arXiv:2004.04650v1 [cs.RO] 7 Apr 2020

State-Only Imitation Learning for Dexterous Manipulation

Ilija Radosavovic [1] **Xiaolong Wang** [1 2] **Lerrel Pinto** [1 3] **Jitendra Malik** [1]

Abstract

Dexterous manipulation has been a long-standing challenge in robotics. Recently, modern model-free RL has demonstrated impressive results on a number of problems. However, complex domains like dexterous manipulation remain a challenge for RL due to the poor sample complexity. To address this, current approaches employ expert demonstrations in the form of state-action pairs, which are difficult to obtain for real-world settings such as learning from videos. In this work, we move toward a more realistic setting and explore state-only imitation learning. To tackle this setting, we train an inverse dynamics model and use it to predict actions for state-only demonstrations. The inverse dynamics model and the policy are trained jointly. Our method performs on par with state-action approaches and considerably outperforms RL alone. By not relying on expert actions, we are able to learn from demonstrations with different dynamics, morphologies, and objects.

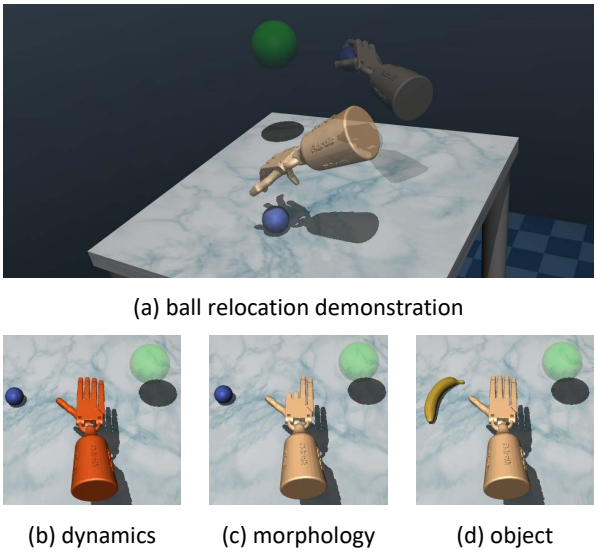

Figure 1. We present *SOIL*, a simple and effective method for *state-only* imitation learning of dexterous manipulation. SOIL enables learning from demonstrations coming from different but related settings. For example, given demonstrations of relocating a ball to a target position (a), we can learn to perform the same task for different dynamics (b), morphologies (c), and objects (d).

1. Introduction

Dexterous manipulation with multi-fingered hands has the potential to equip robots with human-like dexterity and enable them to *generalize* to different tasks, goals, and tools. However, it comes at a significant cost: complex, high-dimensional action spaces. Traditionally, this has been a challenge for standard model-based trajectory optimization approaches. Recently, there has been a renewed interest in using reinforcement learning (RL) techniques for dexterous manipulation. For example, we have witnessed promising results in using large-scale model-free RL for complex in-hand manipulation tasks (OpenAI et al., 2018; 2019).

Despite the favorable properties, model-free RL leaves room for improvements in data efficiency and generalization. In particular, training a model *from scratch* requires a large amount of training data *per task*. This problem of high sample complexity is even more severe in the case of complex tasks like dexterous manipulation. While it may still be possible to train a model-free policy from scratch to perform a task, the learned policy does not typically generalize to different settings. For example, a policy trained to relocate a ball (Figure 1, top) does not generalize well to different dynamics, morphologies, and objects (Figure 1, bottom).

To enable sample-efficient learning of policies that generalize across different settings, one promising avenue lies in imitation learning (Bakker & Kuniyoshi, 1996; Schaal, 1999). Learning by imitation is a well-known and powerful mechanism in the cognitive development of children (Tomasello et al., 1993; Meltzoff, 1995). Similar to learning in children, our robots could acquire motor skills by learning from demonstrations. This general paradigm also has the potential to enable learning from internet-scale videos and thus deliver robots that generalize to diverse environments.

Overall, there has already been a considerable progress toward this goal. In particular, a large body of work has focused on the setting with demonstrations in the form of state-action pairs. For example, Rajeswaran et al. (2018) collect

[1]UC Berkeley [2]UC San Diego [3]New York University. Correspondence to: Ilija Radosavovic <ilija@berkeley.edu>.

Videos available at this project page.

demonstrations for dexterous manipulation using a virtual reality headset and a motion capture glove. They further show that augmenting the RL objective with an imitation term can lead to large improvements in sample complexity compared to RL alone. However, collecting data in such setups is challenging and limited to only a subset of tasks. Moreover, requiring actions makes it hard to leverage the readily available internet-scale videos as demonstrations.

Relying on state-action demonstrations raises another challenge: generalization. Learning a policy that is constrained to output the same action as a demonstrated state-action pair, limits the flexibility of the policy and hinders the ability to generalize across settings (Figure 1). Is relying on demonstrator actions necessary? Can we leverage useful information from state-only demonstrations?

In this work, we move toward the more general setting and explore imitation learning from demonstrations containing only states without the actions. The state space could be entirely or partially shared between the expert and the imitator. Using state-only demonstrations has two major benefits. First, it reduces the burden of data collection. Second, by learning from state trajectories—without mimicking the exact actions—we can leverage demonstrations with different dynamics, morphologies, and objects (Figure 1).

To tackle this setting, we propose a simple and effective method that we call *State-Only Imitation Learning (SOIL)*. We leverage state-only demonstrations by training an inverse dynamics model to predict actions between consecutive states. Given the predicted actions, we incorporate them into the RL objective using an auxiliary behavior cloning-like term (Rajeswaran et al., 2018). The policy and the inverse dynamics model are trained jointly in an alternating fashion. Using an inverse dynamics model is appealing: it is adaptable which enables it to utilize demonstrations across settings (Figure 1) and it is reward-independent which helps it to generalize to new settings (*e.g.*, different tasks).

We validate the proposed method in simulation using the suite of four different dexterous manipulation tasks (Rajeswaran et al., 2018). We perform controlled comparisons to state-action approaches that serve as an upper-bound for our method. Surprisingly, we find that SOIL performs on par with state-action approaches. Moreover, our method achieves considerable improvements compared to RL without demonstrations. Going beyond, we show that SOIL can generalize to environments with different dynamics, morphologies, and objects (Figure 1) while using demonstrator actions degrades the performance. Overall, our results suggest that relying on state-action demonstrations may not be necessary and can even hurt generalization performance.

To facilitate future research we will make the code, models, and videos publicly available at this project page.

