2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Cross Domain Policy Transfer with Effect Cycle-Consistency

Ruiqi Zhu[1], Tianhong Dai[2], Oya Celiktutan[1]

***Abstract*— Training a robotic policy from scratch using deep reinforcement learning methods can be prohibitively expensive due to sample inefficiency. To address this challenge, transferring policies trained in the source domain to the target domain becomes an attractive paradigm. Previous research has typically focused on domains with similar state and action spaces but differing in other aspects. In this paper, our primary focus lies in domains with different state and action spaces, which has broader practical implications, i.e. transfer the policy from robot A to robot B. Unlike prior methods that rely on paired data, we propose a novel approach for learning the mapping functions between state and action spaces across domains using unpaired data. We propose *effect cycle-consistency*, which aligns the effects of transitions across two domains through a *symmetrical optimization* structure for learning these mapping functions. Once the mapping functions are learned, we can seamlessly transfer the policy from the source domain to the target domain. Our approach has been tested on three locomotion tasks and two robotic manipulation tasks. The empirical results demonstrate that our method can reduce alignment errors significantly and achieve better performance compared to the state-of-the-art method. Project page: https://ricky-zhu.github.io/effect_cycle_consistency.**

I. Introduction

Deep reinforcement learning (DRL) has demonstrated impressive performance in sequential decision-making problems, such as video games [1], [2], robotics manipulation [3], [4], and autonomous driving [5]. However, due to the low sample efficiency, training DRL models from scratch has typically limited their application in real-world scenarios where data collection is costly [6], [7]. Consequently, transferring policies trained in the source domain to the target domains has emerged as a promising research direction. Previous works usually focus on the domains with similar state and action spaces but differing in other aspects, such as dynamics [8], [9]. In this paper, we aim to transfer the policy across domains with different state and action spaces, which has a broader applicability, i.e., transferring the pre-trained policy on robot A to a variety of robots with different state spaces and action spaces.

To enable policy transfer across domains with different state spaces and action spaces, a line of prior works has explored the use of graph neural networks (GNN) [10] to address the differences in state and action spaces, leveraging GNN's ability to process graphs of arbitrary sizes. However, these studies often assume that agents possess limbs, each equipped with proprioceptive sensors, to model structural policies with GNN using hand-designed descriptions [11]–[13]. Another line of prior works focuses on learning invariant representations across agents [14], [15]. By training on the latent representation space, policies become invariant to differences in state spaces and action spaces. Nevertheless, these works often hinge on paired trajectories, acquired from pre-trained policies or human labelling, which can be prohibitively expensive to obtain in real-world applications. Furthermore, enforcing invariance may not be universally applicable as various types of invariance may offer advantages for diverse downstream tasks, as demonstrated by recent research in the field of self-supervised visual representation learning [16].

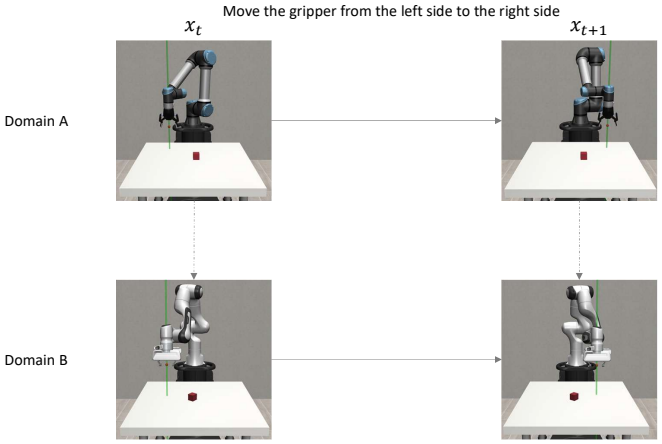

Fig. 1: Illustration of the effect cycle-consistency. We aim to align the effect of the transitions across domains. For instance, the effect of transition in Domain A is moving the gripper from the left side to the right side. The effect of the translated transition in the Domain B is expected to move the gripper from the left side to the right side as well.

Recently, several studies have sought to remove the need for paired data by incorporating the concept of cycle-consistency [17] into the process of learning mapping functions of the state spaces and action spaces across domains [18], [19]. These investigations have focused on establishing mappings between visual scenes, often overlooking the temporal ordering which indicates the dependency of the consecutive states and actions. In an attempt to incorporate the temporal ordering into the mapping functions learning process, Zhang et al. introduced dynamics cycle-consistency constraint [20]. The constraint enforces that the predicted next state by the forward dynamics model corresponds to the translated next state. However, due to inherent approximation errors associated with the mapping functions and learned forward dynamics models, this method is susceptible to compounding errors. As the time step increases, alignment errors accumulate, leading to

[1] R. Zhu and O. Celiktutan are with the Department of Engineering, King's College London.
[2] T. Dai is with the Department of Computing Science, University of Aberdeen.

a degradation in the performance of the transferred policies. To mitigate the compounding error, Wang et al. proposed to apply weak supervision on the learning of mapping functions using the data with paired abstractions [21]. For example, for ant robots with different legs, the data with paired abstractions refers to the states with the same 2D coordinates but may differ in other dimensions. However, collecting such data is still not trivial and the assumption that the domains have the same abstractions may be violated in certain tasks.

