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.

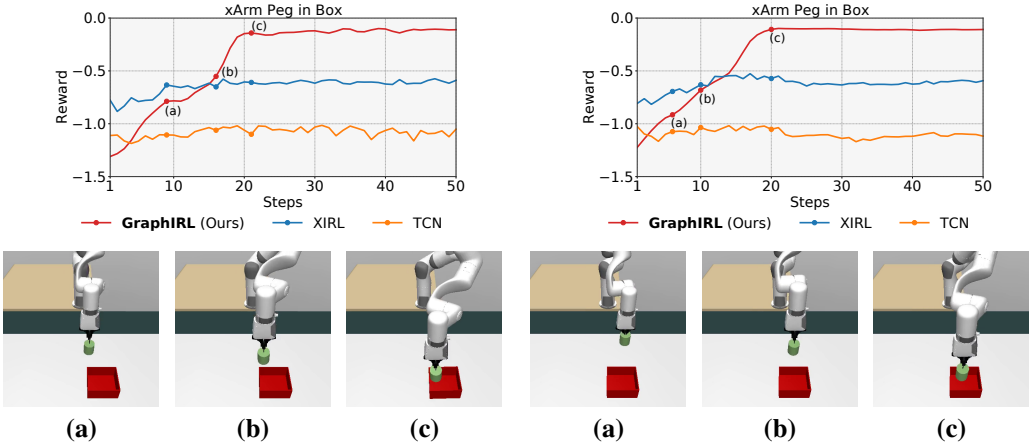

Figure 7: **Peg in Box Task Progress: Success**. For the *Peg in Box* task setting, we find that *GraphIRL* provides an accurate measurement of task progress. Pictured are video frames **(a)**, **(b)**, **(c)** which denote critical points of task progress. Task progress is measured using video frames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

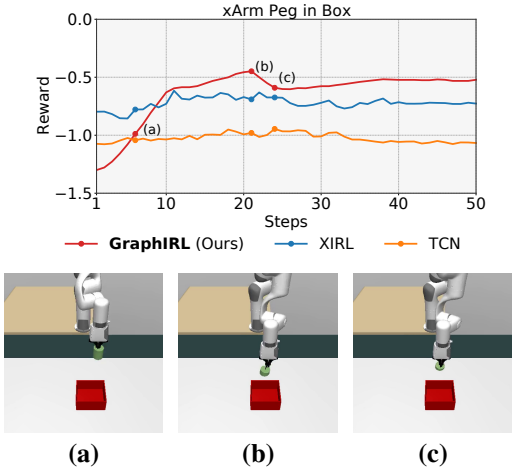

Figure 8: **Peg in Box Task Progress: Failure**. *GraphIRL* measures positive task progress until the peg goes into the table, a critical failure point for the task. The physical interaction between the peg and table is unnatural, and our method succeeds in recognizing this.

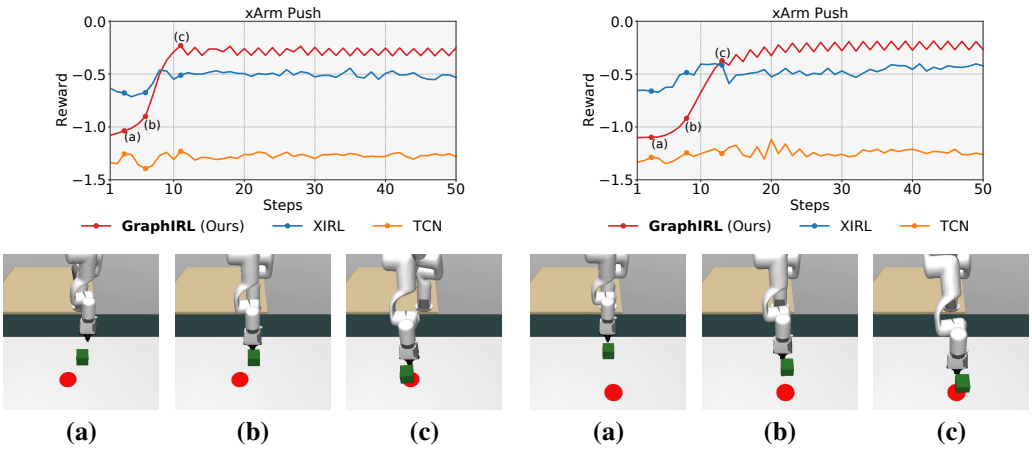

Figure 9: **Push Task Progress: Success**. The *Push* task setting is often completed within the first 10 steps of the evaluation episode, and as shown between Steps 1 through 10 in both success examples, *GraphIRL* measures high task progress. XIRL and TCN on the other hand, incorrectly show much lower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