2. Related Work

Dexterous manipulation. Manipulation with dexterous hands is one of the most challenging control tasks in robotics. There is a wide range of prior work on dexterous manipulation with optimization and planning (Dogar & Srinivasa, 2010; Bai & Liu, 2014; Andrews & Kry, 2013). However, these optimization based approaches have a hard time solving complex dexterous manipulation tasks. Alternatively, we have recently witnessed promising results in dexterous manipulation tasks achieved by deep reinforcement learning in simulation (OpenAI et al., 2018; 2019). These policies can be transferred to the real robot hand with domain randomization (Sadeghi & Levine, 2017; Tobin et al., 2017). However, using pure RL requires large-scale training and is very difficult to generalize to new environments.

Imitation learning with behavior cloning. Instead of relying on pure RL, imitation learning has shown a significant advantage in improving the learning efficiency and solving more complex manipulation tasks. The general paradigm of imitation learning involves using expert demonstrations of successful behavior to learn policies that imitate the expert. A common approach to imitation learning is behavior cloning (BC) (Pomerleau, 1989; Bain & Sammut, 1995; Bojarski et al., 2016; Torabi et al., 2018a). Effectively, BC amounts to learning to mimic the expert demonstrations by performing supervised learning. While BC can work well in certain scenarios, it can lead to failure at test time due to the distribution shift problem (Ross et al., 2011).

RL with demonstrations. To inherit benefits from both reinforcement learning and imitation learning, researchers have looked into combining reinforcement learning with imitation learning from demonstrations (Peters & Schaal, 2008; Duan et al., 2016; Večerík et al., 2017; Peng et al., 2018a). An effective way is to first initialize the policy with behavior cloning and then finetune it with RL (Peters & Schaal, 2008). This approach is compatible with both on-policy and off-policy methods. An alternative approach is to use demonstrations with off-policy methods by adding them to the replay buffer (Večerík et al., 2017). While off-policy methods can be more sample efficient, they are generally less stable and scale worse to high-dimensional spaces (Rajeswaran et al., 2018; Duan et al., 2016). Instead of using off-policy methods, inspired by (Rajeswaran et al., 2018), we adopt the general strategy to incorporate demonstrations with on-policy methods via an auxiliary term. However, our expert demonstrations contain only states without actions. Thus, previous approaches cannot be adopted directly.

Inverse reinforcement learning. If we do not have access to actions, one option would be to create a density model akin to inverse reinforcement learning (IRL) (Russell, 1998; Ng et al., 2000; Abbeel & Ng, 2004; Fu et al., 2017; Aytar et al., 2018; Ho & Ermon, 2016; Torabi et al., 2018b;

Sun et al., 2019; Liu et al., 2020). For example, (Ho & Ermon, 2016) propose to learn a density model to estimate whether the trajectories are from the demonstrations or the policy, and use the output as the reward, in an adversarial training framework. However, learning density models in high dimensional state spaces is often inefficient since the demonstrations only cover a small portion of the full state space. For complex tasks like dexterous manipulation, using adversarial objectives often collapse into sub-optimal modes and makes learning unstable.

Learning dynamics models. One way to improve the sample efficiency of RL is to learn the dynamics model. For example, Nagabandi et al. (2019) showed that model-based RL could provide an efficient way of learning dexterous manipulation tasks. Besides training forward models, researchers have also explored learning the inverse dynamics model (Pinto et al., 2016; Agrawal et al., 2016; Christiano et al., 2016; Nair et al., 2017; Edwards et al., 2018; Torabi et al., 2018a; Kumar et al., 2019). The inverse dynamics model is typically pretrained and then used to complete expert demonstrations. In contrast, we propose to learn the inverse dynamics model *jointly* with the RL policy.

Following demonstrations. Another line of work in imitation learning is to train policies to follow the expert demonstrations (Liu et al., 2017; Peng et al., 2018b; Pathak et al., 2018; Sharma et al., 2018; Sermanet et al., 2018). For example, Peng et al. (2018b) show successful imitation of complex human motion from videos. However, the policy is only trained for repeating one particular trajectory. Instead of training on a single video, Sermanet et al. (2018) created a large-scale dataset and proposed a sequence to sequence model for repeating different human actions. In our work, our policy is goal-conditioned instead of conditioning on the expert trajectories during testing. Instead of being tied to a specific environment, our policy can be generalized to the environment with different dynamics and physics.

3. Preliminaries

We begin by briefly discussing the relevant background on reinforcement learning methods we build upon in this work.

3.1. Reinforcement learning

We consider the standard model-free Reinforcement Learning (RL) setup and model the control problem using a Markov Decision Process (MDP), defined by the tuple $(S, A, T, r, p_0, \gamma)$, where $S \in \mathbb{R}^n$ represents the states, $A \in \mathbb{R}^m$ represents the actions, $T : S \times A \to S$ is the transition dynamics, $r : S \times A \to \mathbb{R}$ is the reward function, p_0 is the initial state distribution, and $\gamma \in (0, 1)$ is the discount factor. We wish to find a parametric policy π_θ that maximizes the expected sum of discounted rewards.

We focus on the policy gradient methods that directly optimize the aforementioned objective using gradient ascent. In its simplest form, the vanilla policy gradient (Williams, 1992) is given by:

$$g = \sum_{(s,a)\in\pi} \nabla_\theta \log \pi_\theta(a|s) A^\pi(s, a), \tag{1}$$

where $A^\pi(s, a)$ is the standard advantage function:

$$A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s). \tag{2}$$

Following Rajeswaran et al. (2018), we build upon the natural policy gradient (NPG) (Kakade, 2002). In particular, we first compute the Fisher Information Matrix:

$$F_\theta = \sum_{(s,a)\in\pi} \nabla_\theta \log \pi_\theta(a|s) \nabla_\theta \log \pi_\theta(a|s)^\top. \tag{3}$$

Then, pre-condition the vanilla policy gradient prior to making the gradient ascent update:

$$\theta_{k+1} = \theta_k + \left(\frac{\delta}{g^\top F_{\theta_k}^{-1} g} \right)^{\frac{1}{2}} F_{\theta_k}^{-1} g. \tag{4}$$

Although NPG stabilizes RL training, it is often still difficult to obtain good performance due to exploration and optimization challenges. This is particularly challenging in complex contact-rich domains like dexterous manipulation.

3.2. Imitation learning

One way to overcome these challenges of pure RL is to perform imitation learning. In the standard imitation learning setup, we assume access to demonstrations in the form of state-action pairs. The simplest way to learn a policy using demonstrations of this form is to perform behavior cloning (Pomerleau, 1989; Bojarski et al., 2016). However, BC is often not very effective for complex tasks due to the distribution shift problem (Ross et al., 2011).

