Reinforcement Learning with Videos: Combining Offline Observations with Interaction

Karl Schmeckpeper[1], **Oleh Rybkin**[1], **Kostas Daniilidis**[1], **Sergey Levine**[2], **and Chelsea Finn**[3]
University of Pennsylvania[1], University of California, Berkeley[2], Stanford University[3]
karls@seas.upenn.edu

Abstract: Reinforcement learning is a powerful framework for robots to acquire skills from experience, but often requires a substantial amount of online data collection. As a result, it is difficult to collect sufficiently diverse experiences that are needed for robots to generalize broadly. Videos of humans, on the other hand, are a readily available source of broad and interesting experiences. In this paper, we consider the question: can we perform reinforcement learning directly on experience collected by humans? This problem is particularly difficult, as such videos are not annotated with actions and exhibit substantial visual domain shift relative to the robot's embodiment. To address these challenges, we propose a framework for reinforcement learning with videos (RLV). RLV learns a policy and value function using experience collected by humans in combination with data collected by robots. In our experiments, we find that RLV is able to leverage such videos to learn challenging vision-based skills with less than half as many samples as RL methods that learn from scratch.

Keywords: reinforcement learning, learning from observation

1 Introduction

Reinforcement learning is a powerful tool for robots to automatically acquire behaviors, but requires a significant amount of online trial-and-error data collection. While one solution is to build more data-efficient algorithms, advances in other areas of deep learning [1, 2] suggest that *large* and *diverse* real-world datasets are critical for broad generalization and high performance. Therefore, we instead look to videos of humans as a readily available source of broad and interesting training data. We propose to use videos of people within a reinforcement learning framework to augment the data acquired by a robot. If this were feasible, it would represent an important first step towards being able to tap into cheap and readily available datasets of diverse human behaviors when training robots in the real world.

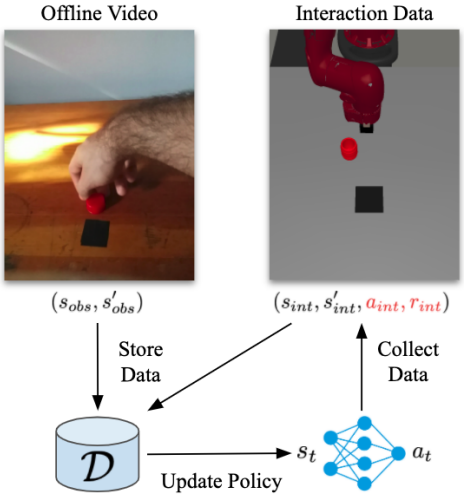


Figure 1: **Reinforcement learning with videos**. We study the setting where observational data is available, in the form of videos (top left). Our method can leverage such data to improve reinforcement learning by adding the videos to the replay buffer and directly performing RL on the observational data, while overcoming the challenges of unknown actions and domain shift between observation and interaction data.

Performing reinforcement learning directly from videos of human-provided behaviors presents multiple challenges. First, the robot must be able to update its policy using observations without any corresponding actions or rewards. Second, the algorithm must be able to account for domain shift from differences in the action space, morphology, viewpoint, and environment, as humans look different and have different degrees of freedom than most robotic manipulators. While prior works have made progress on imitation learning of action-free demonstrations [3, 4], we specifically focus on reinforcement learning since widely-available video data may

4th Conference on Robot Learning (CoRL 2020), Cambridge MA, USA.

not perform the task optimally, may perform tasks in a way that is suboptimal for robot morphologies, or may contain trajectories of many distinct tasks. Even perfect imitation may be unable to learn successful policies from such data, but reinforcement learning, which can learn from both successful and unsuccessful trials, should be able to learn successful policies while better leveraging the information available in the observation data.

Unlike policies learned via imitation learning, value functions acquired with reinforcement learning can capture general-purpose information. For example, a door rotates around its hinges regardless of whether its handle was pulled by a person or a robot. Videos of humans contain insight into object interactions, semantically meaningful tasks, and the physics of the environment. We therefore aim to leverage such observational data in a reinforcement learning framework, and overcome the aforementioned challenges through a simple approach that infers actions and rewards for the observation data from domain-invariant representations of the observed images.

The main contribution of this paper is a framework for reinforcement learning from videos (RLV) that leverages both offline observation data and online interaction. We instantiate this framework and demonstrate that this approach allows a robot to learn from observational robot or human demonstrations, significantly improving the speed of training. In a series of simulation experiments, including challenging vision-based robotic manipulation tasks, we find that RLV can leverage observational demonstrations of varying quality to learn tasks with *half* the number of trials, compared to standard RL and prior works on imitation from observation. Further, we find that, with a small number of offline paired frames, RLV can extend to real videos of humans such as those shown in Figure 1.

2 Related Work

Learning from demonstration. Early work on learning from demonstration focused on high-quality demonstrations and tried to match the policy of the expert [5, 6, 7, 8]. There have been many recent works that study various facets of imitation learning [9, 10, 11, 12, 13, 14, 15, 16, 17]. However, these works assume access to the demonstrator's actions, which often requires extra instrumentation or might even be impossible if morphologies of the demonstrator and the agent are different. In contrast, our method does not require the observation data to contain actions.

Imitation learning without actions. In situations such as learning from human demonstrations, only observations of the demonstrator are available, while the actions are not. Several approaches to learning only from observations exist, such as matching the demonstrated state transitions [18, 3, 19, 20, 4, 21] or training a reward function to encourage behavior similar to that in the demonstrations [22, 23, 24, 25, 26, 27, 28, 29]. However, these methods are all limited by the performance of the demonstrator that they try to imitate. In contrast, our method integrates observation data into a reinforcement learning pipeline and can improve over the observation data.

Reinforcement learning with demonstrations. Recent work has used demonstration trajectories to improve performance of reinforcement learning agents [30, 31, 32, 33, 34, 35]. In contrast to standard imitation approaches, these methods allow improving over the expert performance as the policy can be further fine-tuned via reinforcement learning. We use the same principle to design RLV, however, unlike these works, we don't assume that the demonstration data includes actions.

More closely related to this work, recent approaches have pushed toward removing assumptions on the demonstration data. Edwards et al. [36] propose learning from offline data without actions by training a state-next state value function and a forward dynamics model. Schmeckpeper et al. [37] train a model-based agent that can incorporate data without actions and with domain shift, such as learning robotic tasks from videos of humans. In contrast to these two works, RLV does not require that a good dynamics model be learned, and handles more complex domain shift with different agent morphologies, viewpoints, and background. Concurrent work by Chang et al. [38] leverages offline videos for navigation tasks, but not address the challenge of differences in morphology and the action space between the offline observations and the robot data, which is required to learn object manipulation with human videos. By handling more complex types of domain shift, our approach takes a step toward learning visual manipulation tasks with diverse open-world human data.