In this paper, we propose a novel framework for learning the mapping functions across the domains with different state spaces and action spaces leveraging unpaired data. Unlike dynamics cycle-consistency which aligns the next states in both domains, we propose *effect cycle-consistenct* that aligns the effect of transitions in both domains to learn the mapping functions as illustrated in Fig. 1. Additionally, we utilize *symmetrical optimization* structure which applies identical objectives to the mapping functions from the source domain to the target domain and vice versa. In summary, the contributions of this paper are:

- We propose a novel framework for learning the mapping functions across domains with different state and action spaces.
- We propose *effect cycle-consitency* to learn the mapping functions with *symmetrical optimization* structures.
- We have conducted experiments on 3 locomotion tasks and 2 robotic manipulation tasks. The empirical results demonstrate that our method achieves better performance with the transferred policies and lower alignment errors.

II. Related Works

Cross Morphology Policy Transfer. To transfer a pre-trained policy across agents with different morphologies, a line of prior works model the agents using GNN for leveraging the capability of GNN to process graphs of arbitrary sizes [11], [13]. These works usually assume that the agents have rigid structures such as limbs. Additionally, they need hand-designed descriptions to model the graphs, which is non-trivial. To remove the need for the hand-designed description, Trabucco et al. propose a data-driven method which trains a transferrable policy among a broad set of morphologies [22]. The policy in the approach is conditioned on the agent's state and action, which are used as inputs to a neural network that infers the agent's morphology. The inferred morphology is then used to train a policy that can generalize to new morphologies. However, the approach requires the careful design of the training set of morphologies to ensure the testing morphology representations lie within the support of the distribution. Hejna et al. proposed an approach for transferring the policy by leveraging the similarity of the morphologies [23]. The approach aligns the pre-trained low-level policies and then learns transferrable high-level policies. However, the approach is limited to transfer policies between similar morphologies.

Learning Invariant Representation. To learn the mapping across domains, researchers have proposed to learn the representations which are invariant to the factors that are irrelevant to the downstream tasks and only preserve task-specific information. Domain randomization methods learn invariant task-specific representation by randomizing some factors of the domains, such as lights, and textures [24]–[26]. The policies trained in the augmented domains are supposed to be robust to the variations of these factors. Nevertheless, these methods require expertise to determine what factors to randomize. Also, they assume the distribution of these factors of the target domains lies within the support of the randomization distribution, which could be violated in practical applications. Another line of prior work instead attempts to discover the invariant representation with paired trajectories of the source domains and the target domains [14], [27], [28]. However, collecting the paired trajectories usually requires human annotations, which could be expensive and tedious [29]. In contrast to these methods, we aim to discover the mapping across the source domains and target domains using unpaired data, which would be more applicable.

Cycle-Consistency. To remove the need for paired data in the field of image-to-image translation, Zhu et al. proposed cycle-consistency [17] to discover the correspondence between different image domains leveraging Generative Adversarial Networks [30]. Subsequently, the method has been extended to other fields, such as unsupervised video retargeting [31], and domain adaptation [32]. Recently, the concept of cycle-consistency has been brought to the field of sim-to-real adaptation for robotic tasks [18], [19]. However, these applications typically are limited to visual adaptations. Zhang et al. have proposed dynamics cycle-consistency to learn the mapping functions across the state spaces and the target spaces of different domains [20]. The approach has integrated the dynamics information into learning the mapping functions and hence can distinguish the temporal ordering. The approach achieves state-of-the-art performance in policy transfer across domains with different state spaces and action spaces using unpaired data.

III. Background

Problem Setting. The source domain and target domain are both modelled as Markov Decision Process (MDP). We model the source domain as $\mathcal{M}^1 = \{X, A, \mathcal{T}^1, \mathcal{R}^1, p_0^1, \gamma\}$ and the target domain as $\mathcal{M}^2 = \{Y, U, \mathcal{T}^2, \mathcal{R}^2, p_0^2, \gamma\}$. The source domains and the target domains have different state spaces and action spaces. We aim to transfer the pre-trained policy in the source domain to the target domain for solving the same tasks.

Owing to the mismatch of the state spaces and the action spaces, it is essential to define the mapping functions across the domains. Specifically, we denote the state mapping function and action mapping function from the source domain to the target domain as $F : X \to Y$ and $H : X \times A \to U$ respectively. Similarly, we denote the state mapping function and action mapping function from the target domain to the

