Self-Supervised Disentangled Representation Learning for Third-Person Imitation Learning

Jinghuan Shang[1] and Michael S. Ryoo[1]

***Abstract*— Humans learn to imitate by observing others. However, robot imitation learning generally requires expert demonstrations in the first-person view (FPV). Collecting such FPV videos for every robot could be very expensive.**

Third-person imitation learning (TPIL) is the concept of learning action policies by observing other agents in a third-person view (TPV), similar to what humans do. This ultimately allows utilizing human and robot demonstration videos in TPV from many different data sources, for the policy learning. In this paper, we present a TPIL approach for robot tasks with *egomotion*. Although many robot tasks with ground/aerial mobility often involve actions with camera egomotion, study on TPIL for such tasks has been limited. Here, FPV and TPV observations are visually very different; FPV shows egomotion while the agent appearance is only observable in TPV. To enable better state learning for TPIL, we propose our disentangled representation learning method. We use a dual auto-encoder structure plus representation permutation loss and time-contrastive loss to ensure the state and viewpoint representations are well disentangled. Our experiments show the effectiveness of our approach.

***Index Terms*— Representation learning, imitation learning**

I. INTRODUCTION

Humans learn to imitate by observing others. In robotics, imitation learning enables a robot to learn complex tasks with minimal environmental knowledge [1] based on expert demonstrations. The robot learns a mapping from states (observations) to actions by using the expert trajectories as training data. However, imitation learning is known to be costly in collecting such expert data. Such expert demonstrations should be in the first-person view (FPV) from the same (or very similar) viewpoint to the robot and should include actual action labels. Collecting a sufficient amount of robot data could potentially be very expensive, especially for action labels.

Third-person imitation learning (TPIL) is the concept of learning action policies by observing other agents in a third-person view (TPV) without accessing the action labels. TPIL allows utilizing many human or robot demonstration videos in TPV from different data sources and very different viewpoints. It is similar to that humans could map TPV observations to their egocentric perspective [2], [3] and learn from them. Recent advances in TPIL [4]–[6] solved tasks by learning a joint visual state representation space shared by FPV and TPV from synchronized FPV and TPV demonstrations. Such joint state representation can be used to guide downstream policy learning.

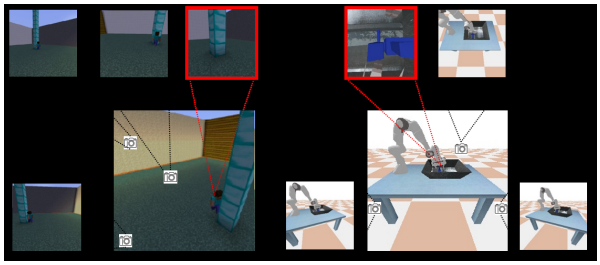

Fig. 1. Examples of third-person imitation tasks using Minecraft **(a)** and Panda robot sim **(b)**. Red box: FPV that the agent actually observes, displaying agent's egomotion. Black box: TPVs that are taken from multiple fixed cameras. In Minecraft environment, the player in Minecraft is trying to walk to the diamond blocks (blue blocks). In Panda environment, the Panda robot is trying to pick up the object in the tray, based on the observations from a camera mounted on the end effector.

In this research, we study a TPIL case where the robot's FPV contains *egomotion* caused by actions (Figure 1). For example, many robot tasks with ground (or aerial) mobility often involve actions with egomotion. Previous research [4], [5] only focused on FPV observations from a static camera, without considering tasks with the egomotion. Learning a joint visual state representation in such setting is very challenging due to the visual differences between egocentric FPV and TPV: (1) FPV videos consist of agent's egomotion while TPV videos consist of a relatively fixed scene with the moving agent, and (2) the agent itself is not visible (or only a very small portion of it is visible) in the FPV videos. The study on TPIL for egocentric videos will further broaden the scope of where TPIL algorithms could be applied.

To this end, we introduce a TPIL approach that learns disentangled state and viewpoint representations to ensure state representations are viewpoint-agnostic. Focusing on such visual differences between FPV and TPV, we propose a dual auto-encoder model to process the FPV and TPV inputs separately. In addition, we split the latent representation z of each auto-encoder into two parts, h and v, to encode disentangled state and viewpoint information, respectively. We propose representation permutation loss to train h and v to be well-disentangled representations. We adapt time-contrastive loss and apply it on h to encode state information. A general reconstruction loss is used to fully train our decoders.

Our visual representation learning is based on synchro-

*This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Ministry of Science and ICT (No.2018-0-00205, Development of Core Technology of Robot Task-Intelligence for Improvement of Labor Condition). This work was also supported by the National Science Foundation (IIS-2104404).

[1]All authors are with the Department of Computer Science, Stony Brook University, 100 Nicolls Road, Stony Brook, NY 11794, USA. {jishang, mryoo}@cs.stonybrook.edu

nized FPV and TPV videos from multiple viewpoints. Once finished, our policy learning is done by providing a single TPV expert demonstration without any FPV experts.

II. RELATED WORK

Our work is mainly related to three categories of research: imitation learning from observation, disentanglement, and contrastive learning.

A. Imitation learning from observations

Imitation learning from observations extends imitation learning to the case that experts' action labels are no longer available. Edwards et al. [7] first learns a latent policy and then remaps outputs from latent policy to actual action space. In [8], [9], a context translation model is used to map across contexts considering context differences between demonstration and agent's observation.

Third-person imitation learning further extends the imitation learning from observations by replacing FPV expert demonstrations with TPV ones. Recent literature on third-person imitation learning focuses on how to make connections between TPV and FPV observations by learning visual representations. TPIL [4] extends GAIL [10] and uses a domain confusion constraint to force features from both FPV and TPV are indistinguishable for a discriminator. TCN [5] uses a time-contrastive way to learn representations by self-supervised metric learning. TCN [5] is also extended to other tasks such as skill transfer [6] and playing hard exploration games [11]. mfTCN [12] extends the time-contrastive method to multiple frames. In this work, we extend TPIL to an egocentric FPV setting.