Domain adaptation. Data collected from different agents can be very dissimilar. To overcome the domain shift between different agents, one group of methods used unpaired image-to-image translation [39, 40, 41, 42] to transform samples in a source domain into samples from the target domain. This approach has been used for sim-to-real adaptation [43, 44] and for learning from demonstrations [24, 45, 26]. A second group of methods, based on domain confusion [46, 47, 48,

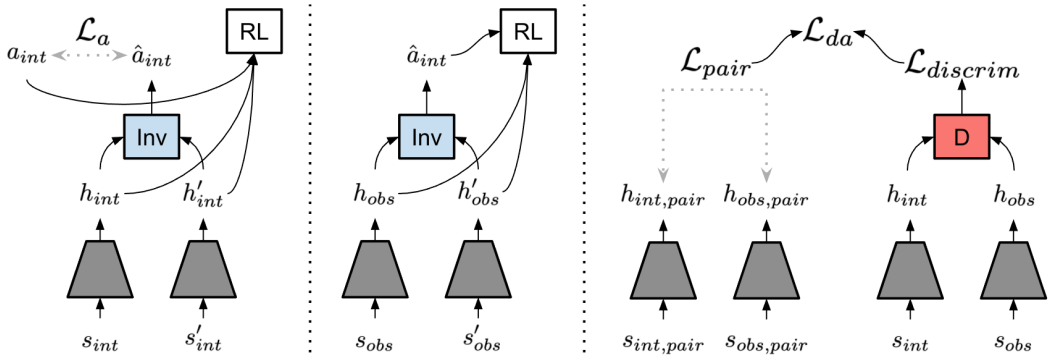


Figure 2: Components of reinforcement learning with offline videos. **Left:** a batch with samples $(\mathbf{s}_{int}, \mathbf{a}_{int}, \mathbf{s}'_{int}, \mathbf{r}_{int})$ is sampled from the action-conditioned replay pool, $\mathcal{D}_{int}$, and the observations are encoded into features $\mathbf{h}_{int}, \mathbf{h}'_{int}$. An inverse model is trained to predict the action $\mathbf{a}_{int}$ from the features $\mathbf{h}_{int}, \mathbf{h}'_{int}$. **Middle:** the inverse model is used to predict the missing actions in the offline videos, $\hat{\mathbf{a}}_{int}$, in the robot's action space, from features $(\mathbf{h}_{obs}, \mathbf{h}'_{obs})$ that were extracted from observations $(\mathbf{s}_{obs}, \mathbf{s}'_{obs})$. To obtain the missing rewards $\hat{\mathbf{r}}_{obs}$, we label the final step in the trajectory with a large reward and other steps with a small reward. **Right:** we use adversarial domain confusion to align the features from the action-conditioned data, $\mathbf{h}_{int}$ with the features from the action-free data, $\mathbf{h}_{obs}$. We find that minimizing the difference between the features of a small amount of off-policy paired data, $\mathbf{h}_{int,pair}, \mathbf{h}_{obs,pair}$ helps to find good features. Finally, we use an off-policy reinforcement learning algorithm on the resulting batch $((\mathbf{h}_{int}, \mathbf{h}_{obs}), (\mathbf{a}_{int}, \hat{\mathbf{a}}_{int}), (\mathbf{h}'_{int}, \mathbf{h}'_{obs}), (\mathbf{r}_{int}, \hat{\mathbf{r}}_{obs}))$. By overcoming the challenges of missing actions, rewards, and the presence of domain shift, we are able to effectively use the observation data to improve performance of a reinforcement learning agent.

49, 50], have sought to learn a set of domain-invariant features that still contain all the required information to complete the task [18, 51, 52, 53, 54]. In contrast to these works, we leverage adversarial domain confusion for the task of reinforcement learning with offline observations.

3 Reinforcement Learning with Videos

To learn from observations of another agent, several challenges must be overcome. The observational data lacks actions and rewards, and may exhibit substantial domain shift relative to the robot. In this section, we describe our framework for learning with observational data as well as the specific techniques we use for estimating the rewards, estimating the actions, and handling domain shift.

3.1 Problem Formulation

We formulate the problem as a Markov decision process (MDP), defined by the tuple $(\mathcal{S}_{int}, \mathcal{A}_{int}, P, R)$ where $\mathbf{s}_{int} \in \mathcal{S}_{int}$ is the state space, $\mathbf{a}_{int} \in \mathcal{A}_{int}$ is the action space, $P(\mathbf{s}'_{int}|\mathbf{a}_{int}, \mathbf{s}_{int})$ is the environment's dynamics, and $R(\mathbf{s}_{int}, \mathbf{a}_{int})$ is the reward function. The agent is initially provided with a set of observations $\{(\mathbf{s}_{obs}, \mathbf{s}'_{obs})_{1:t}\}$ from another agent such as a human, that are modeled as coming from another MDP, $(\mathcal{S}_{obs}, \mathcal{A}_{obs}, P, R)$, which has a distinct state and action space, but whose dynamics and reward function are isomorphic to the original MDP, meaning that there exists some transformation of one state and action space into the other that preserves dynamics and rewards. Formally, this may be written as the assumption that $P(\mathbf{s}'_{int}|\mathbf{a}_{int}, \mathbf{s}_{int}) = P(g_o(\mathbf{s}'_{obs})|g_a(\mathbf{a}_{obs}), g_o(\mathbf{s}_{obs}))$, where g_o and g_a are unknown invertible functions. While this assumption represents a simplification, in that the human's state is not actually isomorphic to that of a robot, it does allow us to bridge the gap between the two MDPs with simple domain adaptation methods, as we discuss in Section 3.5, because the isomorphism property implies that there exists a third representation that is *invariant* with respect to the two MDPs. We find that including a small amount of offline paired data, $(\mathbf{s}_{int,pair}, \mathbf{s}_{obs,pair})$ where $\mathbf{s}_{int,pair} = g_o(\mathbf{s}_{obs,pair})$, significantly improves the ability of our domain adaptation to learn useful representations. We further assume that the agent that collected the observational data was executing a successful, but potentially sub-optimal policy.

The robot may collect additional experience by executing actions, $\mathbf{a}_{int} \in \mathcal{A}_{int}$, to interact with its environment, allowing it to collect observations, $\mathbf{s}_{int} \in \mathcal{S}_{obs}$, and rewards, $\mathbf{r}_{int} \in R_{int}$ The agent seeks to learn a policy that maximizes the expected return in the current environment.

3.2 Overview

An overview of our method is in Figure 2. To learn from both offline observations and online interaction, we maintain two replay pools, one containing the action-free observation data, $(\mathbf{s}_{obs}, \mathbf{s}'_{obs}) \in$

$\mathcal{D}_{obs}$ and one containing the action-conditioned interaction data, $(\mathbf{s}_{int}, \mathbf{a}_{int}, \mathbf{s}'_{int}, \mathbf{r}_{int}) \in \mathcal{D}_{int}$. The pool of interaction data, $\mathcal{D}_{int}$, is updated during training, while the pool of observation data, $\mathcal{D}_{obs}$, only contains the initial set of observations. We also maintain a fixed set of offline paired data $(\mathbf{s}_{obs,pair}, \mathbf{s}_{int,pair}) \in \mathcal{D}_{pair}$ used only for handling domain shift.

To overcome domain shift, we learn to embed each observation $\mathbf{s}$ to a domain-invariant feature vector $\mathbf{h}$. This is discussed in Section 3.5. In order to include the encoded observation tuples, $(\mathbf{h}_{obs}, \mathbf{h}'_{obs})$, in the RL process, they must be annotated with actions and rewards. Drawing on the isomorphism assumption discussed in Section 3.1, we propose to learn an inverse model f_{inv} that can map the invariant feature tuples $(\mathbf{h}, \mathbf{h}')$ to robot actions $\mathbf{a}_{int}$. Since the features are invariant between human and robot observations, we can train this inverse model on the robot data, and then use it to annotate human data, as we will discuss in Section 3.3. We also generate rewards for the observation data using a simple scheme described in Section 3.4. All sampled tuples from *both* replay pools are then combined into a batch and passed into the reinforcement learning algorithm, thereby allowing our method to perform reinforcement learning directly on both the interaction data and the observation data. This approach is detailed in Algorithm 1.