To effectively utilize the demonstrations, researchers have proposed to incorporate demonstration data as an auxiliary loss together with RL. In particular, Rajeswaran et al. (2018) propose augmenting the Eq. 1 with an additional term:

$$g_{aux} = \sum_{(s,a)\in D} \nabla_\theta \log \pi_\theta(a|s) w(s, a), \tag{5}$$

where $w(s, a)$ is a weighting function used to weight the relative contribution of demonstrations and the standard RL objective. In practice, the weight is annealed to zero over the course of training. This enables the policy to keep improving over time, while not forgetting the demonstrations. The resulting method is called the Demo Augmented Policy Gradient (DAPG) (Rajeswaran et al., 2018).

4. State-Only Imitation Learning

Here we describe the state-only imitation learning setting we explore and present our method to tackle this setting.

4.1. Problem Setup

In this work, we focus on a more realistic setting for imitation learning, where the demonstrations only contain states and no actions. Leveraging state-only demonstration allows for practical imitation learning since state-estimation from expert demonstrators is more feasible. One example is learning from videos of human demonstrations (Handa et al., 2019), where state estimates are readily available while human action estimates are not. Hence, this relaxed problem setting brings us closer to real-world settings and potentially allows for utilizing third-person videos.

As a first step toward this goal, we assume our model has direct access to the states represented by the joint angles of the hand, forces applied on the joints, and the speed of the joints. We also provide the locations of the objects. This allows us to investigate the effects of state-only imitation without conflating with state estimation. Our policy is trained in the same state space as the provided demonstrations.

We note that even if the state space is well aligned, our problem is still challenging without access to the actions between every two states. Recall that both behavior cloning and DAPG mentioned in the previous section require the "ground-truth" actions. Thus we cannot directly apply standard imitation learning objectives for training our policy. Can we predict the actions from the provided state-only demonstrations? If we can predict these actions well, we can stand on the shoulders of powerful behavior cloning techniques and achieve good performance. Our learning framework revolves around this idea of action prediction.

Concretely, we train an inverse dynamics model, which takes two consecutive demonstration states as inputs and estimates the action as the output. A naive way to collect the training data for supervising this inverse dynamics model is to sample random trajectories. However, the action space is too large to explore randomly with a robot hand. To acquire informative training data for the inverse model, we need a reasonable policy to explore the action space. On the other hand, to train a reasonable manipulation policy, we need to predict the correct actions with the inverse model.

To overcome this problem, we propose to train the inverse dynamics model and the policy network *jointly* in an iterative manner. Thus, the policy network can help the inverse model better explore the action space, and a better inverse model, in turn, provides better training examples for the policy network. In the following subsections, we introduce the inverse dynamics model and describe our learning framework for joint iterative training.

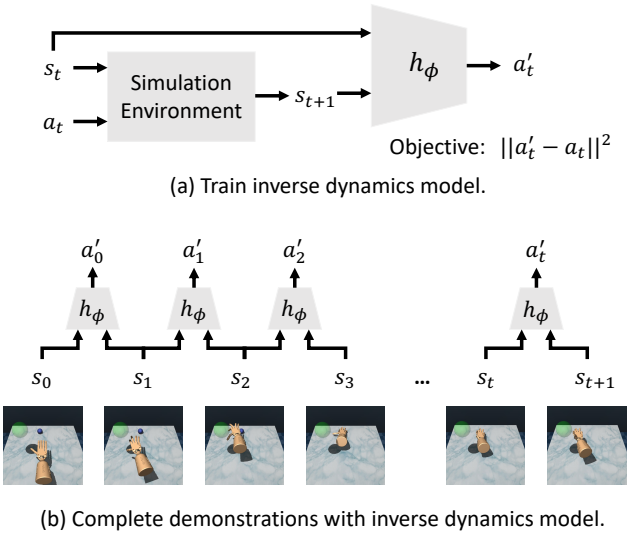

Figure 2. **Inverse dynamics model.** (a) To train the inverse dynamics model, we use the policy to generate the trajectories from the environment and perform supervised learning. (b) Given state-only demonstrations, we use the inverse dynamics model to predict the actions between consecutive states. We then augment the RL objective with an auxiliary term on the predicted state-action pairs.

4.2. Inverse Dynamics Model

We propose to train an inverse dynamics model which takes two consecutive states as inputs and estimates the action between them. As shown in Figure 2a, we assume that we have the state and action triplet of the form (s_t, a_t, s_{t+1}). Our inverse dynamics model is a small MLP network h_ϕ which is parameterized by ϕ. We can then use h_ϕ to estimate the action a'_t given two states as,

$$a'_t = h_\phi(s_t, s_{t+1}). \tag{6}$$

Since we are only applying the actions on the hand, our inverse dynamics model only takes the hand joint states as inputs without the object and environment states. Thus, even the inverse model is trained with one task, it has the potential to generalize to new settings (*e.g.*, different goals, tools, tasks, *etc.*). We train the inverse model in a supervised learning manner using the L2 loss function. Given a policy network, we sample trajectories from the policy to generate the training data for training the inverse model. In particular, we collect the state and action triplets and add them to a reply buffer R. We then sample uniformly from R and obtain a batch of training examples B. The training objective is then defined as,

$$L_{mse} = \sum_{(s_t, a_t, s_{t+1}) \in B} ||a_t - h_\phi(s_t, s_{t+1})||^2. \tag{7}$$

The inverse dynamics model can easily benefit from the advancements in supervised learning (*e.g.*, better network architectures, optimizers, *etc.*).

