Embedding Koopman Optimal Control in Robot Policy Learning

Hang Yin, Michael C. Welle and Danica Kragic

***Abstract*— Embedding an optimization process has been explored for imposing efficient and flexible policy structures. Existing work often build upon nonlinear optimization with explicitly iteration steps, making policy inference prohibitively expensive for online learning and real-time control. Our approach embeds a linear-quadratic-regulator (LQR) formulation with a Koopman representation, thus exhibiting the tractability from a closed-form solution and richness from a non-convex neural network. We use a few auxiliary objectives and reparameterization to enforce optimality conditions of the policy that can be easily integrated to standard gradient-based learning. Our approach is shown to be effective for learning policies rendering an optimality structure and efficient reinforcement learning, including simulated pendulum control, 2D and 3D walking, and manipulation for both rigid and deformable objects. We also demonstrate real world application in a robot pivoting task.**

I. Introduction

Efficient policy learning relies on representing policies with an informative structure. Important contributions have been made by using specific dynamical systems [1] and integrating robot controllers with expressive neural networks [2], [3]. One direction of work has advocated parameterizing an optimization problem instead of a procedural policy. The policy inference is implicit by solving the problem in the inner loop and desired properties can hence be flexibly specified in a *declarative* manner [4]. Early work used derivative-free methods to learn optimization parameters [5] while more recent approaches employ automatic differentiation and functions with almost everywhere differentiability, learning a differentiable optimization [6], [7].

Embedding an optimization process raises the question of how fast one can perform inference for such policies. Existing works focused on differentiating optimality conditions for efficient backward evaluation [8], [9]. However, forward inference still largely relies on iterations, whose number can be up to hundreds [8], and suffers from local optimality [10]. Convex programming exhibits improved performance with a convergence in milliseconds on small or medium numerical problems [9]. However, for a rich policy representation, non-convex optimization involving neural network models is still infeasible as benchmarked in [8], [11]. To this end, challenges remain to embed a non-convex representation and to enable real-time demanding application.

We propose to bridge the gap by integrating Koopman operator to differentiable optimization policies. Koopman theory allows representing nonlinear dynamics with an approximated linear form [12], enabling data-driven control for robot fish and soft manipulators [13], [14]. This effectively turns an original non-convex optimization to an approximated convex programming or even a linear-quadratic-regulator (LQR) problem, thus significantly improves the efficiency of policy evaluation. We show that it is feasible to embed Koopman optimal control for efficient policy inference and gradient-based learning for both optimization and Koopman representation parameters. To ensure the validity of parameters, we propose augmenting the learning task with auxiliary objectives. Our analysis shows that the contributed form is flexible for representing a range of neural policies with an optimality structure. More importantly, it enables using basic automatic differentiation and inference with significantly reduced costs, whose effectiveness is manifested in a series of reinforcement learning tasks, including pendulum control, locomotion, object manipulation and real robot pivoting.

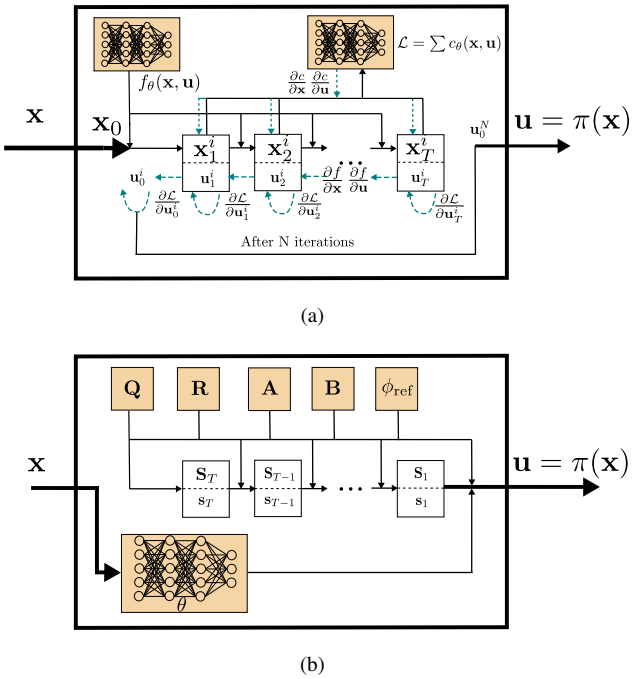


Fig. 1. Embedding optimal control in policy inference: (a) Explicitly expanding gradient steps of iterative optimization [7] or iteratively forming convex optimization with dynamics Jacobian [8]; (b) LQR formulation with a Koopman representation requiring only one backward sweeping.

II. Related Work

Our work is broadly related to topics of improving policy learning efficiency with inductive biases, using Koopman theory in synthesizing robot control and estimating Koopman operators with machine learning approaches particularly learning with deep neural networks.

The authors are with CAS/RPL, KTH Royal Institute of Technology, Stockholm, Sweden. email:{hyin, mwelle, dani}@kth.se

Learning with Structured Policy: A plethora of robot learning research resorts to structured policies such as motion primitives [1] or optimal impedance trajectories [15]. Recent research showcases policies parameterized by state-dependent variable impedance controllers in learning contact-rich tasks [2], [3], with stability guarantees by searching in a constrained parameter space [16] or exploiting special neural network structures [17], [18].

Other research seeks to use an optimization structure. Pioneering work [5] proposes to learn parameters of a quadratic cost function through derivative-free evolutionary strategies, solving the inner loop planning as an optimal inference problem. [19] uses Bayesian optimization to tune parameters of kernels with an LQR structure. Recent machine learning progress opens up ways for gradient evaluation with respect to optimization parameters. This enables plugging policies into gradient-based learning algorithms. Extensive work has been done for learning policies from expert demonstrations, by expanding the inner loop optimization [6], [7], using differentiable LQR under robust constraints [20] or linear switching dynamical systems [21]. [8] computes gradients analytically backward from a fixed point. Obtaining the fixed point from the forward process still relies on iteratively building local convex problems and the gradients can be inaccurate if the iteration is not long enough to reach the fixed point. Another work, [10] further improves the backward efficiency by constructing an auxiliary LQR, resorting to an external solver for forward evaluation. More recent research learns parameters of a convex optimization controller [9]. All of these works focus on numerical examples, assuming known dynamics constraints [9] or their analytical form and a trajectory-based policy [10], and face difficulties when using neural networks to approximate dynamics [8]. Our work embeds optimal control that is non-convex with the original state. Notably, we leverage a Koopman representation to analytically solving an convex problem like [9] but with an approximated form, and is hence scalable to online reinforcement learning with parameterized neural networks.

