VIRL: Self-Supervised Visual Graph Inverse Reinforcement Learning

Lei Huang[1], **Weijia Cai**[1], **Zihan Zhu**[2], **Chen Feng**[3], **Helge Rhodin**[4,5], **Zhengbo Zou**[1]
[1]Columbia University [2]Brown University [3]NYU [4]UBC [5]Bielefeld University

Abstract: Learning dense reward functions from unlabeled videos for reinforcement learning exhibits scalability due to the vast diversity and quantity of video resources. Recent works use visual features or graph abstractions in videos to measure task progress as rewards, which either deteriorate in unseen domains or capture spatial information while overlooking visual details. We propose **V**isual-Graph **I**nverse **R**einforcement **L**earning (VIRL), a self-supervised method that synergizes low-level visual features and high-level graph abstractions from frames to graph representations for reward learning. VIRL utilizes a visual encoder that extracts object-wise features for graph nodes and a graph encoder that derives properties from graphs constructed from detected objects in each frame. The encoded representations are enforced to align videos temporally and reconstruct in-scene objects. The pretrained visual graph encoder is then utilized to construct a dense reward function for policy learning by measuring latent distances between current frames and the goal frame. Our empirical evaluation on the X-MAGICAL and Robot Visual Pusher benchmark demonstrates that VIRL effectively handles tasks necessitating both granular visual attention and broader global feature consideration, and exhibits robust generalization to *extrapolation* tasks and domains not seen in demonstrations. Our policy for the robotic task also achieves the highest success rate in real-world robot experiments. Project website: https://leihhhuang.github.io/VIRL/.

Keywords: Inverse Reinforcement Learning, Learning from Video

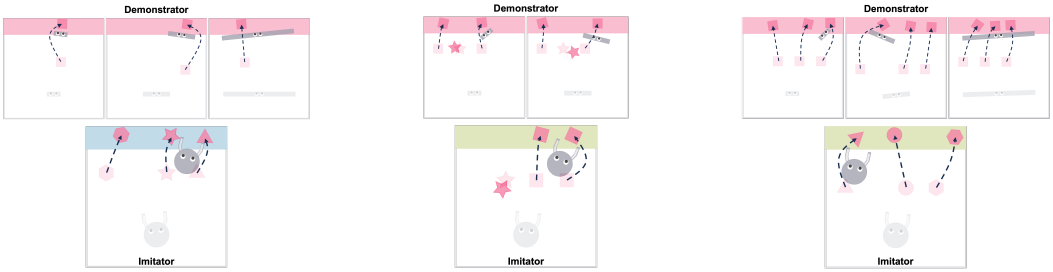
1 Introduction

Intelligent agents have mastered complex tasks by learning policies through reinforcement learning (RL), particularly in the gaming domain, where their performance can match or even surpass champion level [1, 2, 3]. Constructing a precise dense reward function is fundamental and indispensable for appropriately training an agent for a task, as it succinctly defines the task to master [4] and provides dense signals for faster policy learning [5, 6].

However, hand-designing such functions requires substantial domain knowledge and extensive engineering efforts, making it unscalable given the myriad tasks in the real world [7]. Moreover, the process of fine-tuning reward functions [8] is susceptible to overlooking corner cases, leading to agent misbehavior due to unintended sub-optimal rewards [9]. Recently, large language models (LLMs) have been leveraged to assist in generating executable reward functions for RL [10, 11], but they necessitate appropriate prompts and function templates from domain experts and involve considerable training time due to the evolutionary search process.

On the other hand, readily available video demonstrations contain implicit dense rewards in the form of continuous progress towards goals. After watching videos, one can easily estimate how much progress is achieved given a frame and the goal. To this end, inverse reinforcement learning (IRL) has been proposed [12, 13, 14, 15, 16] to learn reward functions from demonstrations by utilizing features of environments and agents. Compared to those requiring ground-truth actions [17, 18] or

8th Conference on Robot Learning (CoRL 2024), Munich, Germany.



(a) *Extrapolation* Task (b) *Visual-feature-dependent* Task (c) Cross-domain Task

Figure 1: Capabilities of VIRL: (a) Generalizable to task **extrapolations**; (b) Adaptable to **visual-feature-dependent** task, such as pushing debris with identical shapes; (c) Robust to **domain shift**. Figures on the top are unlabeled video demonstrations and those on the bottom are tasks for imitators to learn. X-MAGICAL benchmark contains 4 embodiments: short-stick, medium-stick, long-stick as shown in (a) top from left to right, and gripper in the bottom. All tasks are set to be cross-embodiment. **Note**: Long-stick agent is *unsuitable* for (b) visual-feature-dependent tasks due to its excessive length, which hinders task execution.

inferred actions from dynamic models [19, 20], IRL algorithms for action-free videos [21, 22, 23] have garnered greater attention recently due to their scalability and lower cost of data collection.

Recent IRL works [23, 24] achieve impressive performance in building task-specific embodiment-agnostic reward functions using encoders pretrained by temporally aligning videos of varied lengths [25]. Agents are encouraged to reach the provided goal in the latent space by receiving rewards based on the distance between the embeddings of the current state and the goal state. However, employing visual encoder [23] suffers from performance deterioration in unseen domains; and utilizing graph abstractions of object coordinates [24], while robust to domain shifts, sacrifice visual features, which can be crucial for certain tasks. Additionally, since their representations are task-specific, prior works have not been tested beyond demonstrated tasks, such as extrapolations, which may pose challenges even with a minor change in object counts [26]. Therefore, we are interested in two questions: (i) How can we leverage both low-level visual features and high-level graph abstractions to adapt to both geometry-aware and -unaware tasks? (ii) Can a single pretrained encoder be used to build reward functions of tasks beyond those it is trained on, such as task extrapolations?

To this end, we propose a visual graph encoder that combines a visual encoder for capturing object-centric low-level features with a graph encoder for extracting scene-level abstractions. By formulating object-centric graphs that are generalizable across various object counts, we demonstrate that the pretrained visual graph encoder can construct reward functions for tasks of manipulating an unseen number of objects. This capability is inspired by the adaptability and flexibility of graph neural networks to encode graph structures with varying numbers of nodes. Figure 1 illustrates the main properties of our approach, and Table 2 in Appendix A.1 compares advantages over related works.

2 Related Work

For a more comprehensive related work, please see Appendix A.2.

Imitation learning with actions. Imitation learning from expert demonstrations with state and action pairs has demonstrated successes in various approaches [27], from behavioral cloning [28, 29, 30] to inverse reinforcement learning [17, 31, 32]. However, its reliance on ground-truth actions renders data acquisition at scale both costly and challenging, thereby limiting its wider applicability. To leverage unlabeled video demonstrations that are easier to obtain, methods have been developed to infer actions in videos [33, 34, 35, 36, 37, 38, 39]. For example, VPT [39] trains an inverse dynamics model (IDM) on a small labeled dataset and uses the trained IDM to augment a large number of unlabeled videos with actions, which are used for the subsequent policy learning via behavioral cloning. These methods have successfully trained agents using unlabeled videos, but are

subject to well-defined morphologies and dynamics of demonstrators and imitators, making it hard to leverage demonstrations with diverse embodiments.

Domain adaption for imitation learning. To overcome the domain gap between varied agent embodiments, several works [40, 41, 42, 43, 44, 18] attempt to translate the context in demonstrations to the imitators' context in terms of agents and viewpoints, using methods such as generative adversarial networks [45, 46] and context translation. Furthermore, another way to address the domain gap between agents is by inpainting them from video demonstrations and online visual states [47].

Inverse reinforcement learning from videos. On the other hand, a group of works [21, 22, 48, 49, 50, 44, 51, 52, 53, 24] has focused on learning reward functions from videos by extracting latent features that indicate intermediate steps or measure task progress, which are naturally and implicitly present in demonstrations. Learned reward functions are then used for policy training in the regular RL regime. To train on videos with varied lengths and paces, XIRL [23] adopts temporal cycle-consistency loss [25] to learn visual embeddings that measure task progression and then uses distances between current state embedding and the goal embedding as reward signal for policy learning. They demonstrate the pretrained encoder benefits from the diversity of demonstrators' embodiments and generalizes to unseen embodiments. Building on XIRL, our work seeks to train an encoder that generalizes to constructing reward functions for task extrapolations by leveraging graph abstractions, while maintaining the flexibility for tasks requiring attention to low-level visual features by encoding object features.