Algorithm 1 Reinforcement Learning with Videos (RLV)

1: Initialize replay buffer $\mathcal{D}_{int} \leftarrow \{\}$ ▷ Initialize replay pools and networks
2: Fill buffer $\mathcal{D}_{obs} \leftarrow \{(\mathbf{s}_{obs}, \mathbf{s}'_{obs})_{1:t}\}$ with observed data
3: RL.init(), f_{enc}.init(), f_{inv}.init()
4: **for** each iteration **do**
5: **for** each environment step **do**
6: Sample transition $(\mathbf{s}_{int}, \mathbf{a}_{int}, \mathbf{s}'_{int}, \mathbf{r}_{int})$ using RL's exploration policy
7: $\mathcal{D}_{int} \leftarrow \mathcal{D}_{int} \cup \{(\mathbf{s}_{int}, \mathbf{a}_{int}, \mathbf{s}'_{int}, \mathbf{r}_{int})\}$
8: **for** each gradient step **do**
9: $\{(\mathbf{s}_{int}, \mathbf{a}_{int}, \mathbf{s}'_{int}, \mathbf{r}_{int})_{1:n}\} \sim \mathcal{D}_{int}$ ▷ Sample from replay pools
10: $\{(\mathbf{s}_{obs}, \mathbf{s}'_{obs})_{1:m}\} \sim \mathcal{D}_{obs}$
11: $\mathbf{h}_{int} \leftarrow f_{enc}(\mathbf{s}_{int}; \psi)$ ▷ Extract feature representations
12: $\mathbf{h}_{obs} \leftarrow f_{enc}(\mathbf{s}_{obs}; \psi)$
13: $\hat{\mathbf{a}}_{int} = f_{inv}(\mathbf{h}_{obs}, \mathbf{h}'_{obs}; \theta)$ ▷ Generate actions & rewards for observation data
14: $\hat{\mathbf{r}}_{obs} = R(\mathbf{s}_{obs}, \mathbf{s}'_{obs})$ with R defined in Equation 2
15: RL.train_step($\{(\mathbf{h}_{int}, \mathbf{a}_{int}, \mathbf{h}'_{int}, \mathbf{r}_{int})_{1:n}\} \cup \{(\mathbf{h}_{obs}, \hat{\mathbf{a}}_{int}, \mathbf{h}'_{obs}, \hat{\mathbf{r}}_{obs})_{1:m}\}$) ▷ Train RL
16: Update θ using action-prediction loss $\mathcal{L}_a$ (Equation 1) ▷ Update encoder & inverse model
17: Update ψ with joint optimization objective (Equation 6)

This framework can be combined with any method of estimating the actions and the rewards of the observational data. We next present the simple approaches that we use for estimating the action and for estimating the reward.

3.3 Action Prediction

To learn from observational data, our algorithm must be able to estimate the actions used to transition between states. To do so, we train a model to estimate the action via supervised learning using the interaction data. In particular, we train an inverse model, parameterized by θ, to calculate the action in the robot's action space, $\mathbf{a}_{int} \in \mathcal{A}_{int}$, from a pair of invariant feature encodings, $(\mathbf{h}, \mathbf{h}')$. Due to the isomorphism of the environment, we should expect to be able to predict actions for data from either MDP. The inverse model is trained to minimize the mean squared error between the predicted action from the action-conditioned interaction data, $\hat{\mathbf{a}}_{int} = f_{inv}(\mathbf{h}_{int}, \mathbf{h}'_{int}; \theta)$, and the corresponding true action $\mathbf{a}_{int}$:

$$\mathcal{L}_a(\mathbf{a}_{int}, \mathbf{h}_{int}, \mathbf{h}'_{int}, \theta) = \|\mathbf{a}_{int} - f_{inv}\left(\mathbf{h}_{int}, \mathbf{h}'_{int}; \theta\right)\|^2 \tag{1}$$

We then apply this trained inverse model to the action-free observation data. We predict actions $\hat{\mathbf{a}}_{int} = f_{inv}(\mathbf{h}_{obs}, \mathbf{h}'_{obs}; \theta)$ and use them to train the reinforcement learning algorithm. We predict actions in Line 13 and train the inverse model on Line 16 in Algorithm 1.

3.4 Reward Generation

One impediment to using observation data in reinforcement learning is that it lacks rewards. We could use the same approach as in the previous section to predict rewards as well as actions, by training a reward model on top of the invariant features. However, this may perform poorly in sparse reward settings, where initial robot data contains no informative reward supervision. In our implementation, we instead opt for a simple alternative to label the observation data with rewards using the methodology proposed by Reddy et al. [33]. The final timestep in a trajectory is assigned a

large constant reward, c_{large}, while every other timestep is assigned a small constant reward, c_{small}:

$$R(\mathbf{s}, \mathbf{s}') = \begin{cases} c_{large} & \mathbf{s}' \text{ is terminal} \\ c_{small} & \mathbf{s}' \text{ is not terminal} \end{cases} \tag{2}$$

This encourages the agent to try to reach the states at the end of the observed trajectories, implicitly assuming that the observed trajectories are good. We show that despite this assumption, the approach is still robust to observations from sub-optimal trajectories in Section 4.1. We expect this robustness arises because the observation data primarily acts to speed up exploration, while the robot gathers interaction data that can counteract any inaccuracies in the observation data.

3.5 Domain Adaptation

As discussed in the previous sections, utilizing the observational data $(\mathbf{s}_{obs}, \mathbf{s}'_{obs})$ requires mapping it into an invariant representation $\mathbf{h}$. This is necessary both for training the inverse model and for utilizing the resulting tuples $(\mathbf{h}_{obs}, \hat{\mathbf{a}}_{int}, \mathbf{h}'_{obs}, \hat{\mathbf{r}}_{obs})$ alongside samples from the original MDP $(\mathbf{h}_{int}, \mathbf{a}_{int}, \mathbf{h}'_{int}, \mathbf{r}_{int})$. In this section, we describe how we can train such an invariant encoder, following the methodology in the domain adaptation literature [50].

We train our feature extractor, f_{enc}, to learn an encoded representation, $\mathbf{h} = f_{enc}(\mathbf{s}; \psi)$, of an observation $\mathbf{s}$. This encoded representation should contain all relevant information, while being invariant to the domain the observation originated from. To train for such domain invariance, we separately train a discriminator, f_{discr}, to distinguish between encodings from the observational data, $\mathbf{h}_{obs} = f_{enc}(\mathbf{s}_{obs}; \psi)$, and encodings from the interaction data, $\mathbf{h}_{int} = f_{enc}(\mathbf{s}_{int}; \psi)$, and train the encodings in an adversarial manner, following work from [50], according to the following loss:

$$\mathcal{L}_{discrim}(\mathbf{s}_{int}, \mathbf{s}_{obs}, \psi, \phi) = \log\left(f_{discr}(f_{enc}(\mathbf{s}_{int}; \psi); \phi)\right) + \log\left(1 - f_{discr}(f_{enc}(\mathbf{s}_{obs}; \psi); \phi)\right) \tag{3}$$

During training, the encoder attempts to minimize the discriminator's ability to correctly classify the domain of the encoded features, while the discriminator tries to maximize it.

Additionally, we find that a small amount of offline paired data substantially improves the ability of the feature extractor to find a good encoding. We add the paired data with the following loss:

$$\mathcal{L}_{pair}(\mathbf{s}_{obs,pair}, \mathbf{s}_{int,pair}, \psi) = \left\| f_{enc}(\mathbf{s}_{int,pair}; \psi) - f_{enc}(\mathbf{s}_{obs,pair}; \psi) \right\|^2 \tag{4}$$

Combining this with the discriminator loss gives us the following total domain adaptation loss:

$$\mathcal{L}_{da}(\mathbf{s}_{int}, \mathbf{s}_{obs}, \mathbf{s}_{int,pair}, \mathbf{s}_{obs,pair}, \psi, \phi) = \mathcal{L}_{discrim}(\mathbf{s}_{int}, \mathbf{s}_{obs}, \psi, \phi) + \mathcal{L}_{pair}(\mathbf{s}_{obs,pair}, \mathbf{s}_{int,pair}, \psi) \tag{5}$$

Since the paired data is offline, the cost of collecting it is low since one set of paired data can be used across multiple tasks in the same environment. To produce these observations, we annotate the robot videos, collected from random exploration, and the human videos, and leverage the annotations to automatically match observations across domains. This procedure can be performed in a scalable way as it does not make any assumptions on the source of the data and only a small number of frames needs to be annotated. We provide a comparison between domain adaptation with and without paired data in Appendix D.