Koopman Control in Robotics: Koopman operator represents nonlinear dynamical system as linear transformation in a Hilbert space [12]. Modern literature extends the theory to controlled system [22] and explores its application to MPC [23], [24]. These results highlight a data-driven paradigm for controlling high-dimensional nonlinear dynamical systems and have raised interests from the robotics community. Prominent examples include motion control of robotic fish [13] and soft continuum manipulator [14]. Koopman operators are commonly identified with a linearly parameterized estimator, using a fixed set of observable functions [13]. Authors in [25] investigate constraints on Koopman operator and observable basis functions for a stable prediction, showing an improved accuracy in quadrotor control. In our work, these functions can be simultaneously learned alongside policy parameters. The policy is also flexible to be used in a model-free setting thanks to the differentiability of the inner loop optimal control.

Learning Latent Dynamics and Koopman Operators: Learning a latent representation is deemed as an effective way of modeling dynamical data. Deep neural networks enable to learn rich encoders and dynamical models for data like image sequences [26]. Other works argue for learning latent dynamics with an affine structure and local linearity [27], [28], which are more amenable to model-based control [27]. Recently, the significance of structured latent dynamics is further demonstrated by explicitly penalizing space curvature [29] or considering a Newtonian formalism [30].

Koopman theory uses a latent representation that admits linear dynamics, implying a flat space. Learning Koopman observables with neural networks has been explored in [31], [32], [33]. In particular, [32] argues to use an auxiliary network to determine Koopman eigen values so as to account for continuous spectra. Other works employ Koopman representation in the latent space of variational dynamics [34] and image sequences with a compositional structure [35]. All work reviewed above focus on prediction tasks and supervised learning of Koopman representations. In the presented paper we target policy learning for robotic skills in which prediction may be an auxiliary task.

III. Preliminaries

A. Koopman Operator

We consider nonlinear autonomous dynamical systems

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t) \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^n$ and t denote system states and discrete time index, respectively. Koopman operator theory suggests lifting $\mathbf{x}$ to a function space where the evolution can be captured by applying a linear operator $\mathcal{K}$

$$\mathcal{K} \circ g(\mathbf{x}_t) = g \circ f(\mathbf{x}_t) = g(\mathbf{x}_{t+1}) \tag{2}$$

with $g(\cdot)$ denoting elements of the function space as $g : \mathbb{R}^n \to \mathbb{R}$, often known as *observables*. The Koopman operator $\mathcal{K}$ is often infinite-dimensional. In practice, a finite dimensional approximation adopts a vector-valued $\mathbf{g} : \mathbb{R}^n \to \mathbb{R}^k$ and the linear operator $\mathcal{K}$ hence becomes a matrix $\mathbf{K} \in \mathbb{R}^{k \times k}$.

For controlled dynamical systems $\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t)$, one can recover the autonomous case in Equation (1) if control $\mathbf{u} \in \mathbb{R}^m$ is subject to a feedback law or its own dynamics [22]. In the standard case of solely identifying $\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t)$, where predicting $\mathbf{u}_{t+1}$ is not necessary, the Koopman operator can be defined as

$$\mathbf{K}\mathbf{g}(\mathbf{x}_t, \mathbf{u}_t) = \mathbf{g}(f(\mathbf{x}_t, \mathbf{u}_t), \mathbf{0}) \tag{3}$$

where $\mathbf{g} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^k$ and $\mathbf{u}_{t+1}$ is constrained as a zero vector. Equation (13) allows to estimate the operator $\mathbf{K}$ by solving a linear equation when $\mathbf{g}$ is selected for a rich representation. Recent works propose to use neural models to parameterize observables [31], [32]. One common choice is to make $\mathbf{g}$ linear with $\mathbf{u}$ and to decouple state and control

$$\mathbf{g}(\mathbf{x}, \mathbf{u}) = \boldsymbol{\phi}_\theta(\mathbf{x}) + \mathbf{M}\mathbf{u} \tag{4}$$

Here, ϕ denotes the state-dependent part, which can be neural networks with a parameter θ, and $\mathbf{M} \in \mathbb{R}^{k\times m}$ parameterizes a linear transformation of control. Substitute this to Equation (13) and note that $\mathbf{M}$ is a free parameter, one can obtain

$$\boldsymbol{\phi}_\theta(\mathbf{x}_{t+1}) = \mathbf{A}\boldsymbol{\phi}_\theta(\mathbf{x}_t) + \mathbf{B}\mathbf{u}_t \tag{5}$$

with a reparameterization of $\mathbf{A} = \mathbf{K} \in \mathbb{R}^{k\times k}$ and $\mathbf{B} = \mathbf{KM} \in \mathbb{R}^{k\times m}$. The observable design as in Equation (14) might be limited for neglecting the interaction between state and control [22]. However, it yields a standard linear time-invariant system with respect to ϕ and $\mathbf{u}$, which facilitates control analysis and synthesis.

B. Linear Quadratic Regulator

Linear Quadratic Regulator (LQR) derives optimal control for a linear dynamical system under a quadratic instantaneous cost function. Specifically, considering a finite horizon T and non-zero regulation target $\mathbf{x}^r$, it can be formulated as solving

$$\begin{aligned} \min_{\mathbf{u}_{0:T-1}} & \sum_{t=0}^{T-1} [(\mathbf{x}_t - \mathbf{x}^r)^T \mathbf{Q}(\mathbf{x}_t - \mathbf{x}^r) + \mathbf{u}_t^T \mathbf{R}\mathbf{u}_t] \\ \text{s. t.} \quad & \mathbf{x}_{t+1} = \mathbf{A}\mathbf{x}_t + \mathbf{B}\mathbf{u}_t \end{aligned} \tag{6}$$

with $\mathbf{Q} \in \mathbb{R}^{n\times n}$ and $\mathbf{R} \in \mathbb{R}^{m\times m}$ denoting symmetrical positive definite matrices. The regulation target $\mathbf{x}^r$ can be time-dependent for a tracking problem and we choose to omit the terminal cost term. The solution takes a form of

$$\mathbf{u}_t = -\mathbf{K}_t\mathbf{x}_t + \mathbf{k}_t \tag{7}$$

The terms of feedback $\mathbf{K}$ and feedforward $\mathbf{k}$ are obtained by

$$\begin{aligned} \mathbf{K}_t &= (\mathbf{B}^T\mathbf{S}_{t+1}\mathbf{B} + \mathbf{R})^{-1}\mathbf{B}^T\mathbf{S}_{t+1}\mathbf{A} \\ \mathbf{k}_t &= (\mathbf{B}^T\mathbf{S}_{t+1}\mathbf{B} + \mathbf{R})^{-1}\mathbf{B}^T\mathbf{s}_{t+1} \end{aligned} \tag{8}$$

where $\mathbf{S}_t$ and $\mathbf{s}_t$ parameterize the optimal quadratic cost-to-go function up to a constant

$$\mathcal{J}_t^*(\mathbf{x}) = \mathbf{x}^T\mathbf{S}_t\mathbf{x} + 2\mathbf{x}^T\mathbf{s}_t + c \tag{9}$$

