Learning Viewpoint-Agnostic Visual Representations by Recovering Tokens in 3D Space

Jinghuan Shang[1] **Srijan Das**[1,2] **Michael S. Ryoo**[1]
Department of Computer Science
[1]Stony Brook University, [2]University of North Carolina at Charlotte
[1]{jishang, mryoo}@cs.stonybrook.edu, [2]sdas24@uncc.edu

Abstract

Humans are remarkably flexible in understanding viewpoint changes due to visual cortex supporting the perception of 3D structure. In contrast, most of the computer vision models that learn visual representation from a pool of 2D images often fail to generalize over novel camera viewpoints. Recently, the vision architectures have shifted towards convolution-free architectures, visual Transformers, which operate on tokens derived from image patches. However, these Transformers do not perform explicit operations to learn viewpoint-agnostic representation for visual understanding. To this end, we propose a 3D Token Representation Layer (3DTRL) that estimates the 3D positional information of the visual tokens and leverages it for learning viewpoint-agnostic representations. The key elements of 3DTRL include a pseudo-depth estimator and a learned camera matrix to impose geometric transformations on the tokens, trained in an unsupervised fashion. These enable 3DTRL to recover the 3D positional information of the tokens from 2D patches. In practice, 3DTRL is easily plugged-in into a Transformer. Our experiments demonstrate the effectiveness of 3DTRL in many vision tasks including image classification, multi-view video alignment, and action recognition. The models with 3DTRL outperform their backbone Transformers in all the tasks with minimal added computation. Our code is available at `https://github.com/elicassion/3DTRL`.

1 Introduction

Over the past few years, computer vision models have developed rapidly from CNNs [6, 26, 58] to now Transformers [16, 24, 59]. With these models, we can now accurately classify objects in an image, align image frames among video pairs, classify actions in videos, and more. Despite their successes, many of the models neglect that the world is in 3D and do not extend beyond the XY image plane [20]. While humans can readily estimate the 3D structure of a scene from 2D pixels of an image, most of the existing vision models with 2D images do not take the 3D structure of the world into consideration. This is one of the reasons why humans are able to recognize objects in images and actions in videos regardless of their viewpoint, but the vision models often fail to generalize over novel viewpoints [11, 20, 42].

Consequently, in this paper, we develop an approach to learn viewpoint-agnostic representations for a robust understanding of the visual data. Naive solutions to obtain viewpoint-agnostic representation would be either supervising the model with densely annotated 3D data, or learning representation from a large scale 2D datasets with samples encompassing different viewpoints. Given the fact that such high quality data are expensive to acquire and hard to scale, an approach with a higher sample efficiency without 3D supervision is desired.

To this end, we propose a 3D Token Representation Layer (3DTRL), incorporating 3D camera transformations into the recent successful visual Transformers [7, 16, 36, 59]. 3DTRL first recovers camera-centered 3D coordinates of each token by depth estimation. Then 3DTRL estimates a camera

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

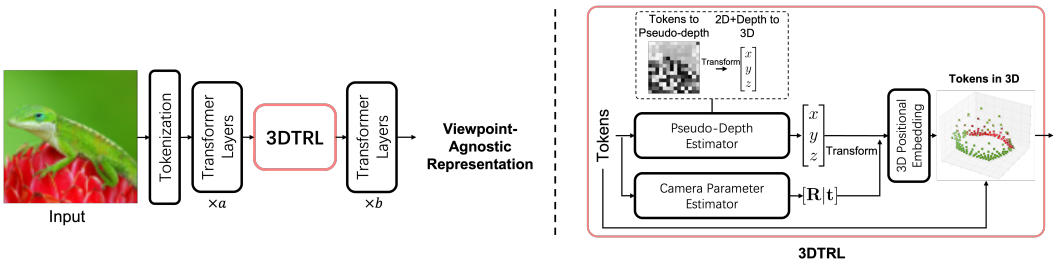

Figure 1: Overview of proposed 3DTRL. **Left**: 3DTRL is a module inserted in between Transformer layers. **Right**: 3DTRL has three parts, Pseudo-Depth Estimator, Camera Parameter Estimator, and 3D Positional Embedding layer. Within Pseudo-Depth Estimator, we first estimate depth of each token and then calculate 3D coordinates from 2D locations and depth.

matrix to transform these camera-centered coordinates to a 3D world space. In this world space, 3D locations of the tokens are absolute and view-invariant, which contain important information for learning viewpoint-agnostic representations. Therefore, 3DTRL incorporates such 3D positional information in the Transformer backbone in the form of 3D positional embeddings, and generates output tokens with the 3D information. Unlike visual Transformers only relying on 2D positional embedding, models with 3DTRL are more compliant with learning viewpoint-agnostic representations.