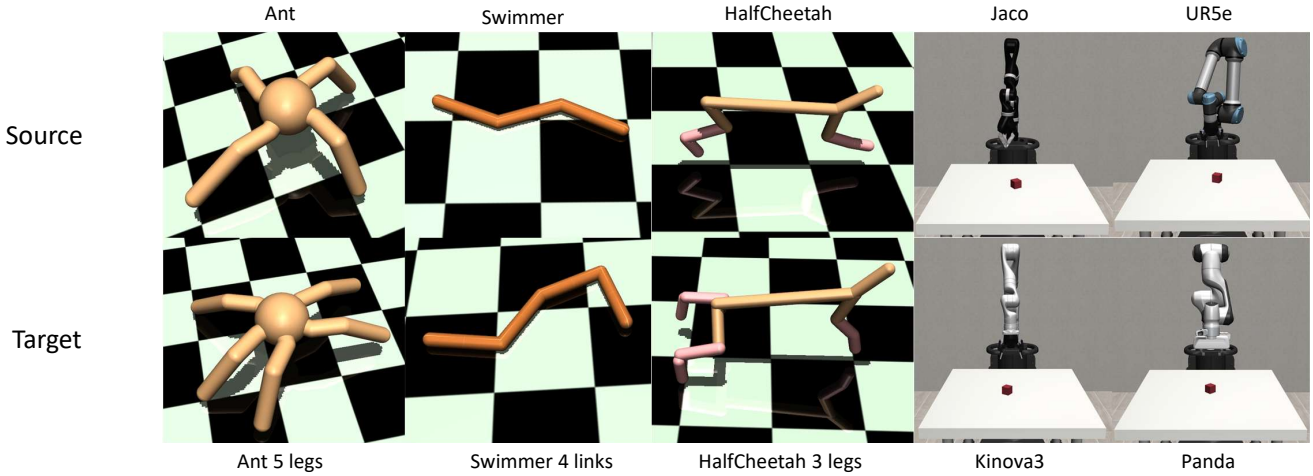

Fig. 2: Visulization of the agents in the source domains and the target domains.

source domain as $G : Y \rightarrow X$ and $P : Y \times U \rightarrow A$ respectively.

We aim to utilize unpaired data to learn the mapping functions. We formulate the dataset of the source domain as $\tau_{\mathcal{M}^1} = \{x_t, a_t, x_{t+1}\}_i$ and the dataset of the target domain as $\tau_{\mathcal{M}^2} = \{y_t, u_t, y_{t+1}\}_i$. The unpaired datasets contain no task-specific behaviours. The mapping functions can be used to transfer the pre-trained policy to the target domain as shown in Algo. 1.

Dynamics cycle consistency. To discover the mapping functions across the domains, Zhang et al. proposed dynamics cycle-consistency which integrates dynamics information into the learning while leveraging adversarial objectives [20]. It learns the state mapping function from the target domain to the source domain using adversarial objectives as:

$$\min_{G} \max_{D_X} \mathbb{E}_{x\sim\tau_{\mathcal{M}^1}}[\log D_X(x)] + \mathbb{E}_{y\sim\tau_{\mathcal{M}^2}}[\log(1 - D_X(G(y)))] \tag{1}$$

where D_X is the discriminator which tries to distinguish the translated sample $G(y)$ against real samples x. Similarly, it utilizes the adversarial objectives to learn the action mapping functions H and P. Additionally, it utilizes cycle-consistency constraints on learning the action mapping functions as:

$$\min_{H,P} \mathbb{E}_{a\sim\tau_{\mathcal{M}^1}}[\|P(y, H(x, a)) - a\|_1] \tag{2}$$

Following that, the approach integrates the dynamics information by imposing the translated next state corresponds to the predicted next state by the forward dynamics model $T_{\mathcal{M}_2}$ as:

$$\min_{F,H} \mathbb{E}_{(x_t,a_t,x_{t+1})\sim\tau_{\mathcal{M}^1}}[\|F(x_{t+1}) - T_{\mathcal{M}_2}(F(x_t), H(x_t, a_t))\|_1] \tag{3}$$

With this objective, the learned mapping function can be aware of the temporal ordering, which would be ignored if simply applying cycle-consistency loss on learning the mapping functions. However, the approach is susceptible to compounding errors. For instance, given a trajectory in the source domain $\tau = \{x_0, a_0, ..., s_T\}$, two methods to derive the translated state at time step T are : (1) $y_T = F(x_T)$; (2) $\hat{y}_T = T_{\mathcal{M}_2}(\cdots T_{\mathcal{M}_2}(F(x_0), H(x_0, a_0)), ..., H(x_T, a_T))$. The second method progressively utilizes the forward dynamics model and the mapping functions to generate the next states. Hence, the approximation errors associated with the forward dynamics model and the mapping functions will enlarge the alignment errors between the translated states using the two methods with longer horizons [21]. Consequently, the large alignment errors could potentially degrade the performance of the transferred policies.

IV. Effect Cycle-Consistency with Symmetrical Optimization

In this section, we present our method for solving policy transfer across domains with different state and action spaces using unpaired data. Our framework is illustrated in Fig. 3.