LQR admits a closed-form solution by following backward Riccati recursions

$$\begin{aligned} \mathbf{S}_t &= \mathbf{A}^T\mathbf{S}_{t+1}(\mathbf{A} - \mathbf{B}\mathbf{K}_t) + \mathbf{Q} \\ \mathbf{s}_t &= (\mathbf{A} - \mathbf{B}\mathbf{K}_t)^T\mathbf{s}_{t+1} + \mathbf{Q}\mathbf{x}^r \end{aligned} \tag{10}$$

with $\mathbf{S}_T = \mathbf{Q}$ and $\mathbf{s}_T = \mathbf{Q}\mathbf{x}^r$. The computational complexity scales with the length of horizon T. One can choose to truncate the planning horizon and only adopt $\mathbf{u}_0$ at the first step. This yields receding horizon model predictive control.

IV. Approach

We first give an intuitive idea about how optimization is used as a differentiable policy and highlight the tractability issue of existing methods. This is followed by a brief introduction to Koopman control which lays foundations for presenting the contributed method, ending with a discussion regarding method properties and implementation.

A. An Intuitive Idea

We consider policies $\mathbf{u} = \pi(\mathbf{x})$ with $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{u} \in \mathbb{R}^m$ denoting state observations and control, respectively. The idea of embedding optimization as a policy representation is determining $\mathbf{u}$ implicitly as the solution to an optimal control problem [5]

$$\begin{aligned} \min_{\mathbf{u}_{0:T}} & \sum_{t=0}^{T} c(\mathbf{x}_t, \mathbf{u}_t) \\ \text{s. t.} \quad & \mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t) \end{aligned} \tag{11}$$

where $c(\mathbf{x}_t, \mathbf{u}_t)$ denotes a running cost and $f(\mathbf{x}_t, \mathbf{u}_t)$ defines a dynamics model with t as the discrete time index for a finite horizon $T+1$. A forward inference of $\pi(\mathbf{x})$ eventually takes the first control $\mathbf{u}_0^*$ and re-runs the optimization for a new state query, working as receding horizon model predictive control (MPC) [8]. The policy is constructed by the cost and dynamics model whose parameters can be learned by a evolutionary strategy [5] or differentiating the optimization process [6], [8], [9].

Embedding optimization could provide useful policy structure apriori, for instance, in learning goal-directed behaviors [5], [7]. However, solving the inner loop optimization can be expensive for evaluating π, especially when the cost or the dynamics are represented by non-convex neural networks [8], [7]. Specifically, as illustrated in Figure 1(a), the policy needs to recursively expand gradient updates for explicit optimization steps [7] or form local quadratic programming [8], resulting in a complexity of $O(NT)$. Moreover, N can be up to a few hundred for policy backwarding with solutions of sufficient optimality [8], which is also not guaranteed in general.

Our idea is to adopt a LQR formulation in the inner-loop optimization, which admits a closed-form solution by following one backward sweep of Riccati recursions. This drastically reduces the complexity from $O(NT)$ to $O(T)$ and avoids local optima, while trade off the policy richness compared to general nonlinear optimization. We propose to use Koopman control as a complement. This essentially aims at an exact solution of an approximated problem, in contrast to seeking approximated solutions to the original problem whose parameters are still subject to learning or estimation [7], [8], [10].

B. Koopman Operators

Koopman operator theory suggests lifting $\mathbf{x}$ to a function space where the evolution can be globally captured by a linear operator $\mathcal{K}$. For the instance of an autonomous system $\mathbf{x}_{t+1} = f(\mathbf{x}_t)$, we have

$$(\mathcal{K} \circ g)(\mathbf{x}_t) = (g \circ f)(\mathbf{x}_t) \tag{12}$$

with every $g : \mathbb{R}^n \to \mathbb{R}$, often known as *observables*, in the function space that is invariant under $\mathcal{K}$. The Koopman operator $\mathcal{K}$ is applied through function composition $\circ$ and the function space is often infinite-dimensional. In practice, a finite dimensional truncation adopts a vector of observables

$\mathbf{g}: \mathbb{R}^n \to \mathbb{R}^k$ and an identification of $\mathcal{K}$ as a matrix $\mathbf{L} \in \mathbb{R}^{k\times k}$ [36].

For controlled dynamical systems $\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t)$, one can recover the autonomous case if control $\mathbf{u}$ is subject to a feedback law or its own dynamics [22]. In the standard case of solely identifying $\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t)$, predicting $\mathbf{u}_{t+1}$ is not necessary and the Koopman operator can be defined as

$$\mathbf{L}\mathbf{g}(\mathbf{x}_t, \mathbf{u}_t) = \mathbf{g}(f(\mathbf{x}_t, \mathbf{u}_t), \mathbf{0}) = g(\mathbf{x}_{t+1}, \mathbf{0}) \tag{13}$$

where $\mathbf{g}: \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^k$ and $\mathbf{u}_{t+1}$ is constrained as a zero vector. Equation (13) allows to estimate the operator $\mathbf{L}$ by solving a linear equation when $\mathbf{g}$ is selected for a rich representation. Recent works propose to use neural models to parameterize observables [31], [32]. One common choice is to decouple state and control [14]

$$\mathbf{g}(\mathbf{x}, \mathbf{u}) = \boldsymbol{\phi}_\theta(\mathbf{x}) + \mathbf{M}\mathbf{u} \tag{14}$$

Here, $\boldsymbol{\phi}$ denotes the state-dependent part, which can be parameterized θ, and $\mathbf{M} \in \mathbb{R}^{k\times m}$ parameterizes a linear transformation of control. Substitute this to Equation (13) and note that $\mathbf{M}$ is a free parameter, one can obtain

$$\boldsymbol{\phi}_\theta(\mathbf{x}_{t+1}) = \mathbf{A}\boldsymbol{\phi}_\theta(\mathbf{x}_t) + \mathbf{B}\mathbf{u}_t \tag{15}$$

with a reparameterization of $\mathbf{A} = \mathbf{L} \in \mathbb{R}^{k\times k}$ and $\mathbf{B} = \mathbf{L}\mathbf{M} \in \mathbb{R}^{k\times m}$. The observable design as in Equation (14) might be limited for neglecting the interaction between state and control [22]. However, it yields a standard linear time-invariant system with respect to $\boldsymbol{\phi}$ and $\mathbf{u}$, which facilitates control analysis and synthesis.