3.6 Joint Optimization

We jointly optimize the domain adaptation loss with the inverse model loss, $\mathcal{L}_a$ and the optimization objective of the chosen reinforcement learning algorithm, $\mathcal{L}_{\text{RL}}$, according to the following objective:

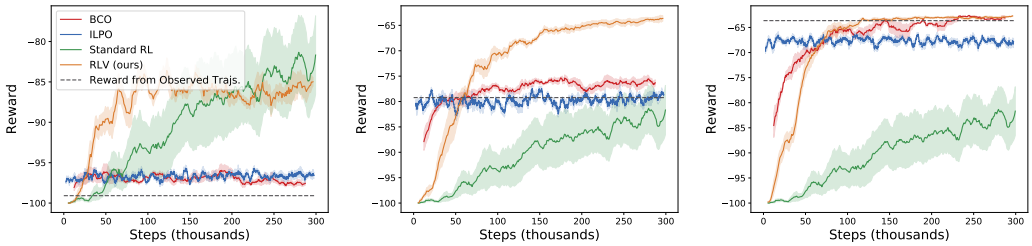
$$\min_{\psi, \theta} \sum_{\substack{\{\mathbf{s}_{obs}, \mathbf{s}'_{obs}\} \in \mathcal{D}_{obs}, \\ \{\mathbf{s}_{int}, \mathbf{a}_{int}, \mathbf{s}'_{int}\} \in \mathcal{D}_{int}, \\ \{\mathbf{s}_{int,pair}, \mathbf{s}_{obs,pair}\} \in \mathcal{D}_{pair}}} c_2 \mathcal{L}_{\text{RL}} + c_1 \mathcal{L}_a(\mathbf{a}_{int}, f_{enc}(\mathbf{s}_{int}; \psi), f_{enc}(\mathbf{s}'_{int}; \psi), \theta) + c_3 \max_{\phi} \mathcal{L}_{da}(\mathbf{s}_{int}, \mathbf{s}_{obs}, \mathbf{s}_{int,pair}, \mathbf{s}_{obs,pair}, \psi, \phi) \tag{6}$$

Constants, c_1, c_2, and c_3 are hyperparameters that control the relative importance of each term.

All of the architecture details and hyperparameters are available in Appendix C.

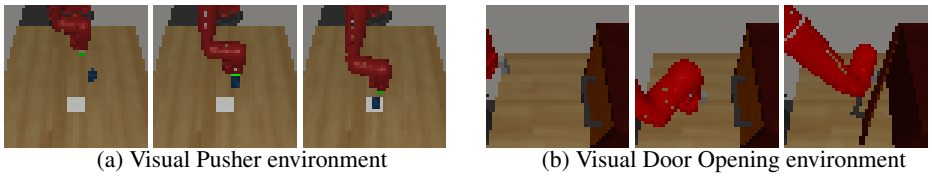
4 Experiments

The goal of our experiments is to evaluate whether RLV can enable more data-efficient reinforcement learning of robotic skills by incorporating video data without actions. To this end, we study the following questions: **(1)** Can RLV learn from observations of sub-optimal policies? **(2)** Can RLV



(a) $\mathcal{D}_{obs}$ from SAC trained for 25K steps (b) $\mathcal{D}_{obs}$ from SAC trained for 225K steps (c) $\mathcal{D}_{obs}$ from SAC trained for 975K steps

Figure 3: Performance on Acrobot with different qualities of observation data. RLV is generally able to achieve equal or higher final rewards than the competing methods while training with fewer samples. The performance is especially notable with medium-quality observation data.



(a) Visual Pusher environment (b) Visual Door Opening environment

Figure 4: Randomly selected trajectories from the policy learned by RLV in different environments. Both trajectories were successful.

learn complex vision-based tasks with sparse rewards from observations and interaction? **(3)** Can RLV learn from videos with large amounts of domain shift, such as videos of humans? We test these questions in three scenarios: learning simple control policies in the Acrobot-v1 environment [55], learning simulated robotic behaviours using simulated observational data, and learning simulated robotic behaviors using real-world human demonstrations. In all experiments, we make use of soft-actor critic (SAC) [56] as the underlying reinforcement learning algorithm. For all methods, we run five seeds and report the mean and standard error.[1]

4.1 Learning from Observations of sub-optimal policies

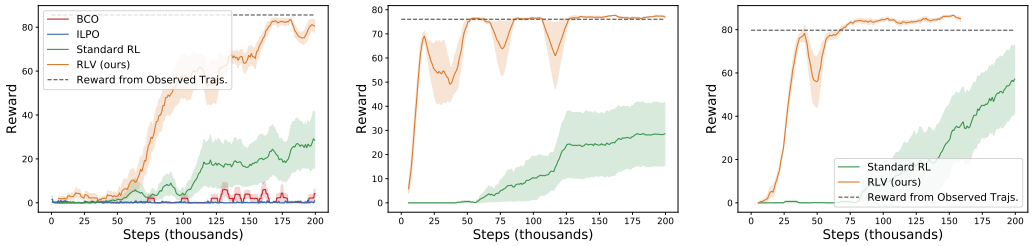
We first evaluate whether RLV can learn from and improve upon observations from sub-optimal policies. We perform these experiments on the Acrobot-v1 environment [55]. To generate observations of sub-optimal trajectories, we train SAC [57] to convergence on the environment and take temporally consecutive blocks of observations from different points in training. We discard the actions and the rewards in this data, so RLV only has access to the data it would have if it was learning from human observations. We compare RLV to two prior methods that can leverage the observational data, ILPO [4] and BCO [3]. For both algorithms, we use the implementation provided by [4] and the hyperparameters the authors tuned for the Acrobot-v1 environment. We additionally compare to SAC [56, 57], a standard off-policy RL algorithm that is unable to make use of the action and reward free observational data. This provides a direct comparison to RLV as we use the same implementation of SAC as the underlying RL algorithm in RLV. For this environment, and the environments in Section 4.2, we disable the domain-adaptation portion of RLV by setting $c_3 = 0$.

The results for these experiments are shown in Figure 3. We find that RLV consistently achieves higher rewards than ILPO [4], BCO [3], and standard reinforcement learning [56]. ILPO is more data-efficient but it is unable to improve upon observations from sub-optimal trajectories, while standard reinforcement learning trains requires more samples than RLV as it cannot leverage the offline observation data. The performance improvements are most significant in Figure 3b, where the observations came from a policy that achieved a medium level of performance. Good performance with medium-quality data is particularly important since acquiring high-quality observation data is difficult and the domain gap will implicitly degrade their quality, while low quality observations contain little information so any method will learn slowly.

4.2 Robotic Tasks

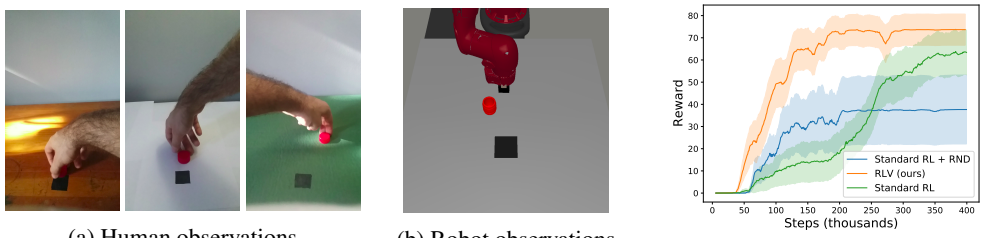
Next, we evaluate whether RLV can learn more complex robotic manipulation tasks using observations without domain shift. We first examine a robotic pushing task [58]. Unlike the Acrobot

[1] Videos and training code are available at our website: https://sites.google.com/view/rl-with-videos



(a) State Pusher. (b) Visual Door Opening. (c) Visual Pusher.

Figure 5: Rewards for the State Pusher, Visual Door Opening, and the Visual Pusher environments. In both simulated environments, the agent trained with RLV requires fewer samples to solve the task than conventional reinforcement learning.