We conduct extensive experiments on various vision tasks to confirm the effectiveness of 3DTRL. Our 3DTRL outperforms the Transformer backbones on 3 image classification, 5 multi-view video alignment, and 2 multi-view action recognition datasets in their respective tasks. Moreover, 3DTRL is a light-weighted, plug-and-play module that achieves the above improvements with minimal (2% computation and 4% parameters) overhead.

In summary, we present a learnable, differentiable layer 3DTRL that efficiently and effectively learns viewpoint-agnostic representations.

2 Background: Pinhole Camera Model

3DTRL is based on the standard pinhole camera model widely used in computer vision. Thus, we first briefly review the pinhole camera model. In homogeneous coordinate system, given a point with world coordinate p^{world}, a camera projects a pixel at p on an image by:

$$p = \mathbf{K}\,[\,\mathbf{R}|\mathbf{t}\,]\,p^{\text{world}}, \tag{1}$$

where $\mathbf{K}$ is the intrinsic matrix and $[\,\mathbf{R}|\mathbf{t}\,]$ is the extrinsic matrix. $\mathbf{K}$ is further represented by

$$\mathbf{K} = \begin{bmatrix} c & 0 & u_0 \\ 0 & c & v_0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{2}$$

where c is the focal length and (u_0, v_0) are offset. In this work, we explore visual understanding in multi-view setting, thus aiming at learning viewpoint-agnostic representations. In this setting, a scene may be captured with different cameras positioned at non-identical locations and viewing angles. Here, the world coordinate p^{world} is the same across all the cameras while pixel projection p is different across cameras. We focus on how to estimate the world coordinates p^{world} from their corresponding image pixels at p. Estimating p^{world} from p involves two transformations that correspond to the *inverse* of $\mathbf{K}$ and $[\,\mathbf{R}|\mathbf{t}\,]$ which might not be known beforehand. Thus, we learn to *estimate* them from image patches instead, which is a key procedure in 3DTRL.

3 3D Token Representation Layer (3DTRL)

In this section, we detail how 3DTRL estimates 3D positional information in a Transformer and is integrated. We will introduce 3DTRL for image analysis, then we adapt it to video models.

3.1 Overview

3DTRL is a simple yet effective plug-in module that can be inserted in between the layers of visual Transformers (Figure 1 left). Given a set of input tokens S, 3DTRL returns a set of tokens with 3D information. The number of tokens and their dimensionality are kept unchanged. Within the

module (Figure 1 right), 3DTRL first performs 3D estimation using two estimators: (1) pseudo-depth estimator and (2) camera parameter estimator. Then the tokens are associated with their recovered world 3D coordinates, which are transformed from estimated depth and camera matrix. Finally, 3DTRL generate 3D positional embeddings from these world coordinates and combine them with input S to generate the output of 3DTRL.

In the aspect that we insert 3DTRL in between the Transformer model, 3DTRL implicitly leverages the Transformer layers *before* it to be a part of the 3D estimators, and layers *after* that to be the actual 3D feature encoder (Figure 1 left). This avoids adding a large number of parameters while resulting in reasonable estimations. We empirically find that placing 3DTRL at a shallow-medium layer of the network yields better results (See Section 4.6), which is a trade-off between model capacity for estimation and 3D feature encoding.

3.2 3D Estimation of the Input Tokens

3DTRL first estimates both the camera-centered 3D coordinates of each token using depth estimation and a camera matrix shared by all the tokens of an image. Then, the camera-centered 3D coordinates are transformed to the "world" coordinates by the camera matrix.

Pseudo-depth Estimation. Given input tokens $S = \{s_1, \ldots, s_N\} \in \mathbb{R}^{N \times m}$, 3DTRL first performs pseudo-depth estimation of each token s_n. The pseudo-depth estimator is a function $f : \mathbb{R}^m \to \mathbb{R}$ that outputs the depth $d_n = f(s_n)$ of each token individually. We implement f using a 2-layer MLP. We call this pseudo-depth estimation since it is similar to depth estimation from monocular images but operates at a very coarse scale, given that each token corresponds to an image patch rather than a single pixel in Transformer.

After pseudo-depth estimation, 3DTRL transforms the pseudo-depth map to camera-centered 3D coordinates. Recall that in ViT [16], an image X is decomposed into N patches $\{X_1, \ldots, X_N\} \in \mathbb{R}^{N \times P \times P \times 3}$, where $P \times P$ is the size of each image patch and the tokens S are obtained from a linear projection of these image patches. Thus, each token is initially associated with a 2D location on the image plane, denoted as (u, v). By depth estimation, 3DTRL associates each token with one more value d. Based on the pinhole camera model explained in Section 2, 3DTRL transforms u, v, d to a camera-centered 3D coordinate (x, y, z) by:

$$p_n^{\text{cam}} = \begin{bmatrix} x_n \\ y_n \\ z_n \end{bmatrix} = \begin{bmatrix} u_n z_n / c \\ v_n z_n / c \\ z_n \end{bmatrix}, \text{ where } z_n = d_n. \tag{3}$$

Since we purely perform the aforementioned estimation from monocular images and the camera intrinsic matrix is unknown, we simply set c to a constant hyperparameter. To define coordinate system of 2D image plane (u, v), we set the center of the original image is the origin $(0, 0)$ for convenience, so that the image plane and camera coordinate system shares the same origin. We use the center of the image patch to represent its associate (u, v) coordinates.