C. Policy with Embedded Koopman Optimal Control

We propose to use a quadratic running cost in the space governed by Equation (15), yielding an LQR formulation with respect to ϕ and $\mathbf{u}$. The original problem in Equation (11) can now be casted as

$$\begin{aligned} \min_{\mathbf{u}_{0:T-1}} \sum_{t=0}^{T-1} & [(\phi_\theta(\mathbf{x}_t) - \phi_\theta(\mathbf{x}^r))^T \mathbf{Q}(\phi_\theta(\mathbf{x}_t) - \phi_\theta(\mathbf{x}^r)) + \mathbf{u}_t^T \mathbf{R}\mathbf{u}_t] \\ \text{s. t.} \quad & \phi_\theta(\mathbf{x}_{t+1}) = \mathbf{A}\phi_\theta(\mathbf{x}_t) + \mathbf{B}\mathbf{u}_t \end{aligned} \tag{16}$$

with $\mathbf{Q} \in \mathbb{R}^{n\times n}$ and $\mathbf{R} \in \mathbb{R}^{m\times m}$ denoting symmetrical positive definite matrices. The regulation target $\mathbf{x}^r$ can be a time-dependent parameter if it is known apriori for a tracking problem. Following the well-known LQR analysis, the solution takes a form of

$$\mathbf{u}_t^* = -\mathbf{K}_t\phi_\theta(\mathbf{x}_t) + \mathbf{k}_t \tag{17}$$

We follow MPC and use $\mathbf{u}_0^*$ at the first step, e.g. $\pi(\mathbf{x}_0) = \mathbf{u}_0^*$ for a deterministic policy. Beside the neural network parameter θ, the policy is also parameterized by the terms of feedback $\mathbf{K}$ and feedforward $\mathbf{k}$, which are obtained by

$$\begin{aligned} \mathbf{K}_t &= (\mathbf{B}^T\mathbf{S}_{t+1}\mathbf{B} + \mathbf{R})^{-1}\mathbf{B}^T\mathbf{S}_{t+1}\mathbf{A} \\ \mathbf{k}_t &= (\mathbf{B}^T\mathbf{S}_{t+1}\mathbf{B} + \mathbf{R})^{-1}\mathbf{B}^T\mathbf{s}_{t+1} \end{aligned} \tag{18}$$

with $\mathbf{S}_t$ and $\mathbf{s}_t$ parameterizing the cost-to-go function up to a constant: $\mathcal{J}_t^*(\mathbf{x}) = \phi_\theta(\mathbf{x})^T\mathbf{S}_t\phi_\theta(\mathbf{x}) + 2\phi_\theta(\mathbf{x})^T\mathbf{s}_t +$ const. These parameters are in turn related to $\{\mathbf{Q}, \mathbf{R}, \mathbf{A}, \mathbf{B}\}$ through Riccati recursions

$$\begin{aligned} \mathbf{S}_t &= \mathbf{A}^T\mathbf{S}_{t+1}(\mathbf{A} - \mathbf{B}\mathbf{K}_t) + \mathbf{Q} \\ \mathbf{s}_t &= (\mathbf{A} - \mathbf{B}\mathbf{K}_t)^T\mathbf{s}_{t+1} + \mathbf{Q}\mathbf{x}^r \end{aligned} \tag{19}$$

with $\mathbf{S}_T = \mathbf{Q}$ and $\mathbf{s}_T = \mathbf{Q}\mathbf{x}^r$.

The form given by Equation (17) can be called *Koopman policy* which is nonlinear with $\mathbf{x}$, see Figure 1(b). It is differentiable with respect to the parameter group $\boldsymbol{\Omega} = \{\mathbf{Q}, \mathbf{R}, \mathbf{A}, \mathbf{B}, \theta, \mathbf{x}^r\}$ and can hence be readily used in gradient-based learning. Some parameters are subject to constraints such as positive-definiteness and the dynamics constraint in Equation (15). For the former, we can use exponential transformation for diagonal $\mathbf{Q}$ and $\mathbf{R}$ or follow the reparameterization in [37] for full matrices. For the latter, we propose to augment auxiliaries to the training loss

$$\mathcal{L} = \mathcal{L}_{\text{train}} + \lambda_{\text{fit}}\mathcal{L}_{\text{fit}} + \lambda_{\text{recons}}\mathcal{L}_{\text{recons}} + \lambda_{\text{reg}}(\|\mathbf{A}\| + \|\mathbf{B}\|) \tag{20}$$

where $\mathcal{L}_{\text{train}}$ is the training loss for example the negative return of a reinforcement learning agent or a regression error of behavior cloning. Given a batch of $\{\mathbf{x}_t, \mathbf{u}_t, \mathbf{x}_{t+1}\}$, $\mathcal{L}_{\text{fit}}(\mathbf{A}, \mathbf{B}, \theta) = \|\phi_\theta(\mathbf{x}_{t+1}) - \mathbf{A}\phi_\theta(\mathbf{x}_t) - \mathbf{B}\mathbf{u}_t\|^2$ penalizes a large fitting error of Equation (15). $\mathcal{L}_{\text{recons}}$ is a reconstruction loss on the Koopman representation similar to [32] and λ_{reg} weighs the regularization of matrix norms as suggested in [14].

D. Remarks on Implications and Implementation

Ensuring linear evolution in the Koopman invariant subspace may require a large k to include sufficient observables. We find it still feasible to learn with k that is comparable to or smaller than the state dimension n. This may be partially explained by the usage of neural network and nonlinear reconstructions, which are believed to allow richer observables [33] and Koopman embeddings with a lower dimension [24]. Using a smaller k also correlates to the idea of learning low-dimensional latent dynamics for a compact model [27], [7]. After all, a model that roughly captures task dynamics could already be useful for efficient policy learning, whose ultimate goal is not necessarily accurate dynamics prediction.