B. Disentanglement

Disentanglement is learning independent attributes encoded into separated dimensions of representation space. Typical methods for disentanglement are based on variational auto-encoders (VAE) [13] and generative-adversarial networks (GAN) [14].

β-TCVAE [15] hypothesis that each dimension in representation z is mutually independent. Both CVAE [16] and Info-GAN [17] provide the model with class label input to learn representations that are independent of these labels. Also, GAN methods like style-GAN [18] get a latent representation as a prior from a normal distribution. CC-VAE [19] adapts CVAE [16] to a self-supervised manner using an extra auto-encoder to learn a condition representation.

In imitation learning context, representations could also be implicitly disentangled. TRAIL [20] extends GAIL [10] to disentangle task-relevant and task-irrelevant information by adding a consistency constraint on samples in an invariant set. The learned representations are agnostic to task-irrelevant factors like colors. Disentangled representations have been also applied to other computer vision tasks and get successful results [21]–[25].

Compared with the above methods, our method explicitly splits latent representation into two components: state and viewpoint representation. We do not assume that exact viewpoint coordinates (i.e. labels of viewpoints) are provided for disentanglement, and rather train the model in a self-supervised manner.

C. Contrastive learning

Contrastive learning performs a comparison across different views that are generated from one single input to learn representations in a self-supervised fashion [26], [27]. The concept of contrastive learning also has been adopted for TPIL [5]. For computer vision tasks, CMC [28] takes advantage of InfoNCE loss [26] on the comparison between anchor sample, positive sample, and a batch of negative samples. Other research on contrastive learning [29]–[33] focus on avoiding trivial solutions to improve representation quality and increasing training efficiency by getting rid of negative samples. Contrastive learning concepts and techniques are also applied to control tasks like [34]. We extend time-contrastive loss [5] (which is based on triplet loss [35]) using InfoNCE loss [26] that includes a batch of negative samples. We also take advantage of the stop-gradient technique from the above literature to avoid trivial solutions.

III. PROBLEM SETUP

A discrete-time finite-horizon discounted Markov decision process (MDP) can be noted by $M = (\mathcal{S}, \mathcal{A}, \mathcal{P}, r, T)$, where $\mathcal{S}$ is a state set, $\mathcal{A}$ is an action set, $\mathcal{P}$ is a transition probability distribution, r is a reward function, and T is the horizon. For imitation learning, the agent is given trajectories $\tau = \{(s_t, a_t)\}$ which are states and actions generated from an unknown expert policy π_E and r is not traceable. The agent is required to learn a policy π_θ that recovers π_E.

In visual third-person imitation learning from observation, we consider an observer is learning by solely watching a demonstrator's demonstration from a different viewpoint than the demonstrator's, and a_t is not traceable. Under this setting, the observer and the demonstrator have visual observations $o_t^T \in \mathcal{O}^T$, $\mathcal{O}^T \subseteq \mathbb{R}^{H \times W \times d}$ and $o_t^F \in \mathcal{O}^F$, $\mathcal{O}^F \subseteq \mathbb{R}^{H \times W \times d}$ respectively, where $d = 3$ if we consider a pure RGB image input. Usually, we call $\mathcal{O}^\mathcal{T}$ third-person view (TPV) and $\mathcal{O}^F$ first-person view (FPV). A fact holds that at any moment t, o_t^T and o_t^F are different since they see from different views but o_t^T and o_t^F correspond to a same state s_t. Therefore, visual representation learning approaches explored by [4], [5] tried to solve this problem by finding a function $\mathcal{F} : \mathcal{O} \to \mathcal{H}$ that learns a latent state representation $h = \mathcal{F}(o)$ satisfying $\mathcal{F}(o_t^T) = \mathcal{F}(o_t^F)$ to estimate the true state space $\mathcal{S}$.

In this research, we investigate an egocentric FPV setting that actions cause agent's egomotion. Moreover, we consider TPV demonstrations from multiple different viewpoints. This is more challenging than learning the state embedding only applicable for one third-person viewpoint.

Formally, we have a set of synchronized FPV-TPV trajectories $\{\tau\}$ for visual representation learning, and a single TPV expert demonstration $\tau_E = \{o_t^T\}$ generated by some unknown expert policy π_E for policy learning. For each FPV-TPV trajectory, it has one FPV video