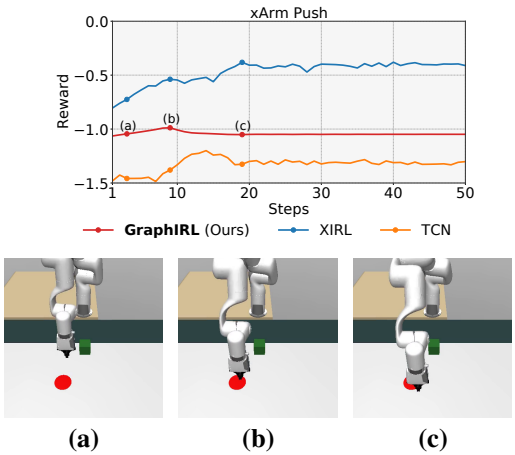

Figure 10: **Push Task Progress: Failure**. *GraphIRL*'s understanding of object relationships is made clear in this *Push* task failure, since without any forward movement of the box object towards the goal, no positive task progress is made. Other baselines rely on direct visual input of the task,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

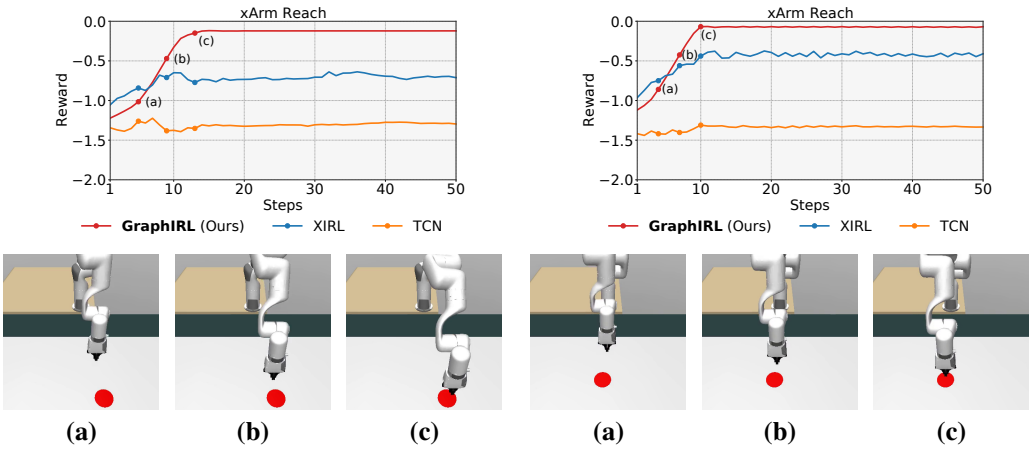

Figure 11: **Reach Task Progress: Success**. In the *Reach* task setting, positive task progress is measured by *GraphIRL* with forward movement of the end-effector gripper towards the goal location. The image frames **(a)**, **(b)**, **(c)** reflect the alignment between measured task progress and visual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

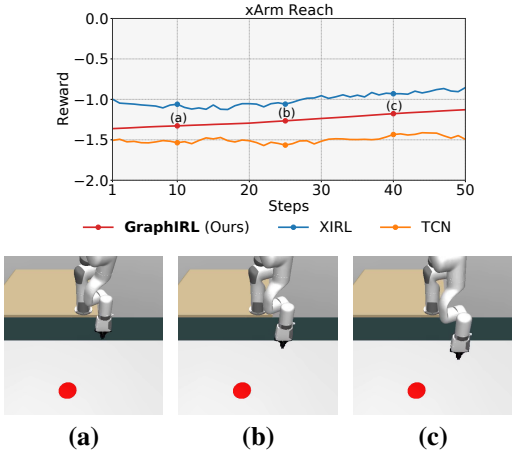

Figure 12: **Reach Task Progress: Failure**. Our *GraphIRL* method measures an approximately linear task progress in this failure example for *Reach*. The gripper's distance to the goal region is indeed minimized over time, though since it does not get within close-enough distance to the goal, the measured 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 84×84 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 ± 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