The time horizon T controls how far the policy is allowed to look ahead. As an alternative perspective, we can view Equation (16) in an infinite horizon setting where T in Figure 1(b) becomes the number of iterations in approximately solving the Discrete Algebraic Riccati Equation. One may differentiate the equation to obtain adjoints of the parameters while we rely on automatic differentiation here. In either case, conditions need to be applied on dynamics matrices to strictly enforce optimality and stability. These are so far not accounted for but may be explicitly added as auxiliary constraints on matrix ranks. Still, we find the approximation and a small T tend to be good enough to yield reasonable policy structure and learning success.

It is also worth to note the relation of a Koopman policy to a "vanilla" neural network policy. Equation (17) is a standard neural network model when constraints imposed by

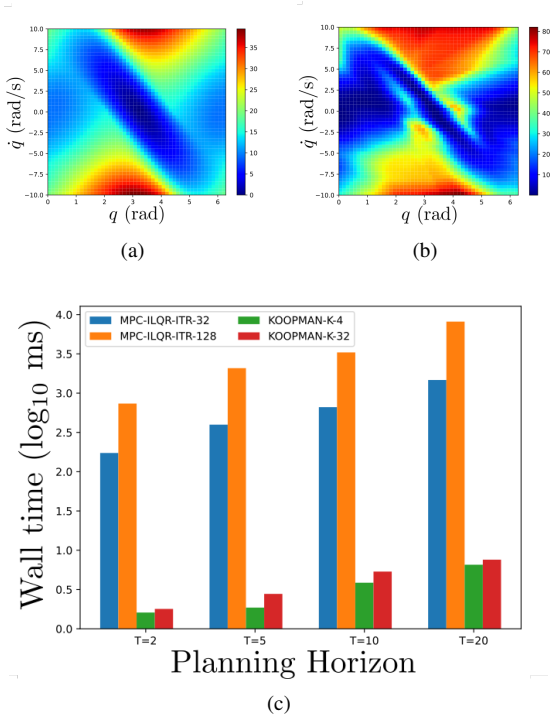


Fig. 2. Policy interpretability and inference efficiency for an inverted pendulum example. (a) Ground truth of cost-to-go obtained from value-iteration. (b) Cost-to-go from learned Koopman policy with $k = 4$ and $T = 5$. (c) Time costs of forwarding a Koopman policy and a differentiable MPC [8] with networks of similar sizes, 32 and 128 iteration steps. Note the metric is wall time of a common logarithmic scale.

Equation (18) and (20) are neglected. Meanwhile, one can also discard the training loss $\mathcal{L}_{\text{train}}$, while determining control from a given cost function and learned dynamics as model-based policy optimization, similar to robotics practice [14]. To this end, our approach implies flexibility of training policies on a spectrum, from model-free neural networks to controllers derived from model-based optimization.

In the implementation, we choose to fix $\mathbf{R}$ to avoid an ill-posed problem. An easy extension to Koopman policy is to merge it with the output from another neural network policy. The neural network policy is expected to complement by learning a "residual" [38]. We call this variant *Koopman Residual Policy* and find it superior in certain task settings. The least square loss of $\mathcal{L}_{\text{fit}}$ can also be implemented with pseudo inverse as in [35] and in such a case, $\mathbf{A}$ and $\mathbf{B}$ are implicitly determined. We find this achieves similar performance and hence stick to an explicit parameterization for the flexibility of regularizing dynamics matrices as in [14]. It is also possible to adopt a multi-step prediction loss as in [32] when longer data snippets are available e.g. in on-policy reinforcement learning. We keep one-step prediction to focus on the basic form of the idea and to minimize the modification on policy learning algorithms.

V. Experimental Results

We set to answer two questions: 1) Can we learn a Koopman policy that resembles optimality structure? 2) Can we use a Koopman policy for efficiently learning and controlling robotic tasks? To answer the first question, we report the interpretablity of a policy learned in a behavior cloning setting. For the second one, we compare the inference efficiency to existing work, highlighting the tractability of Koopman policy and its reinforcement learning results in a series of Gym tasks [39]. Lastly, we show a hardware validation by deploying the proposed policy in a robot pivoting task.

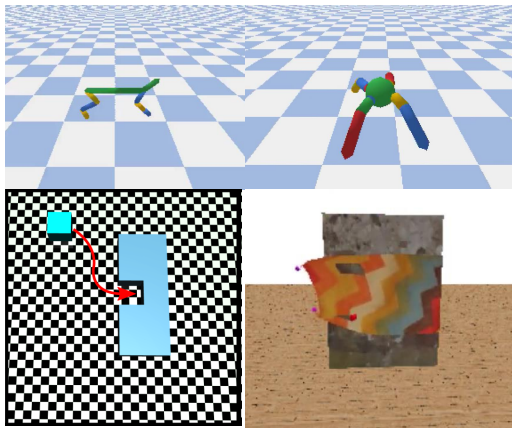


Fig. 3. Locomotion tasks: halfcheetah and ant; Object manipulation tasks: Block insertion and farbic buttoning.

A. Policy Interpretability and Inference Efficiency

We consider an inverted pendulum task as in [8] for the ease of visualization. The dataset contains 60 rollouts, collected by running an expert policy obtained from value iteration over discretized state and action spaces and a discounted factor of 0.99. Each rollout collects 100 time steps and the policy is tasked to regress the expert action, setting $\mathcal{L}_{\text{train}}$ in Equation (20) as a mean-square error. It can be seen from Figure 2(b) that the cost-to-go function constructed by learned parameters resembles a similar layout to the ground truth. Note that the target here is not accurately recovering the underlying quadratic cost function, because the actual form in Equation (17) is with a nonlinear feature ϕ and subject to different problem setups such as a finite horizon. What we would like to point out from Figure 2(b) is that Koopman policy indeed reflects a structure that can be interpreted by the LQR optimality. This may support understanding and reusing the learned policy in a modular manner.