(a) Human observations (b) Robot observations (c) Training curves.

Figure 6: Training curves and example images for learning pushing with human observations. Our model is able to leverage videos of humans to solve the task with significantly fewer samples than conventional reinforcement learning.

environment, which has a discrete action space with three possible values, the pushing task has an action space with two continuous dimensions, significantly increasing the difficulty of determining the actions that caused a sequence of observations. The reward in this environment is sparse: it is one if the puck is within three centimeters of the goal location, and zero otherwise. The observational data is from training SAC for 1M steps and taking observations of trajectories generated by the resulting policy. We present results for the pushing task using state-based observations in Figure 5a. RLV requires 30% as many samples to solve the task as SAC, while BCO and ILPO fail to ever reliably solve the task, as they struggle with the larger action space or more complicated dynamics.

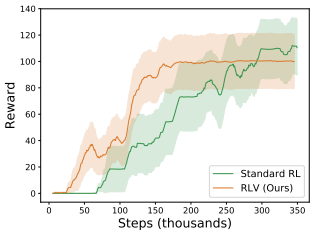
We also investigate image based observations, using the Visual Pusher and the Visual Door Opening tasks from [58]. We again use a sparse binary reward function. The reward for the Visual Pusher is the same as the previous experiment, while the reward for the Visual Door Opening environment is zero if the door was not open to within a five degrees of the goal angle and one if the door is. We were unable to train with BCO or ILPO to achieve non-zero rewards on tasks with image-based observations. The results for these environments are shown in Figure 5. We find that RLV requires 3x fewer samples as SAC in both environments. Further, on the Visual Door Opening environment in Figure 5b, only three out of five of the training seeds of SAC converge to the optimal policy, while all of the training seeds of RLV reach the optimal policy. Randomly sampled trajectories from the policies learned by RLV are shown in Figure 4a for the Visual Pusher and in Figure 4b for the Visual Door Opening environment.

4.3 Learning from Real-World Human Videos

We next seek to study if RLV can learn robotic policies from videos of humans. Given videos of humans completing the task in the real world and paired, off-policy images between the robot and human environments, we train an agent to complete a similar task in simulation. In addition to learning from observations without actions or rewards, the agent must also overcome the domain shift between real and simulated images and between humans and robots. We consider two tasks, pushing an object and opening a drawer. Example observation from the human domain are shown in Figure 6a for the pushing task and Figure 7a for the drawer opening task. Example observations from the robot domain are shown in Figure 6b for the pushing task and Figure 7b for the drawer opening task. Note that there is considerable domain shift in color, lighting conditions, object positions, and embodiment. More information on the datasets are available in Appendix E. For the pushing task, our simulated environment is a recolored and more difficult version of the previous Visual Pusher environment [58] using the same sparse reward function as our previous experiments. For

(a) Human observations

(b) Robot observations



(c) Training curves.

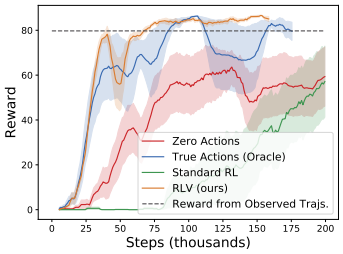
Figure 7: Training curves and example images for learning drawer opening with human observations. Our model is able to leverage videos of humans to solve the task with significantly fewer samples than conventional reinforcement learning.

the drawer opening task, our simulated environment is a modified version of the drawer opening task from Meta-World [59] with a reduced action space and a more difficult initial configuration.

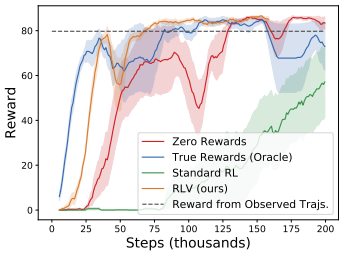
The results for different approaches are shown in Figure 6c for the pushing task and Figure 7c for the drawer opening task. RLV is able to leverage the human observations to significantly speed up training, requiring only 50% of the samples that SAC requires to match its performance in both tasks. Further, to evaluate whether a similar improvement may be achieved with a task-agnostic exploration technique, we compare to a version of SAC that uses the exploration bonus from RND [60]. We observe that RND learns faster than default RL, but still achieves inferior learning speed and final performance compared to RLV.

4.4 Ablations over action and reward prediction

We perform ablations over the actions and the rewards used for the observation data in the Visual Pusher environment. For rewards and actions, we compare using the ground truth values, using the values estimated by our approach, and using all zeros. The results are shown in Figure 8. For the actions, both our approach and using the ground truth actions perform well, while using zero actions initially performs better than the baseline reinforcement learning, but converges to a sub-optimal final score. This indicates that even poor estimates of the actions can speed up training. However, better estimates are required to consistently find the optimal policy and the actions estimated by the inverse model are of sufficient quality to accomplish this. For the rewards, both our approach and using the ground truth rewards perform comparably well, while using zero rewards underperforms but still converges to the optimal solution. This result likely relies on the fact that the ground truth reward is mostly zeros except for at the end, making it such the zero reward contains little incorrect information. This result generally suggests that RLV is fairly robust to the choice of reward labels.



(a) RLV with different action predictors



(b) RLV with different reward predictors

Figure 8: Ablations in the visual pusher environment over different action and reward predictors for the observation data. Even when the predictor always predicts zeros for the actions or rewards, our approach is more data-efficient than standard RL, while our method of estimating the actions and rewards performs comparably to using ground truth values.

5 Conclusion

We present reinforcement learning with videos, a framework for learning robotic policies using both online robotic interaction and offline observations of humans, incorporating the latter data by addressing the lack of actions and rewards and the domain shift with simple mechanisms. By leveraging observations of humans, this framework is able to learn significantly more efficiently than standard reinforcement learning methods that cannot readily leverage that data. It also outperforms prior approaches for imitation learning from observation, as it is able to improve over the demonstrator performance, tackle complex vision-based tasks, and handle large domain shift. Future work includes generalizing to even more visually complex scenes and handling more natural and diverse human demonstrations, while increasing the complexity of the learned behaviors. Overcoming these challenges will allow reinforcement learning to leverage the vast quantities of diverse and interesting observations of humans, making robotic agents both easier to train and more capable of solving challenging tasks.

Acknowledgments

We would like to thank Kenneth Chaney, Wendelyn Bolles, and our anonymous internal and external reviewers. This work was supported by ARL RCTA W911NF-10-2-0016, ARL DCIST CRA W911NF-17-2-0181, ONR grant N00014-20-1-2675, and by Honda Research Institute.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 2012.

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[3] F. Torabi, G. Warnell, and P. Stone. Behavioral cloning from observation. In *IJCAI*, 2018.

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

[5] D. A. Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural computation*, 3(1), 1991.

[6] A. Billard, S. Calinon, R. Dillmann, and S. Schaal. Survey: Robot programming by demonstration. *Handbook of robotics*, 59(BOOK_CHAP), 2008.

[7] B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5), 2009.

[8] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.

[9] C. Finn, S. Levine, and P. Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *ICML*, 2016.

[10] J. Ho and S. Ermon. Generative adversarial imitation learning. In *NeurIPS*, 2016.

[11] C. Finn, P. Christiano, P. Abbeel, and S. Levine. A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. *arXiv preprint arXiv:1611.03852*, 2016.

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

[13] P. Sharma, L. Mohan, L. Pinto, and A. Gupta. Multiple interactions made easy (mime): Large scale demonstrations data for imitation. In *CoRL*, 2018.

[14] T. Zhang, Z. McCarthy, O. Jow, D. Lee, X. Chen, K. Goldberg, and P. Abbeel. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. In *ICRA*. IEEE, 2018.

[15] R. Rahmatizadeh, P. Abolghasemi, L. Bölöni, and S. Levine. Vision-based multi-task manipulation for inexpensive robots using end-to-end learning from demonstration. In *ICRA*. IEEE, 2018.

