On the Utility of Koopman Operator Theory in Learning Dexterous Manipulation Skills

Anonymous Author(s)
Affiliation
Address
email

Abstract: Despite impressive dexterous manipulation capabilities enabled by learning-based approaches, we are yet to witness widespread adoption beyond well-resourced laboratories. This is likely due to practical limitations, such as significant computational burden, inscrutable learned behaviors, sensitivity to initialization, and the considerable technical expertise required for implementation. In this work, we investigate the utility of Koopman operator theory in alleviating these limitations. Koopman operators are simple yet powerful control-theoretic structures to represent complex *nonlinear* dynamics as *linear* systems in higher dimensions. Motivated by the fact that complex nonlinear dynamics underlie dexterous manipulation, we develop a Koopman operator-based imitation learning framework to learn the desired motions of both the robotic hand and the object simultaneously. We show that Koopman operators are surprisingly effective for dexterous manipulation and offer a number of unique benefits. Notably, policies can be learned *analytically*, drastically reducing computation burden and eliminating sensitivity to initialization and the need for painstaking hyperparameter optimization. Our experiments reveal that a Koopman operator-based approach can perform comparably to state-of-the-art imitation learning algorithms in terms of success rate and sample efficiency, while being *an order of magnitude* faster. Policy videos can be viewed at https://sites.google.com/view/kodex-corl.

Keywords: Koopman Operator, Dexterous Manipulation

1 Introduction

Autonomous dexterous manipulation is a necessary skill for robots operating in a physical world built by and for humans. However, achieving reliable robotic dexterous manipulation skills has been a long-standing challenge [1] due to numerous factors, such as complex nonlinear dynamics, high-dimensional action spaces, and expertise required to design bespoke controllers.

Over the past decade, learning-based solutions have emerged as promising solutions that can address the challenges in acquiring dexterous manipulation skills. Indeed, these methods have been shown to be capable of impressive feats, such as solving Rubik's cubes [2], manipulation Baoding balls [3], retrieving tool trays [4], and reorienting complex objects [5]. However, existing learning approaches suffer from practical limitations that hinder their widespread adoption. First, implementing existing algorithms requires significant technical expertise and modern machine learning infrastructure. Second, training policies consume significant computational resources. Third, existing approaches demand painstaking effort in tuning hyperparameters to achieve acceptable performance. Fourth, performance tends to be highly sensitive to parameter initialization.

In this work, we investigate the utility of Koopman operator theory in alleviating the limitations of existing learning-based approaches as identified above. The Koopman operator theory helps represent arbitrary *nonlinear* dynamics in finite dimensional spaces as *linear* dynamics in an infinite-

Submitted to the 7th Conference on Robot Learning (CoRL 2023). Do not distribute.

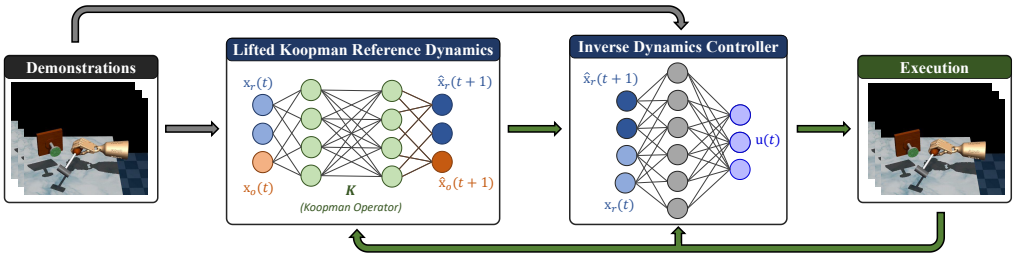


Figure 1: KODex simultaneously encodes complex nonlinear dynamics of the desired motion of both the robot state (x_r) and the object state (x_o) as a *linear* dynamical system in a higher-dimensional space by learning a Koopman operator $\boldsymbol{K}$ directly from demonstrations. Further, KODex learns an inverse dynamics controller to track the robot reference trajectory ($\{\hat{\mathrm{x}}_r(t)\}_{t=1}^{T}$) generated by the lifted linear system.

dimensional Hilbert space [6]. While this equivalence is exact and fascinating from a theoretical standpoint, it is not tractable. However, recent advances have enabled the approximation of this equivalence in higher but *finite*-dimensional spaces by learning the operator directly from data [7].

We develop a novel imitation learning framework, dubbed *Koopman Operator-based Dexterous Manipulation (KODex)*, to evaluate the utility of Koopman operator theory for dexterous manipulation (see Fig. 1). Specifically, we model desired behaviors as solutions to nonlinear dynamical systems and learn Koopman operators that define approximately-equivalent linear dynamics in higher-dimensional spaces. Note that it is insufficient to exclusively focus on the robot's motion as the objective of dexterous manipulation is centered on the object's motion [1]. As such, KODex *simultaneously* learns the desired motions of both the robot and the object from demonstrations. To eliminate the need for an expertly-tuned PID controller, KODex relies on a learned inverse dynamics controller to track the reference trajectory generated by the learned dynamical system.