The time costs for policy inference are illustrated in Figure 2(c), with the official implementation of differentiable MPC [8] as a comparison on the same workstation with an Intel i9 CPU. We get a similar performance as [8] that it takes seconds for differentiable MPC to infer the action to take under neural network dynamics. In contrast, Koopman policy has an overhead at the level of milliseconds and the cost only increases moderately for larger embedding size and planning horizon. This enables running the policy at a rate of 50 to 100Hz which is essential for online policy learning and real-time control. The performance of Koopman policy in inference can further be optimized by precomputing

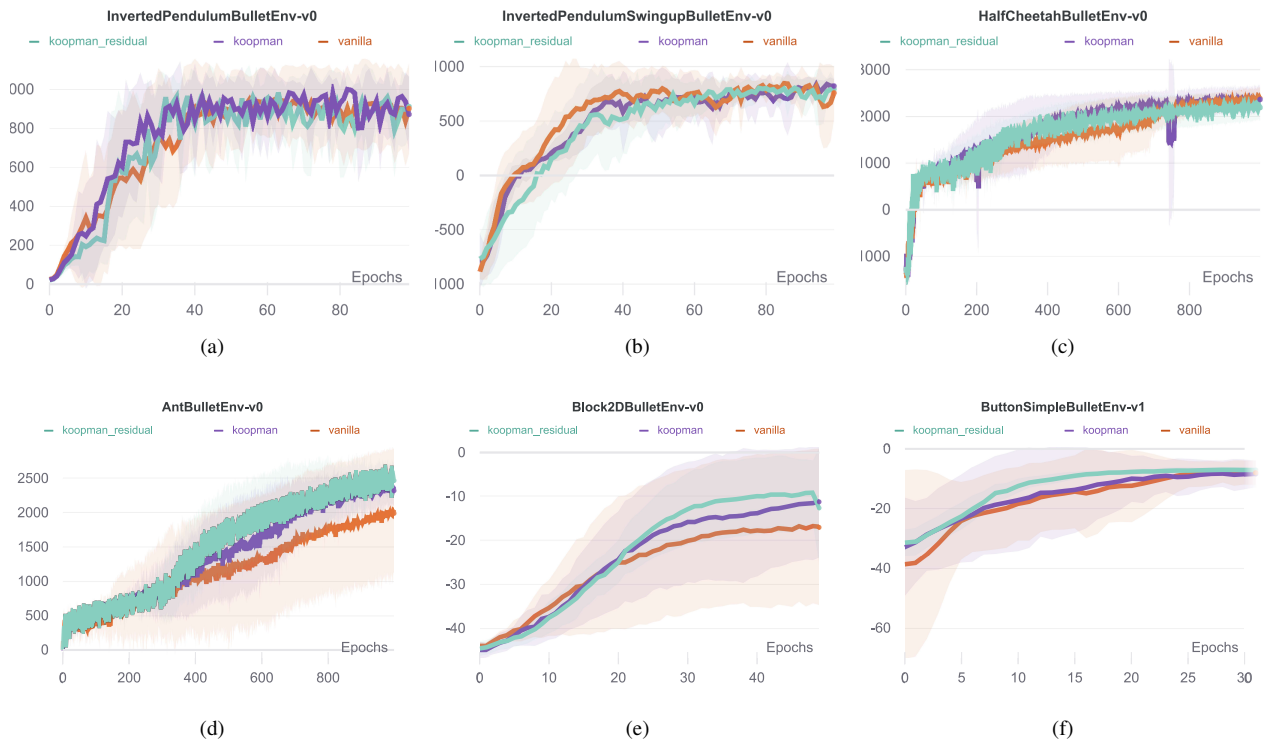


Fig. 4. Average return of reinforcement learning for pendulum control, locomotion and object manipulation tasks, with 10 seeds and shader regions indicating standard deviations.

the Riccati recursions in Figure 1(b), although this is not exploited in this test. In light of the tractability, we will focus on Koopman policies and basic neural network policies in the next subsection for reinforcement learning results.

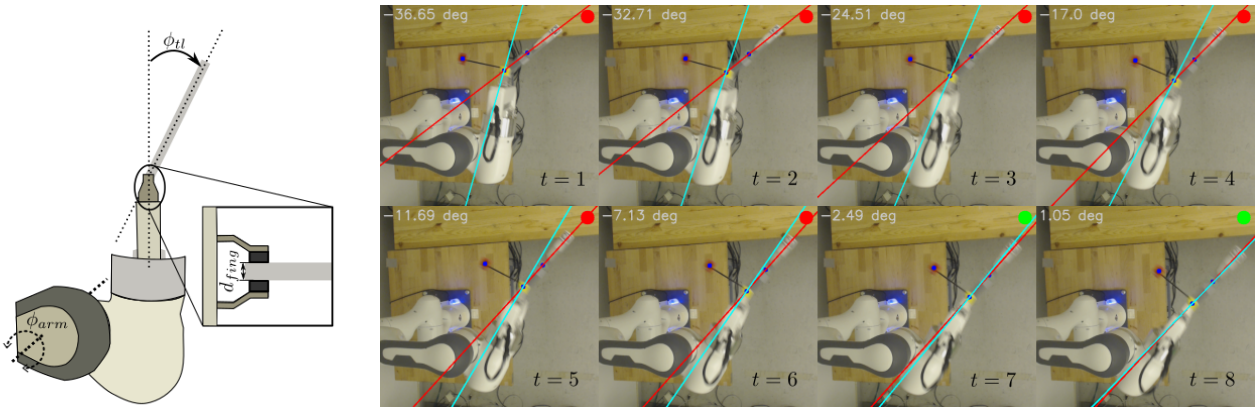
B. Reinforcement Learning for Robotic Tasks

We evaluate the effects of imposed Koopman structure in the context of reinforcement learning. The considered tasks are simulated as Gym environments on top of pybullet [40], ranging from inverted pendulum control, character locomotion to object manipulation (Figure 3). The proposed Koopman policy is applied to both on-policy and off-policy learning including Proximal Policy Optimization (PPO) [41] for pendulum control and object manipulation, and Soft Actor Critic (SAC) [42] for character locomotion, with additive noises compatible to the corresponding algorithms. We use implementation from the garage library [43], default algorithm parameters and the same neural networks for vanilla and Koopman policies, value functions and residual modules. Koopman policies use $k = 4$ and $T = 5$ in all tests since we don't find larger values having major influence when using pendulum as validation tasks. The total loss has $\lambda_{\text{reg}} = 0.01$ and all other coefficients are set to 1, except $\lambda_{\text{fit}} = 0.01$ and $\lambda_{\text{recons}} = 0$ in two object manipulation environments. The tests have a budget of about 500K environment steps for PPO tasks and 1M for SAC tasks. We give a brief introduction to the object manipulation tasks since others are standard Gym environments [39].

Block Insertion (Figure 3 bottom left): this resembles a peg-in-hole task by learning to insert a rigid block over a 2D plane [16]. The policy receives block position and velocity and outputs force actuation. The initial position of the block is randomized while the goal location is fixed with a clearance of 0.5mm. The environment uses a dense reward, measuring the distance to the goal location at each time step and imposing a large weight for the terminal. We migrate this environment to pybullet and the physics engine appears to make the task harder than the original one in [16].

