# Graph Inverse Reinforcement Learning from Diverse Videos

**Sateesh Kumar** **Jonathan Zamora** **Nicklas Hansen**
**Rishabh Jangir** **Xiaolong Wang**
UC San Diego

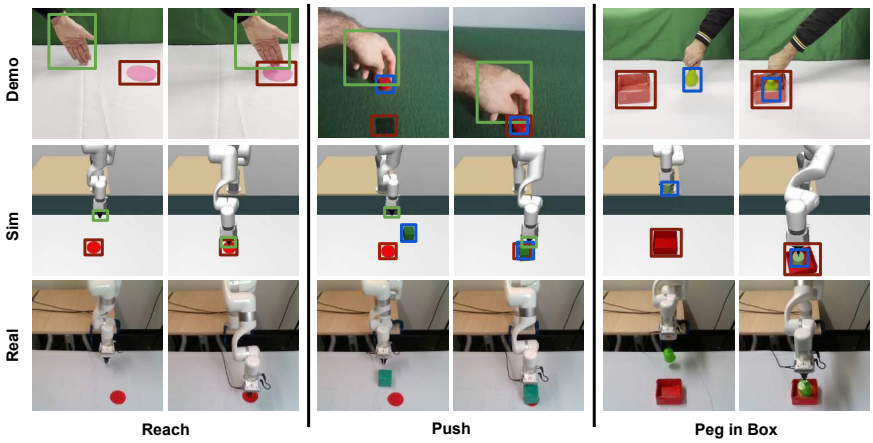


Figure 1: **GraphIRL.** We propose an approach for performing inverse reinforcement learning from diverse third-person videos via graph abstraction. Based on our learned reward functions, we successfully train image-based policies in simulation and deploy them on a real robot.

## Abstract

Research on Inverse Reinforcement Learning (IRL) from third-person videos has shown encouraging results on removing the need for manual reward design for robotic tasks. However, most prior works are still limited by training from a relatively restricted domain of videos. In this paper, we argue that the true potential of third-person IRL lies in increasing the diversity of videos for better scaling. To learn a reward function from diverse videos, we propose to perform graph abstraction on the videos followed by temporal matching in the graph space to measure the task progress. Our insight is that a task can be described by entity interactions that form a graph, and this graph abstraction can help remove irrelevant information such as textures, resulting in more robust reward functions. We evaluate our approach, *GraphIRL*, on cross-embodiment learning in X-MAGICAL and learning from human demonstrations for real-robot manipulation. We show significant improvements in robustness to diverse video demonstrations over previous approaches, and even achieve better results than manual reward design on a real robot pushing task. Videos are available at `https://graphirl.github.io/`.

## 1 Introduction

Deep Reinforcement Learning (RL) is a powerful general-purpose framework for learning behavior policies from high-dimensional interaction data, and has led to a multitude of impressive feats in application areas such as game-playing [29] and robotics [24, 2]. Through interaction with an

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

unknown environment, RL agents iteratively improve their policy by learning to maximize a reward signal, which has the potential to be used in lieu of hand-crafted control policies. However, the performance of policies learned by RL is found to be highly dependent on the careful specification of task-specific reward functions and, as a result, crafting a good reward function may require significant domain knowledge and technical expertise.

As an alternative to manual design of reward functions, *inverse RL* (IRL) has emerged as a promising paradigm for policy learning. By framing the reward specification as a learning problem, operators can specify a reward function based on video examples. While *imitation learning* typically requires demonstrations from a first-person perspective, IRL can in principle learn a reward function, *i.e.*, a measure of task progression, from *any* perspective, including third-person videos of humans performing a task. This has positive implications for data collection, since it is often far easier for humans to capture demonstrations in third-person.

Although IRL from third-person videos is appealing because of its perceived flexibility, learning a good reward function from raw video data comes with a variety of challenges. This is perhaps unsurprising, considering the visual and functional diversity that such data contains. For example, the task of pushing an object across a table may require different motions depending on the embodiment of the agent. A recent method for cross-embodiment IRL, dubbed XIRL [56], learns to capture task progression from videos in a self-supervised manner by enforcing temporal cycle-consistency constraints. While XIRL can in principle consume any video demonstration, we observe that its ability to learn task progression degrades substantially when the visual appearance of the video demonstrations do not match that of the target environment for RL. Therefore, it is natural to ask the question: *can we learn to imitate others from (a limited number of) diverse third-person videos?*

In this work, we demonstrate that it is indeed possible. Our key insight is that, while videos may be of great visual diversity, their underlying scene structure and agent-object interactions can be abstracted via a graph representation. Specifically, instead of directly using images, we extract object bounding boxes from each frame using an off-the-shelf detector, and construct a graph abstraction where each object is represented as a node in the graph. Often – in robotics tasks – the spatial location of an object by itself may not convey the full picture of the task at hand. For instance, to understand a task like *Peg in Box* (shown in Figure 1), we need to also take into account how the agent *interacts* with the object. Therefore, we propose to employ *Interaction Networks* [7] on our graph representation to explicitly model interactions between entities. To train our model, we follow [56, 9] and apply a temporal cycle consistency loss, which (in our framework) yields task-specific yet embodiment- and domain-agnostic feature representations.

We validate our method empirically on a set of simulated cross-domain cross-embodiment tasks from X-MAGICAL [56], as well as three vision-based robotic manipulation tasks. To do so, we collect a diverse set of demonstrations that vary in visual appearance, embodiment, object categories, and scene configuration; X-MAGICAL demonstrations are collected in simulation, whereas our manipulation demonstrations consist of real-world videos of humans performing tasks. We find our method to outperform a set of strong baselines when learning from visually diverse demonstrations, while simultaneously matching their performance in absence of diversity. Further, we demonstrate that vision-based policies trained with our learned reward perform tasks with greater precision than human-designed reward functions, and successfully transfer to a real robot setup with only approximate correspondence to the simulation environment. Thus, our proposed framework completes the cycle of learning rewards from real-world human demonstrations, learning a policy in simulation using learned rewards, and finally deployment of the learned policy on physical hardware.

## 2 Related Work

**Learning from demonstration.** Conventional imitation learning methods require access to expert demonstrations comprised of observations and corresponding ground-truth actions for every time step [32, 5, 3, 36], for which kinesthetic teaching or teleoperation are the primary modes of data collection in robotics. To scale up learning, video demonstrations are recorded with human operating the same gripper that the robot used, which also allows direct behaviro cloning [43, 55]. More recently, researchers have developed methods that instead infer actions from data via a learned forward [31] or inverse [45, 35] dynamics model. However, this approach still makes the implicit assumption that imitator and demonstrator share a common observation and action space, and are therefore not directly applicable to the cross-domain cross-embodiment problem setting that we consider.

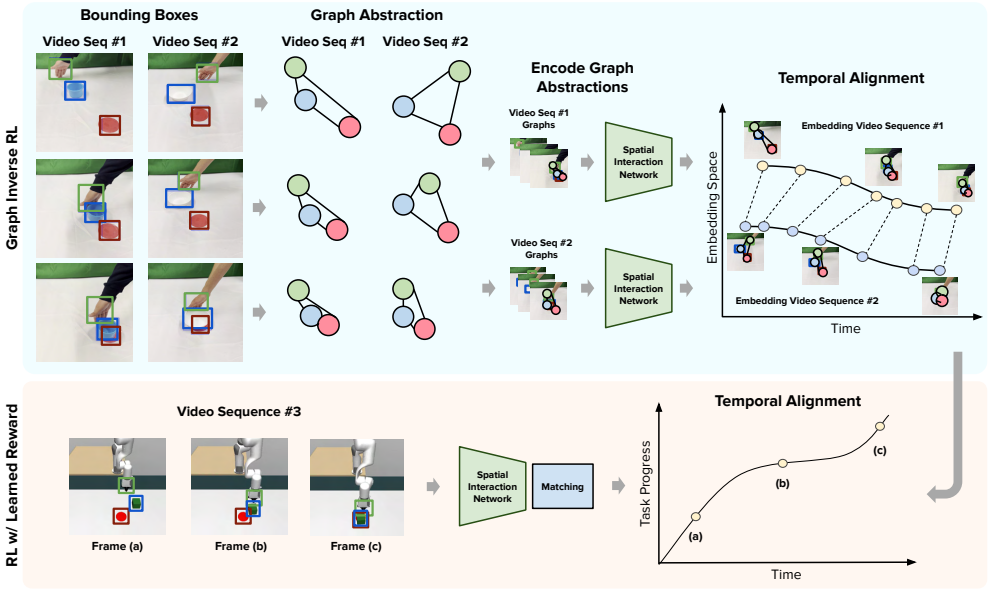


Figure 2: **Overview.** We extract object bounding boxes from video sequences using an off-the-shelf detector, and construct a graph abstraction of the scene. We model graph-abstracted object interactions using *Interaction Networks* [7], and learn a reward function by aligning video embeddings temporally. We then train image-based RL policies using our learned reward function, and deploy on a real robot.

