Robotics: Science and Systems 2021
Held Virtually, July 12–16, 2021

Manipulator-Independent Representations for Visual Imitation

Yuxiang Zhou
DeepMind
London, UK
yuxiangzhou@deepmind.com

Yusuf Aytar
DeepMind
London, UK
yusufaytar@deepmind.com

Konstantinos Bousmalis
DeepMind
London, UK
konstantinos@deepmind.com

***Abstract*—Imitation learning is an effective tool for robotic learning tasks where specifying a reinforcement learning (RL) reward is not feasible or where the exploration problem is particularly difficult. Imitation, typically behavior cloning or inverse RL, derive a policy from a collection of first-person action-state trajectories. This is contrary to how humans and other animals imitate: we observe a behavior, even from other species, understand its perceived effect on the state of the environment, and figure out what actions our body can perform to reach a similar outcome. In this work, we explore the possibility of third-person visual imitation of manipulation trajectories, only from vision and without access to actions, demonstrated by embodiments different to the ones of our imitating agent. Specifically, we investigate what would be an appropriate representation method with which an RL agent can visually track trajectories of complex manipulation behavior —non-planar with multiple-object interactions— demonstrated by experts with different embodiments. We present a way to train manipulator-independent representations (MIR) that primarily focus on the change in the environment and have all the characteristics that make them suitable for cross-embodiment visual imitation with RL: cross-domain alignment, temporal smoothness, and being actionable. We show that with our proposed method our agents are able to imitate, with complex robot control, trajectories from a variety of embodiments and with significant visual and dynamics differences, e.g. simulation-to-reality gap.**

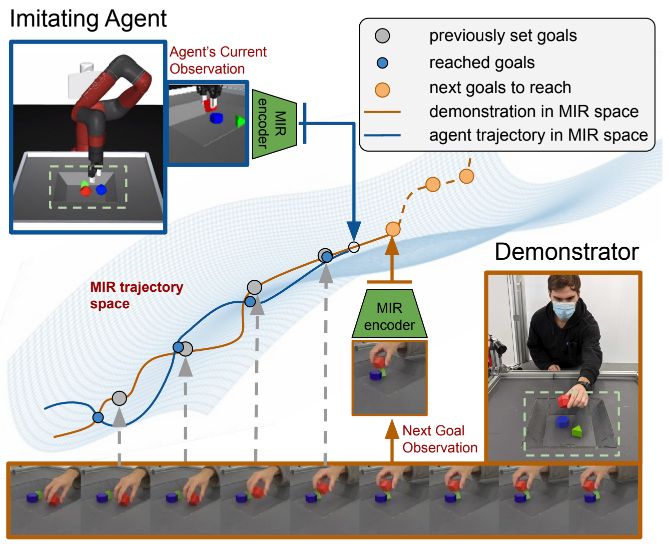

Fig. 1. Our work focuses on learning manipulator-independent representations (MIR) that can be used for imitating trajectories of behavior demonstrated with different embodiments, unseen during training, solely from pixel observations, even in the presence of large visual domain gaps.

I. Introduction

Primates and especially humans depend on understanding actions of others to survive, socially organize, and, in the case of humans, learn new skills by imitation [27]. Thanks to our visuomotor "mirror" neurons, we are able to observe behavior by others, process that visual information, and map it to our own embodiment in order to perform similar actions that arrive at the same results. Replicating a similar imitation performance in robots will surely increase their usefulness, interaction capacity with humans, and significantly decrease the cost of learning new skills. We are motivated by the prospect of a robotic manipulator that can imitate any visually-demonstrated behavior (a sequence of goals) of arbitrary complexity. These goals will need to be perceived, and to provide a context with which an imitating agent can act towards reaching them.

In this work we focus on imitating single trajectories as precisely as possible, *using only vision*, without access to proprioceptive information or actions of the demonstrator[1]. Our primary goal is to explore specifically how we can provide robots with the *perceptual* ability to mirror a demonstrated behavior as accurately as possible, even when that is performed by a "demonstrator" with a different embodiment or in a different domain (simulation vs. reality). In other words we want to equip our robots with the ability of third-person imitation, particularly specialized for cross-embodiment scenarios. We tackle this from a manipulation viewpoint: we learn representations that capture the change effected in the world and details related to it, but ignore some manipulator-specific information, such as the joint angles of a specific robotic arm. A characteristic example from our experiments, shown in Figure 1, is imitating trajectories demonstrated by a human hand, an environment and embodiment both unseen during our perceptual training.

Although the perceptual methods we propose are agnostic to the specific formulation of the cross-embodiment visual imitation task, we study the problem through trajectory tracking with reinforcement learning (RL) [24, 2, 25]. General solutions for reaching visual goals, like goal-conditioned policies [23],

[1]For video examples please visit https://sites.google.com/view/mir4vi

have been shown to work only for simple, short-horizon trajectories in manipulation, and with simple control for the imitating agent [23, 32]. If the behavior demonstrated involves multiple objects with rich dynamics or is not (almost) planar, goal-conditioned visual imitation is much more difficult and has not been shown to work, to the best of our knowledge. The same applies for imitating agents with complex control, for example with actions that correspond to joint velocities, as in this work, as opposed to end-effector positions. For that reason, prior visual imitation work [15] has leaned on trajectory tracking: learning the necessary sequences of actions to regenerate a demonstrated trajectory. This high-fidelity imitation task is in contrast to cloning policies of a demonstrator with different observations but known actions [34]. It is also different from one-shot visual imitation, which focuses on generalizing or adapting existing knowledge to a slight variation of previously seen tasks [15, 6, 5, 8]. This then begs the question: how do we learn pixel-based representations that produce the right information for a cross-embodiment trajectory tracking agent? In this work we identified three characteristics: *(a)* cross-domain alignment, *(b)* temporal smoothness, and *(c)* actionable representations.

Our solution is to learn a common embedding space which has the capacity of aligning trajectories with large domain gaps. We propose training manipulator-independent representations (MIR) in a cross-domain way with a combination of a self-supervised time-contrastive loss, for temporal smoothness, and a goal-conditional skill learning objective, for actionable representations. In order to obtain aligned sequences for cross-domain training, but also to achieve better transferability to the real world and unseen manipulators, we utilize several levels of domain randomization in our simulated environments. This includes an 'invisible arm' environment where the objects move but the manipulator is invisible, as shown in Figure 4. This helps the representation to focus on the actual change in the environment, while other types of randomization help capture the rough position and properties of a manipulator.

Our main contribution, in this work, is a method and a training regime with which we can learn manipulator-independent representations (MIR) for trajectory tracking. Prior work [12, 32] dealing with cross-embodiment visual imitation has been assuming use of data from the demonstrator's environment during perceptual training, and has been focusing on one-shot imitation and not high-fidelity trajectory imitation. Our method for learning MIR representations is a general one, i.e. it is not utilizing specific task or demonstrator information. We show that our MIR approach outperforms other reasonable choices and is able to deal with large domain gaps, difficult control, and complex object dynamics. This is also the first work, to the best of our knowledge, on cross-embodiment trajectory tracking, i.e. visual imitation of robotic manipulation trajectories from unknown embodiments.

The paper is structured as follows: we first provide some background on third-person visual imitation in Section II, particularly in the context of robotics manipulation from pixels. We then describe, in Section III, cross-embodiment visual imitation and the details of our RL based trajectory tracking method. In Section IV we discuss the environments, generated dataset, and demonstrated trajectories. In Section V we give an overview of the evaluations of our claims and discuss our results, and finally we conclude with Section VI.

