultz, and T. D. Murphey, "Trajectory synthesis for Fisher information maximization," *IEEE Trans. Robot.*, vol. 30, no. 6, pp. 1358–1370, Dec. 2014.

[14] K. Ayusawa and E. Yoshida, "Motion retargeting for humanoid robots based on simultaneous morphing parameter identification and motion optimization," *IEEE Trans. Robot.*, vol. 33, no. 6, pp. 1343–1357, Dec. 2017.

[15] V. Mnih *et al.*, "Asynchronous methods for deep reinforcement learning," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1928–1937.

[16] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel, "Benchmarking deep reinforcement learning for continuous control," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 1329–1338.

[17] M. Cutler, T. J. Walsh, and J. P. How, "Real-world reinforcement learning via multifidelity simulators," *IEEE Trans. Robot.*, vol. 31, no. 3, pp. 655–671, Jun. 2015.

[18] W. Yu, J. Tan, C. K. Liu, and G. Turk, "Preparing for the unknown: Learning a universal policy with online system identification," in *Proc. Robot.: Sci. Syst.*, 2017, doi: 10.15607/RSS.2017.XIII.048.

[19] K. S. Sin and G. C. Goodwin, "Stochastic adaptive control using a modified least squares algorithm," *Automatica*, vol. 18, no. 3, pp. 315–321, 1982.

[20] F. Ding, X. Wang, Q. Chen, and Y. Xiao, "Recursive least squares parameter estimation for a class of output nonlinear systems based on the model decomposition," *Circuits, Syst. Signal Process.*, vol. 35, no. 9, pp. 3323–3338, 2016.

[21] F. Ding, D. Meng, J. Dai, Q. Li, A. Alsaedi, and T. Hayat, "Least squares based iterative parameter estimation algorithm for stochastic dynamical systems with ARMA noise using the model equivalence," *Int. J. Control, Autom. Syst.*, vol. 16, no. 2, pp. 630–639, 2018.

[22] V. Bonnet, P. Fraisse, A. Crosnier, M. Gautier, A. González, and G. Venture, "Optimal exciting dance for identifying inertial parameters of an anthropomorphic structure," *IEEE Trans. Robot.*, vol. 32, no. 4, pp. 823–836, Aug. 2016.

[23] J. Jovic, A. Escande, K. Ayusawa, E. Yoshida, A. Kheddar, and G. Venture, "Humanoid and human inertia parameter identification using hierarchical optimization," *IEEE Trans. Robot.*, vol. 32, no. 3, pp. 726–735, Jun. 2016.

[24] J. H. Tu, C. W. Rowley, D. M. Luchtenburg, S. L. Brunton, and J. N. Kutz, "On dynamic mode decomposition: Theory and applications," *J. Comput. Dyn.*, vol. 1, no. 2, pp. 391–421, 2014.

[25] A. Mauroy and I. Mezić, "Global stability analysis using the eigenfunctions of the Koopman operator," *IEEE Trans. Autom. Control*, vol. 61, no. 11, pp. 3356–3369, Nov. 2016.

[26] S. L. Brunton, B. W. Brunton, J. L. Proctor, and J. N. Kutz, "Koopman invariant subspaces and finite linear representations of nonlinear dynamical systems for control," *PloS One*, vol. 11, no. 2, p. e0150171, 2016.

[27] E. Kaiser, J. N. Kutz, and S. L. Brunton, "Data-driven discovery of Koopman eigenfunctions for control," 2017, *arXiv:1707.01146*.

[28] A. Sootla and D. Ernst, "Pulse-based control using Koopman operator under parametric uncertainty," *IEEE Trans. Autom. Control*, vol. 63, no. 3, pp. 791–796, Mar. 2018.

[29] C. W. Rowley, "Low-order models for control of fluids: Balanced models and the Koopman operator," in *Advances Comput., Model. Control Transitional Turbulent Flows*, p. 60, 2015.

[30] J. L. Proctor, S. L. Brunton, and J. N. Kutz, "Dynamic mode decomposition with control," *J. Appl. Dynamical Syst.*, vol. 15, no. 1, pp. 142–161, 2016.