A significant benefit of learning Koopman operators from data is that it lends itself to an *analytical* solution. As such, KODex is simple to implement and does not require expertise and familiarity with state-of-the-art (SOTA) machine learning infrastructure. Instead of painstaking hyperparameter optimization, we show that generic and task-agnostic polynomial lifting functions are sufficient for KODex to learn diverse dexterous manipulation skills. Further, KODex offers consistent and predictable performance since the learning process is analytical and thus not sensitive to parameter initialization. Finally, given that KODex learns a *linear* dynamical system, one can readily inspect the learned behaviors using a wide array of control theoretic tools.

We carry out extensive evaluations of KODex within the context of four dexterous manipulation skills on the simulated Adroit hand, an established experimental platform for dexterous manipulation [8]. Further, we compare KODex against SOTA imitation learning approaches in terms of general efficacy, computational efficiency, and sample efficiency. Our results demonstrate that KODex is at least an *order of magnitude* faster to train than SOTA imitation learning algorithms, while achieving comparable sample efficiency and task success rates. These results suggest that Koopman operators can be effective, efficient, and reliable tools to learn dexterous manipulation skills and to reduce the barriers to wide-spread adoption.

2 Related Work

In this section, we contextualize our contributions within relevant sub-fields.

Learning Manipulation Skills as Dynamical Systems: Our work falls into the category of dynamical-system-based imitation learning methods for manipulation [9]. Made popular by the Dynamics Movement Primitives (DMPs) [10], these methods model robot motions as solutions to a learnable underlying dynamical system. The past decade witnessed a plethora of approaches built upon the same principle (e.g., [11–16]), creating increasingly-capable LfD tools for manipulation. Robustness to perturbations, high sample efficiency, and provable convergence are all but a few ex-

amples of the many advantages of dynamical-system-based approaches. These approaches tend to be highly structured and leverage control-theoretic and topological tools to learn complex desired motions with unparalleled sample efficiency. Recent work also embeded the dynamical systems structure into deep neural networks to enable end-to-end learning [17]. These approaches were primarily designed to capture low-dimensional end-effector skills for serial-link manipulators. In contrast, our work investigates the utility of Koopman operators in learning dexterous manipulation skills on high-DOF platforms.

Learning Dexterous Manipulation Skills: Deep Reinforcement Learning (RL) has been dominating the field of dexterous manipulation recently, enabling an impressive array of skills [3, 18, 19]. A popular approach demonstrated that a multi-finger hand can learn to solve the Rubik's cube OpenAI et al. [2]. Recently, Chen et al. [5] developed a model-free RL framework capable of reorienting over 2000 differently-shaped objects. Despite impressive successes, RL-based algorithms suffer from poor sample efficiency and notoriously-difficult training procedures. In contrast, Imitation learning (IL) aims to improve sample efficiency by leveraging expert demonstrations [9, 20]. However, most existing IL-based methods (including those discussed in Section 2) focus on lower-DOF manipulators and do not scale well to high-dimensional systems. Indeed, there are at least two recent notable exceptions to this limitation: Xie et al. [4] developed a highly structured IL method to learn dexterous manipulation skills from demonstrations in the form of a dynamical system; Arunachalam et al. [21] introduced a mixed-reality framework to collect high-quality demonstrations and learn dexterous manipulation skills by leveraging visual representations and motion retargeting. Researchers have also combined IL with RL to get the best of both approaches and has been able to achieve impressive performance (e.g., [8, 22]). A common attribute of all existing learning approaches to dexterous manipulation is that they rely on significant computational resources and user expertise for implementation and hyperparameter tuning. Further, the effectiveness of these approaches is highly sensitive to parameter initialization [23]. In stark contrast, KODex encodes complex skills as dynamical systems which are *analytically* extracted from demonstrations. As such, KODex incurs a significantly smaller computational burden and eliminates the dependence on painstaking hyperparameter tuning and unreliable numerical optimization. Further, unlike opaque deep neural networks, KODex learns *linear* dynamical systems that can be inspected via control-theoretic tools.

Koopman Operators in Robotics: Recently, Koopman operator theory has proven beneficial in various robotic systems, such as differential drive robots [24], spherical and serial-link manipulators [25], and soft robotic manipulators [26, 27]. However, the systems investigated in these works are low-dimensional. In contrast, our work is focused on evaluating the effectiveness of Koopman operators in learning skills for a high-dimensional system with complex dynamics (i.e., a multi-fingered hand). Further, prior works have not sufficiently investigated the relative benefits of leveraging Koopman operators over SOTA neural network-based approaches, and the circumstances under which these benefits hold. In our work, we thoroughly evaluate KODex against SOTA imitation learning methods within the context of multiple dexterous manipulation tasks.