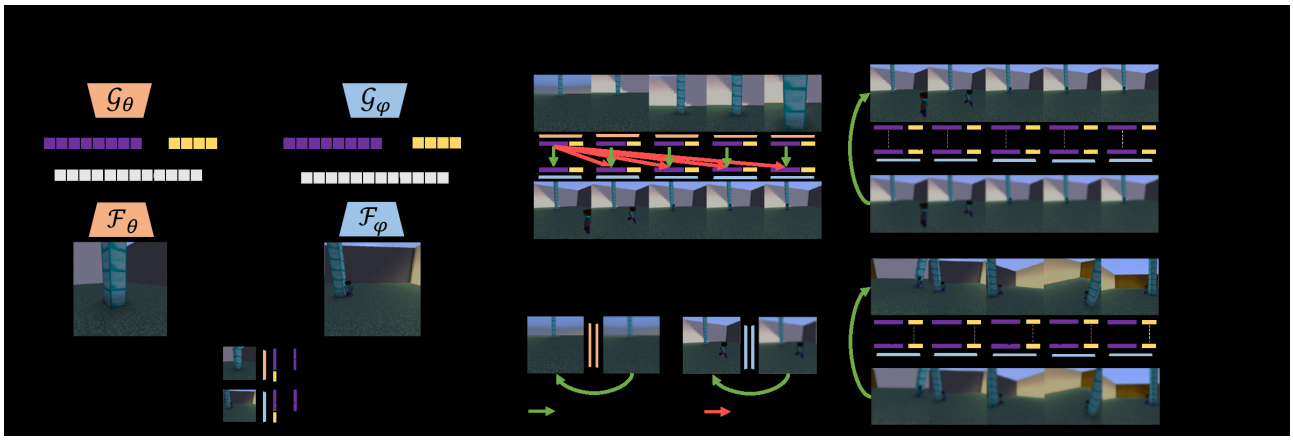

Fig. 2. This figure includes an overview of the dual AE model we propose, the policy learning method we use after having a state representation space, and losses we use to learn our FPV-TPV joint state representation. (a) Proposed dual AE model. (b) When doing policy learning, we discard v and use h. (c) Time-contrastive loss requires an anchor sample, a positive sample, and a batch of negative samples. This figure uses anchor FPV frame as an example. Symmetrically, a TPV frame can be an anchor. (d) We reconstruct every input FPV/TPV frame and calculate the reconstruction loss. (e) We permute viewpoint representation v among a batch of frames from the same TPV but different timesteps. Symmetrically, we permute state representations h among a batch of frames from the same timestep but different TPVs. We compare the reconstructed frames with original inputs accordingly to have our representation permutation loss.

and multiple TPV videos from multiple viewpoints: $\tau = (\{o_t^F\}, \{o_t^{T_1}\}, \ldots, \{o_t^{T_n}\})$. Given the egocentric FPV setting, o^F is visually different from o^T since o^F contains egomotion and o^T contains the agent's appearance which is not in o^F. So the challenge is how we design the map function $\mathcal{F}$, i.e. the representation model, that still preserves $\mathcal{F}(o_t^T) = \mathcal{F}(o_t^F)$. After getting such viewpoint-agnostic state representation h, we train a policy π to maximize imitation reward formulated by τ_E to best recover π_E. We follow the previous literature [5] to give a distance-metric based reward function, e.g. $r(t) = -||h_t^F - h_t^T||_2$, to perform policy learning, where h_t^F is current agent's FPV observation and h_t^T is given TPV expert demonstration at the same timestep.

IV. FPV-TPV JOINT STATE REPRESENTATION LEARNING

Focusing on the visual differences between $\mathcal{O}^F$ and $\mathcal{O}^T$, we propose our dual auto-encoder (AE) model with explicitly disentangled latent representations that learns a joint representation space $\mathcal{H}$. Then, we introduce our modified time-contrastive loss [5] and representation permutation loss to train our proposed model in a self-supervised fashion.

A. Dual AE for Joint State Representation Learning

Considering the visual differences between $\mathcal{O}^F$ and $\mathcal{O}^T$, we design two different mapping functions, $\mathcal{F}_\theta : \mathcal{O}^F \to \mathcal{Z}$ and $\mathcal{F}_\varphi : \mathcal{O}^T \to \mathcal{Z}$, instead of learning a single universal function for both $\mathcal{O}^F$ and $\mathcal{O}^T$. We propose a dual AE structure to model such $\mathcal{F}_\theta$ and $\mathcal{F}_\varphi$ as encoders in AEs along with the decoders $\mathcal{G}_\theta$ and $\mathcal{G}_\varphi$. The decoder $\mathcal{G}_\theta$ takes the full representation $z^F = [h^F, v^F]$ as input and outputs a reconstructed FPV image $\hat{x^F} = \mathcal{G}_\theta([h^F, v^F])$. Similarly $\mathcal{G}_\varphi$ reconstructs TPV image $\hat{x^T} = \mathcal{G}_\varphi([h^T, v^T])$.

We further design the dual AE to disentangle the representation vectors to better learn the joint embedding, taking advantage of multiple training losses. Specifically, each of our AE splits full representation vector z into two vectors, $z = [h, v]$, where $[\cdot, \cdot]$ means concatenation (See Figure 2(a)). Our motivation is to disentangle state and viewpoint from observation. If well-disentangled, the state representation is viewpoint-invariant, i.e. a joint embedding for TPV and FPV. Representation decomposition has been an effective strategy to obtain independent components (representations) in multiple computer vision tasks such as object recognition [15], [18], [21], [36], and our motivation is to extend and take advantage of such concept for TPIL. In our formulation, h (state) and v (viewpoint) are two independent factors we want to split from raw RGB observations. Similar to [21], which decomposed class-irrelevant features from class-relevant features for object recognition, our formulation has only two factors to decompose. Thus, we separate the latent vector into two parts and ensure that they are disentangled by appropriate loss functions.

The main technical challenge is in learning $\mathcal{F}_\theta$ and $\mathcal{F}_\varphi$, while ensuring (1) h to be an FPV-TPV joint state representation and (2) h and v are well disentangled. In the below subsections, we describe how the time-contrastive loss and the proposed representation permutation loss enable this.

B. Self-Supervised Disentangled Representation Learning

1) Time-Contrastive Loss: Time-contrastive [5] was previously proposed to make the state representation follow the temporal order. The temporal order means a state representation h_t should be closed to h_{t+1} but far from h_{t+n} where $n > 1$ in most cases. In practice, we use a triplet that includes anchor (A), positive (P), and negative (N) samples to construct time-contrastive loss. This loss requires representations of A and P samples are close (attraction), and representations of A and N samples are distant (repulsion).