Our initial step involves learning state mapping functions with adversarial training objectives, as defined in Eq. 4 and Eq. 5. Given unpaired samples $x_i \in X$ and $y_i \in Y$ from the dataset, the state mapping functions $G(y)$ and $F(x)$ aim to map the state distribution to that of their counterparts, while the discriminators D_X and D_Y aim to distinguish between the real samples and the translated samples [30]. Additionally, to prevent mode collapse, we include a cycle-consistency loss [17], as defined in Eq. 6, in the training process. Consequently, the translated sample is expected to correspond to the original sample when translated back.

$$\min_{G} \max_{D_X} \mathcal{L}_{adv}(G, D_X) = \mathbb{E}_{x\sim\tau_{\mathcal{M}^1}}[\log D_X(x)] + \mathbb{E}_{y\sim\tau_{\mathcal{M}^2}}[\log(1 - D_X(G(y)))] \tag{4}$$

$$\min_{F} \max_{D_Y} \mathcal{L}_{adv}(F, D_Y) = \mathbb{E}_{y\sim\tau_{\mathcal{M}^2}}[\log D_Y(y)] + \mathbb{E}_{x\sim\tau_{\mathcal{M}^1}}[\log(1 - D_Y(F(x)))] \tag{5}$$

$$\min_{G,F} \mathcal{L}_{cyc}(G, F) = \mathbb{E}_{x\sim\tau_{\mathcal{M}^1}}[\|x - G(F(x))\|_1] + \mathbb{E}_{y\sim\tau_{\mathcal{M}^2}}[\|y - F(G(y))\|_1] \tag{6}$$

For the adversarial training objectives, it achieves the global optimal when the probabilities given by the discriminators equal the ratio of the real samples. For the cycle-consistency objective, it achieves the global optimal when

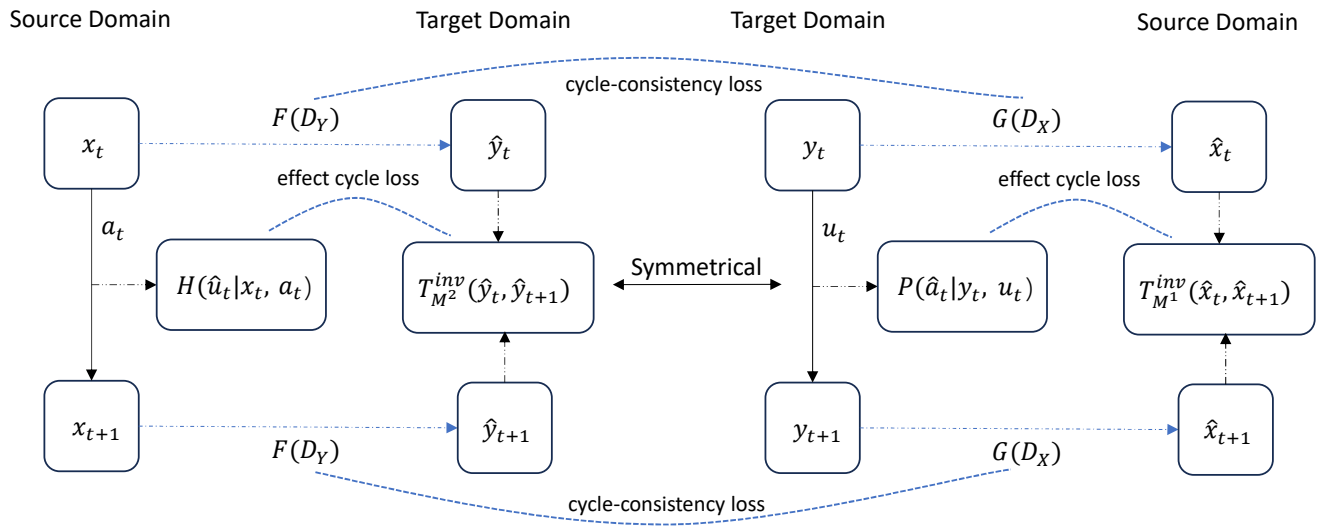

Fig. 3: Illustration of our method.

the state mapping functions successfully establish one-to-one mappings between domains. However, despite achieving the global optimum for both objectives, misalignment issues persist due to the inability to learn the temporal ordering [20]. For instance, given transition tuples (x_t, a_t, x_{t+1}) and (y_t, u_t, y_{t+1}), G can map x_t to y_{t+1} and F can still map back y_{t+1} to x_t, which does not the violate cycle-consitency constraint. However, x_t and x_{t+1} are supposed to be mapped to y_t and y_{t+1} respectively.

To enable the mapping functions to be aware of the temporal ordering, we incorporate dynamics information into the learning by proposing *effect cycle-consistency* constraints which align the effect of the transitions between domains as shown in Eq. 7. $T^{inv}_{\mathcal{M}^1}$ and $T^{inv}_{\mathcal{M}^2}$ represent the inverse dynamics models of the source domain and the target domain respectively. They predict the actions that lead to a given transition. Given the tuple (x_t, a_t, x_{t+1}) from the source domain, the translated action distribution $H(\hat{u}_t|x_t, a_t)$ is supposed to correspond to the predicted target action distribution given by the inverse dynamic model of the target domain $T^{inv}_{\mathcal{M}^2}(u_t|F(x_t), F(x_{t+1}))$ with the translated consecutive states as inputs.