II. Related Work

Imitation learning is a problem with a wide range of methodologies in literature depending on the precise problem formulation and assumptions. Based on the amount and the kind of demonstration data assumed to be available, we can roughly split existing methods into two categories. There are those that assume ample demonstration data, with known actions, and attempt to generalize a policy from them. Then there are those that use a single previously unseen demonstration, usually with unknown actions, as the specific behavior the agent is asked to imitate.

In the first category, there is an implicit assumption of an intent or task that the imitating agent needs to learn. Two main streams of such works are behavioral cloning [3, 28] and inverse reinforcement learning [29, 20, 7, 10]. The former deals with imitating an expert, typically defined by a set of demonstrations with known actions and observations similar to the ones observed by the imitating agent. Inverse RL deals with the problem of recovering a reward function from demonstrated data and training an RL agent with such a reward. The behavior is defined and restricted by the type of demonstration data collected.

In the second category the agent is provided with a trajectory that defines the imitation task. Here we can include works in one-shot imitation and trajectory tracking, which is where our work also belongs. One-shot imitation deals with generalizing or adapting to a novel task at test time. This is typically defined by a first-person trajectory of a slight variation of previously seen tasks, e.g. a new viewpoint or a new scene configuration [15, 21, 6, 5, 8]. While one-shot imitation is about generalizing across task variations, trajectory tracking is about high-fidelity replication of a specific sequence of goals. Feature tracking of trajectories [24, 2, 25, 22] with RL has been successfully used primarily for locomotion and with hand-engineered features. Tracking from such features is not always feasible and is mostly based on task-specific solutions (e.g. mocap data).

Using vision for tracking is desirable but expensive if doing it end-to-end from pixels with RL, even when the demonstrator and imitator domains are known [23, 15]. Visual trajectory tracking for manipulation tasks introduces additional challenges due to the stochasticity of the interaction with the unactuated objects. In our work we also use RL for visual tracking of manipulation trajectories demonstrated with an embodiment and domain unknown during our perceptual training. Although a generalized trajectory-conditioned imitation solution akin to goal-conditional policies would be desirable, they have been shown to be effective only in single-embodiment cases, for simple near-planar tasks [23] and primarily from features [16, 21].

Our biggest challenge is dealing with the domain gap in the manipulator (e.g. human hand vs. robotic gripper), in the low-level characteristics of the visual observations, and even in the physics (e.g. simulation vs. reality). Current imitation learning approaches from pixels are mainly tested within a single domain [23, 24, 17, 38], domains with a limited gap [2, 15, 34], or require manually labelled data [25]. For example, the third-person imitation work by Stadie et al. [34] deals with known actions, observations from the same simulated embodiment, but with a different viewpoint. On the other hand, imitation learning for large domain gap happens in model dependent state space and primarily for locomotion tasks [18, 9, 25, 26, 24].

The works closest to ours are Liu et al. [15], Sharma et al. [32], Smith et al. [33] and James et al. [12]. Similar to our work Liu et al. [15] deals with RL-based trajectory visual tracking with a domain gap. In this particular case the gap has to do with a viewpoint difference. Although the demonstrations are performed by a human, the observations are carefully cropped to leave the manipulators out of the scene. Both [32, 33] deals with cross-embodiment demonstrations in the context of one-shot imitation. Sharma et al. [32] uses temporally aligned human and robot demonstration videos in order to learn how to propose goals for robot given a human video. They operate in a predefined task space and scaling this method to general imitation would be quite challenging as it requires manually collected aligned trajectories. In [33] a technique using CycleGAN [39] is proposed to translate human demonstrations to robot-looking ones at pixel space. More similar to our work, in James et al. [12], domain randomization and models of human hands are used in simulation to be able to train the perceptual stack to generalize to human hand demonstrations in the real world. An important difference to all of these works is that we do not assume a priori the knowledge of the demonstrator domain. With the exception of [12], these works use data collected in the demonstrator domain as part of their perceptual training process. James et al. [12] nevertheless assumes a priori that the demonstrations will be from a human hand. In contrast, our representation are manipulator-independent and our proposed method can successfully track demonstrations from a variety of previously unseen manipulators.

III. Cross-embodiment visual imitation

In this section we explain how we learn to imitate unconstrained manipulation trajectories executed by previously unknown manipulators, such as a human hand, using only *visual observations*. Our imitation method is composed of two main phases: *(a)* learning a manipulator-independent representation (MIR) space (see Figure 2), and *(b)* cross-embodiment visual imitation through reinforcement learning using the pre-trained MIR space (see Figure 1).

A. Manipulator-Independent Representations

Perhaps the most crucial part in visual imitation by trajectory tracking is to have a high-level feature space that can

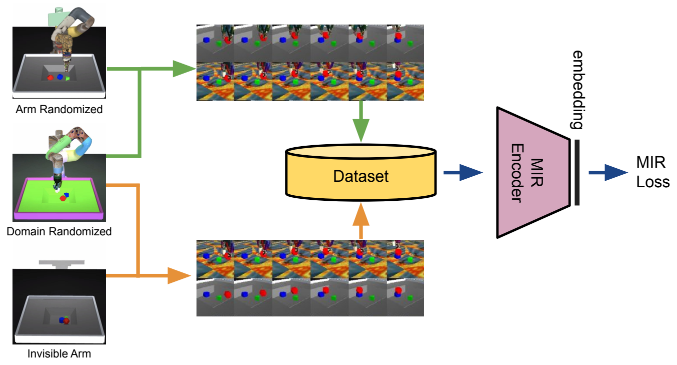

Fig. 2. Learning manipulator-independent representation (MIR) space. MIR is trained on a dataset generated using two pairs of environments: (a) Domain-randomized and 'invisible arm' environment and (b) domain-randomized and arm-only-randomized environments. Please see Section IV-B for the details of dataset collection.

match demonstrator and agent observations adequately. In this work we address learning such a space that specializes in manipulator-independent imitation. We identified three main properties of the desired feature space that are crucial for successful imitation from unseen manipulator trajectories. These are *(a)* cross-domain alignment, *(b)* temporal smoothness, and *(c)* being actionable (suitability for RL).

Cross-Domain Alignment. One of the most daunting challenges in cross-embodiment imitation is the *domain gap* between the demonstrator trajectory and the agent's own environment. The domain gap could be due to the change in observations (e.g. viewpoint, simulation vs. reality), the manipulator (e.g. agent morphology and action space) or dynamics of the environment (e.g. simulation vs. reality). We deal with the domain gap by learning manipulator-independent representations with the help of domain-randomized simulated environments and domain alignment methods.

The main target in domain randomization [30, 36] is to randomize the observations and dynamics of the simulated environment with the hope that an agent trained under this setting will be robust against real-world visual observations and dynamics. In domain randomization the representation space and the policy are typically tightly coupled. Therefore the manipulator's body, action space and the task at hand are strongly ingrained in the representation space. In our work we decouple the domain-randomized representation from manipulator type and task-specific features. Hence the learned representation space can be utilized for any task with any manipulator. This is a crucial step in our work as the target is to transfer behavior across manipulators with unobserved (e.g. human hand) morphology and action space without being constrained by any task. In order to achieve that, we need a representation space that is both domain-invariant and can capture a good high-level understanding of the environment.