Object-centric scene graphs. Abstracting visuals into object-centric scene graphs is beneficial for agents to understand environments and make decisions in various tasks [54], such as dynamics modeling [55, 56], navigation [57, 58, 59], and robotic manipulation [60, 61, 62, 43, 63, 64]. Following up XIRL, GraphIRL [24] enhances robustness to diverse object appearances by abstracting frames into graphs, where each node represents an object and contains features including its bounding box coordinate and distances to other objects. However, limitations arise when tasks require attention to object geometries or when dealing with different numbers of objects. Motivated by their work and aiming to address the limitations, we encode object visual features and locations into nodes in a unified way and build graphs such that the encoder can generalize to scenes with arbitrary numbers of objects. Our work can be considered a generalized version of GraphIRL, suitable for both geometry-aware and -unaware tasks, as well as unseen task extrapolations.

3 Method

Our objective is to learn reward functions from video demonstrations showcasing a specific task, enabling an agent with a different embodiment to learn the task and its extrapolations where there is an unseen number of objects to manipulate (Section 3.1). Our approach involves two phases: Firstly, pretraining a visual graph encoder (Section 3.2) to capture object-wise visual features and global scene features in a self-supervised manner (Section 3.3); Secondly, cross-embodiment policy learning for tasks by training RL algorithms with dense reward functions built on the pretrained encoder (Section 3.4). The overall framework of VIRL is depicted in Figure 2.

3.1 Problem formulation

VIRL takes as input a dataset of action-free videos $D^T = \bigcup_{k=1}^{K} V_k^{e,T}$ of the identical task T, where $V_k^{e,T}$ is the kth video demonstration conducted by an agent with embodiment e. A video is defined to be a sequence of image frames $V_k^{e,T} = \{I_1^k, I_2^k, ..., I_{L_k}^k\}$, where L_k is k^{th} video's length. We design a visual graph encoder ϕ_{vg} to transform D^T into a dataset of graph sequences $G^T = \bigcup_{k=1}^{K} G_k^{e,T}$, where $G_k^{e,T} = \{\mathcal{G}_1^k, \mathcal{G}_2^k, ..., \mathcal{G}_{L_k}^k\}$, and extract graph encodings. We pretrain the encoder via temporal alignment of graph encoding sequences and pixel reconstruction of manipulable objects. The pretrained encoder is then utilized to construct general-form dense reward functions that gauge task progress through low-level visual and high-level structural features in graphs. It merits attention that the visual graph encoder's versatility allows it to not only formulate a reward function for the

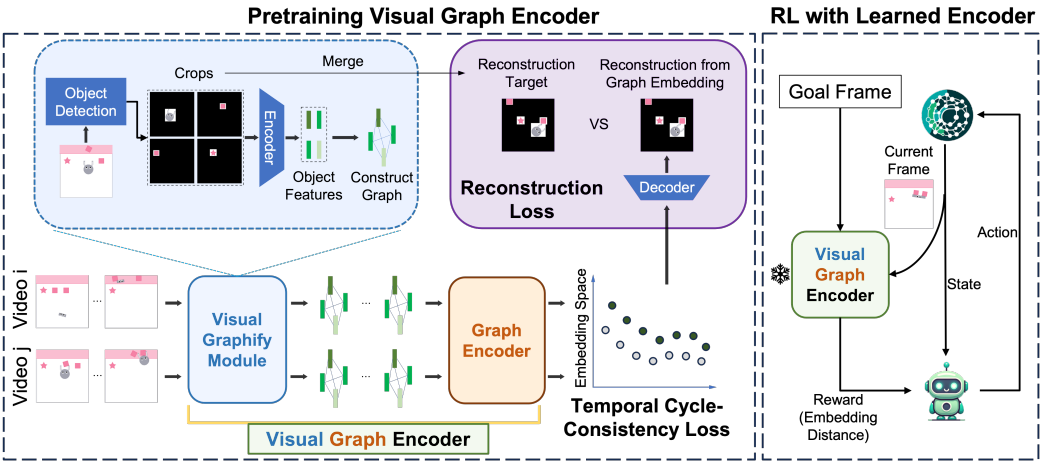


Figure 2: Overview of Visual Graph Inverse Reinforcement Learning. We transform videos to graph sequences containing both low-level visual features and high-level global features, pretrain the visual graph encoder via temporal alignment and object-centric scene reconstruction, and build reward functions using the visual graph encoder for unseen embodiment to learn the demonstrated task and task extrapolations with reinforcement learning.

demonstrated task T but also for its *extrapolation* tasks T' in a zero-shot fashion, attributable to the encoder's capability to process variable-sized graphs. We define extrapolation tasks T' to be those where agents manipulate more objects than demonstrated, which are unseen in videos.

3.2 Visual graph encoder

We propose a visual graph encoder composed of a visual graphify module and a graph encoder. The visual graphify module builds object-centric graphs from images by encoding the spatial and visual features of each object in an image into graph node features and objects interactions into edge features. Then, the graph encoder takes in graphs and outputs the representations, which will be trained via temporal alignment in the latent space and reconstruction of object-centric frames.

Visual graphify module. We use the proposed visual graphify module to transform every sequence of frames $V_k^{e,T} = \{I_1^k, I_2^k, ..., I_{L_k}^k\}$ in the dataset D^T to a sequence of graphs $G_k^{e,T} = \{\mathcal{G}_1^k, \mathcal{G}_2^k, ..., \mathcal{G}_{L_k}^k\}$ to get G^T. A graph is defined as $\mathcal{G}_t^k = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the node set corresponding to objects in the frame and $\mathcal{E}$ is the edge set indicating the relationships between objects. For every frame $I_t^k \in \mathbb{R}^{H \times W \times C}$, an off-the-shelf detector is employed to get N bounding boxes of N objects in the scene. Contrary to approaches that extract visual and positional features using a CNN-based encoder and an MLP separately [65, 66], we encode both features from regions defined by bounding boxes using a shared CNN-based encoder in an unified way. Specifically, for each object in an image, the pixels within its bounding box is kept while all the other pixels being masked so that the *positional* information is encoded implicitly. After this operation, an image is transformed to a stack of images $I_{t,o}^k \in \mathbb{R}^{N \times H \times W \times C}$, each of which contains only one object and is then encoded into an object feature $\mathbf{f}_{o_i} \in \mathbb{R}^{D_{obj}}$ as the feature of a node in $\mathcal{V}$ using the shared ResNet-based [67] visual encoder. The distance between two objects $d_{o_{ij}} \in \mathbb{R}$ represents the edge feature connecting the node pair. We consider all graphs to be fully connected and undirected, allowing for direct interactions between all objects.

Apart from the transformation from images to graphs, we obtain the reconstruction target from the stack of object images $I_{t,o}^k$ to ensure that the embedding after the graph encoder can recover sufficient low-level visual features of objects. It contributes to our work's advantage of not only imitating the spatial interactions between objects but also capturing the low-level visual features of manipulative objects. For example, in the context of the Sweep-to-Goal task in X-MAGICAL [23], a variety of

task variants can be generated, such as pushing multiple objects irrespective of their appearance and selectively pushing duplicates out of three objects present in a scene. With this motivation, we merge all non-agent images in a stack, i.e., ${I_{t,o}^k}' \in \mathbb{R}^{(N-1)\times H\times W\times C}$ into a single RGB image to generate the reconstruction target ${I_t^k}'$ for the original frame I_t^k.