**Inverse RL.** To address the aforementioned limitations, inverse RL has been proposed [30, 1, 19, 12, 6, 46] and it has recently emerged as a promising paradigm for cross-embodiment imitation in particular [39, 21, 49, 23, 8, 11, 34, 56, 4]. For example, Schmeckpeper et al. [39] proposes a method for integrating video demonstrations without corresponding actions into off-policy RL algorithms via a latent inverse dynamics model and heuristic reward assignment, and Zakka et al. [56] (XIRL) learns a reward function from video demonstrations using temporal cycle-consistency and trains an RL agent to maximize the learned rewards. In practice, however, inverse RL methods such as XIRL are found to require limited visual diversity in demonstrations. Our work extends XIRL to the setting of diverse videos by introducing a graph abstraction that models agent-object and object-object interactions while still enforcing temporal cycle-consistency.

**Object-centric representations.** have been proposed in many forms at the intersection of computer vision and robotics. For example, object-centric scene graphs can be constructed for integrated task and motion planning [10, 44, 57], navigation [14, 51], relational inference [50, 25],dynamics modeling [7, 48, 28, 53, 33**?** ], model predictive control [38, 26, 54], grasping [**?** **?** ] or visual imitation learning [**?** **?** 42]. Similar to our work, Sieb et al. [42] propose to abstract video demonstrations as object-centric graphs for the problem of single-video cross-embodiment imitation, and act by minimizing the difference between the demonstration graph and a graph constructed from observations captured at each step. As such, their method is limited to same-domain visual trajectory following, whereas we learn a general alignment function for cross-domain cross-embodiment imitation and leverage *Interaction Networks* [7] for modeling graph-abstracted spatial interactions rather than relying on heuristics.

## 3 Our Approach

In this section, we describe our main contribution, which is a self-supervised method for learning a reward function directly from a set of diverse third-person video demonstrations by applying temporal matching on graph abstractions. Our Graph Inverse Reinforcement Learning (GraphIRL) framework, shown in Figure 2, consists of building an object-centric graph abstraction of the video demonstrations and then learn an embedding space that captures task progression by exploiting the temporal cue in the videos. This embedding space is then used to construct a *domain invariant* and *embodiment invariant* reward function which can be used to train any standard reinforcement learning algorithm.

**Problem Formulation.** Given a task $T$, our approach takes a dataset of video demonstrations $D = \{V_1, V_2, \ldots, V_n\}$. Each video consists of image frames $\{I_1^i, I_2^i, \ldots, I_k^i\}$ where $i$ denotes the video frame index and $k$ denotes the total number of frames in $V_i$. Given $D$, our goal is to learn a reward function that can be used to solve the task $T$ for any robotic environment. Notably, we do *not* assume access to any action information of the expert demonstrations, and our approach does *not* require objects or embodiments in the target environment to share appearance with demonstrations.

### 3.1 Representation Learning

To learn task-specific representations in a self-supervised manner, we take inspiration from Dwibedi et al. [9] and employ a temporal cycle consistency loss. However, instead of directly using images, we propose a novel object-centric graph representation, which allows us to learn an embedding space that not only captures task-specific features, but depends *solely* on the spatial configuration of objects and their interactions. We here detail each component of our approach to representation learning.

**Object-Centric Representation.** Given video frames $\{I_1^i, I_2^i, \ldots, I_k^i\}$, we first extract object bounding boxes from each frame using an off-the-shelf detector. Given $N$ bounding boxes for an image, we represent each bounding box as a $4 + m$ dimensional vector $o_j = \{x_1, y_1, x_2, y_2, d_1, d_2, \ldots, d_m\}$, where the first $4$ dimensions represent the leftmost and rightmost corners of the bounding box, and the remaining $m$ dimensions encode distances between the centroids of the objects. For each frame $I_j^i$ we extract an object-centric representation $I_j'^i = \{o_1, o_2, \ldots, o_m\}$ such that we can represent our dataset of demonstrations as $D' = \{V_1', V_2', \ldots, V_n'\}$ where $V_i'$ is the sequence of bounding boxes corresponding to video $V_i$. Subsequent sections describe how we learn representations given $D'$.

**Spatial Interaction Encoder.** Taking inspiration from recent approaches on modeling physical object dynamics [7, 48], we propose a *Spatial Interaction Encoder Network* to explicitly model object-object interactions. Specifically, given a sequence $V_i'$ from $D'$, we model each element $I'$ as a graph, $G = (O, R)$, where $O$ is the set of objects $\{o_1, o_2, \ldots, o_m\}$, $m$ is the total number of objects in $I'$, and $R$ denotes the relationship between objects (*i.e.*, whether two objects interact with each other). For simplicity, all objects are connected with all other objects in the graph such that $R = \{(i, j) \mid i \neq j \wedge i \leq m \wedge j \leq m\}$. We use a fully-connected graph because this makes the least assumption about the problem and task specific object interaction structure could then be learned directly from the data. We compose an object embedding for each of $o_i \in O$ by combining *self* and *interactional* representations as follows:

$$f_o(o_i) = \phi_{\text{agg}}(f_{\text{s}} + f_{\text{in}}) \quad \text{with} \quad f_s(o_i) = \phi_s(o)\,, \quad f_{\text{in}}(o_i) = \sum_{j=1}^{m} \phi_{\text{in}}((o_i, o_j)) \mid (i, j) \in \mathbb{R}\,, \quad (1)$$

where $f_s(o_i)$ represents the *self* or independent representation of an object, $f_{\text{in}}$ represents the *interactional* representation, *i.e.*, how it interacts with other objects in the scene, $f_o$ is the final object embedding, and $(,)$ represents concatenation. Here, the encoders $\phi_{\text{s}}$, $\phi_{\text{in}}$ and $\phi_{\text{agg}}$ denote Multi layer Perceptron (MLP) networks respectively. We emphasize that the expression for $f_{\text{in}}(\cdot)$ implies that the object embedding $f_o(.)$ depends on *all* other objects in the scene; this term allows us to model relationships of an object with the others. The final output from the spatial interaction encoder $\psi(\cdot)$ for object representation $I'$ is the mean of all object encodings:

$$\psi(I') = \frac{1}{m} \sum_{i}^{m} f(o_i)\,. \quad (2)$$

The spatial interaction encoder is then optimized using the temporal alignment loss introduced next.

**Temporal Alignment Loss**. Taking inspiration from prior works on video representation learning [9, 18, 27, 47, 16], we employ the task of temporal alignment for learning task-specific representations. Given a pair of videos, the task of self-supervised alignment implicitly assumes that there exists true semantic correspondence between the two sequences, *i.e.*, both videos share a common semantic space. These works have shown that optimizing for alignment leads to representations that could be used for tasks that require understanding task progression such as action-classification. This is because in order to solve for alignment, a learning model has to learn features that are (1) common across most videos and (2) exhibit temporal ordering. For a sufficiently large dataset with single task, the most common visual features would be distinct phases of a task that appear in all videos and if the task has small permutations, these distinct features would also exhibit temporal order. In such

scenarios, the representations learned by optimizing for alignment are *task-specific* and invariant to changes in viewpoints, appearances and actor embodiments.