In order to learn such an abstract representation for cross-embodiment imitation, we need to consider what that entails. When a robot imitates a trajectory, it can either *(a)* mimic the movements of the manipulator, or *(b)* replicate the effect of the manipulator on the environment. To illustrate this dilemma consider the demonstrated trajectory of stacking one object on top of another. Assume that during imitation the robot has grasped the first object but dropped it halfway while moving to the other object. Now the robot has to make a decision between following the path of the imitated manipulator blindly or picking up the dropped object. The latter would mean it prioritizes matching the changes in the environment more than following the manipulator. While learning MIR we make sure the representation can capture the changes in the environment without ignoring the rough motion of the manipulator. We achieve this by training our domain alignment methods on two pairs of environments: *(a)* the 'domain-randomized' and 'invisible arm' environments which help better capture the changes in the environment; and *(b)* the 'domain-randomized' and 'arm-randomized' environments which further make sure that our representations encode information about the manipulator without paying attention to its specific characteristics. These environments are illustrated in Figures 2 and 4, and discussed in more detail in the Section V.

Temporal Smoothness implies that observations that are temporally neighbors have similar representations, even when they are from different domains. It is an important property particularly if the representation space will be used for reaching goals specified through visual observations. Using pairs of temporally-synchronized trajectories from the two domain pairs mentioned above we learn our temporally-smooth representations, which we call Temporally-Smooth Contrastive Networks (TSCN), by building upon the widely-used Time-Contrastive Networks (TCN) [31].

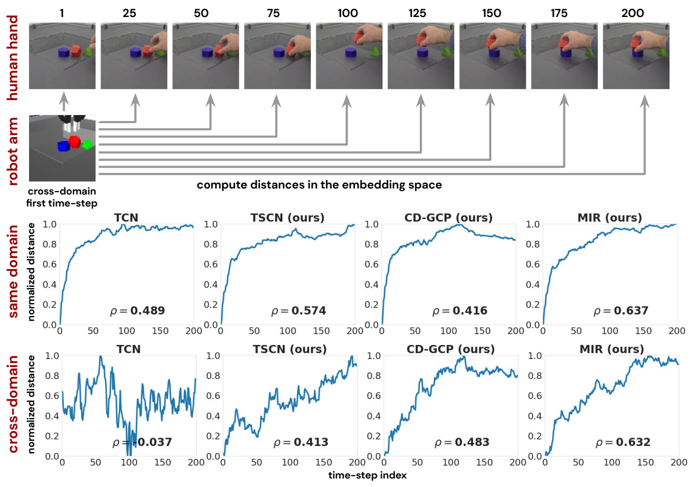

Fig. 3. **Embedding space distances for same-domain and cross-domain goals.** The first time-step is assumed to be the current observation and the goals are selected over the entire demonstration sequence with increasing temporal distance. Note that objects are manually aligned in simulation to match the first observation in the real-world sequence. The distance plots are normalized to $[0, 1]$ for each trajectory and aggregated across 10 trajectories. We also provide the Spearman's rank correlation between the reachability distance (i.e. linear increase of distance over time) and embedding distance for each of the trajectories. The mean rank correlations over all 10 trajectories are displayed on each plot. Note that TCN performs reasonably well in the same domain but not across domains. MIR and its two components separately (i.e. TSCN and CD-GCP) are significantly better correlated with the reachability across domains. MIR has similar performance within and across domains.

TCN learns representations by enforcing higher similarity between temporally-aligned observation pairs across two different camera views, compared to any other pair selected within a single-view sequence. In our context the views are the two randomized domains that we want to align. Formally, given two temporally-aligned trajectories of length N from two domains $\mathbf{o} = \{o_i\}_{i=1}^N$ and $\bar{\mathbf{o}} = \{\bar{o}_i\}_{i=1}^N$, we first apply the encoder ϕ to obtain the latent representations $x_i = \phi(o_i)$ and $\bar{x}_i = \phi(\bar{o}_i)$ for any observation. Given the aligned encoded sequences $\mathbf{x}$ and $\bar{\mathbf{x}}$, the TCN objective that we minimize is $\min_\phi \left(-\sum_i^N \log \frac{\exp(x_i^T \bar{x}_i)}{\sum_j^N \exp(x_i^T \bar{x}_j)} \right)$. We particularly use the n-pairs implementation as opposed to the triplet implementation due to its robustness [31]. In this work we chose $n = 50$. n-pairs TCN essentially classifies the matching pair among all other non-matching negative pairs within the given two aligned trajectories. One drawback of TCN is that it penalizes misclassification of the negative pairs $(x_i, \bar{x}_{i+1})$ and $(x_i, \bar{x}_{i+50})$ equally, though it is more acceptable to confuse temporally nearby observations as opposed to temporally distant pairs.

In order to incorporate this observation in the loss function of TSCN, we update the softmax cross-entropy objective by providing a temporally-smooth distribution p_i as shown below:

$$\min_\phi \left(-\sum_i^N \sum_k^N p_{ik} \log \frac{\exp(x_i^T \bar{x}_k)}{\sum_j^N \exp(x_i^T \bar{x}_j)} \right), \tag{1}$$

$$\text{where} \quad p_{ik} = \frac{exp(-|i-k|)}{\sum_u^N exp(-|i-u|)}$$

This objective encourages the learned representation to be more temporally smooth. We also include negative pairs that are populated from other sequences in the same mini-batch which is commonly applied in many pair-based contrastive learning objectives [4, 35]. This further improves the alignment quality in the presence of distractors from other episodes.

Moving one step further from temporal smoothness, it is desirable if the distance between the current and goal observations in the embedding space correlates well with the *reachability distance*, the minimum time required to move from the current state to the goal state. Of course the reachability distance is very hard to measure as we don't have optimal policies that can take us from one state to another in the shortest available time in the real world. However, assuming that humans are near optimal agents, we can utilise human trajectories to analyse the relationship between the embedding distance and reachability distance. We collected 10 stacking trajectories with different object configurations where a human hand reaches one colored object and stacks it on top of another colored object. Considering there is no going back and forth in a stacking trajectory, it is safe to assume that reachability distance between the first time-step and all the other time-steps are linearly increasing over time for human hand trajectories. In Figure 3 we illustrate the embedding space distance between the first time-step and all the remaining time-steps both within and across domains.

Note that objects are manually aligned in simulation to match the first observation in the real-world sequence. A linear increase of distance would mean a great correlation between the embedding space distance and the reachability distance. As it is clear in the Figure 3, cross-domain distances in our TSCN embedding space has a much smoother increase over time as opposed to the TCN method. For a quantitative measurement we also computed the spearman's rank correlation between the reachability distance (i.e. linear increase of distance over time) and embedding distance for each of the trajectories, and the mean rank correlation over all 10 trajectories are printed on the plots for each method.

Actionable Representations. Another desired property of a good representation space is being actionable, i.e. amenable for RL training. This requires encoding actions while learning the representation. Note that both TCN and TSCN do not utilise actions. To accommodate this we used offline goal-conditioned policy trained in a cross-domain way as an additional method to enrich our representation space. As we have cross-domain aligned trajectories in our dataset, we also introduce offline cross-domain goal conditioned policy training as a way to increase robustness across the domains. Goal conditioned policies are trained to reach a goal that is provided as part of the input [14, 19, 23]. While in theory they can reach any goal, in practice their effectiveness is constrained to nearby goals [19].