Fabric Buttoning (Figure 3 bottom right): this task is one of the Dynamic Environments with Deformable Objects (DEDO) [44] where the policy moves two anchors to match fabric holes to buttons. Since the material is deformable, the hole can only be indirectly manipulated and the whole system is highly underactuated. At the end of each episode, the anchors will release the fabric and incur a large negative reward if the fabric drops, which is deemed as a failure. We discard camera input and use anchor position as the observation. Note that this is the only DEDO task that so far reinforcement learning baselines will not fail [44].

From learning curves in Figure 4, it is clear that Koopman policy and its variant perform at least comparable to a neural network baseline. This is probably not surprising since as discussed in Section IV-D, vanilla neural networks can be seen as a special Koopman policy under certain conditions. Asymptotically the performance converges in most environments except Figure 4(e) which may benefit from the goal-directed structure imposed by the LQR problem. Although locomotion tasks favor limited-cycle behaviors, Koopman policies work equally well for them and appear



(a) (b)

Fig. 5. Pivot task setup, and eight consecutive frames from the observational camera during the $-40°$ pivot to zero task. To safeguard the hardware the simulation sets an operational space of $\approx \pm 71°$ and on the robot a low-level safety controller is triggered when the arm gets close to these joint limits to push the arm back.

to learn faster especially in more challenging environments, e.g., 3D locomotion in Figure 4(d). In the deformable task (Figure 4(f)), the performance boost, in particular from the basic form without a residual, is not as significant as in Figure 4(e). This may be due to the the partial observability from the anchor position whose dynamics are insufficient for fully determining the fabric state.

C. Real-world Pivoting Task

We deploy Koopman residual policy on physical hardware to perform a pivoting, demonstrating thus applicability in a highly dynamic task. An object is placed in Franka Emika Panda's gripper at a non zero angle: we selected $[-40°, -30°, -20°, -10°, 10°, 20°, 30°, 40°]$ for the validation. The acceleration $\ddot{\phi}_{arm}$ of the sixth rotational joint as well as the gripper opening distance d_{fing} are controlled to pivot the object to a $0°$ orientation relative to the gripper. The task is very similar to the pivoting task presented by [45]. We employ the same action space but reduce the observation space to the relative angle ϕ_{tl}, the current arm rotational position ϕ_{arm} as well as the velocity $\dot{\phi}_{arm}$. The system is trained in simulation employing the same dynamic models and randomization as in [45] with the notable difference of changing the friction model such that the pole sticks when the gripper finger distance goes below a minimum threshold $d_{min} = 0.0185$. The model is trained for 20 epochs and then deployed in a zero-shot fashion onto the real robotics system. Figure 5(a) shows the pivoting setup, while Figure 5(b) shows eight consecutive frames from the observational camera during the execution of the $-40°$ pivot to zero task. Successful execution videos of all eight starting configuration can be found on the project website[1].

VI. Conclusions

We contribute a policy form with embedded optimization that is applicable for online learning and inference. The Koopman representation is shown as the key of enabling non-convex nonlinearity with an efficient optimality structure. Our validation finds the policy is versatile and effective in learning a set of robotic tasks, ranging from simulated pendulum control, locomotion to object manipulation and deploying it for a pivoting task on a real robotic system..

Our method does have some limitations that motivate our ongoing research. The current constraint enforcement requires to trade off a few auxiliary objectives. We find it generally works to use default weights from literature with similar terms due to balance of optimizing the auxiliary terms and the policy likelihood. The reconstruction term is also redundant in many cases because only encoder is needed for learning latent dynamics [29]. The method will benefit from replacing auxiliary objectives with less restrictive terms or a parameterization fulfilling these constraints by design. We also observe exploded gradient issues that sometimes halt entire learning process, especially when very large T and m are used. This is caused by inverting poorly conditioned matrices, an issue that appears fundamental for reliably using automatic differentiation and many differentiable structures [46]. Lastly, extensions for including constraints also encoded by the Koopman representation may facilitate more optimization problems and application on real systems.

[1] https://koopman-learning.github.io/web/

References

[1] J. Buchli, F. Stulp, E. Theodorou, and S. Schaal, "Learning variable impedance control," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 820–833, 2011.

[2] R. Martín-Martín, M. A. Lee, R. Gardner, S. Savarese, J. Bohg, and A. Garg, "Variable impedance control in end-effector space: An action space for reinforcement learning in contact-rich tasks," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1010–1017, 2019.

[3] M. Bogdanovic, M. Khadiv, and L. Righetti, "Learning variable impedance control for contact sensitive tasks," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6129–6136, 2020.

[4] S. Boyd, A. Agrawal, and S. Barratt, "Plenary lecture: Embedded convex optimization for control." Conference on Decision and Control (CDC), 2020.

[5] E. Rückert, G. Neumann, M. Toussaint, and W. Maass, "Learned graphical models for probabilistic planning provide a new class of movement primitives," *Frontiers in Computational Neuroscience*, vol. 6, p. 97, 2013.

[6] M. Okada, L. Rigazio, and T. Aoshima, "Path integral networks: End-to-end differentiable optimal control," *CoRR*, vol. abs/1706.09597, 2017.

[7] A. Srinivas, A. Jabri, P. Abbeel, S. Levine, and C. Finn, "Universal planning networks: Learning generalizable representations for visuomotor control," in *Proceedings of the 35th International Conference on Machine Learning (ICML), 2018*, vol. 80 of *Proceedings of Machine Learning Research*, pp. 4739–4748, PMLR, 2018.

[8] B. Amos, I. Jimenez, J. Sacks, B. Boots, and J. Z. Kolter, "Differentiable MPC for End-to-end Planning and Control," in *Advances in Neural Information Processing Systems*, 2018.

[9] A. Agrawal, S. Barratt, S. Boyd, and B. Stellato, "Learning convex optimization control policies," in *Proceedings of the 2nd Conference on Learning for Dynamics and Control*, vol. 120 of *Proceedings of Machine Learning Research*, pp. 361–373, PMLR, 10–11 Jun 2020.

[10] W. Jin, Z. Wang, Z. Yang, and S. Mou, "Pontryagin differentiable programming: An end-to-end learning and control framework," in *Advances in Neural Information Processing Systems* (H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, eds.), vol. 33, pp. 7979–7992, Curran Associates, Inc., 2020.

[11] A. Agrawal, B. Amos, S. Barratt, S. Boyd, S. Diamond, and Z. Kolter, "Differentiable convex optimization layers," in *Advances in Neural Information Processing Systems*, 2019.

[12] B. O. Koopman, "Hamiltonian systems and transformation in hilbert space," *Proceedings of the National Academy of Sciences*, vol. 17, no. 5, pp. 315–318, 1931.