In this work, we employ Temporal Cycle Consistency (TCC) [9] loss to learn temporal alignment. TCC optimizes for alignment by learning an embedding space that maximizes one-to-one nearest neighbour mappings between sequences. This is achieved through a loss that maximizes for cycle-consistent nearest neighbours given a pair of video sequences. In our case, the cycle consistency is applied on the *graph abstraction* instead of image features as done in the aforementioned video alignment methods. Specifically, given $D'$, we sample a pair of bounding box sequences $V'_i = \{I'^i_1, \ldots, I'^i_{m_i}\}$ and $V'_j = \{I'j_1, \ldots, I'^j_{m_j}\}$ and extract embeddings by applying the spatial interaction encoder defined in Equation 2. Thus, we obtain the encoded features $S_i = \{\psi(I'^i_1), \ldots, \psi(I'^i_{m_i})\}$ and $S_j = \{\psi(I'^j_1), \ldots, \psi(I'^j_{m_j})\}$. For the $n$th element in $S_i$, we first compute its nearest neighbour, $\upsilon^n_{ij}$, in $S_j$ and then compute the probability that it cycles-back to the $k$th frame in $S_i$ as:

$$\beta^k_{ijn} = \frac{e^{-||\upsilon^n_{ij} - S^k_i||^2}}{\sum_k^{m_j} e^{-||\upsilon^n_{ij} - S^k_i||^2}}\,, \upsilon^n_{ij} = \sum_k^{m_j} \alpha_k s^k_j\,, \alpha_k = \frac{e^{-||S^n_i - S^k_j||^2}}{\sum_k^{m_j} e^{-||S^n_i - S^k_j||^2}}\,. \tag{3}$$

The cycle consistency loss for $n$th element can be computed as $L^{ij}_n = (\mu^n_{ij} - n)^2$, where $\mu^n_{ij} = \sum_k^{mi} \beta^k_{ijn} k$ is the expected value of frame index $n$ as we cycle back. The overall TCC loss is then defined by summing over all pairs of sequence embeddings $(S_i, S_j)$ in the data, *i.e.*, $L^n_{ij} = \sum_{ijn} L^n_{ij}$.

### 3.2 Reinforcement Learning

We learn a task-specific embedding space by optimizing for temporal alignment. In this section, we define how to go from this embedding space to a reward function that measures task progression. For constructing the reward function, we leverage the insight from Zakka et al. [56] that in a task-specific embedding space, we can use euclidean distance as a notion of task progression, *i.e.*, frames far apart in the embedding space will be far apart in terms of task progression and vice versa. We therefore choose to define our reward function as

$$r(o) = -\frac{1}{c}||\psi(o) - g||^2\,, \quad \text{with} \quad g = \sum_{i=1}^{n} \psi(I'^i_{m_i})\,, \tag{4}$$

where $o$ is the current observation, $\psi$ is the Spatial Interaction Encoder Network from Section 3, $g$ is the representative goal frame, $m_i$ is the length of sequence $V'^i$ and $c$ is a scaling factor. The scaling factor $c$ is computed as the average distance between the first and final observation of all the training videos in the learned embedding space. Note, that the range of the learned reward is $(-\infty, 0]$. Defining the reward function in this way gives us a dense reward because as the observed state gets closer and closer to the goal, the reward starts going down and approaches zero when the goal and current observation are close in embedding space. After constructing the learned reward, we can use it to train any standard RL algorithm. We note that, unlike previous approaches [39, 56], our method does not use *any* environment reward to improve performance, and instead relies *solely* on the learned reward, which our experiments demonstrate is sufficient for solving diverse robotic manipulation tasks.

## 4 Experiments

In this section, we demonstrate how our approach uses diverse video demonstrations to learn a reward function that generalizes to unseen domains. In particular, we are interested in answering the questions: (1) How do vision-based methods for IRL perform when learning from demonstrations that exhibit *domain shift*? and (2) is our approach capable of learning a stronger reward signal under this challenging setting? To that end, we first conduct experiments X-MAGICAL benchmark [56]. We then evaluate our approach on multiple robot manipulation tasks using a diverse set of demonstrations.

**Implementation Details.** All MLPs defined in Equation 2 have 2 layers followed by a ReLU activation, and the embedding layer outputs features of size 128 in all experiments. For training, we use ADAM [22] optimizer with a learning rate of $10^{-5}$. We use Soft Actor-Critic (SAC) [15] as backbone RL algorithm for all methods. For experiments on X-MAGICAL, we follow Zakka et al.

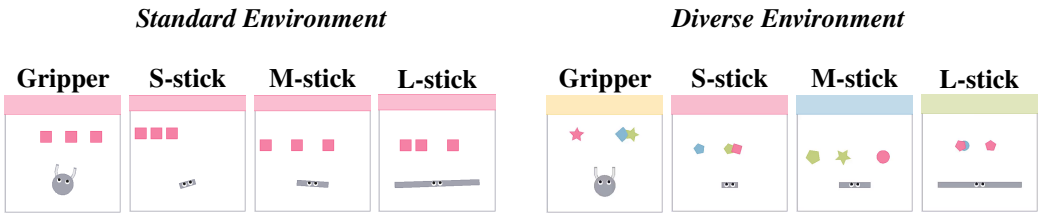


Figure 3: **Overview of X-MAGICAL task variants.** We consider two environment variants and four embodiments for our simulated sweeping task experiments. Our work assesses the performance of IRL algorithms in both the *Diverse* and *Standard* environments across all four embodiments in the *Same-Embodiment* and *Cross-Embodiment settings*.

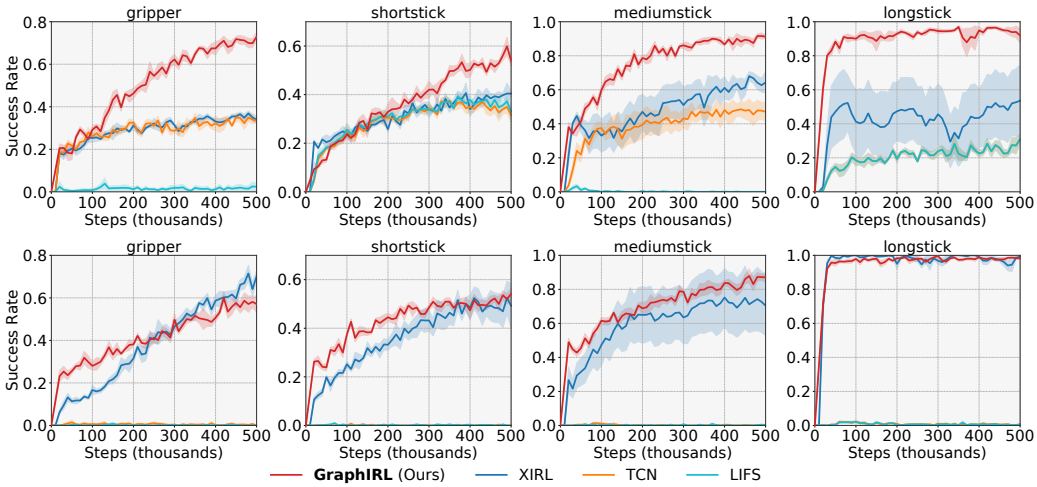


Figure 4: **Cross-Embodiment Cross-Environment.** Success rates of our method *GraphIRL* and baselines on *(top)* Standard Environment Pretraining $\rightarrow$ Diverse Environment RL and *(bottom)* Diverse Environment Pretraining $\rightarrow$ Standard Environment RL. All reported numbers are averaged over 5 seeds. Our approach performs favorably when compared to other baselines on both settings.

[56] and learn a state-based policy; RL training is performed for 500k steps for all embodiments. For robotic manipulation experiments, we learn a multi-view image-based SAC policy [20]. We train RL agent for 300k, 800k and 700k steps for *Reach*, *Push* and *Peg in Box* respectively. For fair comparison, we only change the learned reward function across methods and keep the RL setup identical. The success rates presented for all our experiments are averaged over 50 episodes. Refer to Appendix B for further implementation details.

**Baselines.** We compare against multiple vision-based approaches that learn rewards in a self-supervised manner: **(1) XIRL** [56] that learns a reward function by applying the TCC [9] loss on demonstration video sequences, **(2) TCN** [40] which is a self-supervised contrastive method for video representation learning that optimizes for temporally disentangled representations, and **(3) LIFS** [13] that learns an invariant feature space using a dynamic time warping-based contrastive loss. Lastly, we also compare against the manually designed **(4) Environment Rewards** from Jangir et al. [20]. The environment reward baseline is an oracle method since it is a dense reward and is carefully designed for the task under consideration. For vision-based baselines, we use a ResNet-18 encoder pretrained on ImageNet [37] classification. We use the hyperparameters, data augmentation schemes and network architectures provided in Zakka et al. [56] for all vision-based baselines. Please refer to Appendix E.1 for description of environment rewards and Zakka et al. [56] for details on the vision-based baselines.