Given the current observation o and a goal observation g that is reachable in the near future, a goal conditioned policy $\pi(o, g)$ predicts one step of action $\hat{a}$ that would bring the agent closer to g. We can train such a policy using a large set of episodes collected from the environment. Formally, given any N-step sequence $\{(o_i, a_i), (o_{i+1}, a_{i+1}), ..., (o_{i+N}, a_{i+N})\}$ extracted from an episode, we take o_i as the current observation and pick a goal randomly from one of the N observations in the future. Then we simply train the goal conditioned policy by minimizing the squared Euclidean loss $||\pi(o_i, o_{i+j}) - a_i||^2$ where $1 \leq j \leq N$.

The extension to our Cross-Domain Goal-Conditional Policies (CD-GCP) is then straightforward. Given a paired N-step sequence $\{(o_{i+j}, a_{i+j})\}_{j=0}^{N}$ and $\{(\bar{o}_{i+j}, \bar{a}_{i+j})\}_{j=0}^{N}$ from two domains, we train a cross-domain goal conditioned policy $\pi_\times$ by minimizing the loss $||\pi_\times(o_i, \bar{o}_{i+j}) - a_i||^2$ where $1 \leq j \leq N$. Note that the goals in this objective come from the other domain. We set $N = 20$ in our experiments.

In order to obtain our final **MIR** space we train CD-GCPs together with TSCN. The final CD-GCP loss is $||\pi_\times(\phi(o_i), \phi(\bar{o}_{i+j})) - a_i||^2$ where the embedding network ϕ is shared with the TSCN objective described in Equation (1). As a result the learned representation becomes more effective as the observation and goal embeddings share the same space, which is optimized for reducing the domain gap. All the objectives introduced so far are optimized for all of the aligned sequences in our dataset, which is described in Section IV-B. As CD-GCPs are trained for reaching cross-domain goals, the embedding distances in the learned representation space also correlates well with reachability distance. Moreover, the MIR space, which a combination of both TSCN and CD-GCP, achieves the best rank correlation for reachability distance prediction between the simulated environment observation and real-world human hand trajectories as shown in Figure 3.

B. *Cross-Embodiment Imitation via RL*

In this section we describe how an RL agent operating in a simulated environment learns to follow a given trajectory utilizing the frozen pre-trained MIR embedding space. Given a cross-embodiment trajectory $\{\bar{o}_i\}_{i=1}^{N}$ as a sequence of observations to be followed, we first uniformly sample goals $\{\bar{g}_i\}_{i=1}^{M}$ where $M < N$, and obtain their embeddings by running them through the encoder $\phi(\bar{g}_i)$. In practice, we observed that the optimal goal sampling rate is every 5 to 10 observations. Only one goal is visible to the agents at anytime. Once an agent reaches the goal, the next goal becomes the active one, as illustrated in Figure 1. The agent should reach each intermediate goal sequentially till the end to claim a successful attempt. We define our goal-based reward as described below:

$$r(o, \bar{g}) = \begin{cases} 1, & \text{if } \exp(-w||\phi(o) - \phi(\bar{g})||^2) > \epsilon \\ 0, & \text{otherwise} \end{cases}$$

where o is the current observation of the agent and $\bar{g}$ is the next available goal. w serves as the normalization factor which is defined as the mean of the element-wise Euclidean distances between the neighboring encoded observations in the demonstration. ϵ is randomly sampled in range [0, 1] during parameter sweep and turned out to be optimal around 0.3. Once the agent meets the current goal (i.e. reward above ϵ), the next goal becomes visible to the agent. In this work, we use Maximum a Posteriori Policy Optimization (MPO) [1] for training our RL agent.

IV. Environments and Data

In this section, we present all the environments used for our experiments. We also discuss details about the dataset used for MIR training. Finally, we describe the trajectories used to evaluate MIR and other baselines, when used for RL imitation.

A. *Environment Details*

Our experiments involve 8 domains, as illustrated in Figure 4. Our imitating agents always act in a **canonical** MuJoCo [37] simulated environment. This is designed to have similar appearance and dynamics to our equivalent real environment with a Sawyer arm and a Robotiq 2-finger gripper. The objects used are three geometric props generated procedurally. Two cameras are placed at the front left and right of the basket to generate the 128×128 pixels observations that the agents use.

As a slight variation of our canonical simulation environment the Sawyer and the Robotiq are rendered *visually* **invisible**. This environment is used for learning our manipulator-independent representations that focus on environmental changes and ignore the manipulator. In order to

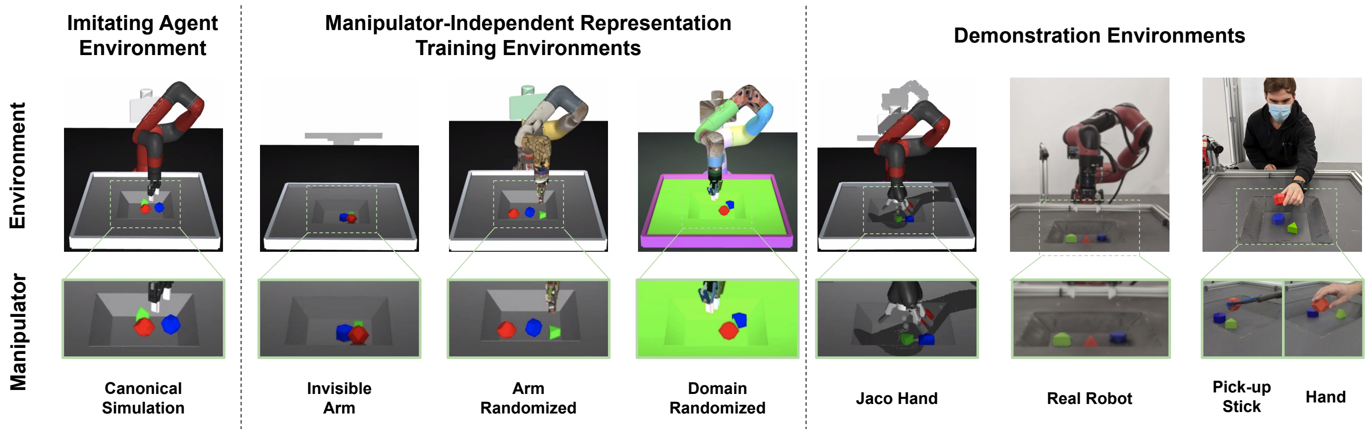

Fig. 4. **Visualization of the different environment used in our work.** Our imitating agent always operates in a canonical simulated environment (left). We have 3 additional versions of it which are used to generate data for perceptual training (middle). Finally we use 4 held-out demonstration domains, both in simulation and in the real world, for cross-embodiment imitation.

aid MIR training we also use two randomized versions of our canonical environment. In both versions we randomize colour, texture, lighting, and camera poses to create a large visual diversity. Physics properties (e.g. mass, friction, armature, damping, gear) are perturbed uniformly within $\pm 10\%$ of their respective values in the canonical environment. In the first, **arm-randomized** version, we only apply the randomizations on the arm and the gripper. In the second, general **domain-randomized** environment, those are applied on every entity in the simulated environment.