3 Preliminary: Koopman Operator Theory

We begin by providing a brief introduction to Koopman operator theory [6].

Koopman Representation: Consider a discrete-time autonomous nonlinear dynamical system

$$\mathrm{x}(t+1) = F(\mathrm{x}(t)), \tag{1}$$

where $\mathrm{x}(t) \in \mathcal{X} \subset \mathbb{R}^n$ is the state at time t, and $F(\cdot) : \mathbb{R}^n \to \mathbb{R}^n$ is a nonlinear function.

To represent the the nonlinear dynamical system in (1) as a linear system, we begin by introducing a set of *observables* using the so-called *lifting function* $g : \mathcal{X} \to \mathcal{O}$, where $\mathcal{O}$ is the space of observables. We can now define the *Koopman Operator* $\mathcal{K}$, an infinite-dimensional operator on the lifting function $g(\cdot)$ for the discrete time system defined in (1) as follows

$$[\mathcal{K}g] = g(F(\mathrm{x}(t))) = g(\mathrm{x}(t+1)) \tag{2}$$

If the observables belong to a vector space, the Operator $\mathcal{K}$ can be seen as an infinite-dimensional linear map that describes the evolution of the observables as follows

$$g(\mathrm{x}(t+1)) = \mathcal{K}g(\mathrm{x}(t)) \tag{3}$$

Therefore, the Koopman operator $\mathcal{K}$ linearly propagates forward the infinite-dimensional lifted states (i.e., observables). In practice, we do not benefit from this representation since it is infinite-dimensional. However, we can approximate $\mathcal{K}$ using a matrix $\mathbf{K} \in \mathbb{R}^{p\times p}$ and define a finite set of observables $\phi(t) \in \mathbb{R}^p$. Thus, we can rewrite the relationship in (3) as

$$\phi(\mathrm{x}(t+1)) = \mathbf{K}\phi(\mathrm{x}(t)) + r(\mathrm{x}(t)), \tag{4}$$

where $r(\mathrm{x}(t)) \in \mathbb{R}^p$ is the residual error caused by the finite dimensional approximation, which can be arbitrarily reduced based on the choice of p.

Learning Koopman Operator from Data: The matrix operator $\mathbf{K}$ can be inferred from a dataset $D = [\mathrm{x}(1), \mathrm{x}(2), \cdots, \mathrm{x}(T)]$, which contains the solution to the system in (1) for T time steps. Given the choice of observables $\phi(\cdot)$, the finite dimensional Koopman matrix $\mathbf{K}$ is computed by minimizing the approximation error defined in (4). Specifically, we can obtain $\mathbf{K}$ from D by minimizing the cost function $\mathbf{J}(\mathbf{K})$ given below

$$\mathbf{J}(\mathbf{K}) = \frac{1}{2}\sum_{t=1}^{t=T-1} \|r(\mathrm{x}(t))\|^2 = \frac{1}{2}\sum_{t=1}^{t=T-1} \|\phi(\mathrm{x}(t+1)) - \mathbf{K}\phi(\mathrm{x}(t))\|^2 \tag{5}$$

Note minimizing $\mathbf{J}(\mathbf{K})$ amounts to solving a least-square problem, whose solution is given by [7]

$$\mathbf{K} = \mathbf{A}\mathbf{G}^{\dagger};\ \ \mathbf{A} = \frac{1}{T-1}\sum_{t=1}^{t=T-1} \phi(\mathrm{x}(t+1)) \otimes \phi(\mathrm{x}(t)),\ \ \mathbf{G} = \frac{1}{T-1}\sum_{t=1}^{t=T-1} \phi(\mathrm{x}(t)) \otimes \phi(\mathrm{x}(t)) \tag{6}$$

where $\mathbf{G}^{\dagger}$ denotes the Moore–Penrose inverse of $\mathbf{G}$, and $\otimes$ denotes the outer product.

4 Learning Koopman Operators for Dexterous Manipulation

We begin by introducing our framework to model dexterous manipulation skills as nonlinear dynamics and discuss the importance of incorporating object states into the system (Section 4.1). Next, we describe how KODex learns the reference dynamics for a given skill from demonstrations (Section 4.2). Finally, we discuss how to learn a low-level controller, also from demonstrations, in order to faithfully track the reference trajectories generated by KODex (Section 4.3). Further, an overall pseudo-code for KODex can be found in Appendices A.