### 4.1 Experimental Setup

We conduct experiments under two settings: the *Sweep-to-Goal* task from X-MAGICAL [56], and robotic manipulation tasks with an xArm robot both in simulation and on a real robot setup. We describe our experimental setup under these two settings in the following.

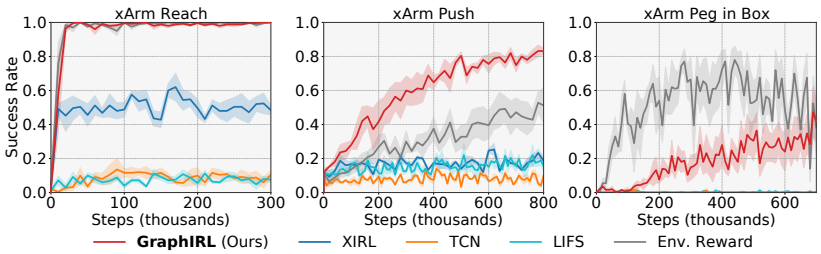


Figure 5: **Robotic Manipulation.** Success rates of our method *GraphIRL* and baselines on the tasks of *Reach*, *Push* and *Peg in Box*. All results are averaged over 5 seeds. We observe significant gains in performance specially over vision-based baselines due to large *domain-gap*

**X-MAGICAL.** We choose to extend X-MAGICAL [56], a 2D simulation environment for cross-embodiment imitation learning. On this benchmark, we consider a multi-object sweeping task, where the agent must push three objects towards a static goal region. We utilize two variants of the X-MAGICAL benchmark, which we denote as ***Standard*** (original) and ***Diverse*** (ours) environments, shown in Figure 3. ***Standard*** only randomizes the position of objects, whereas ***Diverse*** also randomizes visual appearance. We consider a set of four unique embodiments {*gripper, short-stick, medium-stick, long-stick*}. In particular, we conduct experiments in the *cross-environment* and *cross-embodiment setting* where we learn a reward function in the ***Standard*** environment on 3 held-out embodiments and do RL in the ***Diverse*** environment on 1 target embodiment, or vice-versa. This provides an additional layer of difficulty for the RL agent as visual randomizations show the brittleness of vision-based IRL methods. Refer to Appendix C for more details on performed randomizations.

**Robotic Manipulation.** Figure 1 shows initial and success configurations for each of the three task that we consider: **(1) Reach** in which the agent needs to reach a goal (red disc) with its end-effector, **(2) Push** in which the goal is to push a cube to a goal position, and **(3) Peg in Box** where the goal is to put a peg tied to the robot's end-effector inside a box. The last task is particularly difficult because it requires geometric 3D understanding of the objects. Further, a very specific trajectory is required to avoid collision with the box and complete the task. We collect a total of 256 and 162 video demonstrations for *Reach* and *Peg in Box*, respectively, and use 198 videos provided from Schmeckpeper et al. [39] for *Push*. The videos consist of human actors performing the same tasks but with a number of diverse objects and goal markers, as well as varied positions of objects. Unlike the data collected by Schmeckpeper et al. [39], we do not fix the goal position in our demonstrations. In order to detect objects in our training demonstrations, we use a trained model from Shan et al. [41]. The model is trained on a large-scale dataset collected from YouTube and can detect hands and objects in an image.; refer to Appendix E.2 for more details on data collection. Additionally, we do not require the demonstrations to resemble the robotic environment in terms of appearance or distribution of goal location. We use an xArm robot as our robot platform and capture image observations using a static third-person RGB camera in our real setup; details in Appendix G.

### 4.2 Results

**X-MAGICAL.** Results for the *cross-embodiment and cross-environment* setting are shown in Figure 4. When trained on *Standard*, our method performs significantly better than vision-based baselines (*e.g.*, 0.58 GraphIRL for gripper vs 0.35 for XIRL and 0.99 GraphIRL for longstick vs 0.56 XIRL). We conjecture that vision-based baselines struggle with visual variations in the environment, which our method is unaffected by due to its graph abstraction. Additionally, when trained on *diverse* environment, GraphIRL outperforms 3 out of 4 embodiments.

**Robotic manipulation in simulation.** In this section, we answer the core question of our work: *can we learn to imitate others from diverse third-person videos?* In particular, we collect human demonstrations for manipulation tasks as explained in Section 4.1 and learn a reward function as explained in Section 3. This is a challenging setting because as shown in Figure 1, the collected data and robotic environments belong to different domains and do not share any appearance characteristics. Further, unlike previous works [39, 56], we do not use any environment reward as an additional supervision to the reinforcement learning agent. Figure 5 presents our results. For the **Reach** task, GraphIRL and environment reward are able to achieve a success rate of 1.0, while other baseline

| *Real* | XIRL | Env. Reward | GraphIRL (Ours) |
|---|---|---|---|
| Push | 0.27 | 0.47 | **0.60** |
| Reach | 0.26 | **0.93** | 0.86 |
| Peg in Box | 0.06 | **0.60** | 0.53 |

Table 1: **Real robot experiments.** Success rate on robot manipulation tasks on physical hardware. We evaluate each method for 15 trials using a fixed set of goal and start state configurations. Best results are in **bold**.

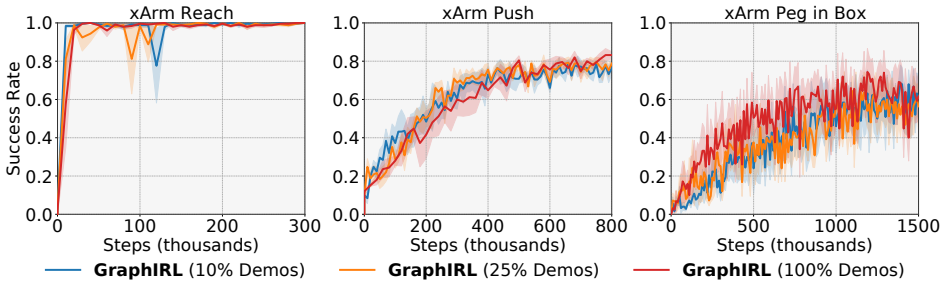


Figure 6: Impact of reducing number of demonstrations. Results averaged over 5 seeds. For experiments with 100% demonstrations, we use 256, 198 and 162 videos for *Reach*, *Push* and *Peg in Box* respectively.

methods are substantially behind GraphIRL (e.g. 0.477 XIRL and 0.155 TCN). The poor performance of vision-based baselines could be attributed to substantial visual domain shift. Due to domain shift, the learned rewards for these baselines produce low rewards for successful episodes, please refer to Appendix A for a more detailed qualitative analysis. In the **Push** setting, we find that vision-based baseline methods still perform poorly. Similar to **Reach**, XIRL performs the best out of the vision-based baselines with a success rate of 0.187, and GraphIRL performs better than environment reward (e.g. 0.832 GraphIRL vs 0.512 Environment Reward). We find that for *Push*, the learned reward gives a much higher reward when the object gets very close to the goal as compared to the environment reward which is linearly proportional to negative distance between goal and object therefore the RL agent ends up learning a more precise policy when trained with the learned reward. This result shows clear advantage of our method as we are able to outperform a hand-designed reward function without using any task specific information. The **Peg in Box** task is rigorous to solve since it requires 3-d reasoning and a precise reward function. Here, while all vision-based methods fail, our GraphIRL method is able to solve the task with a success rate comparable to that achieved with the hand-designed environment reward. Overall, our GraphIRL method is able to solve 2D and 3D reasoning tasks with a real-robot without a hand-designed reward function or access to 3D scene information.

**Real robot experiments.** Finally, we deploy the learned policies on a real robot. For each experiment, we conduct 15 trials per method and report the average success rate. Results are shown in Table 1. Interestingly, we find that GraphIRL outperforms XIRL in all three tasks on the real robot setup (e.g. 0.26 XIRL vs 0.86 GraphIRL on *Reach* and 0.27 XIRL vs 0.60 GraphIRL on *Push*), and on *Push*, GraphIRL performs better than the environment reward specifically designed for the task (e.g. 0.47 Environment Reward vs 0.6 GraphIRL) which is in line with our findings in simulation.