[16] P. Florence, L. Manuelli, and R. Tedrake. Self-supervised correspondence in visuomotor policy learning. *IEEE RAL*, 5(2), 2019.

[17] D. S. Brown, W. Goo, P. Nagarajan, and S. Niekum. Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations. *arXiv preprint arXiv:1904.06387*, 2019.

[18] B. C. Stadie, P. Abbeel, and I. Sutskever. Third-person imitation learning. *arXiv:1703.01703*, 2017.

[19] F. Torabi, G. Warnell, and P. Stone. Generative adversarial imitation from observation. *arXiv preprint arXiv:1807.06158*, 2018.

[20] L. Lee, B. Eysenbach, E. Parisotto, E. Xing, S. Levine, and R. Salakhutdinov. Efficient exploration via state marginal matching. *arXiv preprint arXiv:1906.05274*, 2019.

[21] L. Shao, T. Migimatsu, Q. Zhang, K. Yang, and J. Bohg. Concept2robot: Learning manipulation concepts from instructions and human demonstrations. *Robotics: Science and Systems*, 2020.

[22] P. Sermanet, K. Xu, and S. Levine. Unsupervised perceptual rewards for imitation learning. *arXiv preprint arXiv:1612.06699*, 2016.

[23] P. Sermanet, C. Lynch, Y. Chebotar, J. Hsu, E. Jang, S. Schaal, S. Levine, and G. Brain. Time-contrastive networks: Self-supervised learning from video. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1134–1141. IEEE, 2018.

[24] Y. Liu, A. Gupta, P. Abbeel, and S. Levine. Imitation from observation: Learning to imitate behaviors from raw video via context translation. In *ICRA*. IEEE, 2018.

[25] F. Torabi, G. Warnell, and P. Stone. Imitation learning from video by leveraging proprioception. *arXiv preprint arXiv:1905.09335*, 2019.

[26] L. Smith, N. Dhawan, M. Zhang, P. Abbeel, and S. Levine. Avid: Learning multi-stage tasks via pixel-level translation of human videos. *arXiv preprint arXiv:1912.04443*, 2019.

[27] T. Yu, C. Finn, A. Xie, S. Dasari, T. Zhang, P. Abbeel, and S. Levine. One-shot imitation from observing humans via domain-adaptive meta-learning. *arXiv preprint arXiv:1802.01557*, 2018.

[28] Y. Aytar, T. Pfaff, D. Budden, T. Paine, Z. Wang, and N. de Freitas. Playing hard exploration games by watching youtube. In *Advances in Neural Information Processing Systems*, pages 2930–2941, 2018.

[29] A. Bonardi, S. James, and A. J. Davison. Learning one-shot imitation from humans without humans. *IEEE Robotics and Automation Letters*, 5(2):3533–3539, 2020.

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

[31] 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. *arXiv preprint arXiv:1709.10087*, 2017.

[32] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel. Overcoming exploration in reinforcement learning with demonstrations. In *ICRA*. IEEE, 2018.

[33] S. Reddy, A. D. Dragan, and S. Levine. Sqil: Imitation learning via reinforcement learning with sparse rewards. *arXiv preprint arXiv:1905.11108*, 2019.

[34] X. B. Peng, A. Kumar, G. Zhang, and S. Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

[35] A. Nair, M. Dalal, A. Gupta, and S. Levine. Accelerating online reinforcement learning with offline datasets. *arXiv preprint arXiv:2006.09359*, 2020.

[36] A. D. Edwards, H. Sahni, R. Liu, J. Hung, A. Jain, R. Wang, A. Ecoffet, T. Miconi, C. Isbell, and J. Yosinski. Estimating q (s, s') with deep deterministic dynamics gradients. *arXiv:2002.09505*, 2020.

[37] K. Schmeckpeper, A. Xie, O. Rybkin, S. Tian, K. Daniilidis, S. Levine, and C. Finn. Learning predictive models from observation and interaction. *ECCV*, 2020.

[38] M. Chang, A. Gupta, and S. Gupta. Semantic visual navigation by watching youtube videos. *arXiv preprint arXiv:2006.10034*, 2020.

[39] Y. Taigman, A. Polyak, and L. Wolf. Unsupervised cross-domain image generation. *arXiv preprint arXiv:1611.02200*, 2016.

[40] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, 2017.

[41] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. In *CVPR*, 2017.

[42] J. Hoffman, E. Tzeng, T. Park, J.-Y. Zhu, P. Isola, K. Saenko, A. Efros, and T. Darrell. Cycada: Cycle-consistent adversarial domain adaptation. In *ICML*, 2018.

[43] J. Zhang, L. Tai, P. Yun, Y. Xiong, M. Liu, J. Boedecker, and W. Burgard. Vr-goggles for robots: Real-to-sim domain adaptation for visual control. *IEEE Robotics and Automation Letters*, 4(2), 2019.

[44] K. Rao, C. Harris, A. Irpan, S. Levine, J. Ibarz, and M. Khansari. Rl-cyclegan: Reinforcement learning aware simulation-to-real. In *CVPR*, 2020.

[45] P. Sharma, D. Pathak, and A. Gupta. Third-person visual imitation learning via decoupled hierarchical controller. In *Advances in Neural Information Processing Systems*, pages 2597–2607, 2019.

[46] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell. Deep domain confusion: Maximizing for domain invariance. *arXiv preprint arXiv:1412.3474*, 2014.

[47] Y. Ganin and V. Lempitsky. Unsupervised domain adaptation by backpropagation. In *ICML*, 2015.

[48] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky. Domain-adversarial training of neural networks. *JMLR*, 17(1), 2016.

[49] F. Zhuang, X. Cheng, P. Luo, S. J. Pan, and Q. He. Supervised representation learning with double encoding-layer autoencoder for transfer learning. *ACM TIST*, 9(2), 2017.

[50] E. Tzeng, J. Hoffman, K. Saenko, and T. Darrell. Adversarial discriminative domain adaptation. In *CVPR*, 2017.

[51] E. Tzeng, C. Devin, J. Hoffman, C. Finn, P. Abbeel, S. Levine, K. Saenko, and T. Darrell. Adapting deep visuomotor representations with weak pairwise constraints. *arXiv preprint arXiv:1511.07111*, 2015.

[52] J. Roy and G. Konidaris. Visual transfer for reinforcement learning via wasserstein domain confusion. *arXiv preprint arXiv:2006.03465*, 2020.

[53] S. Choi, S. Han, W. Kim, and Y. Sung. Cross-domain imitation learning with a dual structure. *arXiv preprint arXiv:2006.01494*, 2020.

[54] K. Kim, Y. Gu, J. Song, S. Zhao, and S. Ermon. Domain adaptive imitation learning. *arXiv*, 2020.

[55] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[56] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

[57] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, and S. Levine. Soft actor-critic algorithms and applications. Technical report, 2018.

[58] A. Singh, L. Yang, K. Hartikainen, C. Finn, and S. Levine. End-to-end robotic reinforcement learning without reward engineering. *arXiv preprint arXiv:1904.07854*, 2019.

[59] T. Yu, D. Quillen, Z. He, R. Julian, K. Hausman, C. Finn, and S. Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning (CoRL)*, 2019. URL https://arxiv.org/abs/1910.10897.

[60] Y. Burda, H. Edwards, A. Storkey, and O. Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

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

A Data and Qualitative Results on Supplementary Website

Dataset downloads and videos of our qualitative results are hosted on our website: https://sites.google.com/view/rl-with-videos.

B COVID-19 Statement

Due to COVID-19, we were unable to evaluate RLV on a real robot. However, the experiments in the paper provide significant evidence that the algorithm would be successful on a real robot. First, while there will be domain shift between the human observations and the real robot, this domain shift will likely be no larger than the domain shift betwee real human videos and simulated robot observations in the experiments in Section 4.3. Second, the simulated results show good performance with raw visual observations and without the need for a shaped reward function, both of which make it practical to deploy to a real robot without significant instrumentation of the environment. Third, our results suggest that the simulated robot learns within 75,000 steps or 750 episodes, an amount of data that is practical to collect on a real robot. Finally, RLV builds upon the SAC algorithm, which has been previously demonstrated on a real robot with raw pixel observations [57, 58].