For our cross-embodiment imitation experiments we use 4 domains unseen during MIR training. These are presented in order of increased difficulty, in terms of the differences to the three environments above. Firstly, we evaluate our method with trajectories from a version of our canonical simulated environment which has the 2-finger Robotiq replaced by a 3-finger **Jaco hand** [11]. This already challenges cross-embodiment visual tracking as there is a significant embodiment gap. The following 3 domains are all in the real world. In our **real robot** environment, which is similar to our canonical simulation, we have a Sawyer arm with a Robotiq 2-finger gripper mounted behind a basket with 3 objects. These are the 3D printed versions of the procedurally-generated objects we use in simulation. The gripper is fitted with custom 3D printed fingertips. Similar to the simulation, there are two cameras at the front left and right of the basket which provide 128×128 resolution images to the agents. However, the low-level appearance and the physics are quite different to those in our simulated environment. In our most challenging environments, we use the same real environment but we ourselves instead directly manipulate the objects with a **pick-up stick** or our **hand**.

B. Dataset for MIR Training

As explained in section III, learning manipulator-independent representations with our method requires temporally-aligned sequences. In our work, this is collected by utilizing simulation and domain randomization to create different visual versions of the same trajectories that can be used for our perceptual training with TSCN and CD-GCP.

In total we generated $7,194$ canonical trajectories of reaching, grasping, lifting, and stacking with all combinations of the three objects in the basket. These were replicated in the 'invisible arm', 'domain-randomized', and 'arm-randomized' environments. The canonical trajectories were discarded and, as illustrated in Figure 2, the trajectories from the 'domain-randomized' environment were paired with the equivalent trajectories of each of the other domains for a total of $14,388$ episodes. Out of these, $10,792$ paired trajectories were used for MIR training. The rest were held out for model selection and hyperparameter tuning such as picking the learning rate, batch size and number of epochs. The episodes in our dataset were collected by multiple stacking policies trained with a staged reward for reaching, grasping, lifting, and stacking. They were trained in the canonical simulation environment from fully-observed states, separately for each color pair. As we used policies at multiple points during their training, we have a training dataset that consists of a diverse set of manipulation trajectories, and we assume that most of the state space of object configurations is reasonably represented. A sample trajectory is 200 time-steps, i.e. 10 seconds long with a control frequency of 20Hz. The policies were trained using Maximum a Posteriori Policy Optimization (MPO) [1] following the implementation details of Jeong et al. [13] which solved a similar stacking task.

C. Evaluation Trajectories

In order to evaluate the cross-embodiment aspect of imitation in a principled way we picked a single type of behavior for the trajectories to imitate. We chose that to be stacking one geometric object on top of another in the presence of a third distractor object. There are multiple reasons for this decision. Firstly, having the same type of demonstrations makes it possible to compare the complexity of different environments and how our method deals with increasing levels of domain gap. Secondly, instead of analyzing our results with respect

	Lifting Success					Stacking Success				
Method	invisible	jaco	real robot	stick	hand	invisible	jaco	real robot	stick	hand
TDC [2]	67%	38%	10%	21%	31%	0%	2%	0%	6%	0%
GCP	**100%**	50%	39%	13%	10%	0%	0%	0%	0%	0%
TCN [31]	79%	85%	27%	12%	0%	0%	0%	0%	0%	0%
CMC [2]	80%	**100%**	80%	26%	0%	20%	66%	0%	0%	0%
MIR (ours)	**100%**	**100%**	**100%**	**44%**	**50%**	**38%**	**81%**	**29%**	**17%**	**11%**

TABLE I
QUANTITATIVE COMPARISON OF REPRESENTATION METHODS FOR CROSS-EMBODIMENT IMITATION. EVALUATION IS PERFORMED ON 100 ATTEMPTS OF FOLLOWING FOR EACH OF THE 10 DEMONSTRATED STACKING TRAJECTORIES ON FIVE DOMAINS: INVISIBLE-ARM, JACO HAND, REAL ROBOT, PICK-UP STICK AND HUMAN HAND. THE METRICS ARE THE SUCCESS RATE OF IMITATING LIFTING AND STACKING BEHAVIOURS.

to a tracking reward that is hard to interpret, a multi-stage behavior allows us to intuitively illustrate our analysis of imitation success in terms of achieving the different progressive stages of the demonstrated behaviors. In the Tables I and II, these stages are '**lifting** the top object' and '**stacking** the top object onto the bottom one.' We excluded 'reaching the top object' for clarity, as it is a simple motion that does not involve manipulation. Thirdly, we believe stacking is a task that would challenge existing trajectory tracking methods even in the same-embodiment setting, provided it is performed in an environment with realistic physics and complex control. The action space of our agents is the velocities of the robot joints and gripper, and the physics of our simulated environment are reasonably close to the ones in the real world. Stacking involves multi-object interaction and rich contact dynamics, especially with the geometric objects we have chosen. It clearly exposes the difficulty of imitation once we move away from simple, almost planar tasks typically showcased in works on imitation from observation only. Finally, stacking allows for relatively simple automated evaluations in the real world. Note that the imitating agent is behavior-agnostic and is always just tasked with following a given trajectory with high precision. Behavior generality is only limited by the type of perceptual training and the data used for it. In Appendix B and in our supplementary material we provide examples of how our method is able to generalize beyond the data it was trained with.

We collect 10 sequences from each of the five demonstration domains used during evaluation: 'invisible arm', 'Jaco hand', 'real robot', 'pick-up stick', and 'human hand'. Those were chosen to be of an increasing domain gap with the canonical simulation environment of our imitating agent, in terms of embodiment, vision and dynamics. For the 'invisible arm' domain, which was also used during MIR training, we used held-out trajectories collected with the stacking RL policies as discussed above. The rest of the domains were unseen by the visual feature encoders used for imitation. For most of the held-out demonstration environments ('Jaco hand', 'pick-up stick', and 'human hand') the trajectories were manually collected via teleoperation or by hand. For the 'real robot' we instead used successful trajectories from the zero-shot evaluation of a variant of the policy we used for MIR training, trained in the domain-randomized environment.

V. EXPERIMENTAL EVALUATION

In this section, we demonstrate the efficacy of our method by providing comprehensive quantitative analysis on our design choices and comparing it with a few strong baselines. As discussed above, we evaluate how well each perceptual method aids RL-based imitation of 10 trajectories from each of the five demonstration environments. During imitation, in order to start from a similar initial condition, we manually set the positions of objects similar to the beginning of the target trajectory. Furthermore, our simulation environment exhibits non-deterministic behavior when resetting to a fixed initial condition. Such stochasticity is introduced due to our modelling of the robot. Therefore, for each trajectory collected, we evaluate each agent 100 times and report the lifting success rate and stacking success rate for a total of $1,000$ trajectories for each domain-method combination. As mentioned above, lifting and stacking success are reported to provide a clear picture of what we would like to test for during trajectory following, i.e. that the environmental changes are the same in the demonstration and the trajectory produced by an agent. First we compare our method with several strong baselines for cross-embodiment trajectory tracking, and then we'll discuss the contribution of each component in MIR.

A. *Comparison of Imitation Performance*

We perform cross-embodiment imitation using a few strong baseline representations and our MIR method using an MPO agent as described in Section III-B. We report the performance of successful imitation of trajectories at two stages, lifting and stacking success, in Table I.