Original time-contrastive loss [5], [35] is in a triplet-loss form. We extend it to include multiple negative samples for each single A and P sample using Info-NCE loss (See Figure 2(c)), inspired by recent advances in contrastive learning [28], [32]. Given two synchronized FPV-TPV demonstra-

tion videos, $\{(x_i^F, x_i^T)\}$, we first identify an A(anchor) at timestep a. Suppose x_a^F is the anchor (A) sample, then the x_a^T is the positive (P) sample. We then randomly sample a batch of negative (N) timesteps $\{n_i\}$ that appears ξ away. Then $\{x_{n_i}^T\}$ from a TPV video are N samples of x_a^F. After we get A-P-N samples, we have our modified time-contrastive loss term on FPV anchor:

$$\mathcal{L}_{tc}^F = -\log \frac{d(h_a^F, sg(h_a^T))}{d(h_a^F, sg(h_a^T)) + \sum d(h_a^F, sg(h_{n_i}^T))},$$

where h is state representation vector, $d(\cdot,\cdot)$ is a critic metric on h implemented based on cosine similarity between h vectors [28]:

$$d(h_1, h_2) = \exp\left(\frac{h_1 \cdot h_2}{||h_1|| \cdot ||h_2||} \cdot \tau\right),$$

where $||\cdot||$ is L2-norm, τ is a scaling coefficient [28], $sg(\cdot)$ is a stop-gradient trick [32], [33] to avoid trivial solutions. Symmetrically, if we let the TPV frame x_a^T be anchor, we have $\mathcal{L}_{tc}^T$ by making x_a^F as positive sample and $\{x_{n_i}^F\}$ negative samples.

To force h to be a joint representation space for FPV and TPV, we add a general matching loss (See Figure 2(c) in the middle) for all the $\{(h_i^F, h_i^T)\}$ pairs from the same timestep. We minimize their L2-distance pairwisely:

$$\mathcal{L}_{match} = \mathbb{E}\left[||h^T - sg(h^F)||_2\right],$$

where $sg(\cdot)$ is stop-gradient, viewing the state representations from FPV as references. Together, the total time-contrastive loss is:

$$\mathcal{L}_{tc} = \mathcal{L}_{tc}^F + \mathcal{L}_{tc}^T + \mathcal{L}_{match}.$$

2) Representation Permutation Loss: We introduce our representation permutation loss below to supervise h and v to be disentangled (See Figure 2(e)). Ideally, h and v should be independent, saying that modifying v only changes the viewpoint information, and modifying h only changes the state information. Therefore, we permute h and v among a batch of samples and reconstruct images from permuted full representations $z = [h_i, v_j]$. We expect the reconstructed images should be similar accordingly after the permutation. We will describe how we permute and compare reconstructed images accordingly below.

Given a single TPV demonstration, we can get the representations from two arbitrary timesteps: $(h_i^T, v_i^T) = \mathcal{F}(x_i^T)$ and $(h_j^T, v_j^T) = \mathcal{F}(x_j^T)$. Ideally, we should expect their viewpoint information is the same, so we have $v_i^T = v_j^T$. Then, by exchanging v_i^T to v_j^T and doing reconstruction, we would expect that the reconstruction image should still remain same as input at timestep i: $\mathcal{G}([h_i^T, v_j^T]) = x_i^T$. Symmetrically, $\mathcal{G}([h_j^T, v_i^T]) = x_j^T$. We can do these exchanges in a batch of samples from a single TPV demonstration by randomly pairing v to h. Then our viewpoint permutation loss is formulated as an reconstruction error

$$\mathcal{L}_v = \mathbb{E}_i\left[||\mathcal{G}([h_i^T, v_k^T]) - x_i^T||_2\right], k \neq i,$$

where i, k are time indices of the selected batch.

Symmetrically, we consider permutations on state representations h. Considering samples from the same timestep but different viewpoints, we expect identical h but different v. Similar as $\mathcal{L}_v$ above, we randomly pair h to different v and have this state permutation loss

$$\mathcal{L}_h = \mathbb{E}_j\left[||\mathcal{G}([h_i^{T_k}, v_i^{T_j}]) - x_i^{T_j}||_2\right], k \neq j,$$

where k and j indicate different TPVs. Combine viewpoint and state permutation losses together, we have our representation permutation loss:

$$\mathcal{L}_{permute} = \mathcal{L}_v + \mathcal{L}_h.$$

We do not apply the representation permutation loss above to the latent representation v^F from FPV branch, because the viewpoints can continuously change and are highly correlated with agent's egomotion in its FPV.

3) Reconstruction Loss: We have this reconstruction loss to ensure that (a) h and v are meaningful latent representations instead of trivial solutions and (b) we have a reliable decoder $\mathcal{G}$ that can reconstruct images for computing representation permutation loss above. We compare a reconstructed image $\mathcal{G}(\mathcal{F}(x))$ with input x (See Figure 2(d)):

$$\mathcal{L}_{recon} = \mathbb{E}_x\left[\sum(\mathcal{G}(\mathcal{F}(x)) - x)^2\right]. \quad (1)$$

We apply this loss to all the samples we pass through our model, including the samples used when calculating time-contrastive loss and representation permutation loss.

In conclusion, our overall loss function to learn an FPV-TPV joint representation is

$$\mathcal{L} = \alpha\mathcal{L}_{tc} + \beta\mathcal{L}_{permute} + \mathcal{L}_{recon}, \quad (2)$$

where α and β are hyper-parameters to control the weight of time-contrastive loss and invariant loss.

C. Implementation Details

We implement each of our two AEs by seven convolutional layers and seven deconvolutional layers. As for splitting representation, we can assign any dimension to v. Specifically, if the dimension of v is 0, it means we do not split our full latent representation, i.e. $h := z$ and thus $\mathcal{L}_{permute}$ is not applicable. We use TCN [5] as a baseline model that using one universal encoder for FPV and TPV inputs. It can be regarded as a 0-dimension-v model. Since we use h for policy learning, we control all models to have 16-dimension h for a fair comparison between models. We empirically set $\alpha = 1$, $\beta = 1$ in all our models except ablation studies on these hyper-parameters.

D. Imitation Learning Formulation

We follow the common setting of imitation learning in recent literature [5], [11]. Given synchronized FPV-TPV demonstrations, we first train a joint representation model (See Figure 2(a)). Then, given only one extra demonstration $\{x_i^E\}$, we let the agent execute the policy in the environment to maximize the imitation learning reward (See Figure 2(b)).