$$\begin{aligned} \min \mathcal{L}_{eff}(F, H) &= D_{KL}(T^{inv}_{\mathcal{M}^2}(F(x_t), F(x_{t+1}))||H(x_t, a_t)) \\ \min \mathcal{L}_{eff}(G, P) &= D_{KL}(T^{inv}_{\mathcal{M}^1}(G(y_t), G(y_{t+1}))||P(y_t, u_t)) \end{aligned} \tag{7}$$

However, as the inverse dynamics model represents the physical property of the domain, it is not differentiable for back-propagation. Thus, we train neural networks to approximate the inverse dynamics models of both domains $T^{inv}_{\mathcal{M}^1}$ and $T^{inv}_{\mathcal{M}^2}$. We utilize a supervised objective to train the inverse dynamics models using the dataset to achieve that. As shown in Eq. 8 and Eq. 9, to reduce the variance during the training, we leverage the reparameterization trick [33]. We afterwards fix the parameters of the inverse dynamics models throughout the training of the mapping functions.

$$\begin{aligned} &\min \mathbb{E}_{\substack{(x_t, a_t, x_{t+1}) \sim \tau_{\mathcal{M}^1} \\ \epsilon \sim \mathcal{N}(0,I)}} [\|a_t - (\mu + \epsilon * \sigma)\|_1] \\ s.t. \quad &\mu = mean\ (T^{inv}_{\mathcal{M}^1}(x_t, x_{t+1})) \\ &\sigma = std\ (T^{inv}_{\mathcal{M}^1}(x_t, x_{t+1})) \end{aligned} \tag{8}$$

$$\begin{aligned} &\min \mathbb{E}_{\substack{(y_t, u_t, y_{t+1}) \sim \tau_{\mathcal{M}^2} \\ \epsilon \sim \mathcal{N}(0,I)}} [\|u_t - (\mu + \epsilon * \sigma)\|_1] \\ s.t. \quad &\mu = mean\ (T^{inv}_{\mathcal{M}^2}(y_t, y_{t+1})) \\ &\sigma = std\ (T^{inv}_{\mathcal{M}^2}(y_t, y_{t+1})) \end{aligned} \tag{9}$$

Both the outputs of the action mapping functions and the inverse dynamics models are formulated as Gaussian distributions. We minimize the Kullback-Leibler divergences between the two distributions. Hence, we can have analytic solutions to the objectives. Additionally, using $D_{KL}(T^{inv}_{\mathcal{M}^2}||H)$ instead of $D_{KL}(H||T^{inv}_{\mathcal{M}^2})$ ensures that the learned action mapping function H is mode-covering [34], i.e., represents all actions captured by the inverse dynamics model.

As $H(u_t|x_t, a_t) = \sum T(x_{t+1}|x_t, a_t) H(u_t|x_t, a_t, x_{t+1})$ where T is the characteristic transition model of the source domain, therefore the objective can be interpreted as enabling the action mapping function to predict the action which can lead to the translated transition $(F(x_t), F(x_{t+1}))$ given the transition (x_t, x_{t+1}) conditioned on the action a_t. In contrast to dynamics cycle-consistency which is susceptible to compounding errors caused by the progressive reliance on single-step prediction of the next state, we instead align the effect of the transitions, which is independent of the progressivity. By removing the dependence on the progressivity, the compounding errors will be mitigated as the errors only come from the alignment errors of the mapping functions and will not compound with time steps increase. Additionally, the temporal ordering is embedded as the action distributions given by $T^{inv}_{\mathcal{M}^2}(F(x_t), F(x_{t+1}))$ and $T^{inv}_{\mathcal{M}^2}(F(x_{t+1}), F(x_t))$ are different. As the effect cycle-consistency loss also backpropagates through the state mapping functions, the learned state mapping functions should be aware of the temporal ordering information.

Algorithm 1: Evaluation

Input: state mapping function $G : Y \rightarrow X$, action mapping function $H : X \times A \rightarrow U$, pre-trained policy in the source domain $\pi(a|x)$

for $t = 1...H$ **do**

- Observe y_t ;
- Translate to the source domain $x_t = G(y_t)$;
- Obtain the action in the source domain $a_t \sim \pi(a_t|x_t)$;
- Translate the action to the target domain $u_t \sim H(u_t|x_t, a_t)$;
- Execute u_t;

Note in the training of the mapping functions, the same learning objectives are applied to the mapping functions from the source domain to the target domain and the mapping functions from the target domain to the source domain as shown in Fig. 3. We refer to the optimization structure as *symmetrical optimization* structure. With the optimization structure, the learned mapping functions in the source(target) domains will influence those in the target(source) domains through the cycle-consistency loss. The optimization structure has demonstrated its significance in improving the performance of the transferred policies and stabilizing the training as shown in Section V. Overall. the full objective is given as:

$$\begin{aligned}\mathcal{L}_{full} =& \lambda_1(\mathcal{L}_{adv}(G, D_X) + \mathcal{L}_{adv}(F, D_Y) + \mathcal{L}_{cyc}(G, F)) \\ &+ \lambda_2(\mathcal{L}_{eff}(F, H) + \mathcal{L}_{eff}(G, P)).\end{aligned} \tag{10}$$

As in the training of the mapping functions, it involves joint optimization of multiple neural networks, directly optimizing full objectives could lead to trivial solutions. Hence, we utilize the alternating training procedure. Specifically, we fix the parameters of the action mapping functions when optimizing adversarial objectives and the cycle-consistency objective by setting λ_2 as 0. We set λ_1 as 0 when optimizing the *effect cylce-consistency* objective as shown in Algo. 2.