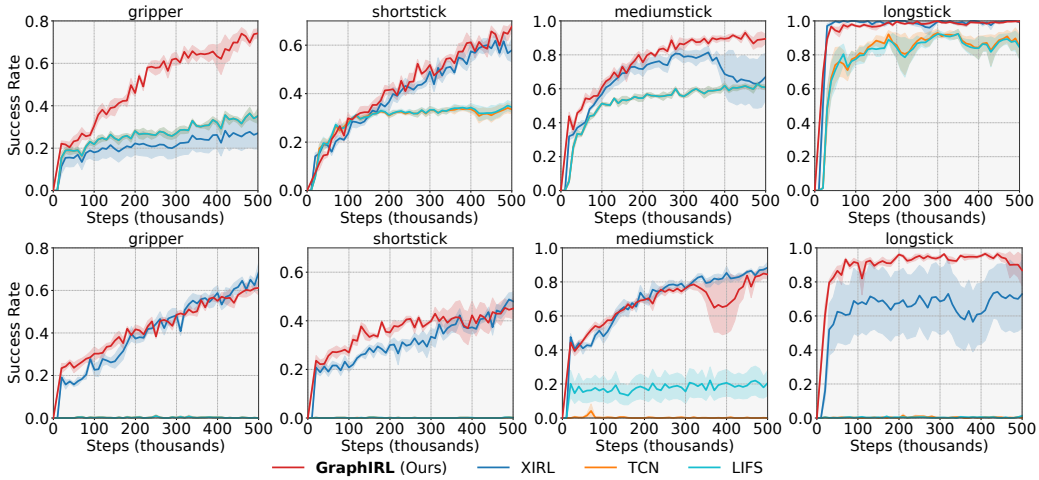

Figure 13: ***Cross-Embodiment Same-Environment:*** We further evaluate *GraphIRL* in the cross-embodiment same-environment setting *(top)* Standard Environment *(bottom)* Diverse Environment, and it continues to provide competitive success rates akin to those achieved by XIRL. These results confirm 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.

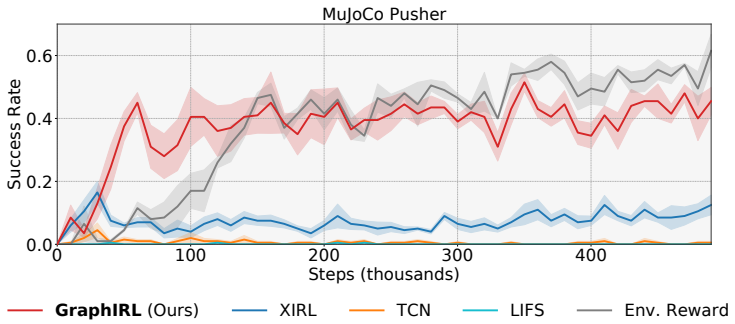

Figure 14: **MuJoCo State Pusher Task Progress: Success** *GraphIRL* provides a reward signal that is both better than all other vision-based baselines and nearly as good as the task-specific environment reward. This indicates that the reward learned from *GraphIRL* could be used across multiple environments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

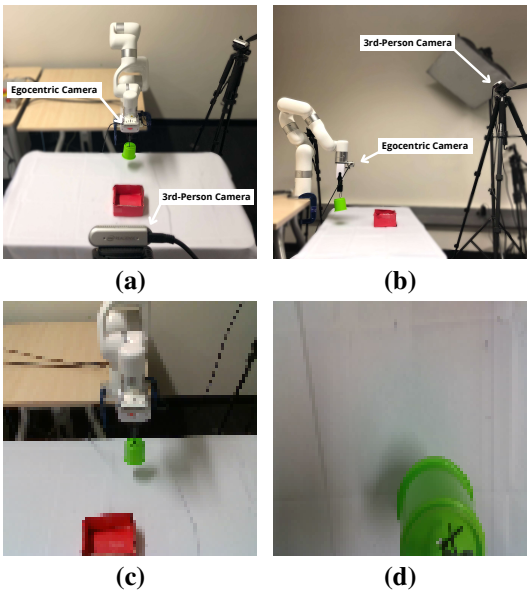

Figure 15: **Real Robot Setup**. In **(a)** and **(b)**, we provide images of our real-world environment for the *Peg in Box* task. We use a static third-person camera and an egocentric camera which moves with the arm while completing the task. Pictured in **(c)** and **(d)** are single image frames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