The imitation learning reward at each step R_t is assigned based the cosine similarity

$$R_t = \begin{cases} 1, & \cos(h_t^F, h_t^E) \geq \xi \\ 0, & \cos(h_t^F, h_t^E) < \xi \end{cases}, \quad (3)$$

where $[h_t^F, v_t^F] = \mathcal{F}_\theta(x_t^F)$ are the latent representations of the agent's first-person observation and h_t^E is the state representation from expert demonstration, both are at current timestep t. h_t^E will be produced by $\mathcal{F}_\theta(x_t^E)$ if we perform a first-person imitation and by $\mathcal{F}_\varphi(x_t^E)$ if it is third-person imitation learning. We use a threshold ξ to discrete the imitation reward, following the formulation as [11], [37].

E. Policy Model and Policy Training

We use a multi-layer perceptron (MLP) as our policy model to map the state representation to agent actions: $\pi : \mathcal{H} \rightarrow \mathcal{A}$. The input layer and two hidden layers contain the same dimension as the input state representation. For continuous action space, the policy outputs means and standard deviations of Gaussian distributions. For discrete action space, the policy outputs log-likelihoods of actions. We use PPO [38] to optimize the policy based on the imitation learning reward above.

V. Experiments

A. Environments and Tasks

We develop and use two simulated environments to collect data, train, and evaluate methods (See Figure 1).

1) Minecraft Environment: We develop a Minecraft game environment based on Project Malmo [39] and MineRL [40]. The agent is a game player and the task is moving itself to the target position. The agent observation is an RGB visual input from the default FPV in the game without user interfaces such that $\mathcal{O} \subseteq R^{W \times H \times 3}$. The action space is discrete, where $|\mathcal{A}| = 6$, including moving forward, backward, left, and right, and turning (along with the camera) towards left and right. The goal is to reach the target labeled as a pillar of diamond blocks (blue ones).

2) Panda Environment: We develop a simulated continuous control environment by PyBullet [41]. The agent is a Panda robot mounted on a desk. The task is reaching an object in a tray in front of the robot and picking up the object. The agent observation is an RGB visual input, $\mathcal{O} \subseteq R^{W \times H \times 3}$. The agent has an action space $\mathcal{A} \subseteq [-0.5, 0.5]^{11}$ which is the force applied to 11 joints. The camera is mounted at the end effector, i.e. the "hand" of the Panda robot. The camera follows the motion of the end effector and emulates an egocentric FPV.

B. Dataset Collection

We collect synchronized FPV-TPV videos from both environments. For the Minecraft environment, we collect 8 different demonstration trajectories with randomized target diamond block locations. Each trajectory contains synchronized 1 FPV video and 8 TPV videos. Third-person viewpoints are not controlled to be the same, so the Minecraft environment is more challenging to get a well-aligned representation. For Panda environment, we collect 30 different demonstration trajectories with randomized initial locations of the target object. The expert policy is driven by an oracle using extra information and inverse kinetics. Each trajectory contains synchronized 1 FPV video and 9 TPV videos corresponding to 9 different viewpoints. All distinct trajectories share the 9 viewpoints. We keep the number of trajectories small to better emulate the realistic setting where the amount of training data is limited. We split 20% data for each dataset as test sets.

C. Quantitative Evaluations on Representation Space

We first investigate the quality of our joint representation model in terms of alignment error [5] between FPV-TPV sequences. Given a FPV frame x_i^F having time index i and its state representation h_i^F, we find its nearest TPV state representation neighbor h_j^T that has time index j. The alignment error is the mean absolute temporal distance of i and j regularized by video frame count L over all indices i:

$$alignment\_error = \mathbb{E}_i \left[\frac{|i-j|}{L} \right]. \quad (4)$$

The lower error means the better quality of imitation reward suggesting a better representation space [5].

TABLE I
Alignment error comparison between different representation models

Model	Environment	
	Minecraft	Panda
Multi-view TCN [5]	0.2725	0.2861
Ours 0-d v (i.e. without v and $\mathcal{L}_{permute}$)	0.2606	0.2036
Ours 4-d v	0.2398	0.1892
Ours 8-d v	**0.2329**	**0.1550**
Ours 12-d v	0.2571	0.1999

As shown in Table I, our model generally has a lower alignment error than the baseline, indicating that our dual AE model can deal with visually different FPV and TPV inputs. Specifically, Minecraft results show that our model can better align videos from various unseen viewpoints than the baseline. And from the comparison between non-zero dimensions of v and 0-d v, having a disentangled representation v yields better performance than not having it, suggesting that our disentangled representation helps learn state representations. We further investigate how the non-zero dimensions of viewpoint representation v affect the representation learning quality. We try 4, 8, and 12 dimensions of v. Table I tells that 8-d has the lowest alignment error compared to 4 and 12. 12-d has worse performance than 4-d (due to the overfitting), but we note that all the variants are better than ours 0-d v baseline. We confirm that having a non-zero-dimension v helps learn a joint representation, while we should choose a suitable dimension of v empirically to ensure the information in v is well encoded, based on how much information we intend v to encode. We also learn that increasing the dimension of v is not always beneficial to learn

TABLE II

ABLATION STUDIES ABOUT LOSS FUNCTIONS ON PANDA ENVIRONMENT BASED ON OUR MODEL WITH 8-DIMENSION v.

Model based on 8-d v	Alignment Error
$\mathcal{L}_{tc} + \mathcal{L}_{permute} + \mathcal{L}_{recon}$	**0.1550**
$\mathcal{L}_{tc} + \mathcal{L}_{vmatch} + \mathcal{L}_{recon}$	0.2518
$\mathcal{L}_{tc} + \mathcal{L}_{recon}$	0.1922
$\mathcal{L}_{tc} + \mathcal{L}_{permute}$	0.1748
$\mathcal{L}_{tc}$	0.1844
$5\mathcal{L}_{tc} + \mathcal{L}_{permute} + \mathcal{L}_{recon}$	0.1695
$\mathcal{L}_{tc} + 5\mathcal{L}_{permute} + \mathcal{L}_{recon}$	0.1616

such disentangled representation, which is the overfitting problem.