### 4.3 Ablations

**Impact of Modelling Spatial Interactions.** We study the impact of modeling object-object spatial interactions using Spatial Interaction Encoder Network described in Section 3.1. Specifically, we replace our proposed encoder component with an Multi-Layer Perceptron (MLP) by concatenating representations of all objects into a single vector and then feeding it to a 3-layer MLP network. We denote this approach as *MLP* and the Spatial Interaction Encoder Network based approach

| Variant | Success Rate |
|---|---|
| MLP | $0.61_{\pm 0.116}$ |
| IN | $0.804_{\pm 0.054}$ |

Table 2: Impact of modelling object-object interaction on *Push* task. **MLP:** Multi-layer perceptron and **IN:** Spatial Interaction Network Encoder. Results over 5 seeds.

as *IN*. Results in Table 2. We observe that modeling object interactions leads to a $20\%$ improvement in the RL success rate (i.e. $0.61$ for MLP vs $0.804$ for Interaction Networks).

**Impact of Decreasing Number of Demonstration Videos.** Results in Figure 6. We find that our approach is very data efficient and is capable of learning meaningful rewards even from a small number of videos. It achieves decent RL success rate with only $10\%$ of total videos used (e.g. $72\%$ for Push and $57.83\%$ for Peg in Box). However, adding more video demonstrations generally leads to improved performance as expected.

## 5 Conclusions and Limitations

**Conclusion.** We demonstrate the effectiveness of our proposed method, *GraphIRL*, in a number of IRL settings with diverse third-person demonstrations. In particular, we show that our method successfully learns reward functions from human demonstrations with diverse objects and scene configurations, that we are able to train image-based policies in simulation using our learned rewards, and that policies trained with our learned rewards are more successful than both prior work and manually designed reward functions on a real robot. We also hereby commit to release our complete code and data to the public.

**Limitations.** while our method relaxes the requirements for human demonstrations, collecting the demonstrations still requires human labor; and although our results indicate that we can learn from relatively few videos, eliminating human labor entirely remains an open problem. Moreover, our approach assumes access to object bounding boxes. This implies that our method's performance is dependent on the performance of the object detector. Fortunately, 2d object detectors have become very reliable as we show in our experiments, we are able to use an off-the-shelf object detector to extract the bounding boxes without having to perform any manual labeling. Furthermore, the proposed graph abstraction allows us to solve tasks despite the large domain gap but it has some potential disadvantages too. In particular, we lose fine grained information such as object poses and precise object interactions which could be useful for complex tasks such as medical procedures. However, we conjecture that 2D images might also not be sufficient for such a task since inferring accurate object pose, 3D geometric information from a 2D image itself is a challenging problem. Finally, there are some forms of reward that cannot be learned using our approach. Specifically, we may not be able to learn rewards where a penalty is imposed for certain actions or rewards for tasks where the progress towards task completion cannot be defined at all by object configuration.

## References


[1] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *ICML*, 2004.

[2] OpenAI Marcin Andrychowicz, Bowen Baker, Maciek Chociej, R. Józefowicz, Bob McGrew, Jakub W. Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, Jonas Schneider, S. Sidor, Joshua Tobin, P. Welinder, Lilian Weng, and Wojciech Zaremba. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39:20 – 3, 2020.

[3] Brenna Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57:469–483, 05 2009.

[4] Sridhar Pandian Arunachalam, Sneha Silwal, Ben Evans, and Lerrel Pinto. Dexterous imitation made easy: A learning-based framework for efficient dexterous manipulation. *arXiv preprint arXiv:2203.13251*, 2022.

[5] Christopher G. Atkeson and Stefan Schaal. Robot learning from demonstration. In *ICML*, 1997.

[6] Yusuf Aytar, Tobias Pfaff, David Budden, Thomas Paine, Ziyu Wang, and Nando de Freitas. Playing hard exploration games by watching youtube. In *NIPS*, 2018.

[7] Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, et al. Interaction networks for learning about objects, relations and physics. *Advances in neural information processing systems*, 29, 2016.

[8] Annie S Chen, Suraj Nair, and Chelsea Finn. Learning generalizable robotic reward functions from" in-the-wild" human videos. *arXiv preprint arXiv:2103.16817*, 2021.

[9] Debidatta Dwibedi, Yusuf Aytar, Jonathan Tompson, Pierre Sermanet, and Andrew Zisserman. Temporal cycle-consistency learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1801–1810, 2019.

[10] Georgios Fainekos, Antoine Girard, Hadas Kress-Gazit, and George J. Pappas. Temporal logic motion planning for dynamic robots. *Autom.*, 45:343–352, 2009.

[11] Arnaud Fickinger, Samuel Cohen, Stuart Russell, and Brandon Amos. Cross-domain imitation learning via optimal transport. *arXiv preprint arXiv:2110.03684*, 2021.

[12] Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adversarial inverse reinforcement learning. *arXiv*, 2017.

[13] Abhishek Gupta, Coline Devin, YuXuan Liu, Pieter Abbeel, and Sergey Levine. Learning invariant feature spaces to transfer skills with reinforcement learning. *arXiv preprint arXiv:1703.02949*, 2017.

[14] Saurabh Gupta, Varun Tolani, James Davidson, Sergey Levine, Rahul Sukthankar, and Jitendra Malik. Cognitive mapping and planning for visual navigation. *International Journal of Computer Vision*, 128:1311–1330, 2019.

[15] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement, 2018.

[16] Isma Hadji, Konstantinos G Derpanis, and Allan D Jepson. Representation learning via global temporal alignment and cycle-consistency. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11068–11077, 2021.

[17] Nicklas Hansen, Hao Su, and Xiaolong Wang. Stabilizing deep q-learning with convnets and vision transformers under data augmentation. *Advances in Neural Information Processing Systems*, 34:3680–3693, 2021.

[18] Sanjay Haresh, Sateesh Kumar, Huseyin Coskun, Shahram N Syed, Andrey Konin, Zeeshan Zia, and Quoc-Huy Tran. Learning by aligning videos in time. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5548–5558, 2021.

[19] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *NeurIPS*, 2016.

[20] Rishabh Jangir, Nicklas Hansen, Sambaran Ghosal, Mohit Jain, and Xiaolong Wang. Look closer: Bridging egocentric and third-person views with transformers for robotic manipulation. *IEEE Robotics and Automation Letters*, 2022.

[21] Jun Jin, Laura Petrich, Zichen Zhang, Masood Dehghan, and Martin Jagersand. Visual geometric skill inference by watching human demonstration. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8985–8991. IEEE, 2020.

[22] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[23] Youngwoon Lee, Andrew Szot, Shao-Hua Sun, and Joseph J Lim. Generalizable imitation learning from observation via inferring goal proximity. *Advances in Neural Information Processing Systems*, 34, 2021.

[24] Sergey Levine, P. Pastor, A. Krizhevsky, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International Journal of Robotics Research*, 37:421 – 436, 2018.

[25] Yikang Li, Wanli Ouyang, Bolei Zhou, Kun Wang, and Xiaogang Wang. Scene graph generation from objects, phrases and region captions. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 1270–1279, 2017.

[26] Yunzhu Li, Hao He, Jiajun Wu, Dina Katabi, and Antonio Torralba. Learning compositional koopman operators for model-based control. *arXiv preprint arXiv:1910.08264*, 2019.

[27] Weizhe Liu, Bugra Tekin, Huseyin Coskun, Vibhav Vineet, Pascal Fua, and Marc Pollefeys. Learning to align sequential actions in the wild. *arXiv preprint arXiv:2111.09301*, 2021.

[28] Joanna Materzynska, Tete Xiao, Roei Herzig, Huijuan Xu, Xiaolong Wang, and Trevor Darrell. Something-else: Compositional action recognition with spatial-temporal interaction networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1049–1059, 2020.

[29] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charlie Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.

[30] Andrew Y Ng, Stuart J Russell, et al. Algorithms for inverse reinforcement learning. In *ICML*, 2000.

[31] Deepak Pathak, Parsa Mahmoudieh, Guanghao Luo, Pulkit Agrawal, Dian Chen, Yide Shentu, Evan Shelhamer, Jitendra Malik, Alexei A. Efros, and Trevor Darrell. Zero-shot visual imitation. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2131–21313, 2018.

[32] Dean A. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In D. Touretzky, editor, *Advances in Neural Information Processing Systems*, volume 1. Morgan-Kaufmann, 1988.

[33] Haozhi Qi, Xiaolong Wang, Deepak Pathak, Yi Ma, and Jitendra Malik. Learning long-term visual dynamics with region proposal interaction networks. In *ICLR*, 2021.