[31] A. Surana, "Koopman operator based observer synthesis for control-affine nonlinear systems," in *Proc. IEEE Int. Conf. Decis. Control*, 2016, pp. 6492–6499.

[32] I. Abraham, G. de la Torre, and T. Murphey, "Model-based control using Koopman operators," in *Proc. Robot.: Sci. Syst.*, 2017, doi: 10.15607/RSS.2017.XIII.052.

[33] A. Broad, T. Murphey, and B. Argall, "Learning models for shared control of human-machine systems with unknown dynamics," in *Proc. Robot.: Sci. Syst.*, 2017, doi: 10.15607/RSS.2017.XIII.037.

[34] M. Watter, J. Springenberg, J. Boedecker, and M. Riedmiller, "Embed to control: A locally linear latent dynamics model for control from raw images," in *Advances in Neural Information Processing Systems.* Cambridge, MA, USA: MIT Press, 2015, pp. 2746–2754.

[35] Y. Pu, Z. Gan, R. Henao, X. Yuan, C. Li, A. Stevens, and L. Carin, "Variational autoencoder for deep learning of images, labels and captions," in *Advances in Neural Information Processing Systems*. Cambridge, MA, USA: MIT Press, 2016, pp. 2352–2360.

[36] M. Rattray, D. Saad, and S.-I. Amari, "Natural gradient descent for on-line learning," *Physical Rev. Lett.*, vol. 81, no. 24, p. 5461–5464, 1998.

[37] T. Lai and C.-Z. Wei, "Extended least squares and their applications to adaptive control and prediction in linear systems," *IEEE Trans. Autom. Control*, vol. 31, no. 10, pp. 898–906, Oct. 1986.

[38] M. Egerstedt, Y. Wardi, and F. Delmotte, "Optimal control of switching times in switched dynamical systems," in *Proc. IEEE Int. Conf. Decis. Control*, vol. 3, 2003, pp. 2138–2143.

[39] H. Axelsson, Y. Wardi, M. Egerstedt, and E. Verriest, "Gradient descent approach to optimal mode scheduling in hybrid dynamical systems," *J. Optim. Theory Appl.*, vol. 136, no. 2, pp. 167–186, 2008.

[40] C. G. Atkeson and J. C. Santamaria, "A comparison of direct and model-based reinforcement learning," in *Proc. IEEE Int. Conf. Robot. Autom.*, vol. 4, 1997, pp. 3557–3564.

[41] A. R. Ansari and T. D. Murphey, "Sequential action control: Closed-form optimal control for nonlinear and nonsmooth systems," *IEEE Trans. Robot.*, vol. 32, no. 5, pp. 1196–1214, 2016.

[42] F. Pukelsheim, *Optimal Design of Experiments*. Philadelphia, PA, USA: SIAM, 2006.

[43] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. Hoboken, NJ, USA: Wiley, 2012.

[44] N. Nahi and G. Napjus, "Design of optimal probing signals for vector parameter estimation," in *Proc. IEEE Conf. Decis. Control*, vol. 10, 1971, pp. 162–168.

[45] T. Morimura, E. I. E. J. Uchibe, and K. Doya, "Utilizing the natural gradient in temporal difference reinforcement learning with eligibility traces," in *Proc. Int. Symp. Inf. Geometry Appl.*, 2005, pp. 256–263.

[46] H. Wei, J. Zhang, F. Cousseau, T. Ozeki, and S.-I. Amari, "Dynamics of learning near singularities in layered networks," *Neural Computation*, vol. 20, no. 3, pp. 813–843, 2008.

[47] M. Inoue, H. Park, and M. Okada, "On-line learning theory of soft committee machines with correlated hidden units–steepest gradient descent and natural gradient descent–," *J. Physical Soc. Japan*, vol. 72, no. 4, pp. 805–810, 2003.

[48] X. Yan, V. Indelman, and B. Boots, "Incremental sparse GP regression for continuous-time trajectory estimation and mapping," *Robot. Auton. Syst.*, vol. 87, pp. 120–132, 2017.