Algorithm 1 State-Only Imitation Learning (SOIL)

Input: Inverse model h, Policy π, Replay buffer R, State-Only Demonstration D.
Initialize: Learnable parameters ϕ for h_ϕ, θ for π_θ.
for i=1,2,..N_{iter} **do**
  # Collect trajectories
  $\tau_i \equiv \{s_t, a_t, s_{t+1}, r_{t+1}\}_i \sim \pi_\theta$
  # Add data to the buffer
  $R \leftarrow R \cup \tau_i$
  for j=1,2,..N_{inv} **do**
    # Sample a batch of state-action triplets
    $B_j \equiv \{s_t, a_t, s_{t+1}\}_j \sim R$
    # Update the inverse dynamics model
    $\phi \leftarrow \text{invOpt}(B_j; \phi)$, according to Eq. 7
  end for
  # Predict actions using the inverse model
  $D' \leftarrow$ complete D with h_ϕ, according to Eq. 6.
  # Perform SOIL policy gradient update
  $\theta \leftarrow \text{policyOpt}(\tau_i, D'; \theta)$, according to Eq. 9
end for

4.3. Joint Training Procedure

We propose to jointly train the inverse dynamics model and the policy network. Since the actions are not directly provided from the demonstrations but estimated by the inverse dynamics model, we adjust the DAPG objective from Eq. 5. As shown in Figure 2b, given the states from the demonstration, we use the inverse model to predict the actions a' between the consecutive states (Eq. 6), and generate the new demonstration set with state and action pairs D'. We adjust the gradient in Eq. 5 as,

$$g_s = \lambda_0 \lambda_1^k \sum_{(s,a') \in D'} \nabla_\theta \log \pi_\theta(a'|s), \tag{8}$$

where λ_0 and λ_1 are constants and k increases with the number of training iterations. In this way, we encourage the augmented gradient to be smaller and smaller as the trained policy is becoming better and better. The overall gradient for training the policy combines the policy gradient (Eq. 1) with the auxiliary demonstration objective (Eq. 8) as,

$$g_{soil} = \sum_{(s,a) \in \pi} \nabla_\theta \log \pi_\theta(a|s) A^\pi(s,a) + \lambda_0 \lambda_1^k \sum_{(s,a') \in D'} \nabla_\theta \log \pi_\theta(a'|s). \tag{9}$$

During training, we jointly optimize the objective for the inverse dynamics model (Eq. 7) and the SOIL policy gradient for the policy network (Eq. 9). The details of the joint optimization process are shown in Algorithm 1.

Figure 3. **Tasks.** In addition to the object relocation task, shown in Figure 1, we study three additional dexterous manipulation tasks. *Left:* In-hand manipulation, the task is to position the blue pen to match the orientation of the green pen. *Middle:* Door opening, the task is to undo the latch and open the door. *Right:* Tool use, the task is to pick up the hammer and drive the nail into the board.

5. Experiments

We now evaluate our method SOIL in simulation, perform extensive studies of different design choices, and test generalization to different dynamics, morphologies, and objects.

5.1. Experimental Setup

We follow the setup from (Rajeswaran et al., 2018) and describe the core components next (see also Appendix B).

Simulator. We use the MuJoCo physics simulator that supports stable contact dynamics (Todorov et al., 2012; Erez et al., 2015). We adopt the simulated model of the dexterous hand provided in the ADROIT platform (Kumar et al., 2013), designed for exploring dexterous manipulation tasks. The hand model has five fingers and 24 degrees of freedom, which involve position control and joint angle sensors.

Task. The suite proposed in (Rajeswaran et al., 2018) consists of four tasks: object relocation, in-hand manipulation, door opening, and tool use (see Figure 3). We evaluate our approach on all four tasks and perform the majority of the ablation and generalization studies using the object relation task. We choose the object relocation task due to its generality and difficulty (discussed in the next subsection).

Demonstrations. We use the demonstrations collected and provided by (Rajeswaran et al., 2018). The demonstrations were recorded using a virtual reality headset and a motion capture glove (Kumar & Todorov, 2015). There are 25 demonstrations per task available. Each demonstration is a sequence of state-action pairs. For our experiments with state-only demonstrations, we ignore the provided actions.

State space. We adopt the state space from (Rajeswaran et al., 2018). The dimensionality varies across tasks. For example, the hammer task has a higher dimensional state than in-hand manipulation due to the arm movement. In all cases, however, the state is high-dimensional (~40 dim).

Reward. Due to the complexity of the tasks, Rajeswaran et al. (2018) design a dense reward function for each of the tasks. In all of our experiments, we use their dense reward functions. We also experiment with sparse reward variants and observe that the general trends are consistent.

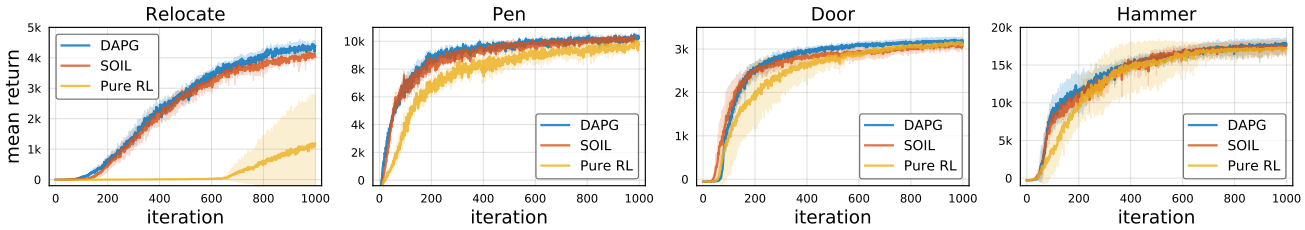

Figure 4. **State-action comparisons.** We evaluate our approach in the standard imitation learning setup. In particular, we compare our state-only method SOIL to DAPG that uses demonstrations in the form of state-action pairs and pure RL that does not use demonstrations. Pure RL and DAPG can be considered as a lower-bound and an upper-bound for our method, respectively. We perform comparisons on four different dexterous manipulation tasks. In all cases, the results are consistent: *SOIL performs comparably to DAPG while being considerably better than pure RL.* Note that the gap compared to pure RL is the largest for the most challenging object relocation task.

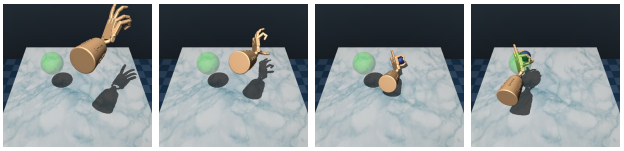

Figure 5. **Qualitative, pure RL.** We show an episode of the policy learnt using pure RL. Although the policy can achieve high return in some cases, it does not exhibit realistic behavior. For example, it completes the object relocation task by exploiting an imperfection in the simulator which is unlikely to transfer to the real-world.

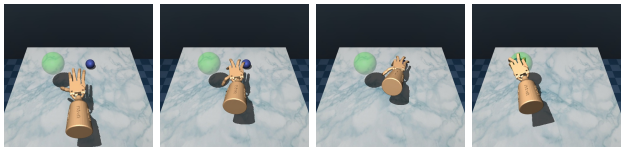

Figure 6. **Qualitative, DAPG.** We show an example episode of the DAPG policy. We observe that the policy exhibits realistic human-like behavior (*e.g.*, reaching, grasping, *etc.*). This suggests that using demonstrations can be a powerful mechanism for learning policies that are more likely to transfer to the real-world.