Graph transformer encoder. Subsequent to obtaining G^T, we use a graph network to extract frame embeddings for temporal alignment and reconstruction. Considering the impressive performance of Transformer models in computer vision and natural language processing [68, 69, 70], we employ a Graph Transformer [71] network as the encoder. Given a graph with node features $H = \{\mathbf{h}_1^l, \mathbf{h}_2^l, ..., \mathbf{h}_N^l\}$ and edge features $E = \{\mathbf{d}_{ij} | i, j \in [N] \wedge i \neq j\}$, we update the node features using the graph transformer operator by:

$$\mathbf{h}_i^{l+1} = \mathbf{W}_1^l \mathbf{h}_i^l + \sum_{j\in\mathcal{N}_i} \alpha_{ij}(\mathbf{W}_2^l \mathbf{h}_j^l + \mathbf{W}_3^l \mathbf{d}_{ij}), \text{where } \alpha_{ij} = \sigma\left(\frac{(\mathbf{W}_4^l \mathbf{h}_i^l)^\top + (\mathbf{W}_5^l \mathbf{h}_j^l + \mathbf{W}_3^l \mathbf{d}_{ij})}{\sqrt{d}}\right).$$

All $\mathbf{W}$s are different trainable parameters. α_{ij} is the attention coefficient and σ is the softmax function. After the graph transformer operations, we apply a global mean pooling layer to extract the average of node encodings as the final embeddings for temporal alignment and reconstruction.

3.3 Loss function for pretraining

The visual graph encoder is trained in a self-supervised way. Given each video V, we obtain a sequences of graph embeddings $\phi_{vg}(V)$ using the proposed visual graph encoder. For simplicity, we omit subscripts and superscripts for video index k, embodiment e and task T. We adopt TCC loss [25] to temporally align the graph embeddings between sampled videos for measuring task progress. Meanwhile, we enforce the embeddings to keep objects' low-level visual features by adopting a reconstruction loss, which is inspired by He et al. [72] and Ye et al. [62].

Temporal alignment. Given a pair of randomly sampled videos U and V, the visual graph encoder computes their embeddings $\phi_{vg}(U) = (u_1, u_2, ..., u_{L_U})$ and $\phi_{vg}(V) = (v_1, v_2, ..., v_{L_V})$. To compute the TCC regression loss, we randomly select an embedding u_i in $\phi_{vg}(U)$ and find its soft nearest neighbor $\widetilde{v}$ in $\phi_{vg}(V)$ by estimating the similarity between u_i and every embedding in $\phi_{vg}(V)$ and conducting a weighted average over $\phi_{vg}(V)$:

$$\widetilde{v} = \sum_j^{L_V} \alpha_j v_j, \text{ where } \alpha_j = \frac{e^{-||u_i - v_j||^2}}{\sum_k^{L_V} e^{-||u_i - v_k||^2}}.$$

We then operate cycle back from $\widetilde{v}$ and find the index of $\widetilde{v}$'s soft nearest neighbor in $\phi_{vg}(U)$:

$$\widetilde{\mu} = \sum_k^{L_U} \beta_k k, \text{ where } \beta_k = \frac{e^{-||\widetilde{v} - u_k||^2}}{\sum_j^{L_U} e^{-||\widetilde{v} - u_j||^2}}.$$

Since the index of the starting point u_i is known to be i, the loss can be computed using the squared distance $(\widetilde{\mu} - i)^2$. Additionally, variance $\sigma^2 = \sum_k^{L_U} \beta_k (k - \widetilde{\mu})^2$ is added to the loss function as a regularization term to make predictions less dispersed [25]. Thus, the final objective function for temporal alignment is $L_{tcc} = \frac{(i-\widetilde{\mu})^2}{\sigma^2} + \lambda \log(\sigma)$, where λ is a constant weight.

Object-centric reconstruction. When coming to tasks that require reasoning about low-level features, such as pushing duplicates among several objects, we want the visual graph encoder to keep as much object-centric low-level information as possible [73]. Thus, we define a decoder $\text{Dec}(\phi_{vg}(V))$ that maps from graph embeddings back to frames. We would like to highlight that, instead of reconstructing the original frame I_t^k, our reconstruction target is ${I_t^k}'$ generated in the *visual graphify module* as introduced in Section 3.2, which keeps pixels within the bounding boxes of $N-1$ objects, excluding the agent. We define the reconstruction loss as $L_{rec} = ||{I_t^k}' - \text{Dec}(\phi_{vg}(I_t^k))||_2$.

Finally, we combine the two losses to get the overall loss function for pretraining the visual graph encoder: $L = L_{tcc} + \lambda_1 L_{rec}$, where λ_1 is a weight term.

3.4 Reinforcement learning with learned reward

We use the visual graph encoder ϕ_{vg} pretrained on videos of task T to build reward functions for T and its extrapolations T', which are then used for an agent with an embodiment different from demonstrators to learn the tasks with RL algorithms. Since the embedding sequences are temporally aligned in the embedding space, we can build a reward function for a task by measuring the current progress in the embedding space [23], i.e., the negative distance between the current frame embedding $\phi_{vg}(I)$ and the goal embedding g: $r(I) = -1/s||\phi_{vg}(I) - g||_2^2$, where g is defined by the average embedding of all last frames in videos, and s is a parameter that rescales the reward to a reasonable range for policy learning [74]. For the most challenging extrapolation task where agents learn to push 3 debris from videos of pushing 1 debris, combining our approach with a reformated reward function: $r(I) = -1/s \times \log(||\phi_{vg}(I) - g||_2^2 + 1)$, can achieve better performance.

Compared to the most related prior works [23, 24], the main advantages of building reward functions using our approach are twofold: (i) it reserves the ability to learn tasks that require attention to low-level features while maintaining generalization to unseen domains by leveraging graph abstraction, (ii) apart from the demonstrated task T, it generalizes better to task extrapolations T' because of the flexibility of the encoder to consume graphs consisting of various numbers of objects.

4 Experiments

We conduct experiments on the *Sweep-to-Goal* task in X-MAGICAL [75, 23] and Robot Visual Pusher [44, 24] benchmark to answer following questions: (i) Can a pretrained visual graph encoder construct reward functions for extrapolations of the demonstrated task in zero-shot? (ii) Does VIRL retain low-level visual features for learning? (iii) How does VIRL perform in shifted domains [1]? We also deploy policies trained in simulation on a real robot for robotic tasks. For implementation details, please see Appendix A.3.

4.1 Experiment setup

Baselines. We compare our work against self-supervised approaches that learn reward functions from unlabeled videos: (i) XIRL [23] pretrains a task-specific visual encoder using TCC loss on videos and builds reward functions using the pretrained encoder. (ii) GraphIRL [24] transforms videos to graphs of object coordinates, uses a spatial interaction graph encoder to extract features, and follows a similar pipeline of XIRL. (iii) TCN [22] uses triplet loss to pretrain representations on videos such that frames at the same time are close in the embedding space while those that are temporally different are far apart. (iv) LIFS [73] trains an invariant feature space to transfer skills between agents with different embodiments using a contrastive loss and a reconstruction loss.

Task descriptions. *Sweep-to-Goal* in X-MAGICAL. All tasks are set to be cross-embodiment, i.e., an agent learns from videos of other 3 agents. Task I (Figure 1a): learning to sweep 3 debris to the colored goal region on the top, given videos of sweeping 1 or 2 debris. Task II (Figure 1b): learning to sweep 2 debris with the same shape to the goal region while leaving the rest outside, given videos of the same task. Task III (Figure 1c): learning to sweep 3 debris regardless of shapes and colors, given videos of sweeping 3 squares. *Robot Visual Pusher* tasks. Task I (Figure 5): xArm learning to push an object to the goal, given videos of the same task by human. Task II: xArm learning to push 1 specific object out of 2 objects in-scene to the goal, only given the set of human videos in Task I.