Algorithm 2: Training

Input: state mapping function $G : Y \rightarrow X$ and $F : X \rightarrow Y$, action mapping function $H : X \times A \rightarrow U$ and $P : Y \times U \rightarrow A$, dataset of the source domain $\tau_{\mathcal{M}^1}$ and the dataset of the target domain $\tau_{\mathcal{M}^2}$.

Train the inverse dynamics models $T^{inv}_{\mathcal{M}^1}$ and $T^{inv}_{\mathcal{M}^2}$ using Eq. 8 and Eq. 9;

for $i = 1..e$ **do**

- reset λ_1, set $\lambda_2 = 0$;
- **for** $i = 1...e_1$ **do**
  - Train G and F using $\mathcal{L}_{full}$ as shown in Eq. 10;
- reset λ_2, set $\lambda_1 = 0$;
- **for** $i = 1...e_2$ **do**
  - Train all the mapping functions using $\mathcal{L}_{full}$ as shown in Eq. 10;

V. Experiment

We aim to answer the following questions through our experiments: **(1)** How is the efficacy of our methods compared to the baselines? **(2)** Does our method reduce the alignment errors compared to the baselines? **(3)** What is the impact of the dataset size? **(4)** What is the importance of the *symmetrical optimization* structure in our methods?

A. Experiment Setup

The pre-trained policies in the source domain are trained with TD3 algorithms [35] for all tasks. The dataset used in the experiment contains $1k$ unpaired trajectories collected by random policies in both domains. To answer the above questions, we have designed our experiments from 3 aspects.

1) The efficacy of our proposed method: To compare the proposed method with baselines, we have evaluated the performance of the transferred policy in the target domains through the state mapping functions and action mapping functions as depicted in Algo. 1.

We have conducted experiments on 3 locomotion tasks based on Mujoco [36] and 2 robotic reaching tasks based on Robosuite [37]. Additionally, we have added 2D coordinates of the agents to the observation spaces for the locomotion tasks. For each task, the source domains and the target domains have different state and action spaces. The specifications of the state and action spaces are shown in Table II. And the visualization of the agents in the source domains and the target domains are shown in Fig. 2.

Baselines. We have compared our methods with: **(1) Random**, in which the mapping functions across source and target domains are random; **(2) Cycle-GAN**, which imposes cycle-consistency loss [17] on learning the state mapping functions and the action mapping functions respectively; **(3) DCC [20]**, which utilizes dynamic cycle-consistency in learning the mapping functions. We follow the official implementation of this baseline. **(4) Ours w.o. symmetrical**, which ablates the symmetrical optimization structure in our method by removing the action mapping function P as it is not used in the evaluation as shown in Algo.1.

2) Alignment error evaluation: To evaluate the alignment errors, we focus on two locomotion tasks (HalfCheetah to HalfCheetah 3 legs, Ant to Ant 5 legs) as we do not have access to the ground truth of translated states in the target domain for the evaluation. We assume the 2D coordinates of the agents should remain the same after being translated by the state mapping functions. Thus, We utilize the 1-norm error between the coordinates of the original states and the translated states as a proxy to evaluate the alignment errors.

3) The impact of different dataset sizes: To investigate the impact of dataset size, We have trained the mapping functions with the datasets containing $2k$, $5k$, $10k$ and $20k$ unpaired trajectories respectively for the task (HalfCheetah to HalfCheetah 3 legs). We then evaluate the performance of the transferred policy to demonstrate the impact.

TABLE I: The performance of the transferred policy under different morphologies. (w.o. denotes without)

Source X	Target Y	Oracle, X	Random	Cycle-GAN	DCC	Ours, $X \rightarrow Y$	Ours w.o. symmetrical
HalfCheetah	HalfCheetah 3 legs	$6773.36_{\pm 43.65}$	$-266.03_{\pm 115.24}$	$-65.76_{\pm 32.23}$	$1361.65_{\pm 175.96}$	**1981.36±72.81**	$1427.25_{\pm 102.42}$
Swimmer	Swimmer 4 links	$301.81_{\pm 6.25}$	$-4.86_{\pm 7.53}$	$8.43_{\pm 7.97}$	$53.73_{\pm 18.29}$	**207.28±21.87**	$55.09_{\pm 32.52}$
Ant	Ant 5 legs	$4319.85_{\pm 49.39}$	$-213.55_{\pm 195.65}$	$28.97_{\pm 52.47}$	$670.53_{\pm 189.89}$	**872.28±77.80**	$720.21_{\pm 159.12}$
Jaco	Kinova3	$45.32_{\pm 3.74}$	$0.57_{\pm 0.53}$	$1.57_{\pm 1.23}$	$8.53_{\pm 7.25}$	**11.25±6.35**	$7.12_{\pm 5.13}$
UR5e	Panda	$45.99_{\pm 1.44}$	$1.12_{\pm 0.34}$	$1.87_{\pm 0.73}$	$6.89_{\pm 5.36}$	**7.34±5.15**	$7.01_{\pm 5.82}$

TABLE II: Dimensions of state spaces and action spaces of the source domains and the target domains.