4.1 Modeling Dexterous Manipulation Skills

A central principle behind KODex is that the desired behavior of a robot can be represented using a dynamical system. Note that, unlike other kinds of manipulation skills (e.g., end-effector skills of multi-link manipulators), dexterous manipulation is explicitly concerned with how an object moves as a result of the robot's motion [1]. As such, KODex captures the desired motion of the robot along with that of the object. To this end, we define the state at time t as $\mathrm{x}(t) = [\mathrm{x}_r(t)^\top, \mathrm{x}_o(t)^\top]^\top$, where $\mathrm{x}_r(t) \in \mathcal{X}_r \subset \mathbb{R}^n$ and $\mathrm{x}_o(t) \in \mathcal{X}_o \subset \mathbb{R}^m$ represent the state of the robot and the object, respectively, at time t. As such, the dynamical system we wish to capture is

$$\mathrm{x}(t+1) = F^*(\mathrm{x}(t)) \tag{7}$$

where $F^*(\cdot) : \mathcal{X}_r \times \mathcal{X}_o \to \mathcal{X}_r \times \mathcal{X}_o$ denotes the true dynamics that govern the *interdependent* motions of the robot and the object. Note that this system is time-invariant. Indeed, time-invariant dynamical systems provide a natural way to capture manipulation skills that are more robust to intermittent perturbations than those that explicitly depend on time [9].

A key challenge in learning the dynamical system in (7) is that it can be arbitrarily complex and highly nonlinear, depending on the particular skill of interest. KODex leverages Koopman operator

theory to learn a *linear* dynamical system that can effectively approximate such complex nonlinear dynamics. To this end, we first define a set of observables as follows

$$\phi(\mathrm{x}(t)) = [\mathrm{x}_r(t)^\top, \psi_r(\mathrm{x}_r(t)), \mathrm{x}_o(t)^\top, \psi_o(\mathrm{x}_o(t))]^\top, \forall t \tag{8}$$

where $\psi_r : \mathbb{R}^n \to \mathbb{R}^{n'}$ and $\psi_o : \mathbb{R}^m \to \mathbb{R}^{m'}$ are vector-valued lifting functions that transform the robot and object state respectively.

In our implementation, we use polynomial functions up to a finite degree in our lifting function since polynomial functions allow for flexible definition of complex functions. However, it is important to note that KODex is agnostic to the specific choice of observables. Further, we do not assume that we know the ideal set of observables for any given skill. Instead, as we demonstrate in Section 5, KODex can learn different dexterous manipulation skills on the same space of observables.

4.2 Learning Reference Dynamics

We now turn to the challenge of learning the Koopman operator $\mathbf{K}$ from demonstrations. Let $D = [\{\mathrm{x}^{(1)}(t), \tau^{(1)}(t)\}_{t=1}^{t=T^{(1)}}, \cdots, \{\mathrm{x}^{(N)}(t), \tau^{(N)}(t)\}_{t=1}^{t=T^{(N)}}]$ denote a set of N demonstrations containing trajectories of state-torque pairs. Now, we can compute the Koopman matrix as $\mathbf{K} = \mathbf{A}\mathbf{G}^\dagger$, where $\mathbf{A}$ and $\mathbf{G}$ can be computed by modifying the expressions in (6) as follows

$$\mathbf{A} = \sum_{n=1}^{n=N} \sum_{t=1}^{t=T^{(n)}-1} \frac{\phi(\mathrm{x}^n(t+1)) \otimes \phi(\mathrm{x}^n(t))}{N(T^{(n)}-1)}, \ \mathbf{G} = \sum_{n=1}^{n=N} \sum_{t=1}^{t=T^{(n)}-1} \frac{\phi(\mathrm{x}^n(t)) \otimes \phi(\mathrm{x}^n(t))}{N(T^{(n)}-1)} \tag{9}$$

It is worth noting that KODex can also leverage partial trajectories that do not complete the task, as long as all the state pairs $(\mathrm{x}^n(t), \mathrm{x}^n(t+1))$ are temporally consecutive. Additionally, we also record the actuated torque $\tau(t)$ at each time step for the controller design discussed in Section 4.3.

We use the learned reference dynamics to generate rollouts in the observables space. However, we need to obtain the rollouts in the original robot states to command the robot. Since we designed $\phi(\mathrm{x}(t))$ such that both robot state $\mathrm{x}_r(t)$ is a part of the observables in (8), we can retrieve the desired robot trajectory $\{\hat{\mathrm{x}}_r(t)\}$ by selecting the corresponding elements in $\phi(\mathrm{x}(t))$.

Indeed, the data distribution in D has a considerable effect on the generalizability of the computed Koopman matrix $\mathbf{K}$. Therefore, there is an inevitable trade-off between the number of demonstrations and the cost of data collection - a challenge shared by most imitation learning algorithms.

4.3 Learning a Tracking Controller

To track the desired trajectories generated from the learned reference dynamics, we design an inverse dynamics controller C [28] [17]. Indeed, a PID controller could be used instead of learning a tracking controller. However, one would have to painstakingly tune the control gains and frequency, and do so for each task independently.