We also do ablation studies about our loss functions on Panda environment (see Table II). We first compare our $\mathcal{L}_{permute}$ with a vanilla loss function $\mathcal{L}_{vmatch}$ that could be applied to v. Recall that v should be similar for inputs from the same TPV and different for inputs from different TPVs. We define $\mathcal{L}_{vmatch} = \mathcal{L}_{vsim} + \mathcal{L}_{vdissim}$ where $\mathcal{L}_{vsim} = \cos(v_i, v_j)$ and $\mathcal{L}_{vdissim} = \max\{\cos(v_i, v_j), 0\}$ for similarities and dissimilarities between v respectively. Results show that using $\mathcal{L}_{permute}$ has a lower alignment error (0.1550) than $\mathcal{L}_{vmatch}$ (0.2518), indicating $\mathcal{L}_{permute}$ is reasonable and effective on supervising v. Using $\mathcal{L}_{vmatch}$ leads to a higher alignment error than not using $\mathcal{L}_{permute}$ (0.1922), suggesting that $\mathcal{L}_{vmatch}$ provides a poor supervise signal. We infer that the $\mathcal{L}_{vdissim}$ part overemphasizes the dissimilarity among v to satisfy $\cos(v_i, v_j) \leq 0$. We also investigate the performance if we remove certain loss functions. By removing $\mathcal{L}_{recon}$ and keep $\mathcal{L}_{permute}$ (0.1748), we know that adding a general reconstruction error would help the training of the decoder. But when we compare using $\mathcal{L}_{tc}$ and $\mathcal{L}_{recon}$ (0.1922) and using $\mathcal{L}_{tc}$ (0.1844), we find that using this $\mathcal{L}_{recon}$ without $\mathcal{L}_{permute}$ is not sufficient to supervise h and v. A well-trained decoder ensures that $\mathcal{L}_{permute}$ is not affected by the under-trained decoder. As for hyper-parameters α and β, increasing α does not yield a better result while the result of increasing β is comparable. We always keep $\mathcal{L}_{tc}$, otherwise the model will easily fail to capture temporal information between states.

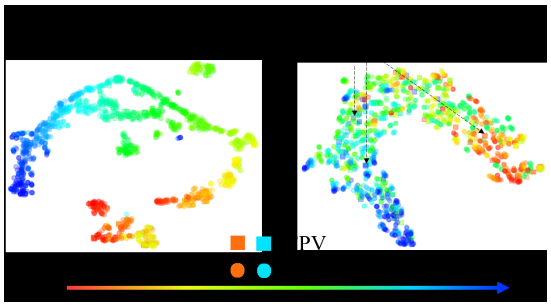

Fig. 3. Visualization of state representation spaces $\mathcal{H}$ of Panda environment. Squares are FPV representations and dots are TPV representations. The colors from red to blue indicate the timesteps from 0 to T (end of a trajectory).

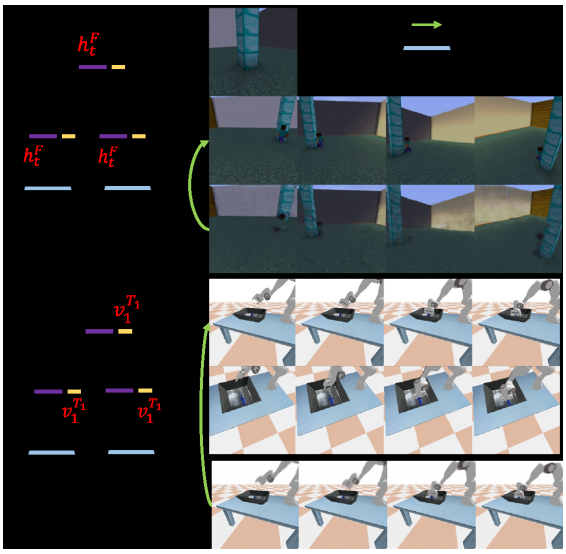

Fig. 4. Reconstruction results of representation permutation on the test sets. The green arrow means the reconstructed images (from) should be similar with ground truth (to). Up: replacing h^{T_j} of TPV frames (2nd row) by h^F from an FPV frame (1st row) at the same timestep. Bottom: replacing v^{T_j} (of 2nd row) by v^{T_i} (of 1st row) from a different viewpoint.

D. Qualitative Evaluations on Representation Space

To evaluate the state representation space qualitatively, we first show a t-SNE visualization [42] of the learned representation space on Panda environment (Figure 3). We observe a generally clear temporal order from both methods. Moreover, FPV and TPV frames are aligned together by temporal order by our method, which indicates a good joint representation for FPV and TPV. However, in the visualization of TCN [5]'s representation space, we observe many FPV representations are scattered in the space and are misaligned. This will affect the joint representation and generate a misleading reward function in third-person imitation learning. Therefore our model is more suitable to deal with the visual differences between $\mathcal{O}^F$ and $\mathcal{O}^T$.

We then show how our state representation h and viewpoint representation v encode corresponding information i.e. they are well disentangled. We follow the same permutation and reconstruction procedures as calculating $\mathcal{L}_{permute}$ in Section IV-B.2 on frames in the test set the model has never seen.

Figure 4 shows the result of replacing state representation h^{T_j} with h^F in Minecraft frames and replacing viewpoint representation v^{T_2} with v^{T_1} in Panda frames. From the result in Panda environment, we could observe the viewpoint in reconstructed images is coherent, which means changing h does not affect the viewpoint. The states are also well reconstructed if we see the poses of the robot. In the result of Minecraft environment, we could see the model reconstructs corresponding viewpoints by changing v. Even though the reconstructed agents are blurred, their positions are closed to the ground truth. We note that reconstruction is not our ultimate goal in this research, but it provides us an informative self-supervised objective to train disentangled latent representations. The clear disentanglement meets our

TABLE III

SUCCESS RATE AND CUMULATIVE REWARDS FROM (SINGLE EXAMPLE) IMITATION LEARNING. *: FIRST-PERSON IMITATION LEARNING. †: THIRD-PERSON IMITATION LEARNING.