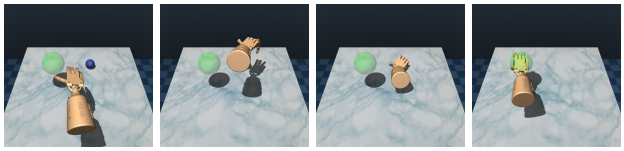

Figure 7. **Qualitative, SOIL.** We observe that the SOIL policy is reasonably close to the DAPG one while being considerably more realistic than the pure RL policy. This suggests that relying on expert actions may not be necessary for learning realistic policies.

5.2. State-Action Comparisons

We begin by evaluating our state-only method in the standard imitation learning setup. We compare with two methods: DAPG that uses demonstrations in the form of state-action pairs and pure RL that does not use demonstrations.

Pure RL comparisons. We consider pure RL without demonstrations as a lower-bound for our approach. Intuitively, using demonstrations, even without actions, should not hurt the performance and should work at least as well as pure RL by ignoring the demonstrations. In Figure 4, we compare our state-only approach to pure RL on four different tasks. In all cases, we observe that our method SOIL outperforms pure RL. We highlight two cases next.

First, the gap is the largest for the hardest task of object relocation (left). This is a promising signal for the applicability of our method to more challenging settings. Second, pure RL works surprisingly well on the hammer task (right). Upon further inspection, we find that the demonstrations may bias the imitation-based approaches to human-like jerky movements which is likely suboptimal in terms of the reward but may be more likely to generalize to the real-world. This highlights the importance of not relying on the overall returns alone when evaluating methods in simulation.

DAPG comparisons. Similarly, we consider DAPG that uses state-action demonstrations as an upper-bound for our approach. Namely, our inverse dynamics model trained from scratch is unlikely to predict actions superior to the ground-truth actions. In Figure 4, we compare our method SOIL to DAPG on four different tasks. In all cases, the findings are again consistent: SOIL comes surprisingly close to DAPG that uses ground-truth actions. This is very encouraging and suggests that having access to actions may not be critical for developing well-performing imitation learning approaches.

Qualitative comparisons. As discussed earlier, the returns alone do not paint a complete picture. Thus, to get a better sense of the learnt policies, we show representative example episodes. In Figure 5, we observe that the pure RL policy does not exhibit realistic behavior. In contrast, the DAPG policy, shown in Figure 6, is more realistic. Lastly, in Figure 7, we see that our state-only learnt policy is comparable to DAPG and considerably more realistic than pure RL.

Summary. We evaluate our method in a controlled setting where state-action demonstrations are available. We find that our state-only method reaches the performance of standard state-action approaches while considerably outperforming RL without demonstrations. Overall, our results suggest that relying on expert actions may not be necessary.

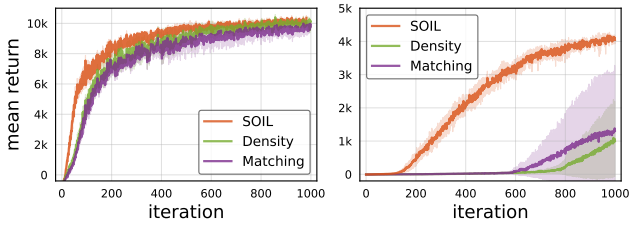

Figure 8. **State-only comparisons.** We compare our method SOIL to two state-only baselines: state matching and density estimation. SOIL outperforms the baselines on both pen in-hand manipulation task (left) and the more challenging object relocation task (right).

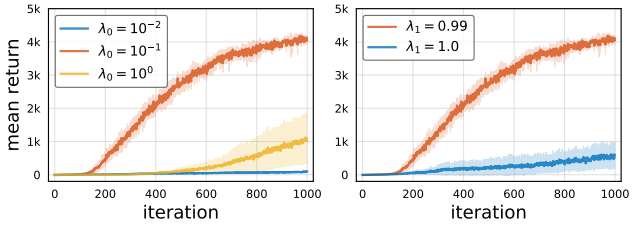

Figure 9. **Auxiliary objective weight.** We study the impact of the auxiliary loss weights from Eq. 8. *Left:* The scale of the auxiliary term impacts the performance significantly. *Right:* Annealing the demonstrations term to zero ($\lambda_1 = 0.99$) is beneficial. *Relocate.*

5.3. State-Only Comparisons

We now compare our method to state-only baselines inspired by methods from the literature (see also Appendix A).

State matching. One way to incorporate state-only demonstrations is to perform state matching between sampled trajectories and demonstrations (Abbeel et al., 2010; Peng et al., 2018a). We experiment with a number of variants based on standard distance functions including DTW (Needleman & Wunsch, 1970; Sakoe & Chiba, 1978), nearest neighbor, and Chamfer distance. We find the approach based on Chamfer distance to work the best (not shown). Consequently, we adopt it as our state matching baseline in the comparisons.

Density estimation. While state matching approaches can work well in certain settings, they rely on domain knowledge and careful tuning of the distance function, which limits their applicability in practice. Inspired by (Ho & Ermon, 2016), we explore a data-driven alternative. In particular, we train a density model to differentiate between states coming from the current policy and the demonstrations distribution.

Comparisons. In Figure 8, we compare our method SOIL to the two aforementioned baselines. First, we observe that both baselines work reasonably well on the easier pen in-hand manipulation task (left). Nevertheless, there is still a gap compared to our approach. Second, we see that in case of the more challenging object relocation task our method outperforms the baselines considerably. This is a promising signal for applicability of SOIL to harder settings. We note that we tuned both baselines analogously to our method.

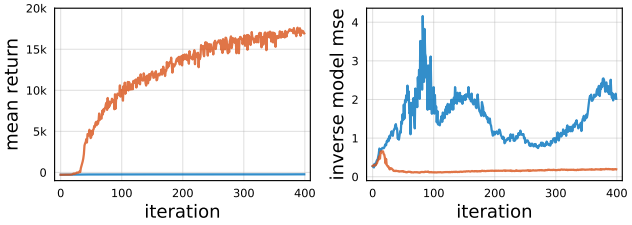

Figure 10. **Inverse dynamics model.** We study the impact of the quality of the inverse dynamics model (right) on the policy (left). We observe a strong correlation between the two: a good inverse model (low error) results in a good policy (high reward). *Hammer.*

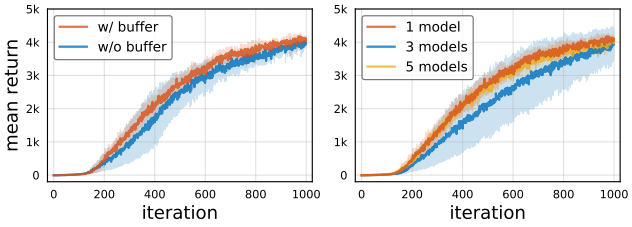

Figure 11. **Model-based RL enhancements.** *Left:* Using a replay buffer to aggregate the training data for the inverse dynamics model reduces the variance. *Right:* Using an ensemble of inverse dynamics models does not improve the performance. *Relocate.*

5.4. Ablation Studies

Next, we perform ablation studies to get a better understanding of different design choices involved in our method.