We use a multi-layer perception (MLP) as the tracking controller and train it using the recorded state-torque pairs $(\mathrm{x}_r^n(t), \mathrm{x}_r^n(t+1), \tau^n(t))$ by minimizing

$$\mathcal{L}_{\text{control}} = \sum_{n=1}^{n=N} \sum_{t=1}^{t=T^{(n)}-1} \frac{|C(\mathrm{x}_r^n(t), \mathrm{x}_r^n(t+1)) - \tau^n(t)|^2}{N(T^{(n)}-1)} \tag{10}$$

The learned controller takes as input the current robot state $\mathrm{x}_r(t)$ and the desired next state from the reference trajectory $\hat{\mathrm{x}}_r(t+1)$, and generates the torque $\tau(t)$ required for actuation.

5 Experimental Evaluation

We evaluated KODex along with existing approaches in terms of their general efficacy, computational efficiency, sample efficiency, and scalability when learning dexterous manipulation skills.

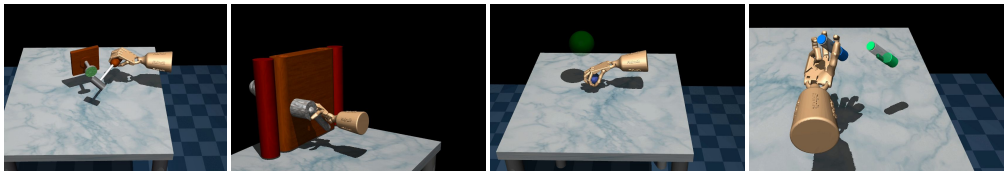


Figure 2: We evaluate KODex on four tasks from [8]: Tool Use, Door Opening, Relocation, and Reorientation.

5.1 Experimental Design

Evaluation Platform: We conducted all our experiments on the widely-used ADROIT Hand [8] – a 30-DoF simulated system (24-DoF hand + 6-DoF floating wrist base) built with MuJoCo [29].

Baselines: We compared KODex against the following baselines:

- *NN*: Fully-connected neural network policy
- *LSTM*: Recurrent neural network policy with Long Short-Term Memory (LSTM) units
- *NDP*: Neural Dynamic policy [17]
- *NGF*: Neural Geometric Fabrics policy [4]

Note that NN and LSTM are unstructured baselines which help question the need for structured policies. NDP and NGF are highly-structured SOTA imitation learning methods for manipulation.

We undertook several precautions to ensure a fair comparison. First, we designed the robot and object state space for all baselines and KODex to be identical. Second, we carefully designed the baselines policies and tuned their hyper-parameters for each baseline method (Appendices E and F). Third, we trained each baseline policy over five random seeds to control for initialization effects. For all tasks, we saved the baseline policies that performed the best on a validation set of 50 held-out demonstrations. Note that KODex utilizes an analytical solution and thus does not require parameter initialization or hyper parameter optimization.

Tasks: We evaluated all algorithms on a set of four tasks originally proposed in [8] (see Fig. 2).

- *Tool use*: Pick up the hammer to drive the nail into the board placed at a randomized height.
- *Door opening*: Given a randomized door position, undo the latch and drag the door open.
- *Object relocation*: Move the blue ball to a randomized target location (green sphere).
- *In-hand reorientation*: Reorient the blue pen to a randomized goal orientation (green pen).

See Appendices B for the state space design of all tasks. For all tasks, the reference dynamics was queried at 100HZ, and the controller ran at 500HZ.

Metrics: We quantify performance in terms of i) *Training time*: Time taken to train a policy, ii) *Imitation error*: The $L1$ distance between generated joint trajectories and the demonstrations, and iii) *Task success rate*: Percentage of successful trials (see Appendices C for success criteria).

Expert Policy: For each task, we trained an expert RL agent using DAPG [8] to generate 250 expert demonstrations (200 for training and 50 for validation). See Appendices D for further details.

Inverse Dynamics Controller: To standardize controller performance across methods, we trained a common inverse dynamics controller for each task using 250 demonstrations (see Appendices G).

5.2 General Efficacy

In Fig. 3, we report the training time, imitation error, and task success rate for each method on each task, when trained on 200 demonstrations and tested on 10,000 testing instances.

Training time: As can be seen, KODex is *an order of magnitude* faster than both unstructured baselines (NN, LSTM) and SOTA IL methods (NDP, NGF). Further, this trend holds across all the tasks. This is to be expected since KODex analytically computes the Koopman operator unlike all the baselines, which rely on gradient descent and numerical optimization.