Model	Environment	
	Minecraft (Succ. Rate / Reward Mean ± Std)	Panda (Succ. Rate / Reward Mean ± Std)
Random Policy	0 / 10.3±9.47	0 / 48.4±11.0
Single-view TCN [5] *	0.12 / 12.6±8.33	0.16 / 64.0±8.42
Ours*	0.46 / 19.8±9.34	0.36 / 69.3±8.45
Multi-view TCN [5]†	0.31 / 18.9±12.4	0.32 / 66.4±9.00
Ours†	**0.52 / 21.9±10.5**	**0.44 / 71.2±7.01**

initial motivation to separate h and v from full representation z.

E. Imitation Learning Evaluation

We finally evaluate the quality of our state representation in terms of success rates and rewards by applying it for the actual policy learning. We try both first-person imitation learning (FPIL) and third-person imitation learning (TPIL) to show that our joint state representation generalizes to both FPV and TPV domains.

1) Experiment Settings: All representation models are trained by expert trajectories in the training set. We have 6 distinct trajectories for Minecraft environment and 24 for Panda environment. We implemented two baselines for two different settings: FPIL and TPIL. For FPIL, we implement a single-view TCN [5] baseline where the representation is trained solely on FPV demonstrations. For TPIL, we implement a multi-view TCN [5] baseline where the representation is trained by FPV-TPV demonstrations. Our representation models are trained by FPV-TPV demonstrations as in TCN [5], and we apply them to both FPIL and TPIL because our representations are generalizable to both FPV and TPV inputs once learned. All the policies are trained by providing only one FPV expert demonstration for FPIL, or one expert TPV demonstration for TPIL, from the testing set.

We train each policy on Panda for 10^5 steps and on Minecraft for 5×10^4 steps by PPO [38]. We run 100 evaluation epochs for the best model achieved during training steps. Successful execution of a task is defined by reaching to the diamond block in Minecraft environment and picking and lifting the object in Panda environment. A continuous reward is defined by how much distance between the target and the agent has been minimized at any timestep in an epoch. We regularize the reward according to different randomized target locations to a fair comparison. The larger reward means a closer approaching to the target for both environments.

2) Experiment Results: Table III shows that our method outperforms the multi-view TCN [5] baseline in both FPIL and TPIL, suggesting our joint state representation space is better aligned in general and is capable of learning the policy in this egocentric TPIL setting.

In FPIL setting (2nd and 3rd rows in Table III), our method outperforms the single-view TCN [5] baseline indicating

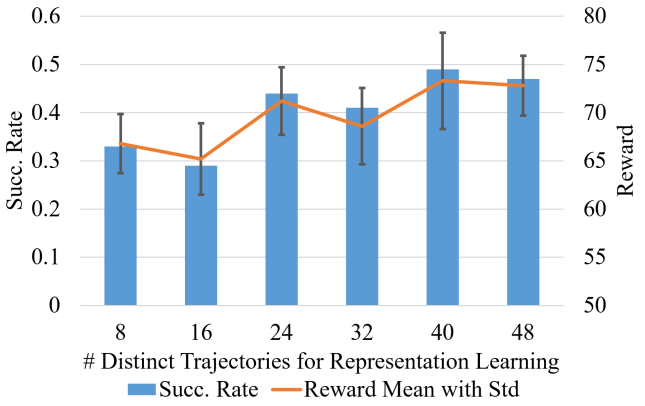

Fig. 5. Performance under different number of distinct trajectories used in representation learning. The experiment is taken in Panda environment

that (1) the representation learned from only few FPV demonstrations is limited and (2) our method takes advantage of the FPV-TPV demonstrations to shape a better state representation space than using pure FPV demonstration. By introducing third-person demonstrations from multiple viewpoints, we could learn a better representation while keeping the total number of distinct trajectories is small.

We also show how the number of trajectories used for representation model training will affect the final policy. This experiment is done using Panda environment and results are shown in Figure 5. Our method benefits from more trajectories when the trajectory count is small and the performance saturates after providing 40 trajectories. Even with 8 and 16 FPV-TPV trajectories (0.33 and 0.29 Succ. Rate), our method can reach a better performance than Single-view TCN [5] (0.16), which is trained by 24 FPV trajectories. This highlights the value of our approach, as it uses fewer trajectories to gain better performance.

VI. CONCLUSION

In this research, we consider a third-person imitation learning setting where the agent performs tasks that actions may cause agents' egomotion. We solve this egocentric third-person imitation learning by learning a joint state representation space for FPV and TPV inputs. We introduce a dual AE model to encode FPV and TPV inputs separately considering the visual differences between FPV and TPV videos. We explicitly split latent representation into state and viewpoint representations and train them to be disentangled by applying time-contrastive, representation permutation, and reconstruction losses in a self-supervised way. Results show that our representation space successfully encodes the state information with viewpoint information disentangled. We apply our joint state representation to both third-person and first-person imitation learning, and results show that our state representation is effective for learning a task out of either TPV or FPV expert demonstrations.

ACKNOWLEDGMENT

We thank the reviewers for their comments that greatly improved the manuscript. We would also show our gratitude

to Xiang Li and Ryan Burgert for inspiring discussions, figure improvements, and writing.

REFERENCES

[1] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne, "Imitation learning: A survey of learning methods," *ACM Computing Surveys (CSUR)*, vol. 50, no. 2, pp. 1–35, 2017.

[2] D. Premack and G. Woodruff, "Does the chimpanzee have a theory of mind?" *Behavioral and brain sciences*, vol. 1, no. 4, pp. 515–526, 1978.

[3] A. N. Meltzoff, "Imitation of televised models by infants," *Child development*, vol. 59, no. 5, p. 1221, 1988.