Auxiliary objective weight. In Figure 9, we study the impact of the auxiliary objective weights from Eq. 8. First, we see that the scale of the auxiliary term plays an important role (left). In particular, using too small or too large of a weight leads to suboptimal performance. Thus, the scale needs to be tuned carefully. Second, we observe that annealing the auxiliary term to zero results in considerably better performance (right). This suggests that demonstrations are helpful during the initial exploration but start to hurt as the policy gets better and starts to exceed demonstrations.

Inverse dynamics model. In Figure 10, we study the impact of the quality of the inverse dynamics model (right) on the policy (left). A good inverse dynamics model (low error) results in a good policy (high return), and vice versa. This suggests that the quality of the inverse dynamics model plays a key role in the effectiveness of our method.

Model-based RL enhancements. Motivated by the advancements from the model-based RL literature (Chua et al., 2018; Nagabandi et al., 2018), we explore inverse dynamics model training enhancements in Figure 11. First, we find that using a replay buffer reduces the variance (left). Consequently, we adopt the replay buffer as part of our method. Second, we see no benefit from using an ensemble of inverse dynamics models. Thus, we do not use an ensemble of inverse dynamics models in the rest of the experiments.

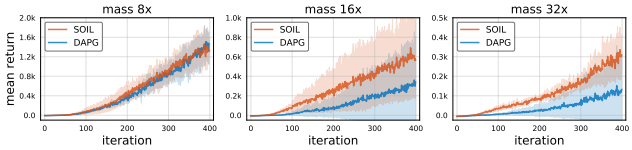

Figure 12. **Generalization, dynamics.** We study generalization in settings with different dynamics. In particular, we consider hands with larger mass. We observe that our state-only method generalizes better than DAPG that utilizes state-action demonstrations. This suggests that using state-only imitations may be preferable for leveraging demonstrations with different dynamics.

5.5. Generalization

Finally, we demonstrate the favorable generalization properties of our state-only approach in different settings.

Experimental setup. Our setup is as follows. We consider training to perform the object relocation task using *original* demonstrations under *different* conditions. In particular, we study generalizations to different dynamics, morphologies, and objects. In these settings, using original actions may not be helpful and may even hurt. The motivation for these experiments is to test if our state-only method can generalize better to such settings than state-action approaches.

Different dynamics. First, we consider generalization to different dynamics. Specifically, we perform training with hands of increased mass. In this setting the model needs to learn a different action distribution. For example, in order to move a heavier hand the model needs to apply more force and in turn account for a larger momentum. In Figure 12, we show the results for three variants of increasing difficulty (mass increased by $8\times$, $16\times$, and $32\times$). We observe a clear trend: our state-only method SOIL outperforms state-action based DAPG for different dynamics. As the task becomes more difficult, our gains become larger.

Different morphologies. Next, we evaluate generalization to different morphologies. In particular, we consider training to perform the ball relocation task using hands with a subset of fingers. In this setting, the model must learn a considerably different strategy compared to using the full hand. In Figure 13, we show the results for three hand configurations of increasing difficulty with one to three fingers removed. In all cases, we observe that our state-only method performs considerably better than DAPG.

Different objects. Lastly, we study generalization across settings with different objects. For example, we consider the task of learning to relocate a banana from demonstrations of relocating a ball. In this setting, the grasping strategies suitable for different objects can vary considerably. In Figure 14, we report the results for three object variants of increasing difficulty. In all cases, we observe that our state-only method SOIL outperforms DAPG that uses actions. As before, our gains are larger for harder settings. Surprisingly,

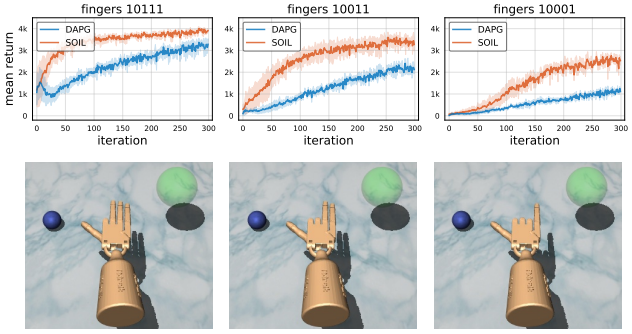

Figure 13. **Generalization, morphologies.** We explore generalization to different morphologies. Namely, we consider learning to perform the task using hand variants with a subset of fingers removed. Each subplot corresponds to a different morphology. The zeros and ones in the subplot titles indicate inactive and active fingers, respectively. In all cases, we observe that our state-only method generalizes better than state-action based DAPG.

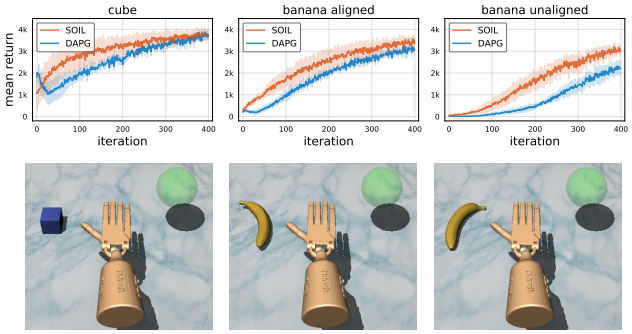

Figure 14. **Generalization, objects.** We study generalization to different objects. For example, we use the demonstrations of relocating a ball to learn to relocate a banana. Each subplot corresponds to a different object variant, ordered from left to right in increasing difficulty. Aligned and unaligned indicate if the banana is conveniently positioned for a grasp. In all cases, we observe that our method SOIL performs considerably better than DAPG that uses actions. Note that the gap is larger for harder settings.

even in the relatively easier case of the cube there is still a clear gap. This suggests that relying on demonstrator actions can hurt the performance even for objects that require fairly similar grasping strategies like a ball and a cube.