4.2 Experiment results on X-MAGICAL

Results in extrapolation tasks. We validate a *key feature* of our method: the visual graph encoder's ability to construct reward functions for extrapolations of the demonstrated tasks and train agents to manipulate an unseen number of objects. We begin by testing the results of policy learning from videos showcasing cross-embodiment demonstrations of pushing 2 debris. We then increase the

[1] For experiment results of X-MAGICAL in domain-shift environment, please see Appendix A.4

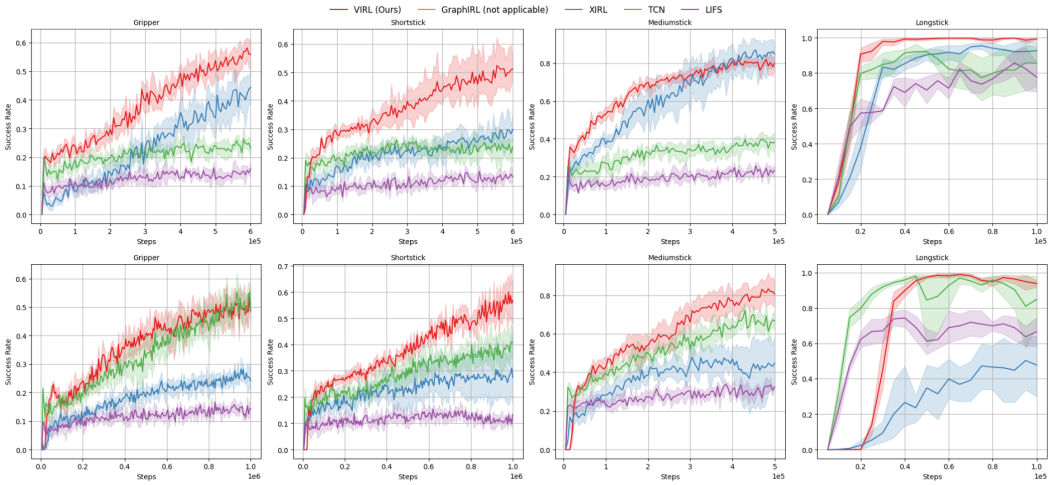


Figure 3: **Cross-embodiment extrapolation tasks.** Learning to push 3 debris given videos of pushing 2 debris (top), and videos of pushing 1 debris (bottom). **Note**: GraphIRL is *not applicable* when agents are learning to manipulate a different number of objects than in the video demonstrations.

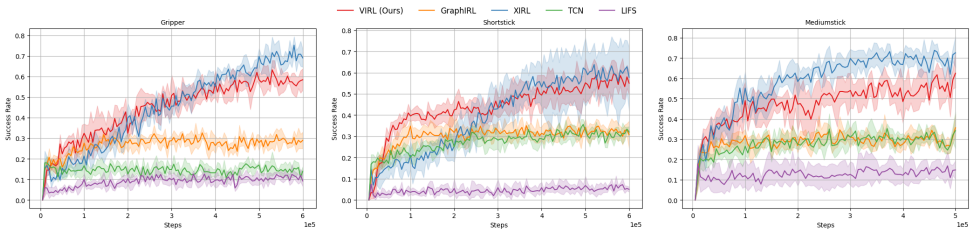


Figure 4: **Cross-embodiment visual-feature-dependent task.** Our approach can be applied for visual-feature-dependent tasks. For example, pushing 2 debris with the identical shape out of 3. **Note**: *long-stick* is *not included* for this task because its excessive length hinders the task execution.

difficulty by providing demonstrations of pushing only 1 debris, as shown in Figure 1a (top). In Figure 3 (top), we illustrate that in the first scenario, our approach outperforms baselines for 3 out of 4 embodiments and is on par with XIRL for medium-stick. In the more challenging scenario, where demonstrators push 1 debris, policy learning becomes significantly harder, as it is not demonstrated that returning to fetch additional debris leads to greater rewards in the long term. Figure 3 (bottom) shows that our approach generalizes to manipulate the unseen number of objects with higher success rates than others. Surprisingly, TCN outperforms XIRL in the most challenging setting and delivers performance comparable to our approach for the gripper and long-stick configurations.

Results in visual-feature-dependent task. We evaluate the second *key feature* of our method: the adaptability to visual-feature-dependent tasks by encoding visual features of objects into nodes. The task for imitators to learn is identical to what is demonstrated, as shown in Figure 1b. Hence, it can be anticipated that the performance of XIRL should be comparable to our approach. Figure 4 reveals that our approach is capable of policy learning for geometry-aware tasks, achieving similar performance to XIRL for two embodiments, yet slightly lower for medium-stick. On the other hand, GraphIRL shows inferior performance in this task because of the lack of visual information.

4.3 Experiment results on Robot Visual Pusher

Learning robotic manipulation from human videos. We build reward functions on encoders pretrained on human videos and use them for policy learning of demonstrated task. Task I in Figure 5 shows the result. Although XIRL can improve the sample efficiency when combined with sparse

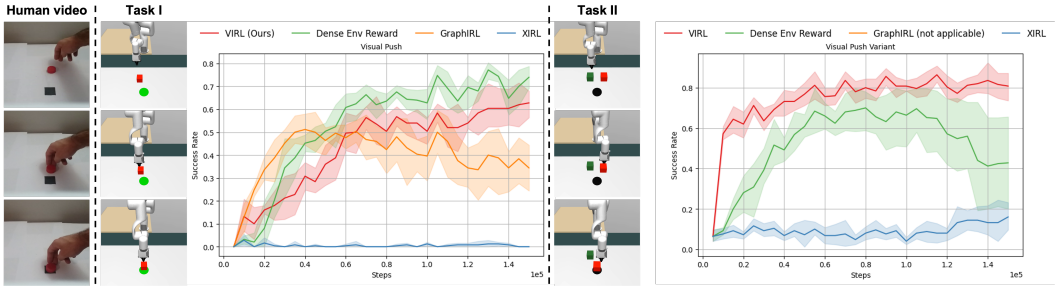


Figure 5: **Robot Visual Pusher.** Task I: robot learns the demonstrated task. Task II: robot learns variant task to push a specific object among 2 objects to goal. Thresholds for success, measured by distance between positions of object and goal, are 5cm for Task I and 10cm [24] for Task II.

environment reward [23], we notice that its learned rewards alone are insufficient for policy learning. In contrast, our approach provides a learned reward function closest to dense environment rewards.

Learning visual-feature-dependent task variant. We use the *same encoders* in the first task to build reward functions for the task variant: pushing a specific object among two objects to the goal. It merits attention that GraphIRL is not applicable for this task since it encodes object counts in demonstrations into node features and cannot generalize to scenarios with unseen numbers of objects. The result of learning Task II in Figure 5 shows that our approach VIRL surpasses the performance of using handcrafted dense environment rewards, while it is hard for agents to learn functional policies given reward signals from the baseline XIRL.

Additionally, we ablate the detector's accuracy to study the impact of missing detection in both robot tasks. For experiment details and results, please see Appendix A.5.

Sim2Real transfer. We evaluate policies trained in simulation, without domain randomization, on a real-world xArm 6 robot in the Robot Visual Pusher - Task II. Each policy runs for 20 trials: the green object is placed on the left and the red on the right for the first 10 trials, then their positions are switched. Results are in Table 1. The performance gap between simulation and real robot likely stems from unaligned physics properties of objects and lack of domain randomization. For details of robot setup and episode visualization, please see Appendix A.6.

Method	Success rate
VIRL (ours)	**0.55**
XIRL	0.05
Env. reward	0.2
GraphIRL	-

Table 1: Real robot test results.

5 Discussion

Conclusion. We present VIRL, an IRL method utilizing a visual graph encoder to construct reward functions for demonstrated tasks in videos and their extrapolation variants. VIRL integrates object-level spatial and visual features unifiedly into nodes for graph construction from frames. Pretraining the visual graph encoder is self-supervised using TCC loss and reconstruction loss. The design of the visual graph encoder enables effective reward function generation for the demonstrated task and extrapolations involving unseen object numbers, which is challenging in robotic tasks [26]. Additionally, our approach is flexibly adaptable for visual-feature-dependent tasks.