[13] G. Mamakoukas, M. L. Castano, X. Tan, and T. D. Murphey, "Derivative-based koopman operators for real-time control of robotic systems," *IEEE Transactions on Robotics*, 2021.

[14] D. Bruder, X. Fu, R. B. Gillespie, C. D. Remy, and R. Vasudevan, "Data-driven control of soft robots using koopman operator theory," *IEEE Transactions on Robotics*, vol. 37, no. 3, pp. 948–961, 2021.

[15] L. Rozo, D. Bruno, S. Calinon, and D. G. Caldwell, "Learning optimal controllers in human-robot cooperative transportation tasks with position and force constraints," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1024–1030, 2015.

[16] S. A. Khader, H. Yin, P. Falco, and D. Kragic, "Stability-guaranteed reinforcement learning for contact-rich manipulation," *IEEE Robotics and Automation Letters*, vol. 6, no. 1, pp. 1–8, 2020.

[17] S. A. Khader, H. Yin, P. Falco, and D. Kragic, "Learning stable normalizing-flow control for robotic manipulation," in *International Conference on Robotics and Automation (ICRA)*, IEEE, 2021.

[18] S. A. Khader, H. Yin, P. Falco, and D. Kragic, "Learning deep energy shaping policies for stability-guaranteed manipulation," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 8583–8590, 2021.

[19] A. Marco, P. Hennig, S. Schaal, and S. Trimpe, "On the design of lqr kernels for efficient controller learning," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pp. 5193–5200, 2017.

[20] N. A. Vien and G. Neumann, "Differentiable robust LQR layers," *CoRR*, vol. abs/2106.05535, 2021.

[21] S. Saxena, A. LaGrassa, and O. Kroemer, "Learning reactive and predictive differentiable controllers for switching linear dynamical models," in *Proceedings of (ICRA) International Conference on Robotics and Automation*, May 2021.

[22] J. L. Proctor, S. L. Brunton, and J. N. Kutz, "Generalizing koopman theory to allow for inputs and control," *SIAM Journal on Applied Dynamical Systems*, vol. 17, no. 1, pp. 909–930, 2018.

[23] M. Korda and I. Mezić, "Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control," *Automatica*, vol. 93, pp. 149–160, 2018.

[24] M. Korda and I. Mezić, "Optimal construction of koopman eigenfunctions for prediction and control," *IEEE Transactions on Automatic Control*, vol. 65, no. 12, pp. 5114–5129, 2020.

[25] G. Mamakoukas, I. Abraham, and T. D. Murphey, "Learning data-driven stable koopman operators," in *Advances in Neural Information Processing Systems*, Curran Associates, Inc., 2020.

[26] J. Chung, K. Kastner, L. Dinh, K. Goel, A. C. Courville, and Y. Bengio, "A recurrent latent variable model for sequential data," in *Advances in Neural Information Processing Systems* (C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, eds.), vol. 28, Curran Associates, Inc., 2015.

[27] 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* (C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, eds.), vol. 28, Curran Associates, Inc., 2015.

[28] M. Karl, M. Soelch, J. Bayer, and P. van der Smagt, "Deep variational bayes filters: Unsupervised learning of state space models from raw data," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, OpenReview.net, 2017.

[29] R. Shu, T. Nguyen, Y. Chow, T. Pham, K. Than, M. Ghavamzadeh, S. Ermon, and H. Bui, "Predictive coding for locally-linear control," in *Proceedings of the 37th International Conference on Machine Learning*, vol. 119 of *Proceedings of Machine Learning Research*, PMLR, 2020.

[30] M. Jaques, M. Burke, and T. M. Hospedales, "Newtonianvae: Proportional control and goal identification from pixels via physical latent spaces," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4454–4463, June 2021.

[31] N. Takeishi, Y. Kawahara, and T. Yairi, "Learning koopman invariant subspaces for dynamic mode decomposition," in *Advances in Neural Information Processing Systems*, vol. 30, Curran Associates, Inc., 2017.

[32] B. Lusch, J. N. Kutz, and S. L. Brunton, "Deep learning for universal linear embeddings of nonlinear dynamics," *Nature Communications*, vol. 9, p. 4950, Nov 2018.

[33] E. Yeung, S. Kundu, and N. Hodas, "Learning deep neural network representations for koopman operators of nonlinear dynamical systems," in *2019 American Control Conference (ACC)*, pp. 4832–4839, 2019.

[34] J. Morton, F. D. Witherden, and M. J. Kochenderfer, "Deep variational koopman models: Inferring koopman observations for uncertainty-aware dynamics modeling and control," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pp. 3173–3179, International Joint Conferences on Artificial Intelligence Organization, 7 2019.

[35] Y. Li, H. He, J. Wu, D. Katabi, and A. Torralba, "Learning compositional koopman operators for model-based control," in *International Conference on Learning Representations*, 2020.

[36] M. O. Williams, I. G. Kevrekidis, and C. W. Rowley, "A data-driven approximation of the koopman operator: Extending dynamic mode decomposition," *Journal of Nonlinear Science*, vol. 25, no. 6, pp. 1307–1346, 2015.

[37] M. Lutter, C. Ritter, and J. Peters, "Deep lagrangian networks: Using physics as model prior for deep learning," in *International Conference on Learning Representations (ICLR)*, 2019.

[38] A. Zeng, S. Song, J. Lee, A. Rodriguez, and T. Funkhouser, "Tossingbot: Learning to throw arbitrary objects with residual physics," *IEEE Transactions on Robotics*, vol. 36, no. 4, pp. 1307–1319, 2020.

[39] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *CoRR*, vol. abs/1606.01540, 2016.

[40] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning." `http://pybullet.org`, 2016–2021.

[41] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017.

[42] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," *CoRR*, vol. abs/1801.01290, 2018.

[43] T. garage contributors, "Garage: A toolkit for reproducible reinforcement learning research." `https://github.com/rlworkgroup/garage`, 2019.

[44] R. Antonova, P. Shi, H. Yin, Z. Weng, and D. K. Jensfelt, "Dynamic environments with deformable objects," in *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

[45] R. Antonova, S. Cruciani, C. Smith, and D. Kragic, "Reinforcement learning for pivoting task," *arXiv preprint arXiv:1703.00472*, 2017.

[46] L. Metz, C. D. Freeman, S. S. Schoenholz, and T. Kachman, "Gradients are not all you need," *arXiv preprint arXiv:2111.05803*, 2021.