6. Conclusion

In this work, we explore state-only imitation learning. To tackle this setting, we propose a simple and effective method that we call SOIL. Our method achieves results on par with standard state-action approaches and considerably outperforms RL without demonstrations. Going beyond, we show that SOIL can effectively leverage demonstrations with different dynamics, morphologies, and objects. We hope that our work serves as a step toward the more general setting of imitation learning from real-world third-person videos.

Acknowledgements

This research was supported in part by DARPA Machines with Common Sense program.

Appendix A: State-Only Baselines

In §5.3 of the main text, we compare our state-only method to state-only baselines inspired by methods from the literature. Namely, state matching and density estimation. Here we provide more information about the state-only baselines.

Auxiliary objective. We begin by describing the auxiliary objective formulation we use for these methods. In the case of SOIL, we use an inverse dynamics model to predict actions for state-only demonstrations. Thus, we can apply the auxiliary objective proposed in (Rajeswaran et al., 2018) on the predicted actions. However, if we do not have access to any actions this auxiliary objective is not applicable.

To overcome this, we use an auxiliary objective that is based on the idea of state similarity. In particular, we augment the vanilla policy gradient with a similarity score that is added to the advantage function:

$$g = \sum_{(s,a)\in\pi} \nabla_\theta \log \pi_\theta(a|s)[A^\pi(s,a) + \lambda_0 \lambda_1^k \, \text{sim}(s, D)]$$

The similarity score measures the similarity of the sampled states to the demonstration states. Intuitively, we want to make the sampled trajectories that are similar to demonstrations more likely. Following DAPG, we allow scaling and annealing the contribution of demonstrations over the course of training. Our two state-only baselines correspond to two different strategies for computing the similarity score.

State matching. In the case of state matching, we compute the similarity score based on standard distance functions, including DTW, nearest neighbor, and Chamfer distance.

Density estimation. An alternative strategy to define the similarity function is to employ a data-driven approach. In particular, learn a density model to differentiate between trajectories sampled from the policy and demonstrations.

To train the density model, we employ a joint optimization procedure similar to SOIL. In particular, we train the policy and the density model jointly in an alternating fashion. We treat the demonstration trajectories as positives and use the trajectories sampled from the current policy as negatives for training the density model. Thus, a better policy generates better data for training the density model, and a better density model helps the policy explore the space better.

While we find SOIL to work considerably better, we believe this is still an interesting method for state-only imitation learning. We also note that this method is closely related to GAIL (Ho & Ermon, 2016) and contrastive learning.

Appendix B: Implementation Details

In this section we report additional implementation details. We closely follow the training settings from (Rajeswaran et al., 2018) unless specified in the following.

Policy network. Our policy is a diagonal Gaussian MLP with standalone log standard deviation parameters. In particular, the policy network is a 2-layer MLP with 32 hidden units and tanh activation function. For fair comparisons, we use the same policy network structure for all of the methods.

Policy initialization. In our state-action comparisons in §5.2, we initialize the policy from scratch for all methods. We note that in the case of DAPG one can initialize the policy using BC pretraining on state-action demonstrations, which leads to considerably better performance. However, as our goal in these experiments is not necessarily to obtain the best possible performance on these tasks, but rather to study the potential of state-only methods under controlled settings we do not use BC pretraining in our experiments.

Training schedule. In our comparisons to state-action and state-only methods in §5.2 and §5.3, respectively, we train all methods for 1000 iterations. We use a batch size of 200 trajectories and the horizon length of 200 steps.

Auxiliary objective weights. We set auxiliary objective weights for SOIL following §5.4. For fair comparisons, we perform the equivalent optimization of auxiliary objective weights for both state-action and state-only methods. We note that in the case of DAPG this leads to considerably better performance than using the original settings, which are likely optimized for DAPG with BC initialization. We adopt the improved settings in all of the comparisons.

Inverse dynamics model. Our inverse dynamics model is a 2-layer MLP with 64 hidden units and tanh nonlinearities. We optimize the MSE loss using Adam (Kingma & Ba, 2015) with a learning rate of 1e-3 and a batch size of 32. We train the inverse model for 500 steps per iteration of the outer loop. We tune the inverse dynamics model hyperparameters on two heldout demonstrations for the relocate task.

Replay buffer. To store examples for training the inverse dynamics model, we use a replay buffer of size 1M. We sample examples uniformly at random. When the buffer fills up we drop the least recently added examples.

Density model. Our density model training settings closely follow the inverse dynamics model settings except for two main differences. First, we use the binary cross-entropy loss function. Second, in each batch we sample positives and negatives in equal proportion (ratio of one half).

Experimental setup. For all experiments, we report the mean return and standard deviation across three random seeds. All of our experiments are run on CPUs.

References

Abbeel, P. and Ng, A. Y. Apprenticeship learning via inverse reinforcement learning. In *ICML*, 2004.

Abbeel, P., Coates, A., and Ng, A. Y. Autonomous helicopter aerobatics through apprenticeship learning. *IJRR*, 2010.

Agrawal, P., Nair, A. V., Abbeel, P., Malik, J., and Levine, S. Learning to poke by poking: Experiential learning of intuitive physics. In *NIPS*, 2016.

Andrews, S. and Kry, P. G. Goal directed multi-finger manipulation: Control policies and analysis. *Computers & Graphics*, 2013.

Aytar, Y., Pfaff, T., Budden, D., Paine, T., Wang, Z., and de Freitas, N. Playing hard exploration games by watching youtube. In *NIPS*, 2018.

Bai, Y. and Liu, C. K. Dexterous manipulation using both palm and fingers. In *ICRA*, 2014.

Bain, M. and Sammut, C. A framework for behavioural cloning. In *Machine Intelligence 15*, 1995.

Bakker, P. and Kuniyoshi, Y. Robot see, robot do: An overview of robot imitation. In *AISB96 Workshop on Learning in Robots and Animals*, 1996.

Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U., Zhang, J., et al. End to end learning for self-driving cars. *arXiv:1604.07316*, 2016.

Christiano, P., Shah, Z., Mordatch, I., Schneider, J., Blackwell, T., Tobin, J., Abbeel, P., and Zaremba, W. Transfer from simulation to real world through learning deep inverse dynamics model. *arXiv:1610.03518*, 2016.

Chua, K., Calandra, R., McAllister, R., and Levine, S. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *NIPS*, 2018.

Dogar, M. R. and Srinivasa, S. S. Push-grasping with dexterous hands: Mechanics and a method. In *IROS*, 2010.

Duan, Y., Chen, X., Houthooft, R., Schulman, J., and Abbeel, P. Benchmarking deep reinforcement learning for continuous control. In *ICML*, 2016.