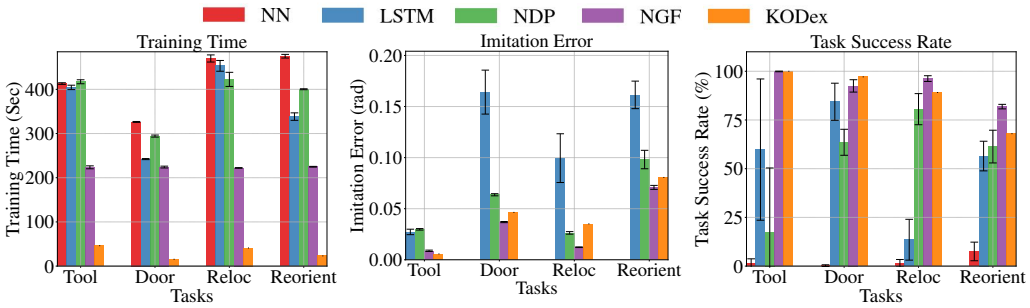
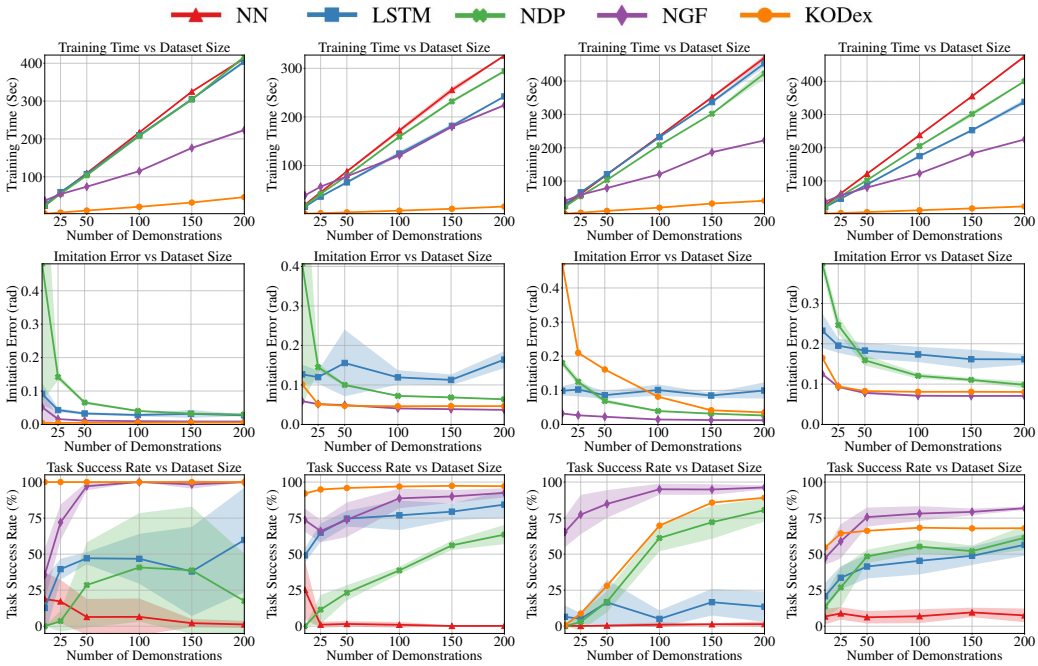


Figure 3: We report training time (left), imitation error (middle), and success rate (right) for methods on each task when trained on 200 demonstrations and evaluated on an independent set of 10,000 samples. Error bars for baseline methods, show the standard deviation over five random seeds.



(a) Tool Use (Hammer) (b) Door Opening (c) Object Relocation (d) In-hand Reorientation

Figure 4: The effects of number of demonstrations on training time (top row), imitation error (middle row), and success rate (bottom row) for all methods on each task. Solid lines indicate median trends and shaded areas show $\pm$ standard deviation over five random seeds.

Imitation error: In general, all methods (except NN) achieve low imitation error for the Tool Use task with negligible difference across methods. In the three remaining tasks, we see that all structured methods (NDP, NGF, KODex) considerably outperform the unstructured baseline (LSTM). We have excluded the significantly larger imitation errors generated by NN from the plot to preserve the resolution necessary to distinguish between the other methods. These results reinforce the effectiveness of structured methods in imitating demonstrations. Importantly, KODex is able to achieve imitation performance comparable to the SOTA IL methods despite its simplicity, while remaining an order of magnitude more computationally efficient.

Task success rate: As expected, the NN policy performs significantly worse than all other methods. On the other hand, LSTM achieves impressive task success rates, even outperforming NDP in two of the four tasks. This is in stark contrast to its high imitation error. While counter-intuitive, this observation follows the recent finding that imitation error and task success rate might not necessarily be correlated [30]. We observe that KODex and NGF perform comparably, with one achieving a

higher task success rate than the other in two of the four tasks. Importantly, KODex results in the most consistent and predictable performance due to its lack of sensitivity to initialization.

5.3 Scalability and Sample Efficiency

To investigate scalability and sample efficiency, we trained policies on a varying number of demonstrations ([10, 25, 50, 100, 150, 200]). In Fig. 4, we report the training time, imitation error, and task success rate for each method as a function of the number of demonstrations when tested on the same 10,000 instances used to evaluate general efficacy.

Training time: We observe that KODex scales with the number of demonstrations significantly better than the baselines, as evidenced by its training time growing at considerably lower rates.