[34] Yuzhe Qin, Yueh-Hua Wu, Shaowei Liu, Hanwen Jiang, Ruihan Yang, Yang Fu, and Xiaolong Wang. Dexmv: Imitation learning for dexterous manipulation from human videos. *arXiv preprint arXiv:2108.05877*, 2021.

[35] Ilija Radosavovic, Xiaolong Wang, Lerrel Pinto, and Jitendra Malik. State-only imitation learning for dexterous manipulation. *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7865–7871, 2021.

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

[37] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

[38] Alvaro Sanchez-Gonzalez, Nicolas Heess, Jost Tobias Springenberg, Josh Merel, Martin Riedmiller, Raia Hadsell, and Peter Battaglia. Graph networks as learnable physics engines for inference and control. In *International Conference on Machine Learning*, pages 4470–4479. PMLR, 2018.

[39] Karl Schmeckpeper, Oleh Rybkin, Kostas Daniilidis, Sergey Levine, and Chelsea Finn. Reinforcement learning with videos: Combining offline observations with interaction. *arXiv preprint arXiv:2011.06507*, 2020.

[40] Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, Sergey Levine, and Google Brain. Time-contrastive networks: Self-supervised learning from video. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 1134–1141. IEEE, 2018.

[41] Dandan Shan, Jiaqi Geng, Michelle Shu, and David F Fouhey. Understanding human hands in contact at internet scale. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9869–9878, 2020.

[42] Maximilian Sieb, Zhou Xian, Audrey Huang, Oliver Kroemer, and Katerina Fragkiadaki. Graph-structured visual imitation. In *Conference on Robot Learning*, pages 979–989. PMLR, 2020.

[43] Shuran Song, Andy Zeng, Johnny Lee, and Thomas Funkhouser. Grasping in the wild: Learning 6dof closed-loop grasping from low-cost demonstrations. *Robotics and Automation Letters*, 2020.

[44] Siddharth Srivastava, Eugene Fang, Lorenzo Riano, Rohan Chitnis, Stuart Russell, and Pieter Abbeel. Combined task and motion planning through an extensible planner-independent interface layer. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 639–646, 2014.

[45] Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. *ArXiv*, abs/1805.01954, 2018.

[46] Faraz Torabi, Garrett Warnell, and Peter Stone. Generative adversarial imitation from observation. *arXiv*, 2018.

[47] Junyan Wang, Yang Long, Maurice Pagnucco, and Yang Song. Dynamic graph warping transformer for video alignment. 2020.

[48] Nicholas Watters, Daniel Zoran, Theophane Weber, Peter Battaglia, Razvan Pascanu, and Andrea Tacchetti. Visual interaction networks: Learning a physics simulator from video. *Advances in neural information processing systems*, 30, 2017.

[49] Haoyu Xiong, Quanzhou Li, Yun-Chun Chen, Homanga Bharadhwaj, Samarth Sinha, and Animesh Garg. Learning by watching: Physical imitation of manipulation skills from human videos. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7827–7834. IEEE, 2021.

[50] Danfei Xu, Yuke Zhu, Christopher Bongsoo Choy, and Li Fei-Fei. Scene graph generation by iterative message passing. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3097–3106, 2017.

[51] Wei Yang, X. Wang, Ali Farhadi, Abhinav Kumar Gupta, and Roozbeh Mottaghi. Visual semantic navigation using scene priors. *ArXiv*, abs/1810.06543, 2019.

[52] Denis Yarats, Ilya Kostrikov, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. In *International Conference on Learning Representations*, 2020.

[53] Yufei Ye, Maneesh Singh, Abhinav Gupta, and Shubham Tulsiani. Compositional video prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10353–10362, 2019.

[54] Yufei Ye, Dhiraj Gandhi, Abhinav Gupta, and Shubham Tulsiani. Object-centric forward modeling for model predictive control. In *Conference on Robot Learning*, pages 100–109. PMLR, 2020.

[55] Sarah Young, Dhiraj Gandhi, Shubham Tulsiani, Abhinav Gupta, Pieter Abbeel, and Lerrel Pinto. Visual imitation made easy. *arXiv*, 2020.

[56] Kevin Zakka, Andy Zeng, Pete Florence, Jonathan Tompson, Jeannette Bohg, and Debidatta Dwibedi. Xirl: Cross-embodiment inverse reinforcement learning. In *Conference on Robot Learning*, pages 537–546. PMLR, 2022.

[57] Yifeng Zhu, Jonathan Tremblay, Stan Birchfield, and Yuke Zhu. Hierarchical planning for long-horizon manipulation with geometric and symbolic scene graphs. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6541–6548. IEEE, 2021.

# A Qualitative Analysis of Learned Reward

In this section, we present qualitative analysis of the reward learned using *GraphIRL*. We plot the reward as defined in Equation 4 for *GraphIRL* and two baseline IRL methods for three test examples across three tasks. The tasks we evaluate with are *Peg in Box*, *Push*, and *Reach*. For each task, we use show two successful episodes and one unsuccessful episode. The length of each episode is 50, and for each figure we have included, we provide images that align with critical points in the completion of the task.

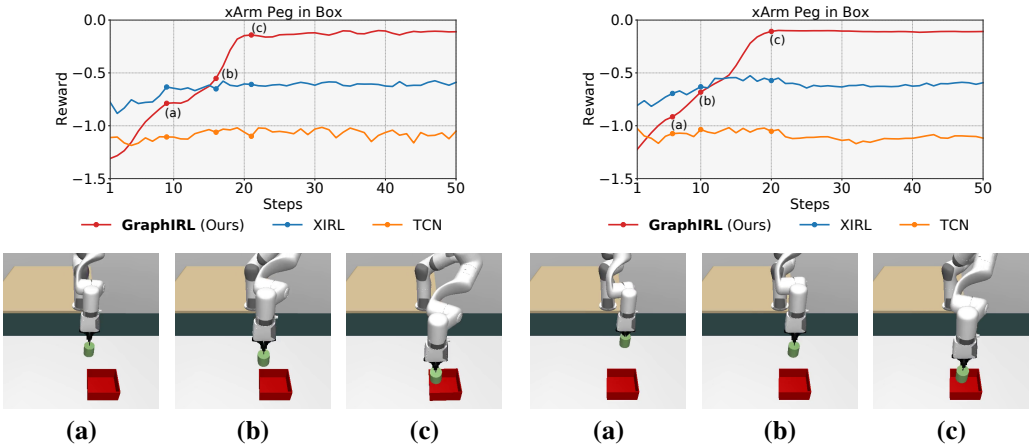


Figure 7: **Peg in Box Task Progress: Success**. For the *Peg in Box* task setting, we find that *GraphIRL* provides an accurate measurement of task progress. Pictured are video frames **(a)**, **(b)**, **(c)** which denote critical points of task progress. Task progress is measured using video frames from a 50-step evaluation episode.

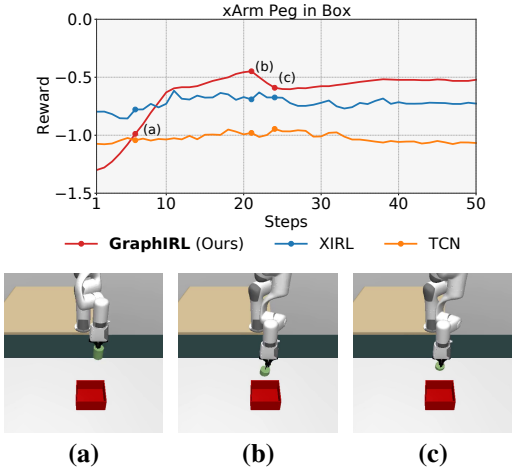


Figure 8: **Peg in Box Task Progress: Failure**. *GraphIRL* measures positive task progress until the peg goes into the table, a critical failure point for the task. The physical interaction between the peg and table is unnatural, and our method succeeds in recognizing this.