The first group of baselines doesn't utilize the paired nature of the trajectories across domains. These are: naïve Goal-Conditioned Policies (GCP), as described in section III-A; and Temporal Distance Classification (TDC) [2], which learns a representation by classifying temporal distance between any given pair of observations within an episode. For a more detailed summary of these baselines see Appendix C. Note that TDC uses all three training domains but not the temporally paired nature of the data. GCP is trained using target goals within the same domain, and used the 'domain-randomized' and 'arm-randomized' data, as the 'invisible arm' data in a single-domain setting would hurt its performance given that GCP is tasked with predicting actions given, among others, the position of the gripper. The second group of baselines is

	Lifting Success					Stacking Success				
Method	**invisible**	**jaco**	**real robot**	**stick**	**hand**	**invisible**	**jaco**	**real robot**	**stick**	**hand**
TCN [31]	79%	85%	27%	12%	0%	0%	0%	0%	0%	0%
TSCN	**100%**	**100%**	**100%**	20%	40%	0%	67%	0%	0%	7%
CD-GCP	86%	44%	50%	20%	0%	0%	8%	0%	0%	0%
CD-GCP + TCN	90%	**100%**	89%	22%	20%	20%	38%	8%	0%.	8%
MIR (CD-GCP + TSCN)	**100%**	**100%**	**100%**	**44%**	**50%**	**38%**	**81%**	**29%**	**17%**	**11%**

TABLE II
ABLATION STUDY OF EACH COMPONENT IN OUR METHOD. EVALUATION IS PERFORMED ON 100 ATTEMPTS OF FOLLOWING FOR EACH OF THE 10 DEMONSTRATED STACKING TRAJECTORIES ON FIVE DOMAINS: INVISIBLE-ARM, JACO HAND, REAL ROBOT, PICK-UP STICK AND HUMAN HAND. THE METRICS ARE THE SUCCESS RATE OF IMITATING LIFTING AND STACKING BEHAVIOURS.

domain alignment methods which explicitly utilize paired trajectories. These are: Time-Contrastive Networks (TCN) [31], as described in section III-A; and Cross-Modal Distance Classification (CMC)[2], which learns a representation by classifying temporal distance between any given pair of observations across domains (see Appendix C). Our MIR method also belongs in this category. Overall, both for imitating lifting and stacking behaviors, domain alignment methods perform much better. MIR clearly achieves the best performance in all test domains for both lifting and stacking stages.

MIR perfectly imitates simulated ‘Jaco hand’ and ‘invisible arm’ sequences for lifting, and achieves 81% success rate for ‘Jaco hand’ and 38% performance for ‘invisible arm’ for completing the full trajectory of stacking. Note that ‘Jaco hand’ is never seen during training of MIR. Although ‘invisible arm’ is used during MIR training, following demonstrations in this environment remains challenging as the manipulator is invisible, and the camera observation of the scene remains static while there is no contact with the objects. Because of that, the first goals are likely to be identical to the first frame and not informative. As imitation relies solely on vision, this makes it harder for the arm to reach the object that needs to move later in the demonstration.

As the domain gap increases in both visual perception and dynamics the performance of methods decrease across the board for real-world sequences. This clearly demonstrates how hard it is to generalize to unseen real-world settings. In addition, the increasing complexity of the gripper morphology (i.e. real robot, pick-up stick, and human hand) also affects the performance in a negative way for all methods. However, MIR still achieves excellent imitation performance for the lifting stage of real sequences. More importantly, MIR is the only method that can successfully imitate some of the real-world sequences for the full stacking trajectories in all three real-world settings.

Imitating a stacking behavior (i.e. a long horizon episode with multi-object contact-rich dynamics) proves to be significantly harder than lifting, and only CMC and MIR achieve considerable performance when it comes to stacking. This clearly demonstrates the need for further research in visual imitation of multi-stage contact-rich sequences.

B. Ablation Study

As shown in Table I our proposed MIR method is able to achieve better imitation compared to other baselines we investigated. As MIR is a combination of TSCN and CD-GCP training, this begs the following question: are all the components of MIR training crucial?

We investigated whether certain aspects of our proposed MIR are indeed important and present our findings in Table II. We first examined whether combining TSCN and CD-GCP for our proposed MIR method is indeed obtaining better representations for imitation, compared to the two components individually. We also looked into the importance of the temporal smoothness aspect of TSCN. We did so by replacing our TSCN loss with the standard TCN [31] one, since former is a temporally-smooth variation of the latter, as discussed in Section III-A. It is clear from the results that using TSCN is preferable to TCN, both when using it on its own and when combining it with CD-GCP, with performance gains that are quite significant and consistent across the domains and stages of imitation.

In this experimental section we have demonstrated that our MIR method has the right characteristics for cross-embodiment visual imitation with RL. The interested reader can find a t-SNE visualization of the learned embedding spaces and more qualitative examples in the Appendix and our project website[2].

VI. CONCLUSION

In this work we explored cross-embodiment visual imitation of robotic manipulation trajectories. We demonstrated the importance of the representation space in visual imitation, and introduced manipulation-independent representations (MIR) as a suitable candidate that can support successful imitation of behaviors demonstrated by previously unseen manipulator morphologies. This includes imitating a real human hand with a simulated robot. Through MIR, we emphasised what is important in learning manipulator and task independent representations for imitation. They need to be robust to domain gaps, and mainly focus on the change of object configurations, while maintaining the ability to capture the rough notion of a manipulator. They also need to be temporally smooth and actionable. These are addressed by introducing TSCN for the former and CD-GCPs for the latter. All these claims are validated through extensive quantitative experimentation.

[2]https://sites.google.com/view/mir4vi

Acknowledgments

We gratefully acknowledge Rae Chan Jeong for helping with simulator modification, David Khosid, Claudio Fantacci, and Federico Casarini for their support in manipulating real robots and collecting human demonstrations and Serkan Cabi for fruitful discussions and proofreading.

References

[1] Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a posteriori policy optimisation. *arXiv preprint arXiv:1806.06920*, 2018.

[2] Yusuf Aytar, Tobias Pfaff, David Budden, Thomas Paine, Ziyu Wang, and Nando de Freitas. Playing hard exploration games by watching youtube. In *Advances in Neural Information Processing Systems*, pages 2930–2941, 2018.

[3] Michael Bain and Claude Sammut. A framework for behavioural cloning. In *Machine Intelligence 15*, pages 103–129, 1995.

[4] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[5] Sudeep Dasari and Abhinav Gupta. Transformers for one-shot visual imitation. *arXiv preprint arXiv:2011.05970*, 2020.

[6] Yan Duan, Marcin Andrychowicz, Bradly Stadie, Openai Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba. One-Shot imitation learning. In I Guyon, U V Luxburg, S Bengio, H Wallach, R Fergus, S Vishwanathan, and R Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 1087–1098. Curran Associates, Inc., 2017.

[7] Chelsea Finn, Sergey Levine, and Pieter Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *International conference on machine learning*, pages 49–58, 2016.

[8] Chelsea Finn, Tianhe Yu, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. One-shot visual imitation learning via meta-learning. In *Conference on Robot Learning*, pages 357–368. PMLR, 2017.

[9] Leonard Hasenclever, Fabio Pardo, Raia Hadsell, Nicolas Heess, and Josh Merel. CoMic: Complementary task learning & mimicry for reusable skills. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 4105–4115. PMLR, 13–18 Jul 2020. URL http://proceedings.mlr.press/v119/hasenclever20a.html.

[10] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. In *Advances in neural information processing systems*, pages 4565–4573, 2016.

[11] JACOArm. JACO Arm. https://robots.ieee.org/robots/jaco/.

[12] Stephen James, Paul Wohlhart, Mrinal Kalakrishnan, Dmitry Kalashnikov, Alex Irpan, Julian Ibarz, Sergey Levine, Raia Hadsell, and Konstantinos Bousmalis. Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12627–12637, 2019.