Edwards, A. D., Sahni, H., Schroecker, Y., and Isbell, C. L. Imitating latent policies from observation. *arXiv:1805.07914*, 2018.

Erez, T., Tassa, Y., and Todorov, E. Simulation tools for model-based robotics: Comparison of bullet, havok, mujoco, ode and physx. In *ICRA*, 2015.

Fu, J., Luo, K., and Levine, S. Learning robust rewards with adversarial inverse reinforcement learning. *arXiv:1710.11248*, 2017.

Handa, A., Van Wyk, K., Yang, W., Liang, J., Chao, Y.-W., Wan, Q., Birchfield, S., Ratliff, N., and Fox, D. Dexpilot: Vision based teleoperation of dexterous robotic hand-arm system. *arXiv:1910.03135*, 2019.

Ho, J. and Ermon, S. Generative adversarial imitation learning. In *NIPS*, 2016.

Kakade, S. M. A natural policy gradient. In *NIPS*, 2002.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *ICLR*, 2015.

Kumar, A., Gupta, S., and Malik, J. Learning navigation subroutines by watching videos. *arXiv:1905.12612*, 2019.

Kumar, V. and Todorov, E. Mujoco haptix: A virtual reality system for hand manipulation. In *Humanoids*, 2015.

Kumar, V., Xu, Z., and Todorov, E. Fast, strong and compliant pneumatic actuation for dexterous tendon-driven hands. In *ICRA*, 2013.

Liu, F., Ling, Z., Mu, T., and Su, H. State alignment-based imitation learning. In *ICLR*, 2020.

Liu, Y., Gupta, A., Abbeel, P., and Levine, S. Imitation from observation: Learning to imitate behaviors from raw video via context translation. *arXiv:1707.03374*, 2017.

Meltzoff, A. N. Understanding the intentions of others: re-enactment of intended acts by 18-month-old children. *Developmental psychology*, 1995.

Nagabandi, A., Kahn, G., Fearing, R. S., and Levine, S. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *ICRA*, 2018.

Nagabandi, A., Konoglie, K., Levine, S., and Kumar, V. Deep dynamics models for learning dexterous manipulation. *arXiv:1909.11652*, 2019.

Nair, A., Chen, D., Agrawal, P., Isola, P., Abbeel, P., Malik, J., and Levine, S. Combining self-supervised learning and imitation for vision-based rope manipulation. In *ICRA*, 2017.

Needleman, S. B. and Wunsch, C. D. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 1970.

Ng, A. Y., Russell, S. J., et al. Algorithms for inverse reinforcement learning. In *ICML*, 2000.

OpenAI, Andrychowicz, M., Baker, B., Chociej, M., Jzefowicz, R., McGrew, B., Pachocki, J., Petron, A., Plappert, M., Powell, G., Ray, A., Schneider, J., Sidor, S., Tobin, J., Welinder, P., Weng, L., and Zaremba, W. Learning dexterous in-hand manipulation. *arXiv:1808.00177*, 2018.

OpenAI, Akkaya, I., Andrychowicz, M., Chociej, M., Litwin, M., McGrew, B., Petron, A., Paino, A., Plappert, M., Powell, G., Ribas, R., Schneider, J., Tezak, N., Tworek, J., Welinder, P., Weng, L., Yuan, Q., Zaremba, W., and Zhang, L. Solving rubik's cube with a robot hand. *arXiv:1910.07113*, 2019.

Pathak, D., Mahmoudieh, P., Luo, G., Agrawal, P., Chen, D., Shentu, Y., Shelhamer, E., Malik, J., Efros, A. A., and Darrell, T. Zero-shot visual imitation. In *ICLR*, 2018.

Peng, X. B., Abbeel, P., Levine, S., and van de Panne, M. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *TOG*, 2018a.

Peng, X. B., Kanazawa, A., Malik, J., Abbeel, P., and Levine, S. Sfv: Reinforcement learning of physical skills from videos. *TOG*, 2018b.

Peters, J. and Schaal, S. Reinforcement learning of motor skills with policy gradients. *Neural networks*, 2008.

Pinto, L., Gandhi, D., Han, Y., Park, Y.-L., and Gupta, A. The curious robot: Learning visual representations via physical interactions. In *ECCV*, 2016.

Pomerleau, D. A. Alvinn: An autonomous land vehicle in a neural network. In *NIPS*, 1989.

Rajeswaran, A., Kumar, V., Gupta, A., Vezzani, G., Schulman, J., Todorov, E., and Levine, S. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. In *RSS*, 2018.

Ross, S., Gordon, G., and Bagnell, D. A reduction of imitation learning and structured prediction to no-regret online learning. In *AISTATS*, 2011.

Russell, S. Learning agents for uncertain environments. In *COLT*, 1998.

Sadeghi, F. and Levine, S. Cad2rl: Real single-image flight without a single real image. In *RSS*, 2017.

Sakoe, H. and Chiba, S. Dynamic programming algorithm optimization for spoken word recognition. *IEEE transactions on acoustics, speech, and signal processing*, 1978.

Schaal, S. Is imitation learning the route to humanoid robots? *Trends in cognitive sciences*, 1999.

Sermanet, P., Lynch, C., Chebotar, Y., Hsu, J., Jang, E., Schaal, S., Levine, S., and Brain, G. Time-contrastive networks: Self-supervised learning from video. In *ICRA*, 2018.

Sharma, P., Mohan, L., Pinto, L., and Gupta, A. Multiple interactions made easy (mime): Large scale demonstrations data for imitation. *arXiv:1810.07121*, 2018.

Sun, W., Vemula, A., Boots, B., and Bagnell, J. A. Provably efficient imitation learning from observation alone. *arXiv:1905.10948*, 2019.

Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., and Abbeel, P. Domain randomization for transferring deep neural networks from simulation to the real world. In *IROS*, 2017.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *IROS*, 2012.

Tomasello, M., Kruger, A. C., and Ratner, H. H. Cultural learning. *Behavioral and brain sciences*, 1993.

Torabi, F., Warnell, G., and Stone, P. Behavioral cloning from observation. *arXiv:1805.01954*, 2018a.

Torabi, F., Warnell, G., and Stone, P. Generative adversarial imitation from observation. *arXiv:1807.06158*, 2018b.

Večerík, M., Hester, T., Scholz, J., Wang, F., Pietquin, O., Piot, B., Heess, N., Rothörl, T., Lampe, T., and Riedmiller, M. Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards. *arXiv:1707.08817*, 2017.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 1992.