Limitations. Our approach depends on reliable object detectors. While occasional detection failures have minimal impact, missing detection in keyframes can be critical. Additionally, the tested tasks don't fully capture the complexity of kinematics and scene variations in benchmarks like RoboSuite [76], MetaWorld [77], and BridgeDataV2 [78]. Third, it remains unclear how VIRL would generalize to deformable object manipulation [79, 80]. Lastly, focusing only on object-centric information may limit performance on tasks requiring broader scene context. [2]

[2] For a more comprehensive discussion of limitations, please see Appendix A.9

Acknowledgments

We would like to thank the reviewers for the insightful comments and thank Omar Swei, Renjie Liao, Tengda Han, Qi Zhu, Runsheng Xu, and Binghao Huang for the fruitful discussions.

References

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, Feb 2015. ISSN 1476-4687.

[2] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, Jan 2016. ISSN 1476-4687.

[3] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, Nov 2019. ISSN 1476-4687.

[4] Y. Hu, Q. Xie, V. Jain, J. Francis, J. Patrikar, N. Keetha, S. Kim, Y. Xie, T. Zhang, Z. Zhao, et al. Toward general-purpose robots via foundation models: A survey and meta-analysis. *arXiv preprint arXiv:2312.08782*, 2023.

[5] A. Y. Ng, D. Harada, and S. J. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the Sixteenth International Conference on Machine Learning*, page 278–287, San Francisco, CA, USA, 1999. ISBN 1558606122.

[6] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA, 2018. ISBN 0262039249.

[7] T. Xiao, I. Radosavovic, T. Darrell, and J. Malik. Masked visual pre-training for motor control. *arXiv preprint arXiv:2203.06173*, 2022.

[8] S. Booth, W. B. Knox, J. Shah, S. Niekum, P. Stone, and A. Allievi. The perils of trial-and-error reward design: Misdesign through overfitting and invalid task specifications. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(5):5920–5929, Jun. 2023.

[9] J. Clark and D. Amodei. Faulty reward functions in the wild. *Internet: https://blog. openai. com/faulty-reward-functions*, 2016.

[10] W. Yu, N. Gileadi, C. Fu, S. Kirmani, K.-H. Lee, M. G. Arenas, H.-T. L. Chiang, T. Erez, L. Hasenclever, J. Humplik, et al. Language to rewards for robotic skill synthesis. *arXiv preprint arXiv:2306.08647*, 2023.

[11] Y. J. Ma, W. Liang, G. Wang, D.-A. Huang, O. Bastani, D. Jayaraman, Y. Zhu, L. Fan, and A. Anandkumar. Eureka: Human-level reward design via coding large language models. *arXiv preprint arXiv:2310.12931*, 2023.

[12] A. Y. Ng and S. J. Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, page 663–670, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc. ISBN 1558607072.

[13] P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the Twenty-First International Conference on Machine Learning*, ICML '04, page 1, New York, NY, USA, 2004. Association for Computing Machinery. ISBN 1581138385. doi:10.1145/1015330.1015430.

[14] U. Syed and R. E. Schapire. A game-theoretic approach to apprenticeship learning. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc., 2007.

[15] B. D. Ziebart, A. Maas, J. A. Bagnell, and A. K. Dey. Maximum entropy inverse reinforcement learning. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, volume 8, pages 1433–1438, 2008.

[16] D. Hadfield-Menell, S. Milli, P. Abbeel, S. Russell, and A. D. Dragan. Inverse reward design. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6768–6777, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

[17] C. Finn, S. Levine, and P. Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In M. F. Balcan and K. Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 49–58, New York, New York, USA, 20–22 Jun 2016. PMLR.

[18] A. Fickinger, S. Cohen, S. Russell, and B. Amos. Cross-domain imitation learning via optimal transport. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=xP3cPq2hQC.

[19] A. Edwards, H. Sahni, Y. Schroecker, and C. Isbell. Imitating latent policies from observation. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1755–1763. PMLR, 09–15 Jun 2019.

[20] I. Radosavovic, X. Wang, L. Pinto, and J. Malik. State-only imitation learning for dexterous manipulation. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7865–7871, 2021.

[21] P. Sermanet, K. Xu, and S. Levine. Unsupervised perceptual rewards for imitation learning. In *Proceedings of Robotics: Science and Systems*, Cambridge, Massachusetts, July 2017. doi: 10.15607/RSS.2017.XIII.050.

[22] 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, 2018.

[23] K. Zakka, A. Zeng, P. Florence, J. Tompson, J. Bohg, and D. Dwibedi. Xirl: Cross-embodiment inverse reinforcement learning. In A. Faust, D. Hsu, and G. Neumann, editors, *Proceedings of the 5th Conference on Robot Learning*, volume 164 of *Proceedings of Machine Learning Research*, pages 537–546. PMLR, 08–11 Nov 2022.

[24] S. Kumar, J. Zamora, N. Hansen, R. Jangir, and X. Wang. Graph inverse reinforcement learning from diverse videos. In K. Liu, D. Kulic, and J. Ichnowski, editors, *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pages 55–66. PMLR, 14–18 Dec 2023.

[25] D. Dwibedi, Y. Aytar, J. Tompson, P. Sermanet, and A. Zisserman. Temporal cycle-consistency learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[26] Z. Fu, T. Z. Zhao, and C. Finn. Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation. In *arXiv*, 2024.

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

[28] D. A. Pomerleau. Alvinn: an autonomous land vehicle in a neural network. In *Proceedings of the 1st International Conference on Neural Information Processing Systems*, NIPS'88, page 305–313, Cambridge, MA, USA, 1988. MIT Press.

[29] S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In G. Gordon, D. Dunson, and M. Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 627–635, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR. URL https://proceedings.mlr.press/v15/ross11a.html.

[30] A. Zhou, M. J. Kim, L. Wang, P. Florence, and C. Finn. Nerf in the palm of your hand: Corrective augmentation for robotics via novel-view synthesis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 17907–17917, June 2023.

[31] J. Ho and S. Ermon. Generative adversarial imitation learning. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[32] S. Haldar, V. Mathur, D. Yarats, and L. Pinto. Watch and match: Supercharging imitation with regularized optimal transport. In *Conference on Robot Learning*, pages 32–43. PMLR, 2023.

[33] F. Torabi, G. Warnell, and P. Stone. Behavioral cloning from observation. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 4950–4957. International Joint Conferences on Artificial Intelligence Organization, 7 2018. doi:10.24963/ijcai.2018/687. URL https://doi.org/10.24963/ijcai.2018/687.

[34] D. Pathak, P. Mahmoudieh, M. Luo, P. Agrawal, D. Chen, F. Shentu, E. Shelhamer, J. Malik, A. A. Efros, and T. Darrell. Zero-shot visual imitation. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=BkisuzWRW.

[35] C. Yang, X. Ma, W. Huang, F. Sun, H. Liu, J. Huang, and C. Gan. Imitation learning from observations by minimizing inverse dynamics disagreement. *Advances in neural information processing systems*, 32, 2019.

[36] A. Edwards, H. Sahni, Y. Schroecker, and C. Isbell. Imitating latent policies from observation. In *International conference on machine learning*, pages 1755–1763. PMLR, 2019.

[37] L. Shao, T. Migimatsu, Q. Zhang, K. Yang, and J. Bohg. Concept2Robot: Learning Manipulation Concepts from Instructions and Human Demonstrations. In *Proceedings of Robotics: Science and Systems*, Corvalis, Oregon, USA, July 2020. doi:10.15607/RSS.2020.XVI.082.

[38] I. Radosavovic, X. Wang, L. Pinto, and J. Malik. State-only imitation learning for dexterous manipulation. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7865–7871. IEEE, 2021.

[39] B. Baker, I. Akkaya, P. Zhokov, J. Huizinga, J. Tang, A. Ecoffet, B. Houghton, R. Sampedro, and J. Clune. Video pretraining (vpt): Learning to act by watching unlabeled online videos. *Advances in Neural Information Processing Systems*, 35:24639–24654, 2022.

[40] Y. Liu, A. Gupta, P. Abbeel, and S. Levine. Imitation from observation: Learning to imitate behaviors from raw video via context translation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1118–1125. IEEE, 2018.