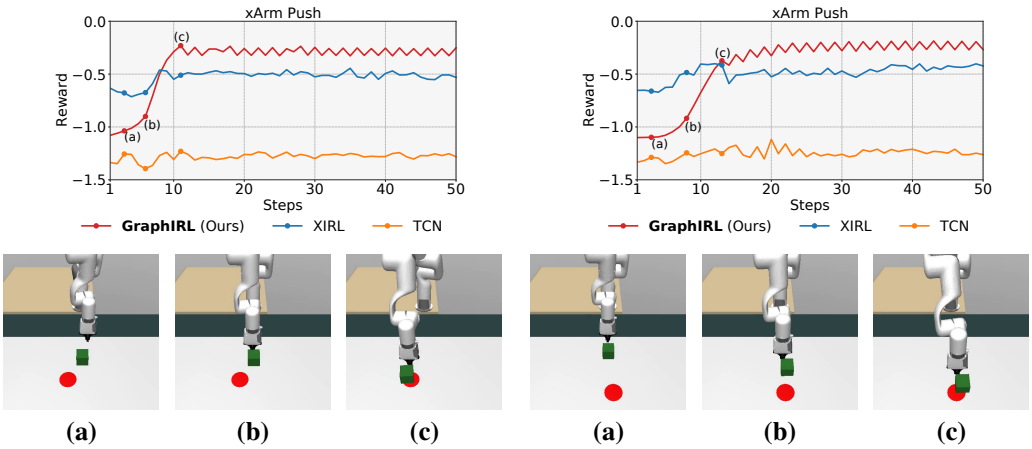


Figure 9: **Push Task Progress: Success**. The *Push* task setting is often completed within the first 10 steps of the evaluation episode, and as shown between Steps 1 through 10 in both success examples, *GraphIRL* measures high task progress. XIRL and TCN on the other hand, incorrectly show much lower task progress.

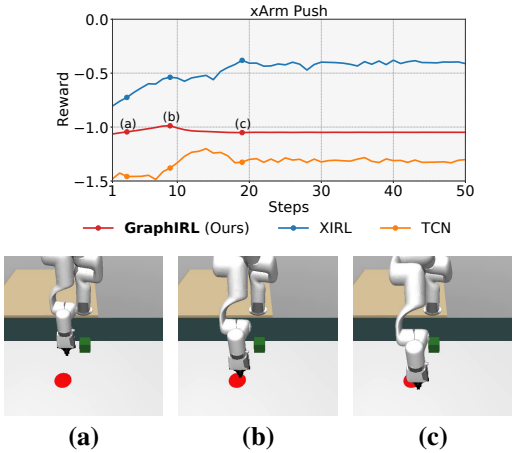


Figure 10: **Push Task Progress: Failure**. *GraphIRL*'s understanding of object relationships is made clear in this *Push* task failure, since without any forward movement of the box object towards the goal, no positive task progress is made. Other baselines rely on direct visual input of the task, and because of this, they inaccurately align visual states **(a)**, **(b)**, **(c)** of the task with positive task progress.

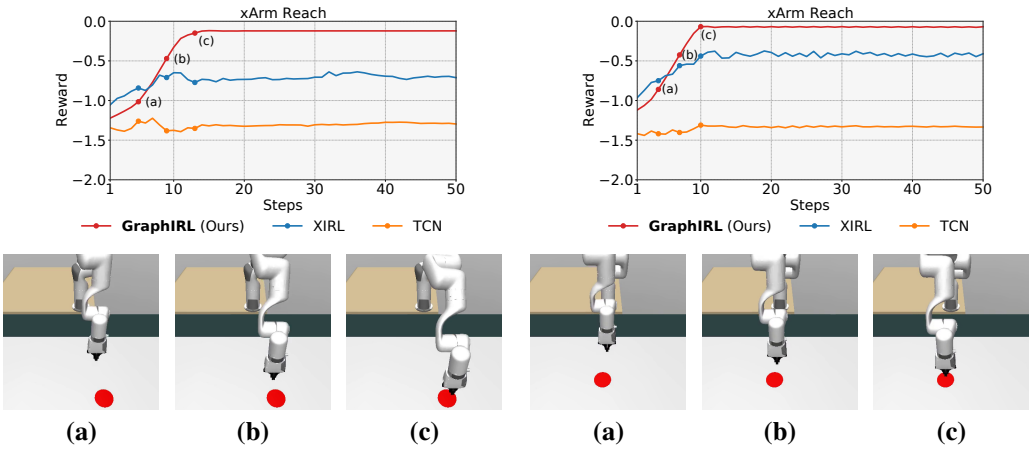


Figure 11: **Reach Task Progress: Success**. In the *Reach* task setting, positive task progress is measured by *GraphIRL* with forward movement of the end-effector gripper towards the goal location. The image frames **(a)**, **(b)**, **(c)** reflect the alignment between measured task progress and visual state of the task.

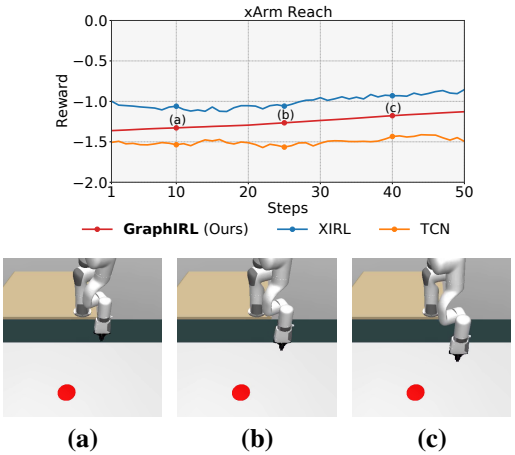


Figure 12: **Reach Task Progress: Failure**. Our *GraphIRL* method measures an approximately linear task progress in this failure example for *Reach*. The gripper's distance to the goal region is indeed minimized over time, though since it does not get within close-enough distance to the goal, the measured task progress is lower compared to success examples shown in Figure 11.

We find that our method provides a superior and accurate reward signal to the agent compared to the baseline visual IRL methods. We observe that if a task is being completed successfully or unsuccessfully in a video, our method can obtain a reward that accurately reflects how close the agent is to completing the task. Additionally, both *XIRL* and *TCN* yield low reward even for successful episodes due to large distance between the current observation and the representative goal observation in the embedding space which could be attributed to visual domain shift.

## B Additional Implementation Details

**Representation Learning.** Each MLP in the Spatial Interaction Encoder Network defined in Equation 3.1 is implemented as a 2-layer network with a ReLU activation. The size of the final embedding $\psi(\cdot)$ is 128 in our experiments. Please see Table 3 for a detailed list of hyperparameters for representation learning. All the hyperparameters in Table 3 are kept fixed for all tasks considered in this work.

**Reinforcement Learning.** For X-MAGICAL, we follow Zakka et al. [56] and learn a state based policy. The state vector has dimensions of 16 and 17 for the *Standard* and *Diverse* environments

| Hyperparameter | Value |
|---|---|
| # of sampled frames | 90 |
| Batch Size | 2 |
| Learning Rate | $10^{-5}$ |
| Weight Decay | $10^{-5}$ |
| # of training iterations | 12000 |
| Embedding Size | 128 |
| Softmax Temperature | 0.1 |

Table 3: Hyperparameters for Representation Learning with GraphIRL.

respectively. The *Diverse* environment state has an additional dimension to represent the size of blocks. For xArm, we learn an image based policy. Specifically, we use first-person and third-person cameras to learn a policy from multi-view image data. We extract $84 \times 84$ image from both cameras and concatenate them channel-wise. We use the network architecture and attention mechanism proposed in Jangir et al. [20]. Additionally, we apply data augmentation techniques: random $\pm 4$ pixel shift [52] and color jitter [17].

**Extracting Reward.** In order to compute the reward during Reinforcement Learning (RL) training, we use the locations of objects available in simulation to extract the bounding boxes corresponding to the current observation. The bounding boxes are used to construct the object representation which is then passed to the trained Spatial Interaction Encoder Network to get the reward.

**Criterion for Success.** We use distance threshold to determine the success of an episode. The thresholds are 5cms, 10cms and 8cms for *Reach*, *Push* and *Peg in Box* respectively. The distance refers to distance between goal position and end-effector for *Reach*, and goal position and object position for *Push* and *Peg in Box*.

**Baseline Implementation Details.** For all the vision-based baselines, we use the hyperparameters, data augmentation schemes and network architectures provided in Zakka et al. [56]. Readers are encouraged to read Zakka et al. [56] for more details on the vision-based baselines.

# C X-MAGICAL Experiment Details

## C.1 Demonstration Data

For collecting demonstration data in the X-MAGICAL *Diverse* environment, we trained 5 uniquely-seeded Soft Actor-Critic (SAC) RL policies for 2 million steps for each embodiment using the environment reward. We collect 1000 successful episode rollouts for each embodiment using the 5 trained policies. In particular, each policy is used to produce 200 episode rollouts for a given embodiment.