C Architecture Details and Hyperparameters

We instantiate our model with deep neural networks. We use SAC as the underlying RL algorithm in RLV. Hyperparameters for our different experiments are shown in Tables 1-5.

We used the default hyperparameters for SAC wherever possible, and performed a hyperparameter sweep to find the remaining parameters. The most sensitive hyperparameters are the ones in Table 5, which control the balance between domain invariance and the other losses for training the feature representation. The reward generation parameter, c_{large}, is robust, capable of using the same value in different environments with different reward scales. The other reward generation parameter, c_{small}, is less robust, but can simply be set to the reward for taking a random step at the starting state in each environment. The network architecture hyperparameters are shown in Tables 3 and 4; these hyperparameters have a range of viable values.

Name	Value
Learning Rate	3e-4
Num Initial Exploration Steps	1000
Batch Size	256
Optimizer	ADAM [61]
Nonlinearity	ReLU
Gradient Steps per environment step	1

Table 1: General Hyperparameters. These hyperparameters are used for RLV and for SAC in all environments.

Hyperparameter	Acrobot Environment	MuJoCo Environments
c_{large}	10.0	10.0
c_{small}	-1.0	0.0

Table 2: Reward generation parameters. These hyperparameters were used to generate the rewards for RLV in different environments. The MuJoCo environments include the State Pusher, Visual Pusher, Visual Door Opening, and the visual drawer opening environments.

For learning from observational data of humans, we augmented the images with random crops for both SAC and RLV. We pad the image on all sides with four black pixels, then randomly crop an image of the original size from the padded image.

While ILPO and BCO are designed for tasks with discrete actions, we were able to make them run on the robotic pushing task by discretizing the action space. We ran a hyperparamter sweep to determine the correct resolution of the discretization, selecting a discretization that broke the two-dimensional action space of the State Pusher environment into 49 discrete bins.

Name	Value
Q Function Fully Connected Layer Features	[256, 256]
Policy Network Fully Connected Layer Features	[256, 256]
Inverse Model Fully Connected Layer Features	[64, 64, 64]

Table 3: State Observation Architecture Hyperparameters. These hyperparameters are used when the agent receives state based observations (Acrobot-v0 and State Pusher).

Name	Value
Feature Extractor Conv Number of Filters	[16, 16, 32]
Feature Extractor Conv Filter Size	5
Input image shape	[48, 48, 3]
Pooling Type	MaxPool
Pooling Stride	2
Q Function Fully Connected Layer Features	[512, 256, 256]
Policy Network Fully Connected Layer Features	[512, 256, 256]
Inverse Model Fully Connected Layer Features	[64, 64, 64]

Table 4: Image Observation Architecture Hyperparameters. These hyperparameters are used when the agent receives image based observations (visual pusher, visual door opening, visual drawer opening).

D Domain adaptation without paired data

We additionally performed ablations over the effectiveness of our domain adaptation approach without paired data. These results are shown in Figure 9.

Without paired images RLV is sometimes able to learn a feature embedding that allows for transfer between the human and the robot environments. This allows some seeds of RLV to learn quickly, exceeding the performance of the existing RL algorithms. However, without paired images, RLV is sometimes unable to learn a good feature embedding, limiting its average asymptotic performance.

Having access to off-policy paired data ensures that RLV can always learn a good representation. This enables RLV to learn much faster and to have higher asymptotic performance that the default RL algorithm.

E Human Dataset

For our experiments in Section 4.3, we collected a dataset of a human pushing a puck and of a human opening a drawer. The pushing component of the dataset is composed of 198 videos, totaling 806 seconds. The data includes four backgrounds, two hands, two pucks, and a variety of lighting conditions. The drawer opening component of the dataset is composed of 108 videos, totaling 376 seconds. The data includes two drawers, one hand, and a variety of lighting conditions.

We additionally include paired images between random robot exploration and the human videos for both tasks. There are 314 paired images between random robot exploration and the human pushing data and 578 paired images between the random robot exploration and the human drawer opening data.

Name	Value
c_1	1
c_2	1
c_3	0.001
Discriminator Fully Connected Layer Features	[64, 64, 64]
Discriminator learning rate	3e-8
Paired data weight	1e-6

Table 5: Human Observation Hyperparameters. These hyperparameters are used for RLV learning from human demonstrations.

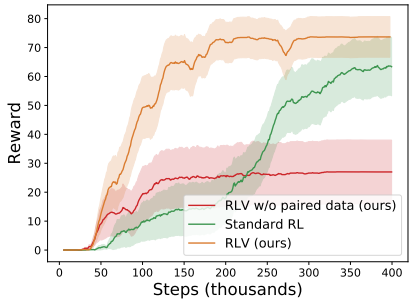


Figure 9: Training curves and example images for learning pushing with human observations. Without paired images, RLV is able to initially learn much faster than default RL, but fails to converge to as good of a solution. With paired images, RLV is able to learn much faster than standard RL, and is able to converge to a higher reward.

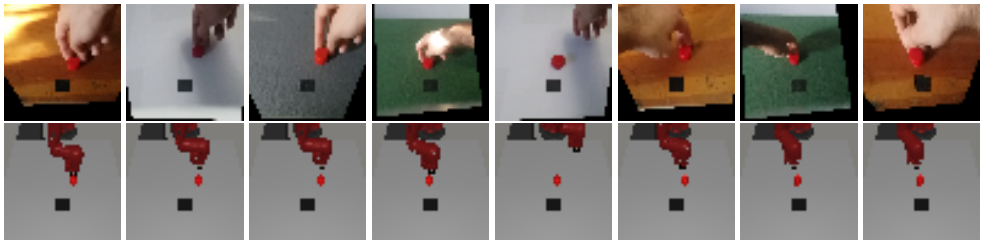


Figure 10: Example paired frames. The paired frames were found by selecting frames where the puck and manipulator were in similar locations between the robot and the human data. Due to the differences in the morphologies of the agents, there are numerous instances where the human hand is in contact with the puck, while the robot manipulator is not in contact with the puck.

In the pushing domain, the paired images were found by choosing images where the puck and the human hand appeared in in the same locations as the puck and the robot arm. Examples of these paired images are shown in Figure 10. Due to the differences in the shape and appearance of the two agents, there are numerous sets of paired frames where the human's hand is in contact with the puck while the robot's hand is not in contact. The difference in contact would cause the paired states to have significantly different dynamics. Fortunately, RLV is robust to these errors in pairing.

In the drawer opening data, the paired images were found by choosing images where the robot's end effector was located at the same image location as the human's hand. Due to the coloration of the simulated scene, we were able to automatically label the position of the robot, making finding the paired frames significantly easier. Examples of these paired frames are shown in Figure 11.

We post-processed the collected videos by transforming them to align the goal square in the pushing data and the drawer handle in the drawer opening data. The resulting images are then cropped to 48x48 pixels. Example sequences from the human pushing dataset are shown in Figure 12 and example post-processed images from the human pushing dataset are shown in Figure 13. Example sequences from the human drawer opening dataset are shown in Figure 14 and example post-processed images from the human drawer opening dataset are shown in Figure 15. The full dataset

Figure 11: Example paired frames for the drawer opening task. None of the paired frames included the either the human or the robot making contact with the drawer.