[4] B. C. Stadie, P. Abbeel, and I. Sutskever, "Third-person imitation learning," *ArXiv*, vol. abs/1703.01703, 2017.

[5] 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.

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

[7] A. D. Edwards, H. Sahni, Y. Schroecker, and C. L. Isbell, "Imitating latent policies from observation," *arXiv preprint arXiv:1805.07914*, 2018.

[8] Y. Liu, A. Gupta, P. Abbeel, and S. Levine, "Imitation from observation: Learning to imitate behaviors from raw video via context translation," *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018. [Online]. Available: http://dx.doi.org/10.1109/ICRA.2018.8462901

[9] S. Yang, W. Zhang, W. Lu, H. Wang, and Y. Li, "Cross-context visual imitation learning from demonstrations," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 5467–5473.

[10] J. Ho and S. Ermon, "Generative adversarial imitation learning," *CoRR*, vol. abs/1606.03476, 2016. [Online]. Available: http://arxiv.org/abs/1606.03476

[11] Y. Aytar, T. Pfaff, D. Budden, T. L. Paine, Z. Wang, and N. de Freitas, "Playing hard exploration games by watching youtube," 2018.

[12] D. Dwibedi, J. Tompson, C. Lynch, and P. Sermanet, "Learning actionable representations from visual observations," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 1577–1584.

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

[14] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio, "Generative adversarial nets," in *NIPS*, 2014.

[15] R. T. Q. Chen, X. Li, R. Grosse, and D. Duvenaud, "Isolating sources of disentanglement in variational autoencoders," in *Advances in Neural Information Processing Systems*, 2018.

[16] K. Sohn, X. Yan, and H. Lee, "Learning structured output representation using deep conditional generative models," in *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 2*, 2015, pp. 3483–3491.

[17] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "Infogan: interpretable representation learning by information maximizing generative adversarial nets," in *Neural Information Processing Systems (NIPS)*, 2016.

[18] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2019. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2019.00453

[19] A. Nair, S. Bahl, A. Khazatsky, V. Pong, G. Berseth, and S. Levine, "Contextual imagined goals for self-supervised robotic learning," in *Conference on Robot Learning*. PMLR, 2020, pp. 530–539.

[20] K. Zolna, S. Reed, A. Novikov, S. G. Colmenarej, D. Budden, S. Cabi, M. Denil, N. de Freitas, and Z. Wang, "Task-relevant adversarial imitation learning," *arXiv preprint arXiv:1910.01077*, 2019.

[21] X. Peng, Z. Huang, X. Sun, and K. Saenko, "Domain agnostic learning with disentangled representations," in *International Conference on Machine Learning*. PMLR, 2019, pp. 5102–5112.

[22] Z.-H. Jiang, Q. Wu, K. Chen, and J. Zhang, "Disentangled representation learning for 3d face shape," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 11 957–11 966.

[23] Z. Shu, E. Yumer, S. Hadap, K. Sunkavalli, E. Shechtman, and D. Samaras, "Neural face editing with intrinsic image disentangling," in *Computer Vision and Pattern Recognition, 2017. CVPR 2017. IEEE Conference on*. IEEE, 2017, pp. –.

[24] Z. Zhang, L. Tran, X. Yin, Y. Atoum, X. Liu, J. Wan, and N. Wang, "Gait recognition via disentangled representation learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4710–4719.

[25] E. Denton and V. Birodkar, "Unsupervised learning of disentangled representations from video," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 4417–4426.

[26] Y. Aytar, L. Castrejon, C. Vondrick, H. Pirsiavash, and A. Torralba, "Cross-modal scene networks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 10, pp. 2303–2314, 2017.

[27] C. Doersch, A. Gupta, and A. A. Efros, "Unsupervised visual representation learning by context prediction," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1422–1430.

[28] Y. Tian, D. Krishnan, and P. Isola, "Contrastive multiview coding," in *Computer Vision – ECCV 2020*, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds. Cham: Springer International Publishing, 2020, pp. 776–794.

[29] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," *arXiv preprint arXiv:2002.05709*, 2020.

[30] T. Chen, S. Kornblith, K. Swersky, M. Norouzi, and G. Hinton, "Big self-supervised models are strong semi-supervised learners," *arXiv preprint arXiv:2006.10029*, 2020.

[31] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2020. [Online]. Available: http://dx.doi.org/10.1109/cvpr42600.2020.00975

[32] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. H. Richemond, E. Buchatskaya, C. Doersch, B. A. Pires, Z. D. Guo, M. G. Azar, B. Piot, K. Kavukcuoglu, R. Munos, and M. Valko, "Bootstrap your own latent: A new approach to self-supervised learning," 2020.

[33] X. Chen and K. He, "Exploring simple siamese representation learning," 2020.

[34] A. Srinivas, M. Laskin, and P. Abbeel, "Curl: Contrastive unsupervised representations for reinforcement learning," 2020.

[35] A. Hermans, L. Beyer, and B. Leibe, "In defense of the triplet loss for person re-identification," *arXiv preprint arXiv:1703.07737*, 2017.

[36] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "beta-vae: Learning basic visual concepts with a constrained variational framework," in *ICLR*, 2017.

[37] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, and W. Zaremba, "Hindsight experience replay," 2018.

[38] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *ArXiv*, vol. abs/1707.06347, 2017.

[39] M. Johnson, K. Hofmann, T. Hutton, and D. Bignell, "The malmo platform for artificial intelligence experimentation," in *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, ser. IJCAI'16. AAAI Press, 2016, p. 4246–4247.

[40] W. H. Guss, B. Houghton, N. Topin, P. Wang, C. Codel, M. Veloso, and R. Salakhutdinov, "MineRL: A large-scale dataset of Minecraft demonstrations," *Twenty-Eighth International Joint Conference on Artificial Intelligence*, 2019. [Online]. Available: http://minerl.io

[41] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," http://pybullet.org, 2016–2019.

[42] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of machine learning research*, vol. 9, no. 11, 2008.