## C.2 Diverse Environment

Below, we explain the randomization performed on the blocks in the diverse environment that we use in our experiments:

- **Color:** We randomly assign 1 out of 4 colors to each block.
- **Shape:** Each block is randomly assigned 1 out of 6 shapes.
- **Size:** The block sizes are also varied. In particular, we generate a number between 0.75 and 1.25 and multiply the default block size by that factor.
- **Initial Orientation:** The initial orientation of the blocks is also randomized. We randomly pick a value between 0 to 360 degrees.
- **Initial Location:** The initial location of the boxes is randomized by first randomly picking a position for the y-coordinate for all blocks and then randomly selecting x-coordinate separately for each block. This randomization is also performed in the standard environment.

# D Additional Results on X-MAGICAL Benchmark

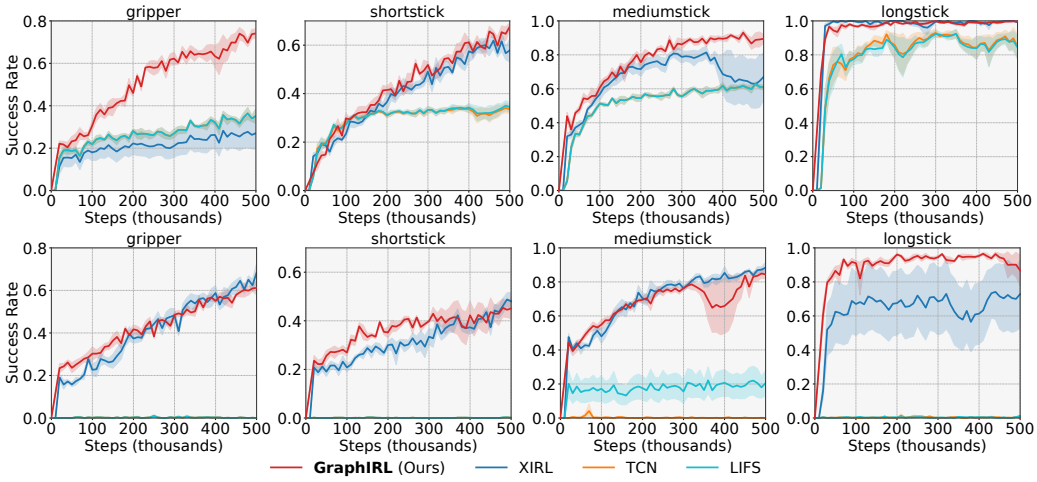


Figure 13: ***Cross-Embodiment Same-Environment:*** We further evaluate *GraphIRL* in the cross-embodiment same-environment setting *(top)* Standard Environment *(bottom)* Diverse Environment, and it continues to provide competitive success rates akin to those achieved by XIRL. These results confirm that *GraphIRL* is a consistent and reliable method for learning from video demonstrations in visually similar environments.

To complement our cross-embodiment cross-environment results from the main paper, we also report results for *X-MAGICAL* in the *cross-embodiment same-environment* setting. As shown in Figure 13, we outperform *TCN* and *LIFS* by significant margins and achieve comparable results to *XIRL*. These results reflect the effectiveness of *GraphIRL* when learning in a visually similar environment with visually different agents.

# E Appendix E: xArm Experiment Details

## E.1 Description of Environment Rewards

In this section, we define the environment rewards for xArm environments that were compared against GraphIRL in robot manipulation experiments under Section **??**. We define $p_g$, $p_o$, and $p_e$ as the positions of the goal, object and robot end-effector respectively. The reward for *Push* is defined as $||p_o - p_g||^2$, for reach it becomes $||p_e - p_g||^2$ and finally for *Peg in Box*, the reward is $||p_o - p_g||^2$. Note that the distances are computed using 2-d positions in the case of *Reach* and *Push* and 3-d positions in the case of *Peg in Box*.

## E.2 Demonstration Data

We use data from [39] for *Push*. We collect 256 and 162 demonstrations respectively for *Reach* and *Peg in Box*. For *Reach*, we use 18 visually distinct goal position markers *i.e.* 3 different shapes and each shape with 6 different colors in order to ensure visual diversity. *Reach* demonstrations have minimum, average and maximum demonstration lengths of 1.76 seconds, 4.51 seconds and 9.23 seconds respectively. For *Peg in Box*, we use 4 visually distinct objects. In this case, the minimum, average and maximum demonstration lengths are 1.73 seconds, 4.74 seconds and 11.7 seconds respectively. For both *Reach* and *Peg in Box*, the goal and object positions are also varied across demonstrations to diversify trajectories. Please see `https://sateeshkumar21.github.io/GraphIRL/` for examples of collected demonstrations.

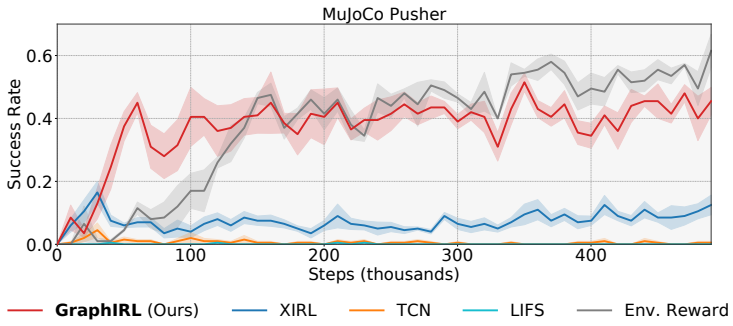


Figure 14: **MuJoCo State Pusher Task Progress: Success** *GraphIRL* provides a reward signal that is both better than all other vision-based baselines and nearly as good as the task-specific environment reward. This indicates that the reward learned from *GraphIRL* could be used across multiple environments of the same task, showing strong generalization capabilities.

## F Additional Results on Robot Manipulation in Simulation

We also experiment with the *MuJoCo State Pusher* environment used by Schmeckpeper et al. [39] and Zakka et al. [56]. However, we make two changes, (1) Instead of using a fixed goal position, we use a randomized goal position and learn a goal-conditioned policy and (2) we do not use the sparse environment reward and instead only use the learned rewards for GraphIRL and learning-based baselines. Figure 14 presents our results, we note that GraphIRL achieves slightly lower success rate than the task-specific environment reward (e.g. GraphIRL $0.455$ vs Environment Reward $0.6133$). Further, all vision-based baselines perform significantly lower than GraphIRL (e.g. GraphIRL $0.455$ vs XIRL $0.125$ and TCN $0.005$). For all learning-based methods, we use the data from Schmeckpeper et al. [39] as training demonstrations similar to *Push* experiments conducted in Section **??**.

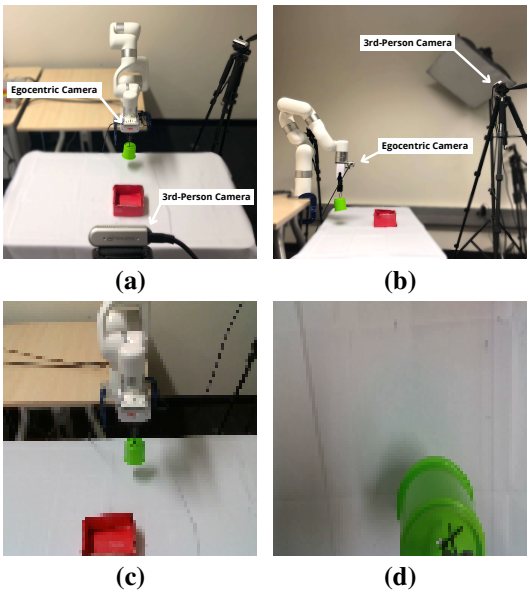


Figure 15: **Real Robot Setup**. In **(a)** and **(b)**, we provide images of our real-world environment for the *Peg in Box* task. We use a static third-person camera and an egocentric camera which moves with the arm while completing the task. Pictured in **(c)** and **(d)** are single image frames captured by our third-person and egocentric cameras.

## G Robot Setup

We use a Ufactory xArm 7 robot for our real robot experiments. As shown in Figure 15, we use a fixed third-person camera and an egocentric camera that is attached above the robot's gripper. Example images of the egocentric and third-person camera feeds passed to the RL agent are shown in Figure 15 (c) and Figure 15 (d).