Imitation error and Success rate: We find that unstructured models (NN and LSTM) fail to demonstrate a consistent monotonic decrease (increase) in imitation error (task success rate) as the number of demonstrations increase. In stark contrast, structured methods (NDP, NGF, and KODex) are able to consistently drive down imitation error and improve task success rate. KODex almost consistently achieves the lowest imitation error and the highest task success rate with the fewest number of demonstrations, and is closely followed by NGF. These observations suggest that KODex tends to be comparably, if not more sample efficient than the baselines, thanks to the rich structure induced by the Koopman operator and the resulting effectiveness in capturing nonlinear dynamics. The only exception to this trend is the Object Relocation task, in which KODex requires 150 demonstrations to perform comparably to NGF. We speculate this is because the demonstrations for this task exhibit high variance as the hand base moves across a large space, and KODex requires more demonstrations to capture the underlying dynamics.

5.4 Additional Experiments

Additional experiments reported in the appendix suggest that KODex learns policies that i) have zero-shot out-of-distribution generalization comparable to SOTA IL methods (Appendices H), ii) are robust to changes in physical properties (Appendices I), and iii) are not overly sensitive to the choice of basis functions (Appendices J), and iv) are nearly-stable linear dynamical systems that generate safe and smooth robot trajectories (Appendices K).

6 Conclusions

We investigated the utility of Koopman operator theory in learning dexterous manipulation skills by encoding complex underlying nonlinear dynamics as linear dynamical systems in higher-dimensional spaces. Our investigations conclusively show that a Koopman-based framework can i) analytically learn dexterous manipulation skills, eliminating the sensitivity to initialization and reducing the need for user expertise, and ii) match or outperform SOTA imitation learning approaches on various dexterous manipulation tasks, while being an order of magnitude faster.

7 Limitations and Future Work

While our work offers promise for the utility of Koopman operators in dexterous manipulation, it reveals numerous avenues for further improvement. First, we did not deploy KODex on physical robots. Although our results on robustness to changes in physical properties show promise, we plan to deploy KODex on physical platforms to translate our findings to hardware. Second, KODex has limited out-of-distribution generalization, much like most existing imitation learning approaches. Future work can investigate if additional data collection [31] and learned lifting functions [32–34] alleviate this concern. Third, KODex relies on demonstrated action trajectories to learn the tracking controller and reduce human effort. It might be possible to instead use reinforcement learning [35], thereby enabling the ability to learn from state-only observations [36]. Further, Koopman operators can be used to learn system dynamics via self play to enable model-based reinforcement learning.

References

[1] A. M. Okamura, N. Smaby, and M. R. Cutkosky. An overview of dexterous manipulation. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 1, pages 255–262, 2000.

[2] OpenAI, M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba. Learning Dexterous In-Hand Manipulation. arXiv, 2018. doi:10.48550/ARXIV.1808.00177.

[3] A. Nagabandi, K. Konolige, S. Levine, and V. Kumar. Deep dynamics models for learning dexterous manipulation. In *Conference on Robot Learning*, pages 1101–1112. PMLR, 2020.

[4] M. Xie, A. Handa, S. Tyree, D. Fox, H. Ravichandar, N. D. Ratliff, and K. Van Wyk. Neural geometric fabrics: Efficiently learning high-dimensional policies from demonstration. In *Conference on Robot Learning*, pages 1355–1367. PMLR, 2023.

[5] T. Chen, J. Xu, and P. Agrawal. A System for General In-Hand Object Re-Orientation. In *5th Annual Conference on Robot Learning*, 2021.

[6] B. O. Koopman. Hamiltonian Systems and Transformation in Hilbert Space. *Proceedings of the National Academy of Sciences*, 17(5):315–318, 1931. doi:10.1073/pnas.17.5.315.

[7] 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*, 25(6):1307–1346, 2015.

[8] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine. Learning Complex Dexterous Manipulation with Deep Reinforcement Learning and Demonstrations. In *Proceedings of Robotics: Science and Systems (RSS)*, 2018.

[9] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard. Recent advances in robot learning from demonstration. *Annual Review of Control, Robotics, and Autonomous Systems*, 3(1):297–330, 2020.

[10] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal. Dynamical movement primitives: learning attractor models for motor behaviors. *Neural computation*, 25(2):328–373, 2013.

[11] S. M. Khansari-Zadeh and A. Billard. Learning stable nonlinear dynamical systems with gaussian mixture models. *IEEE Transactions on Robotics*, 27(5):943–957, 2011.

[12] K. Neumann and J. J. Steil. Learning robot motions with stable dynamical systems under diffeomorphic transformations. *Robotics and Autonomous Systems*, 70:1–15, 2015.

[13] H. Ravichandar, I. Salehi, and A. Dani. Learning Partially Contracting Dynamical Systems from Demonstrations. In *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78, pages 369–378, 13–15 Nov 2017.