Source $\mathcal{M}^1$	Target $\mathcal{M}^2$	$\mathcal{M}^1$		$\mathcal{M}^2$	
		State	Action	State	Action
HalfCheetah	HalfCheetah 3 legs	19	6	25	9
Swimmer	Swimmer 4 links	10	2	12	3
Ant	Ant 5 legs	113	8	115	10
Jaco	Kinova3	50	7	50	8
UR5e	Panda	47	7	42	8

B. Experiment Results

1) The efficacy of our proposed method: For each task, we trained our methods and baselines with 5 different seeds and evaluated the performance with 10 episodes. We report the average and the standard deviation of performances in Table I. As shown in the table, since cycle-GAN cannot recover the temporal ordering information in learning the mapping functions even though the state and action distributions of both domains in the dataset can be translated, the performance is poor. Compared to DCC, the state of the arts under this problem setting, our proposed method outperforms it in all the locomotion tasks and robotic reaching tasks. The outperformance ranges from **6%** (UR5e to Panda) to **285.8%** (Swimmer to Swimmer 4 links). Additionally, the performance of our proposed methods presents lower variance, which reflects the stability of the performance. We have also evaluated the performance of our proposed method without the symmetrical optimization structure (Ours w.o. symmetrical). The results show that the performance has dropped dramatically and the variance has increased, which demonstrates the importance of the optimization structure.

2) The alignment error analysis: We have compared our proposed method to DCC with respect to the alignment errors. We rollout the transferred policies in the target domain for 5 different seeds using the trained mapping functions and report the smoothed average of the running averaging of the alignment errors v.s. the time steps. With the time steps increase, the alignment errors of DCC present a trend of increasing which corresponds to the compounding errors as shown in Fig. 4. In contrast, our proposed method presents lower alignment errors, and the compounding errors have been significantly reduced.

3) The impact of different dataset sizes: We have trained our proposed method for 5 different seeds using the dataset of different sizes. We report the average and the standard deviation of the performance in Table. III. With the size increase, the inverse dynamics models can have better approximations and therefore lead to better action mapping functions and state mapping functions through *effect cycle-consistency* constraints, which further offers a better performance of the transferred policies. However, as the distribution of the dataset does not match the distribution of the encountered data during evaluation, the trained mapping functions cannot generalize to the unseen data in the evaluation. Consequently, as the scale increases from $10k$ to $20k$, the performance does not change much.

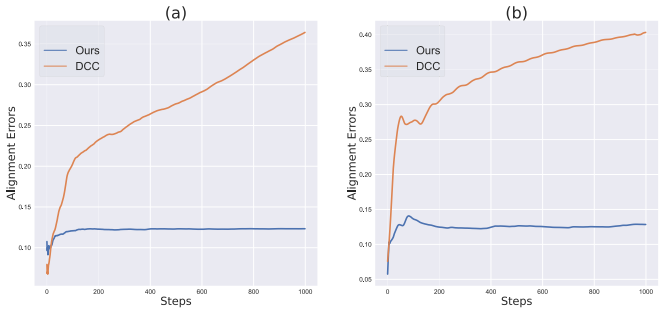

Fig. 4: The alignment errors v.s. time steps. (a) HalfCheetah to HalfCheetah 3 legs; (b) Ant to Ant 5 legs.

TABLE III: The performance of using different dataset sizes.

Scale	Performance
1k	$1981.36_{\pm 72.81}$
2k	$2061.41_{\pm 106.96}$
5k	$2113.12_{\pm 85.67}$
10k	$2307.63_{\pm 101.91}$
20k	$2282.77_{\pm 186.41}$

VI. Conclusion

In this paper, we present a novel framework that leverages unpaired data to learn the mapping functions across domains with different state spaces and action spaces. Within this framework, we propose *effect cycle-consistency* that align the effects of the original transitions and the translated transitions and *symmetrical optimization structure* in learning the mapping functions. The empirical results demonstrate that our proposed method obtains lower alignment errors and better performance compared to the baselines. With this framework, a learned policy can be seamlessly transferred to other agents without requiring task-specific datasets, which greatly alleviates the burden brought by the sample inefficiency of DRL methods. An exciting direction for future work would be generalizing the mapping functions to unseen data in the evaluation to further improve the performance.

VII. Acknowledgement

This work of Ruiqi Zhu is supported by the King's China Scholarship Council (K-CSC) PhD Scholarship programme.

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*, vol. 518, no. 7540, pp. 529–533, 2015.

[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*, vol. 529, no. 7587, pp. 484–489, 2016.

[3] R. Zhu, D. Zhang, and B. Lo, "Deep reinforcement learning based semi-autonomous control for robotic surgery," *arXiv preprint arXiv:2204.05433*, 2022.