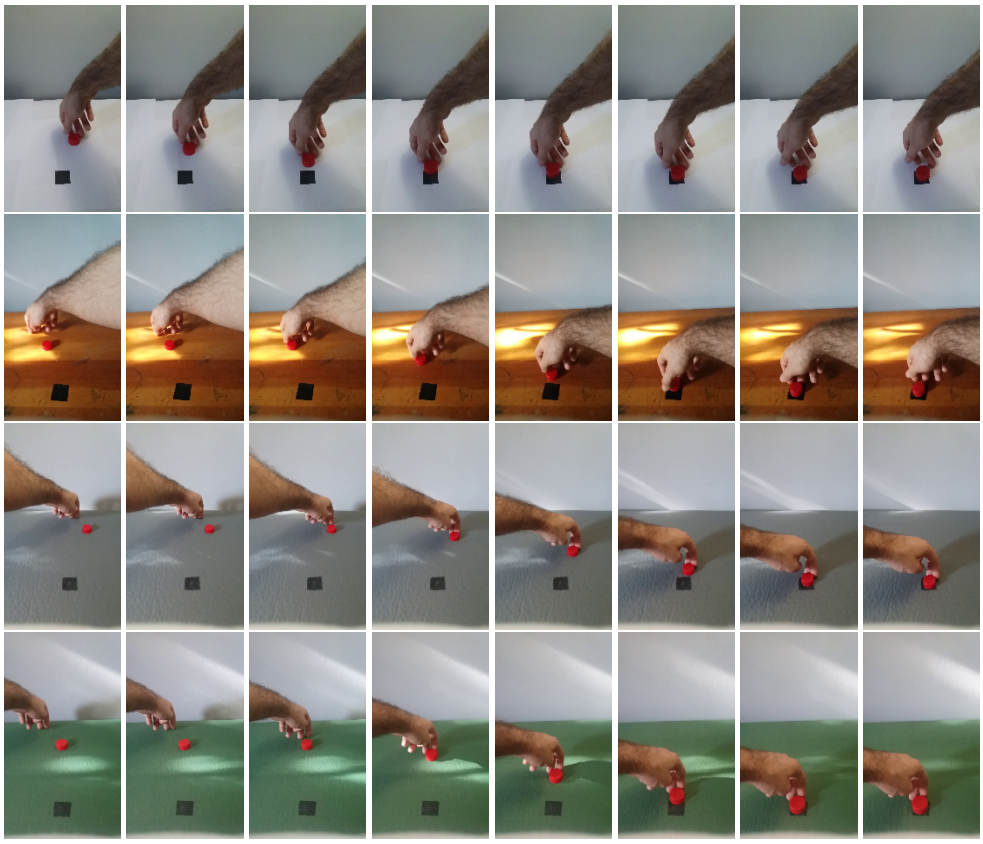


Figure 12: Example images from the human pushing observation data used by RLV.

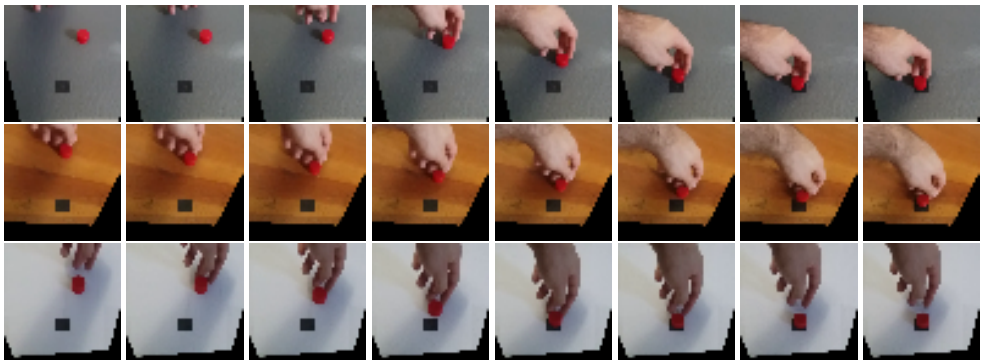


Figure 13: Example post-processed images from the human observation data used by RLV.

is available for download on the supplementary website: https://sites.google.com/view/rl-with-videos.

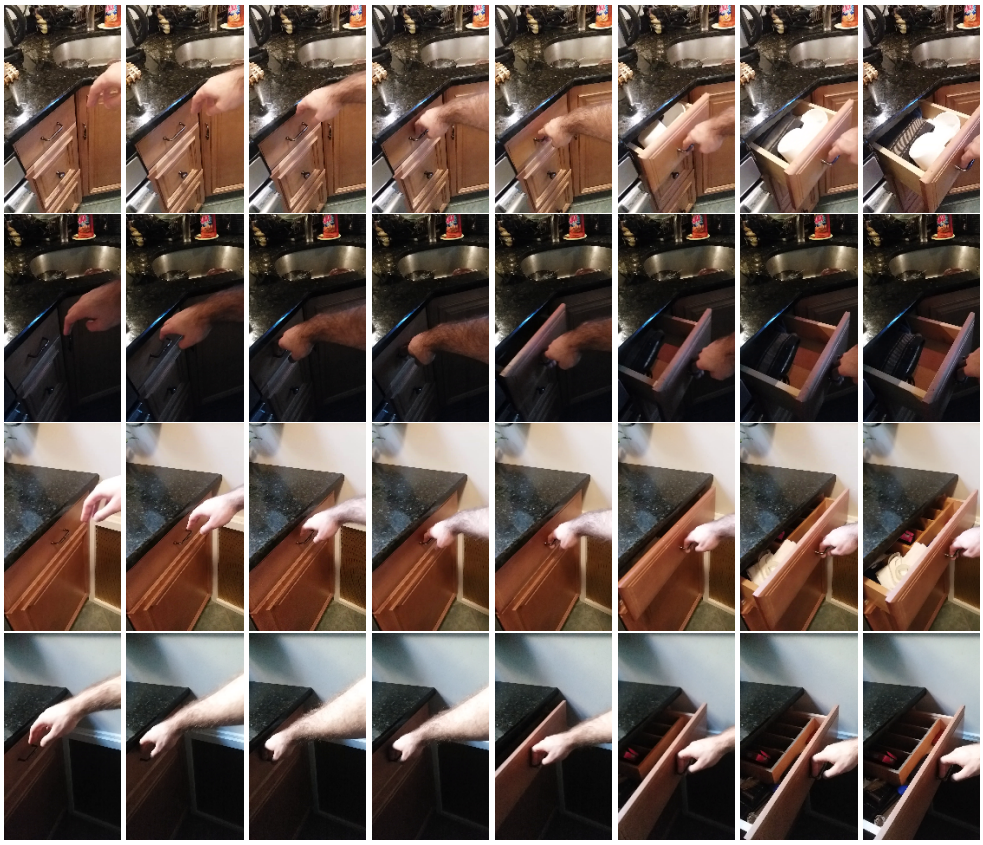


Figure 14: Example images from the human drawer opening observation data used by RLV.

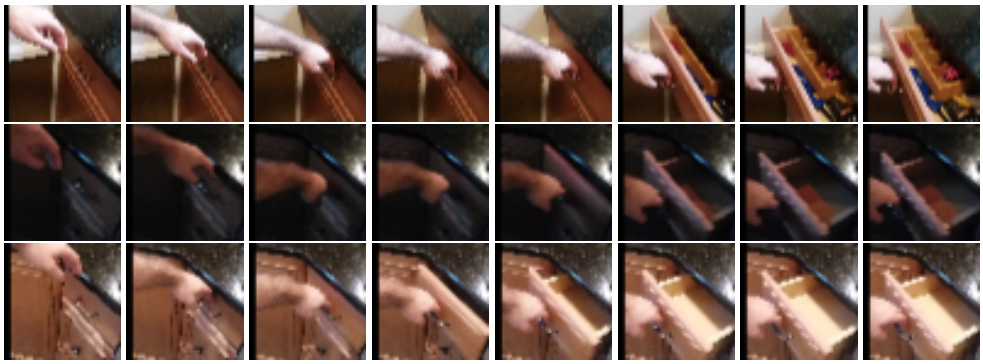


Figure 15: Example post-processed images from the human drawer opening observation data used by RLV.