[41] P. Sharma, D. Pathak, and A. Gupta. Third-person visual imitation learning via decoupled hierarchical controller. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[42] L. Smith, N. Dhawan, M. Zhang, P. Abbeel, and S. Levine. Avid: Learning multi-stage tasks via pixel-level translation of human videos. In *Proceedings of Robotics: Science and Systems*, 07 2020. doi:10.15607/RSS.2020.XVI.024.

[43] M. Sieb, Z. Xian, A. Huang, O. Kroemer, and K. Fragkiadaki. Graph-structured visual imitation. In L. P. Kaelbling, D. Kragic, and K. Sugiura, editors, *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pages 979–989. PMLR, 30 Oct–01 Nov 2020.

[44] K. Schmeckpeper, O. Rybkin, K. Daniilidis, S. Levine, and C. Finn. Reinforcement learning with videos: Combining offline observations with interaction. In J. Kober, F. Ramos, and C. Tomlin, editors, *Proceedings of the 2020 Conference on Robot Learning*, volume 155 of *Proceedings of Machine Learning Research*, pages 339–354. PMLR, 16–18 Nov 2021.

[45] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017. doi:10.1109/ICCV.2017.244.

[46] 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 J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1989–1998. PMLR, 10–15 Jul 2018.

[47] S. Bahl, A. Gupta, and D. Pathak. Human-to-Robot Imitation in the Wild. In *Proceedings of Robotics: Science and Systems*, New York City, NY, USA, June 2022. doi:10.15607/RSS.2022.XVIII.026.

[48] Y. Aytar, T. Pfaff, D. Budden, T. Paine, Z. Wang, and N. de Freitas. Playing hard exploration games by watching youtube. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[49] A. Singh, L. Yang, C. Finn, and S. Levine. End-to-end robotic reinforcement learning without reward engineering. In *Proceedings of Robotics: Science and Systems*, FreiburgimBreisgau, Germany, June 2019. doi:10.15607/RSS.2019.XV.073.

[50] O. Mees, M. Merklinger, G. Kalweit, and W. Burgard. Adversarial skill networks: Unsupervised robot skill learning from video. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4188–4194. IEEE, 2020.

[51] H. Xiong, Q. Li, Y.-C. Chen, H. Bharadhwaj, S. Sinha, and A. Garg. Learning by watching: Physical imitation of manipulation skills from human videos. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, page 7827–7834. IEEE Press, 2021. doi:10.1109/IROS51168.2021.9636080.

[52] Y. Lee, A. Szot, S.-H. Sun, and J. J. Lim. Generalizable imitation learning from observation via inferring goal proximity. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 16118–16130. Curran Associates, Inc., 2021.

[53] A. S. Chen, S. Nair, and C. Finn. Learning Generalizable Robotic Reward Functions from "In-The-Wild" Human Videos. In *Proceedings of Robotics: Science and Systems*, Virtual, July 2021. doi:10.15607/RSS.2021.XVII.012.

[54] D. Bear, C. Fan, D. Mrowca, Y. Li, S. Alter, A. Nayebi, J. Schwartz, L. F. Fei-Fei, J. Wu, J. Tenenbaum, and D. L. Yamins. Learning physical graph representations from visual scenes. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6027–6039. Curran Associates, Inc., 2020.

[55] P. Battaglia, R. Pascanu, M. Lai, D. Jimenez Rezende, and k. kavukcuoglu. Interaction networks for learning about objects, relations and physics. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[56] H. Qi, X. Wang, D. Pathak, Y. Ma, and J. Malik. Learning long-term visual dynamics with region proposal interaction networks. *arXiv preprint arXiv:2008.02265*, 2020.

[57] W. Yang, X. Wang, A. Farhadi, A. Gupta, and R. Mottaghi. Visual semantic navigation using scene priors. *CoRR*, abs/1810.06543, 2018. URL http://arxiv.org/abs/1810.06543.

[58] S. Zhang, X. Song, Y. Bai, W. Li, Y. Chu, and S. Jiang. Hierarchical object-to-zone graph for object navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15130–15140, October 2021.

[59] N. Hughes, Y. Chang, and L. Carlone. Hydra: A real-time spatial perception system for 3D scene graph construction and optimization. *Robotics: Science and Systems (RSS)*, 2022.

[60] A. Sanchez-Gonzalez, N. Heess, J. T. Springenberg, J. Merel, M. Riedmiller, R. Hadsell, and P. Battaglia. Graph networks as learnable physics engines for inference and control. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4470–4479. PMLR, 10–15 Jul 2018.

[61] Y. Li, H. He, J. Wu, D. Katabi, and A. Torralba. Learning compositional koopman operators for model-based control. In *International Conference on Learning Representations*, 2020.

[62] Y. Ye, D. Gandhi, A. Gupta, and S. Tulsiani. Object-centric forward modeling for model predictive control. In L. P. Kaelbling, D. Kragic, and K. Sugiura, editors, *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pages 100–109. PMLR, 30 Oct–01 Nov 2020.

[63] Y. Zhu, J. Tremblay, S. Birchfield, and Y. 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.

[64] H. Jiang, B. Huang, R. Wu, Z. Li, S. Garg, H. Nayyeri, S. Wang, and Y. Li. Roboexp: Action-conditioned scene graph via interactive exploration for robotic manipulation. *arXiv preprint arXiv:2402.15487*, 2024.

[65] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich. Superglue: Learning feature matching with graph neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[66] C. Elich, I. Armeni, M. R. Oswald, M. Pollefeys, and J. Stueckler. Learning-based relational object matching across views. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5999–6005, 2023. doi:10.1109/ICRA48891.2023.10161393.

[67] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[68] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[69] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In J. Burstein, C. Doran, and T. Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, jun 2019. Association for Computational Linguistics.

[70] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021.

[71] Y. Shi, Z. Huang, S. Feng, H. Zhong, W. Wang, and Y. Sun. Masked label prediction: Unified message passing model for semi-supervised classification. In Z.-H. Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 1548–1554. International Joint Conferences on Artificial Intelligence Organization, 8 2021. Main Track.

[72] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16000–16009, June 2022.

[73] A. Gupta, C. Devin, Y. Liu, P. Abbeel, and S. Levine. Learning invariant feature spaces to transfer skills with reinforcement learning. In *International Conference on Learning Representations*, 2017.

[74] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger. Deep reinforcement learning that matters. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), Apr. 2018.

[75] S. Toyer, R. Shah, A. Critch, and S. Russell. The magical benchmark for robust imitation. In H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 18284–18295. Curran Associates, Inc., 2020.

[76] Y. Zhu, J. Wong, A. Mandlekar, R. Martín-Martín, A. Joshi, S. Nasiriany, and Y. Zhu. robosuite: A modular simulation framework and benchmark for robot learning. In *arXiv preprint arXiv:2009.12293*, 2020.

[77] 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.

[78] H. Walke, K. Black, A. Lee, M. J. Kim, M. Du, C. Zheng, T. Zhao, P. Hansen-Estruch, Q. Vuong, A. He, V. Myers, K. Fang, C. Finn, and S. Levine. Bridgedata v2: A dataset for robot learning at scale. In *Conference on Robot Learning (CoRL)*, 2023.

[79] Y. Li, J. Wu, R. Tedrake, J. B. Tenenbaum, and A. Torralba. Learning particle dynamics for manipulating rigid bodies, deformable objects, and fluids. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJgbSn09Ym.

[80] T. Weng, S. Bajracharya, Y. Wang, K. Agrawal, and D. Held. Fabricflownet: Bimanual cloth manipulation with a flow-based policy. In *Conference on Robot Learning*, 2021.

[81] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

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

[83] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR, 2018.

[84] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

[85] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 2024.

[86] G. Jocher, A. Chaurasia, and J. Qiu. Ultralytics YOLO, Jan 2023. URL https://github.com/ultralytics/ultralytics.

[87] N. Watters, D. Zoran, T. Weber, P. Battaglia, R. Pascanu, and A. Tacchetti. Visual interaction networks: Learning a physics simulator from video. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/8cbd005a556ccd4211ce43f309bc0eac-Paper.pdf.