[14] M. A. Rana, A. Li, H. Ravichandar, M. Mukadam, S. Chernova, D. Fox, B. Boots, and N. Ratliff. Learning reactive motion policies in multiple task spaces from human demonstrations. In *Conference on Robot Learning*, pages 1457–1468. PMLR, 2020.

[15] M. A. Rana, A. Li, D. Fox, B. Boots, F. Ramos, and N. Ratliff. Euclideanizing flows: Diffeomorphic reduction for learning stable dynamical systems. In *Learning for Dynamics and Control*, pages 630–639. PMLR, 2020.

[16] N. Figueroa and A. Billard. Locally active globally stable dynamical systems: Theory, learning, and experiments. *The International Journal of Robotics Research*, 41(3):312–347, 2022.

[17] S. Bahl, M. Mukadam, A. a. Gupta, and D. Pathak. Neural dynamic policies for end-to-end sensorimotor learning. *Advances in Neural Information Processing Systems*, 33:5058–5069, 2020.

[18] H. Qi, A. Kumar, R. Calandra, Y. Ma, and J. Malik. In-hand object rotation via rapid motor adaptation. *arXiv preprint arXiv:2210.04887*, 2022.

[19] H. Zhu, A. Gupta, A. Rajeswaran, S. Levine, and V. Kumar. Dexterous manipulation with deep reinforcement learning: Efficient, general, and low-cost. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 3651–3657. IEEE, 2019.

[20] T. Osa, J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel, J. Peters, et al. An algorithmic perspective on imitation learning. *Foundations and Trends® in Robotics*, 7(1-2):1–179, 2018.

[21] S. P. Arunachalam, I. Güzey, S. Chintala, and L. Pinto. Holo-dex: Teaching dexterity with immersive mixed reality. *arXiv preprint arXiv:2210.06463*, 2022.

[22] V. Kumar, A. Gupta, E. Todorov, and S. Levine. Learning dexterous manipulation policies from experience and imitation. *arXiv preprint arXiv:1611.05095*, 2016.

[23] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger. Deep reinforcement learning that matters. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[24] I. Abraham, G. De La Torre, and T. D. Murphey. Model-based control using Koopman operators. *arXiv preprint arXiv:1709.01568*, 2017.

[25] I. Abraham and T. D. Murphey. Active Learning of Dynamics for Data-Driven Control Using Koopman Operators. *IEEE Transactions on Robotics*, 35(5):1071–1083, 2019. doi:10.1109/TRO.2019.2923880.

[26] D. Bruder, B. Gillespie, C. D. Remy, and R. Vasudevan. Modeling and control of soft robots using the koopman operator and model predictive control. *arXiv preprint arXiv:1902.02827*, 2019.

[27] D. Bruder, X. Fu, R. B. Gillespie, C. D. Remy, and R. Vasudevan. Koopman-Based Control of a Soft Continuum Manipulator Under Variable Loading Conditions. *IEEE Robotics and Automation Letters*, 6(4):6852–6859, 2021. doi:10.1109/LRA.2021.3095268.

[28] J. P. Hanna and P. Stone. Grounded action transformation for robot learning in simulation. In *Thirty-first AAAI conference on artificial intelligence*, 2017.

[29] Todorov, Emanuel and Erez, Tom and Tassa, Yuval. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033, 2012.

[30] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. *arXiv preprint arXiv:2108.03298*, 2021.

[31] S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

[32] M. Weissenbacher, S. Sinha, A. Garg, and K. Yoshinobu. Koopman q-learning: Offline reinforcement learning via symmetries of dynamics. In *International Conference on Machine Learning*, pages 23645–23667. PMLR, 2022.

[33] B. Lusch, J. N. Kutz, and S. L. Brunton. Deep learning for universal linear embeddings of nonlinear dynamics. *Nature communications*, 9(1):1–10, 2018.

[34] 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. URL https://openreview.net/forum?id=H1ldzA4tPr.

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

[36] F. Torabi, G. Warnell, and P. Stone. Recent advances in imitation learning from observation. *arXiv preprint arXiv:1905.13566*, 2019.

[37] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber. Lstm: A search space odyssey. *IEEE transactions on neural networks and learning systems*, 28(10): 2222–2232, 2016.

[38] K. Van Wyk, M. Xie, A. Li, M. A. Rana, B. Babich, B. Peele, Q. Wan, I. Akinola, B. Sundaralingam, D. Fox, et al. Geometric fabrics: Generalizing classical mechanics to capture the physics of behavior. *IEEE Robotics and Automation Letters*, 2022.

[39] C.-A. Cheng, M. Mukadam, J. Issac, S. Birchfield, D. Fox, B. Boots, and N. Ratliff. Rmpflow: A geometric framework for generation of multitask motion policies. *IEEE Transactions on Automation Science and Engineering*, 18(3):968–987, 2021.