[13] Rae Jeong, Yusuf Aytar, David Khosid, Yuxiang Zhou, Jackie Kay, Thomas Lampe, Konstantinos Bousmalis, and Francesco Nori. Self-supervised sim-to-real adaptation for visual robotic manipulation. *arXiv preprint arXiv:1910.09470*, 2019.

[14] Leslie Pack Kaelbling. Learning to achieve goals. In *IJCAI*, pages 1094–1099. Citeseer, 1993.

[15] Yuxuan Liu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Imitation from observation: Learning to imitate behaviors from raw video via context translation. July 2017.

[16] Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, Sergey Levine, and Pierre Sermanet. Learning latent plans from play. In *Conference on Robot Learning*, pages 1113–1132, 2020.

[17] Ajay Mandlekar, Danfei Xu, Roberto Martín-Martín, Silvio Savarese, and Li Fei-Fei. Learning to generalize across long-horizon tasks from human demonstrations. *arXiv preprint arXiv:2003.06085*, 2020.

[18] Josh Merel, Leonard Hasenclever, Alexandre Galashov, Arun Ahuja, Vu Pham, Greg Wayne, Yee Whye Teh, and Nicolas Heess. Neural probabilistic motor primitives for humanoid control. In *International Conference on Learning Representations*, 2018.

[19] Soroush Nasiriany, Vitchyr Pong, Steven Lin, and Sergey Levine. Planning with goal-conditioned policies. In *Advances in Neural Information Processing Systems*, pages 14843–14854, 2019.

[20] Andrew Y Ng, Stuart J Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, page 2, 2000.

[21] Openai OpenAI, Matthias Plappert, Raul Sampedro, Tao Xu, Ilge Akkaya, Vineet Kosaraju, Peter Welinder, Ruben D'Sa, Arthur Petron, Henrique Ponde de Oliveira Pinto, Alex Paino, Hyeonwoo Noh, Lilian Weng, Qiming Yuan, Casey Chu, and Wojciech Zaremba. Asymmetric self-play for automatic goal discovery in robotic manipulation. January 2021.

[22] Tom Le Paine, Sergio Gómez Colmenarejo, Ziyu Wang, Scott Reed, Yusuf Aytar, Tobias Pfaff, Matt W Hoffman, Gabriel Barth-Maron, Serkan Cabi, David Budden, et al. One-shot high-fidelity imitation: Training large-scale deep nets with rl. *arXiv preprint arXiv:1810.05017*, 2018.

[23] 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. In *Proceedings of the IEEE Conference on Computer Vision and Pattern*

Recognition Workshops, pages 2050–2053, 2018.

[24] Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions on Graphics (TOG)*, 37(4):1–14, 2018.

[25] Xue Bin Peng, Angjoo Kanazawa, Jitendra Malik, Pieter Abbeel, and Sergey Levine. Sfv: Reinforcement learning of physical skills from videos. *ACM Transactions on Graphics (TOG)*, 37(6):1–14, 2018.

[26] Xue Bin Peng, Erwin Coumans, Tingnan Zhang, Tsang-Wei Lee, Jie Tan, and Sergey Levine. Learning agile robotic locomotion skills by imitating animals. *arXiv preprint arXiv:2004.00784*, 2020.

[27] Giacomo Rizzolatti and Laila Craighero. The mirror-neuron system. *Annu. Rev. Neurosci.*, 27:169–192, 2004.

[28] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635, 2011.

[29] Stuart Russell. Learning agents for uncertain environments. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 101–103, 1998.

[30] Fereshteh Sadeghi and Sergey Levine. Cad2rl: Real single-image flight without a single real image. *arXiv preprint arXiv:1611.04201*, 2016.

[31] 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.

[32] Pratyusha Sharma, Deepak Pathak, and Abhinav Gupta. Third-person visual imitation learning via decoupled hierarchical controller. *arXiv preprint arXiv:1911.09676*, 2019.

[33] Laura Smith, Nikita Dhawan, Marvin Zhang, Pieter Abbeel, and Sergey Levine. Avid: Learning multi-stage tasks via pixel-level translation of human videos. In *Robotics: Science and systems*, 2020.

[34] Bradly C Stadie, Pieter Abbeel, and Ilya Sutskever. Third-person imitation learning. *arXiv preprint arXiv:1703.01703*, 2017.

[35] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. *arXiv preprint arXiv:1906.05849*, 2019.

[36] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017.

[37] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

[38] Tianhe Yu, Chelsea Finn, Annie Xie, Sudeep Dasari, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. One-shot imitation from observing humans via domain-adaptive meta-learning. *arXiv preprint arXiv:1802.01557*, 2018.

[39] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei 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.

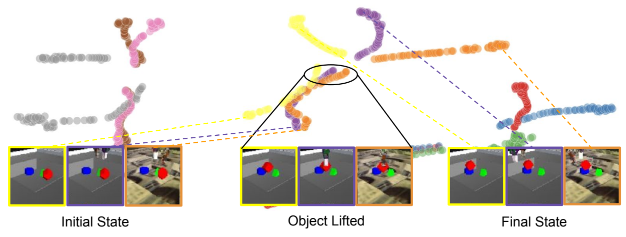

Fig. 5. t-SNE Projection of 3 triplets of trajectories, where each triplet consists of paired trajectories rendered in 3 visually different domains: with an 'invisible arm', with full domain randomization and with domain randomization only for the robotic arm. The features used were learned with MIR. The 3 triplets are clearly separated from each other and the trajectories from the different domains appear to align well, especially when there is an environment change that doesn't include the arm, a desired feature for our representations.

Appendix

A. *t-SNE Visualization*

In order to gain better insight into embeddings we learn with the MIR objective, we plotted the t-SNE projection for each method on three aligned triplets, for a total of 9 trajectories from the three different domains used for training our manipulator-independent representations (MIR): domain-randomized, arm-randomized, and 'invisible arm'. Figure 5 depicts the alignment achieved by learning the representations of MIR. As one can see in the figure, where we depict the frames at certain points in the trajectories for a given triplet, the trajectories are aligning fairly well even though they are from different domains, while staying further from the other two trajectory-triplets. Alignment appears to be particularly successful at the point of an environmental change, i.e. when the red object is being lifted.

B. *Qualitative Evaluation Videos*

We provide example imitation trajectories[3] that are part of our quantitative evaluation. All the experiments are using the pretrained MIR as described in the main paper. Our method successfully learns to imitate each trajectory despite being out of distribution. A brief description of the qualitative evaluations are provided below:

Example trajectory following for each domains. We firstly show examples of third-person demonstrated trajectories and how our agents managed to follow them. A few examples are presented for each of the testing domains, namely 'invisible arm' simulation, 'Jaco hand' simulation, shadow hand simulation and real-world robot.

Stack and unstack. Stacking red object on top of blue object, releasing the blue object and rising up, grasping the blue object again and returning it to its original location.

Stack with obstacles. A fixed obstacle is added into the basket as a distractor, while the obstacle does not exist in the demonstrated trajectory. Note that obstacles are not seen during MIR training.