[4] R. Zhu, S. Li, T. Dai, C. Zhang, and O. Celiktutan, "Learning to solve tasks with exploring prior behaviours," *arXiv preprint arXiv:2307.02889*, 2023.

[5] B. R. Kiran, I. Sobh, V. Talpaert, P. Mannion, A. A. Al Sallab, S. Yogamani, and P. Pérez, "Deep reinforcement learning for autonomous driving: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 6, pp. 4909–4926, 2021.

[6] Y. Yu, "Towards sample efficient reinforcement learning." in *IJCAI*, 2018, pp. 5739–5743.

[7] D. Zhang, Z. Wu, J. Chen, R. Zhu, A. Munawar, B. Xiao, Y. Guan, H. Su, W. Hong, Y. Guo, *et al.*, "Human-robot shared control for surgical robot based on context-aware sim-to-real adaptation," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 7694–7700.

[8] J. Hanna and P. Stone, "Grounded action transformation for robot learning in simulation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, no. 1, 2017.

[9] S. Desai, I. Durugkar, H. Karnan, G. Warnell, J. Hanna, and P. Stone, "An imitation from observation approach to transfer learning with dynamics mismatch," *Advances in Neural Information Processing Systems*, vol. 33, pp. 3917–3929, 2020.

[10] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *AI open*, vol. 1, pp. 57–81, 2020.

[11] T. Wang, R. Liao, R. Zemel, J. Ba, and S. Fidler, "Nervenet: Learning structured policy with graph neural networks," *Diss. University of Toronto*, 2017.

[12] W. Huang, I. Mordatch, and D. Pathak, "One policy to control them all: Shared modular policies for agent-agnostic control," in *International Conference on Machine Learning*. PMLR, 2020, pp. 4455–4464.

[13] V. Kurin, M. Igl, T. Rocktäschel, W. Boehmer, and S. Whiteson, "My body is a cage: the role of morphology in graph-based incompatible control," *arXiv preprint arXiv:2010.01856*, 2020.

[14] A. Gupta, C. Devin, Y. Liu, P. Abbeel, and S. Levine, "Learning invariant feature spaces to transfer skills with reinforcement learning," *arXiv preprint arXiv:1703.02949*, 2017.

[15] 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)*. IEEE, 2018, pp. 1134–1141.

[16] Y. Tian, C. Sun, B. Poole, D. Krishnan, C. Schmid, and P. Isola, "What makes for good views for contrastive learning?" *Advances in neural information processing systems*, vol. 33, pp. 6827–6839, 2020.

[17] 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*, 2017, pp. 2223–2232.

[18] K. Rao, C. Harris, A. Irpan, S. Levine, J. Ibarz, and M. Khansari, "Rl-cyclegan: Reinforcement learning aware simulation-to-real," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 11 157–11 166.

[19] D. Ho, K. Rao, Z. Xu, E. Jang, M. Khansari, and Y. Bai, "Retinagan: An object-aware approach to sim-to-real transfer," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 10 920–10 926.

[20] Q. Zhang, T. Xiao, A. A. Efros, L. Pinto, and X. Wang, "Learning cross-domain correspondence for control with dynamics cycle-consistency," *arXiv preprint arXiv:2012.09811*, 2020.

[21] Z. Wang, Z. Cao, Y. Hao, and D. Sadigh, "Weakly supervised correspondence learning," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 469–476.

[22] B. Trabucco, M. Phielipp, and G. Berseth, "Anymorph: Learning transferable polices by inferring agent morphology," in *International Conference on Machine Learning*. PMLR, 2022, pp. 21 677–21 691.

[23] D. Hejna, L. Pinto, and P. Abbeel, "Hierarchically decoupled imitation for morphological transfer," in *International Conference on Machine Learning*. PMLR, 2020, pp. 4159–4171.

[24] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 3803–3810.

[25] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. 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)*. IEEE, 2017, pp. 23–30.

[26] O. M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, *et al.*, "Learning dexterous in-hand manipulation," *The International Journal of Robotics Research*, vol. 39, no. 1, pp. 3–20, 2020.

[27] W. Yan, A. Vangipuram, P. Abbeel, and L. Pinto, "Learning predictive representations for deformable objects using contrastive estimation," in *Conference on Robot Learning*. PMLR, 2021, pp. 564–574.

[28] 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)*. IEEE, 2018, pp. 1118–1125.

[29] M. E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey." *Journal of Machine Learning Research*, vol. 10, no. 7, 2009.

[30] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," *Communications of the ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[31] A. Bansal, S. Ma, D. Ramanan, and Y. Sheikh, "Recycle-gan: Unsupervised video retargeting," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 119–135.

[32] 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 *International conference on machine learning*. Pmlr, 2018, pp. 1989–1998.

[33] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[34] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*. Springer, 2006, vol. 4, no. 4.

[35] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International conference on machine learning*. PMLR, 2018, pp. 1587–1596.

[36] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2012, pp. 5026–5033.

[37] 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," *arXiv preprint arXiv:2009.12293*, 2020.