[88] M. Minderer, A. Gritsenko, A. Stone, M. Neumann, D. Weissenborn, A. Dosovitskiy, A. Mahendran, A. Arnab, M. Dehghani, Z. Shen, X. Wang, X. Zhai, T. Kipf, and N. Houlsby. Simple open-vocabulary object detection. In S. Avidan, G. Brostow, M. Cissé, G. M. Farinella, and T. Hassner, editors, *Computer Vision – ECCV 2022*, pages 728–755, Cham, 2022. Springer Nature Switzerland. ISBN 978-3-031-20080-9.

[89] T. Cheng, L. Song, Y. Ge, W. Liu, X. Wang, and Y. Shan. Yolo-world: Real-time open-vocabulary object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16901–16911, June 2024.

A Appendix

A.1 Property Comparison between VIRL and Baselines

Method	Extrapolation task	Visual-feature-dependent task	Task in shifted domain	Cross-embodiment
XIRL [23]	×	✓	×	✓
GraphIRL [24]	×	×	✓	✓
VIRL (ours)	✓	✓	✓	✓

Table 2: Property comparison in different task settings between VIRL and the most related works.

A.2 Extended Related Work

Imitation learning with actions. Imitation learning from expert demonstrations with state and action pairs has demonstrated successes in various approaches [27], from behavioral cloning [28, 29, 30] to inverse reinforcement learning [17, 31, 32]. However, its reliance on ground-truth actions renders data acquisition at scale both costly and challenging, thereby limiting its wider applicability. To leverage unlabeled video demonstrations that are easier to obtain, methods have been developed to infer actions in videos [33, 34, 35, 36, 37, 38, 39]. For example, Pathak et al. [34] utilize a forward model so that, given the current state and the policy's predicted action, it outputs the next state. With the state sequences in state-only demonstrations, both the model and the policy can be trained using the difference between the forward model's output and the ground truth of the next state. A similar forward dynamics model is also used in Edwards et al. [36]. In another example, VPT [39] trains an inverse dynamics model (IDM) on a small labeled dataset and then uses the trained IDM to augment a large amount of unlabeled videos with actions, which are used for the subsequent policy learning via behavioral cloning. These methods have successfully trained agents using unlabeled videos, but are subject to well-defined morphologies and dynamics of demonstrators and imitators, making it hard to leverage demonstrations with diverse embodiments.

Domain adaption for imitation learning. To overcome the domain gap between varied agent embodiments, several works [40, 41, 42, 43, 44, 18] attempt to translate the context in demonstrations to the imitators' context in terms of agents and viewpoints, using methods such as generative adversarial networks [45, 46] and context translation. Furthermore, another way to address the domain gap between agents is by inpainting them from video demonstrations and online visual states [47].

Inverse reinforcement learning from videos. On the other hand, a group of works [21, 22, 48, 49, 50, 44, 51, 52, 53, 24] has focused on learning reward functions from videos by, instead of using any additional models for action prediction or domain translation, extracting latent features that indicate intermediate steps or measure task progress, which are naturally and implicitly present in demonstrations. These learned reward functions are then used for policy training in the regular RL regime. For example, Aytar et al. [48] pretrain a visual encoder on classifying temporal distances between frames in a single video, retrieve a sequence of embedding checkpoints in the video, and then generate reward signals by comparing observations and embedding checkpoints. For learning robotic tasks from human videos, Sermanet et al. [22] train viewpoint-invariant representations on pairs of simultaneous videos with an objective of attracting positive frame pairs while repulsing negative frame pairs in the latent space. They demonstrate that the agent-invariant representations can directly provide reward signals for robot arms. To handle video demonstrations with varied lengths and paces, XIRL [23] adopts temporal cycle-consistency loss [25] to learn visual embeddings that measure task progression and then uses distances between current state embeddings and the goal embedding as reward signal for policy learning via RL. They demonstrate the pretrained encoder benefits from the diversity of demonstrators' embodiments and generalizes to agents with unseen embodiments. Building on XIRL, our work seeks to train an encoder that generalizes to constructing reward functions for task extrapolations by leveraging graph abstractions, while maintaining the flexibility for tasks requiring attention to low-level visual features by encoding object features.

Object-centric scene graphs. Abstracting visual scenes into object-centric scene graphs is beneficial for agents to understand environments and make decisions in various tasks [54], such as dynamics modeling [55, 56], navigation [57, 58, 59], and robotic manipulation [60, 61, 62, 43, 63, 24, 64]. As a follow-up work to XIRL, GraphIRL [24] enhances robustness to diverse object appearances by abstracting frames into graphs, where each node represents an object and contains features including its bounding box coordinate and distances to other objects. However, limitations arise when tasks require attention to object geometries or when dealing with different numbers of objects. Motivated by their work and aiming to address the limitations, we encode object visual features and locations into nodes in a unified way and build graphs such that the encoder can generalize to scenes with arbitrary numbers of objects. Our work can be considered a generalized version of GraphIRL, suitable for both visual-feature-dependent and -independent tasks, as well as unseen task extrapolations.

A.3 Implementation Details

The visual encoder borrows from a pretrained ResNet-18 [81]. We take 512-dimensional embeddings as node features to construct complete graphs and use graph transformer [71] to extract 512-dimensional graph embeddings. We use the graph embeddings for reconstruction from decoder and temporal alignment. We adopt ADAM [82] optimizer and Soft Actor-Critic (SAC) [83] RL algorithm. Hyperparameters are from Zakka et al. [23] with minimum fine-tuning on the number of frames per sequence in pretraining due to memory limitation. Learning rates are 10^{-5} for pretraining and 10^{-4} for RL. Our approach does not rely on any data augmentation. All results represent the mean performance over 5 random seeds.

A.4 Additional Experiment Results of X-MAGICAL

Results in domain-shifted environment. We assess VIRL's performance in shifted domains, specifically with respect to unseen goal region colors and debris shapes, as depicted in Figure 1c. In contrast to GraphIRL, which enhances robustness to unseen environments by disregarding visual features and focusing solely on graph abstractions of coordinates of bounding boxes, we encode both spatial and visual features into embeddings. Our aim is to validate that graph embeddings remain effective for measuring progress despite the incorporation of unseen visual features. Figure 6 shows that our approach significantly outperforms baselines in the short-stick scenario. For gripper, our approach and GraphIRL achieve similar performance, outperforming the rest, which is expected considering that GraphIRL's abstracted spatial information of objects is sufficient for learning this task and is robust to shifted domains. For medium-stick and long-stick, results show that our approach is on par with the best baselines.

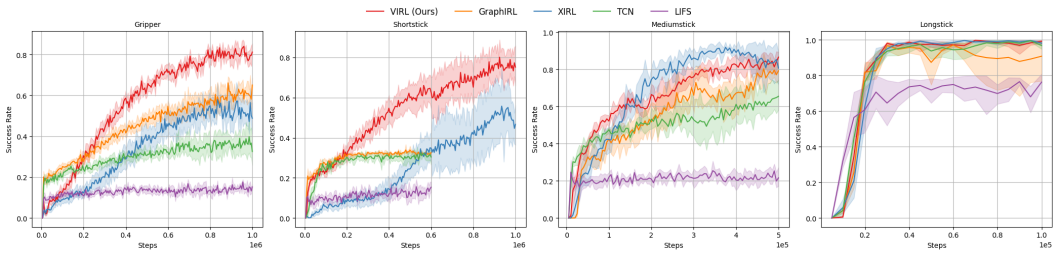
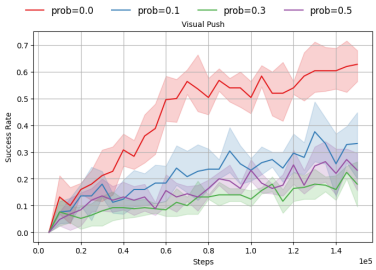
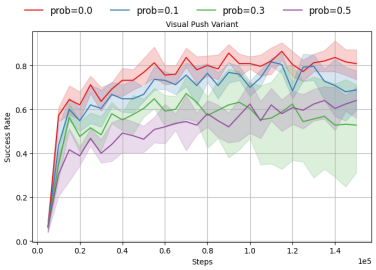


Figure 6: **Cross-embodiment cross-environment.** Encoder is pretrained on demonstrations as shown in Figure 1c (top). Policy learning in diverse environments with randomized goal region color and debris shape as shown in Figure 1c (bottom). Each agent (gripper, short-stick, medium-stick, long-stick) learns from videos of agents with the other 3 embodiments. For example, gripper learns from videos by short-stick, medium-stick, and long-stick.