[3]Videos are accessible from: https://sites.google.com/view/mir4vi/qualitative-videos

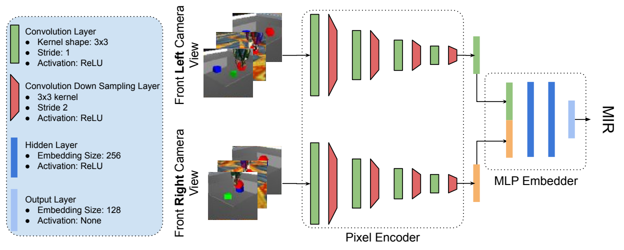

Fig. 6. MIR Encoder Structure. Our agents can access both front cameras in the environment. For each camera observation, a convolutional encoder is applied to extract a 128-dimension feature. A 3-layer MLP is then applied to the concatenated features and output a 128-dimension representation for the final MIR. Configurations of different layers are shown one the left.

C. *Baselines*

As TCN is already covered in the main text, here we'll describe the other two baselines compared to our method.

TDC [2]. As it is evident from its name, temporal distance classification (TDC) predicts the temporal distance between any given observation pair within the same sequence. TDC sets the distance prediction as a classification problem where the classes are different distance intervals. We follow the same setting described in [2] and set the distance interval categories as $\{[1], [2], [3-4], [5-20], [21-200]\}$, which suits well to our setting as we operate over episodes of 200 time-steps. To define the loss, given the observation pair (x_i, x_{i+d}) we concatenate the features and run them through a two-level MLP to predict the distance interval. We use the distance d to set the true label. The optimized loss function is a softmax cross-entropy for this multi-class classification problem.

CMC[2]. Cross-modal distance classification (CMC) predicts the temporal distance between any given cross-modal (i.e. cross-domain) observation pair. [2] uses this method to learn representations by aligning audio and images in game videos. In our setting modalities are the pairs of randomized domains. For instance in the cross-domain pair $(x_i, \bar{x}_{i+2})$ the temporal distance would be 2. CMC also sets the distance prediction as a classification problem where $[0]$ distance is also included in the distance interval categories used for TDC. To define the loss, given the cross-domain pair $(x_i, \bar{x}_{i+d})$ we concatenate the features and run them through a two-level MLP to predict the distance interval. We use the distance d to set the true label. The optimized loss function is a softmax cross-entropy for this multi-class classification problem.

D. *Network Structure for MIR*

In our environment, an agent can access two front cameras. The two cameras are beneficial to help disambiguate 3D positions of the robot arm and the objects. We extract image representations separately from each camera using the same convolutional encoder. As shown in Figure 6, the encoder has a 4-step repetition of one 3x3 convolutions each follow by a down-sampling 3x3 convolution layer with stride 2. A rectified linear unit (ReLU) is applied after each convolution layer. At each down-sampling step we double the number of feature channels, where first step has channel size 8. The output of

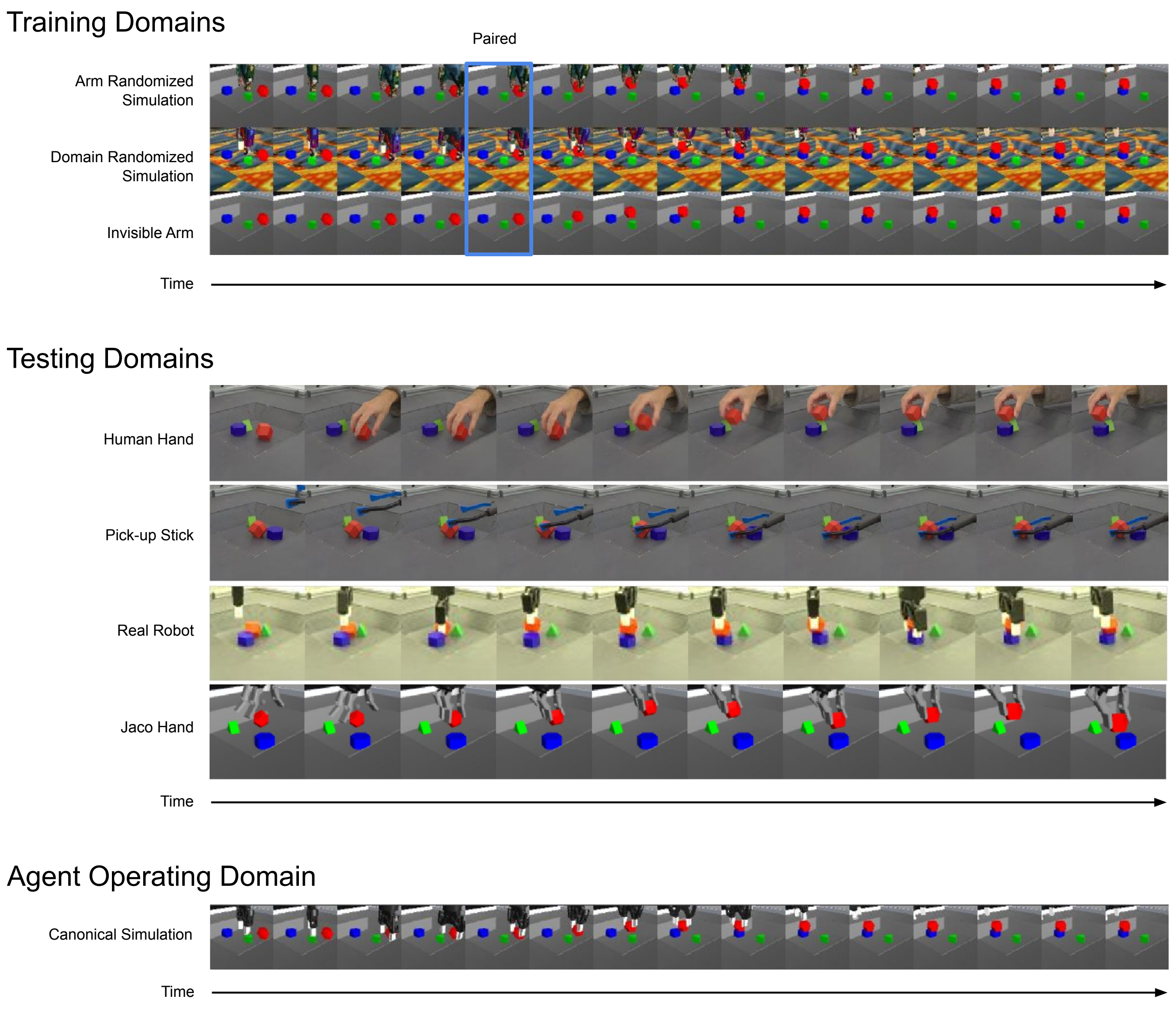

Fig. 7. Filmstrips of MIR training and testing domains. **In training domains**, examples of paired trajectories used for manipulator-independent representation learning are shown. A sub-sampled filmstrip with 10 steps interval is presented from left to right. From top to bottom, the environment involved are arm randomized simulation, domain randomized simulation and simulation with invisible hand. **In testing domains**, examples of demonstrated trajectories are presented. Similarly, sub-sampled filmstrips are shown for human hand, pick-up stick, real robot, and Jaco hand domains. Note that there is no correlation between any trajectories in testing domains. The canonical environment, where our simulated robot agent is operating on, is also shown at the bottom.

a pixel encoder is a 128-dimension feature which is obtained through a final linear layer. Features of both camera views are concatenated channel-wise followed by a 3-layer multilayer perceptron (MLP) network. Each hidden layer is the MLP has 256 channels and the output of the MLP is a 128-channel embedding. The configuration of layers are shown in Figure 6. For GCP and CD-GCP training we also use a separate goal encoder which shares the same structure described above but weights are not shared. Additionally a 3 layer MLP with $[256, 256, 5]$ channels is used as the policy head to predict the actions.