[49] D. Nguyen-Tuong and J. Peters, "Incremental online sparsification for model learning in real-time robot control," *Neurocomputing*, vol. 74, no. 11, pp. 1859–1867, 2011.

[50] 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.

[51] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1238–1274, 2013.

[52] P. Kormushev, S. Calinon, and D. G. Caldwell, "Robot motor skill coordination with em-based reinforcement learning," in *Proc. Int. Conf. Intell. Robots Syst.*, 2010, pp. 3232–3237.

[53] R. Saegusa, G. Metta, G. Sandini, and S. Sakka, "Active motor babbling for sensorimotor learning," in *Proc. Int. Conf. Robot. Biomimetics*, 2009, pp. 794–799.

[54] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine, "Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning," 2017, *arXiv:1708.02596*.

[55] R. F. Reinhart, "Autonomous exploration of motor skills by skill babbling," *Auton. Robots*, vol. 41, no. 7, pp. 1521–1537, 2017.

[56] T. Fan and T. Murphey, "Online feedback control for input-saturated robotic systems on lie groups," in *Proc. Robot.: Sci. Syst.*, 2016, doi: 10.15607/RSS.2016.XII.027.

[57] F. Berkenkamp, A. P. Schoellig, and A. Krause, "Safe controller optimization for quadrotors with Gaussian processes," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2016, pp. 491–496.

[58] J. Schreiter, D. Nguyen-Tuong, M. Eberts, B. Bischoff, H. Markert, and M. Toussaint, "Safe exploration for active learning with gaussian processes," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discovery Databases*, 2015, pp. 133–149.

[59] B. Kramer, P. Grover, P. Boufounos, S. Nabi, and M. Benosman, "Sparse sensing and DMD-based identification of flow regimes and bifurcations in complex flows," *J. Appl. Dynamical Syst.*, vol. 16, no. 2, pp. 1164–1196, 2017.

[60] E. Yeung, S. Kundu, and N. Hodas, "Learning deep neural network representations for Koopman operators of nonlinear dynamical systems," 2017, *arXiv:1708.06850*.

[61] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

[62] B. Lusch, J. N. Kutz, and S. L. Brunton, "Deep learning for universal linear embeddings of nonlinear dynamics," *Nature Commun.*, vol. 9, no. 1, 2018, Art. no. 4950.

[63] Itseez, Open Source Computer Vision Library. (2015). [Online]. Available: https://github.com/itseez/opencv

[64] M. Quigley *et al.*, "ROS: An open-source robot operating system," in *Proc. ICRA Workshop Open Source Softw.*, Kobe, Japan, vol. 3, 2009, p. 5.

[65] O. Klimov and J. Shulman, Roboschool. (2017). Available: https://github.com/openai/roboschool

[66] B. D. Anderson and J. B. Moore, *Optimal Control: Linear Quadratic Methods*. Chelmsford, MA, USA: Courier Corporation, 2007.

Ian Abraham received the B.S. degree in mechanical and aerospace engineering from Rutgers University, NJ, USA, in 2014, and the M.S. degree in mechanical engineering from Northwestern University, Evanston, IL, USA, in 2017. He is currently working toward the Ph.D. degree on the intersection of robot learning and optimal control theory, focusing on active sensing and efficient robot learning, with the Neuroscience and Robotics Lab, Northwestern University.

Todd D. Murphey received the B.S. degree in mathematics from the University of Arizona, Tucson, AZ, USA, in 1997, and the Ph.D. degree in control and dynamical systems from the California Institute of Technology, Pasadena, CA, USA, in 2002.

He is a Professor of Mechanical Engineering with Northwestern University, Evanston, IL, USA. His laboratory is part of the Neuroscience and Robotics Laboratory, Northwestern University. His research interests include robotics, control, computational methods for biomechanical systems, and computational neuroscience.

Dr. Murphey was a member of the 2014–2015 DARPA/IDA Defense Science Study Group. He was the recipient of the National Science Foundation CAREER Award in 2006, and Northwesterns Professorship of Teaching Excellence. He was a Senior Editor for the IEEE TRANSACTIONS ON ROBOTICS.