(a) Robot Visual Pusher - Task I



(b) Robot Visual Pusher - Task II

Figure 7: Impact of object detection failures. Each object is set to be not detected with a probability. We test with probabilities being 0.1, 0.3, and 0.5 for both Task I and Task II.

A.5 Ablation Study

Impact of missed object detection. Our approach requires a reliable object detector in both pre-training and policy learning. The detection performance on collected demonstrations can be exhaustively evaluated, while that on scenes during policy learning is prone to uncertainty. Therefore, we study the effect of object detection failures during policy learning. We randomly mask an object's visual and positional features by setting its crop to be zeroed out with a probability. The objects subject to mask include the robot end effector, the red and green cubes, and the goal. We test with probabilities being 0.1, 0.3, and 0.5 for Robot Visual Pusher Task I and Task II, and the results are shown in Figure 7. The effect brought by failing to detect all objects in scene at a step is that the corresponding reward signal may not accurately reflect the actual task progress. Interestingly, performance drop by detection failure varies between tasks.

A.6 Real-World Robot Setup

In the real-world robot experiment, we apply the trained policies of the variant task of Robot Visual Pusher, i.e., Task II in Figure 5, on a Ufactory xArm 6 robot arm. The observation is captured by a fixed Intel RealSense D435 camera. The robot setup is shown in Figure 8. We utilize the codebase of Diffusion Policy [84, 85] for the policy deployment on the real robot. We also visualize a successful trial and a failed trial as shown in Figure 9a and Figure 9b, respectively.

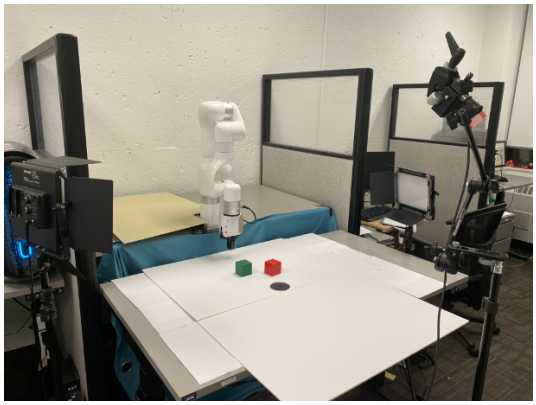
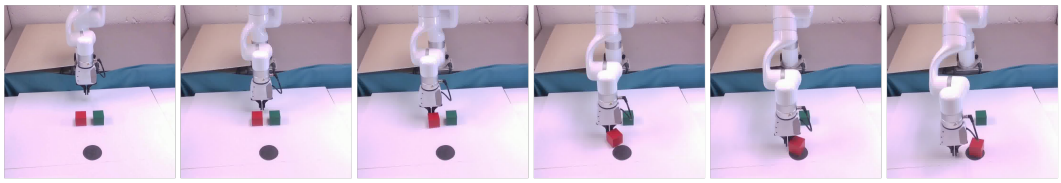


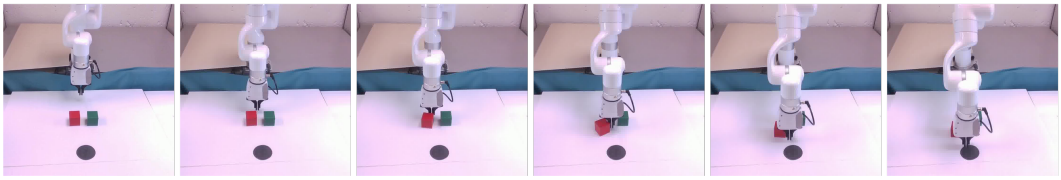
Figure 8: Real-world robot experiment steup.

A.7 Details of Object Detector

Detection for X-MAGICAL. We use YOLO-v8 [86] as the object detector for X-MAGICAL experiments. For encoder pretraining, we train the detector on a small set of images from demonstration



(a) Real-world robot successful trial.



(b) Real-world robot failed trial.

Figure 9: Real-world robot experiment visualization.

data by XIRL [23], and evaluate it to ensure that detection is successful for all demonstration videos. For policy learning, we adopt the trained detector from the pretraining step. Given an image, we apply the detector to detect objects and build the corresponding graph of the image. However, the detection is not always successful during policy learning. We use the image from the previous step when detection failure happens.

Detection for Robot Visual Pusher. We also use YOLO-v8 [86] as the object detector for Robot Visual Pusher experiments, but only in pretraining. Similarly, for encoder pretraining, we train the detector on a small set of images from demonstration in [44], and evaluate it for successful detection of the object and the robot end effector over all demonstration videos. For scenes where the goal is occluded by the object and cannot be detected, we set the goal bounding box to be the summation of the object bounding box and Gaussian noise. For policy learning, we do not use any object detector. Instead, we follow GraphIRL [24] and compute objects bounding boxes in 2D image.

A.8 Technical Comparison between VIRL and GraphIRL

VIRL differs from and improves upon GraphIRL [24] mainly in three aspects: (i) the construction of graphs from images, (ii) the choice of graph encoder, and (iii) the loss function for pretraining.

Graph construction. GraphIRL concatenates an object's bounding box coordinate with its distances to other detected objects as the node feature $\{x_1, y_1, x_2, y_2, d_1, d_2, ..., d_{N-1}\}$. Although robust to domain shift and visual distraction, their design has two downsides: First, it keeps object spatial information via coordinates while disregarding the visual features; Second, a pretrained graph encoder cannot generalize to policy learning for tasks where the number of objects is different from that in demonstrations because of the change of node feature length.

To resolve the first issue, VIRL encodes object-wise visual features into graph nodes. Specifically, VIRL creates an image for each detected object by zeroing out pixels outside of the object bounding box, keeping images shape identical to the original scene image. We then use a convolutional neural network to extract the object's visual feature $\mathbf{f}_{o_i} = \text{ResNet}(I_o)$. In this way, we make our approach applicable for tasks dependent on visual features. To resolve the second issue, we simply make the distance between two nodes the feature of the edge connecting them.

Graph encoder. GraphIRL proposes a Spatial Interaction Encoder for their graphs following [55] and [87] to extract abstract information. In contrast, we adopt Graph Transformer [71] as the graph encoder. We argue that any graph neural network that is capable of extracting decent global features from graphs with edges should be applicable in our approach.

Loss function for pretraining. GraphIRL follows XIRL [23] and uses Temporal Cycle-Consistency (TCC) loss as the loss function for pretraining their graph encoder. In contrast, we follow [73] and

add a reconstruction loss apart from TCC loss to ensure that sufficient object-centric low-level visual features are kept in the graph embedding for learning policies of visual-feature-dependent tasks.

A.9 Extended Limitations and Opportunities

First, our approach relies on reliable object detectors. Although occasional nonconsecutive detection failure may result in limited influence, detection failure in keyframes can be influential. The opportunity brought by this limitation is that our approach may be applicable for more complex tasks by leveraging more powerful open-vocabulary detectors, such as OWL-ViT [88] and YOLO-World [89]. By incorporating object tracking methods, the occlusion cases may be also improved. Second, while our approach is tested on X-MAGICAL, Robot Visual Pusher, and real-robot, which provided valuable initial insights, we acknowledge that it does not fully capture the complexity of kinematics and scene variations present in more challenging benchmarks, such as tasks in RoboSuite [76], Meta-World [77], and BridgeDataV2 [78]. Third, our approach is proposed for rigid object manipulation in the first place. Thus, it is unclear how it would generalize to tasks of deformable object manipulation [79, 80]. Fourth, since our approach focuses on object-centric information while disregarding non-object-centric information, it might struggle with certain tasks that require an understanding of the scene context.