We believe that this depth-based 3D coordinate estimation best leverages the known 2D geometry, which is beneficial for later representation learning. We later confirm this in our ablation study (in Section 4.6), where we compare it against a variant of directly estimating (x, y, z) instead of depth.

Camera Parameter Estimation. The camera parameters are required to transform the estimated camera-centered 3D coordinates p^{cam} to the world coordinate system. These camera parameters are estimated jointly from all input tokens S. This involves estimation of two matrices, a 3×3 rotation matrix $\mathbf{R}$ and a 3×1 and translation matrix $\mathbf{t}$ through an estimator g. We implement g using a MLP. Specifically, we use a shared MLP stem to aggregate all the tokens into an intermediate representation. Then, we use two separated fully connected heads to estimate the parameters in $\mathbf{R}$ and $\mathbf{t}$ respectively. We note the camera parameter estimator as a whole: $[\ \mathbf{R}|\mathbf{t}\ ] = g(S)$. To ensure $\mathbf{R}$ is mathematically valid, we first estimate the three values corresponding to yaw, pitch and roll angles of the camera pose, and then convert them into a 3×3 rotation matrix. Research has shown that the discontinuity occurs at the boundary cases in rotation representation [63, 69], however, such corner cases are rare.

In case of generic visual understanding tasks like object classifications, we expect the camera parameter estimation to perform an "object-centric canonicalization" of images with respect to the "common" poses of the class object. This is qualitatively shown by Figure 10 in the Appendix.

Transform to World Coordinates. Now, with the camera parameters, 3DTRL transforms estimated camera-centered coordinates p^{cam} into the world space, a 3D space where 3D coordinates of the

tokens are absolute and viewpoint-invariant. Following the pinhole camera model, we recover p^{world}:

$$p_n^{\text{world}} = [\ \mathbf{R}^T | \mathbf{R}^T \mathbf{t}\ ]\ p_n^{\text{cam}}. \tag{4}$$

3.3 Incorporating 3D Positional Information in Transformers

The last step of 3DTRL is to leverage the estimated 3D positional information in Transformer backbone. For this, we choose to adopt a typical technique of incorporating positional embedding that is already used in Transformers [16, 60]. In contrast to 2D positional embedding in ViTs [16], 3DTRL learns a 3D embedding function $h : \mathbb{R}^3 \rightarrow \mathbb{R}^m$ to transform estimated world coordinates p^{world} to positional embeddings p^{3D}. This 3D embedding function h is implemented using a two-layer MLP. Then, the obtained 3D positional embedding is incorporated in the Transformer backbone by combining it with the token representations. The outcome is the final token representations $\{s^{\text{3D}}\}$:

$$s_n^{\text{3D}} = s_n + p_n^{\text{3D}},\ \text{where}\ p^{\text{3D}} = h(p^{\text{world}}). \tag{5}$$

After 3D embedding, the resultant token representations are associated with a 3D space, thus enabling the remaining Transformer layers to encode viewpoint-agnostic token representations. We ablate other ways of incorporating the 3D positional information of the tokens in Section 4.6.

3.4 3DTRL in Video Models

Notably, 3DTRL can be also easily generalized to video models. For video models, the input to 3DTRL is a set of spatial-temporal tokens $\{S_1, \ldots, S_T\}$ corresponding to a video clip containing T frames, where $S_t = \{s_{t1}, \ldots, s_{tN}\}$ are N tokens from t-th frame. We simply extend our module to operate on an additional time dimension, where depth estimation and 3D positional embedding are done for each spatial-temporal tokens s_{tn} individually: $d_{tn} = f(s_{tn}),\ p_{tn}^{\text{3D}} = h(p_{tn}^{\text{world}})$. Camera parameters are estimated per input frame (S_t) in a dissociated manner, namely $[\ \mathbf{R}|\mathbf{t}\ ]_t = g(S_t)$, resulting in a total of T camera matrices per video. We investigate another strategy of camera estimation in the supplementary, where only one camera matrix is learned for all frames. However, our studies have substantiated the effectiveness of learning dissociated camera matrices per frame.

4 Experiments

We conduct extensive experiments to demonstrate the efficacy of viewpoint-agnostic representations learned by 3DTRL in multiple vision tasks: (i) image classification, (ii) multi-view video alignment, and (iii) video action classification. We also qualitatively evaluate the pseudo-depth and camera estimation to confirm 3DTRL works.

4.1 Image Classification

In order to validate the power of 3DTRL, we first evaluate ViT [16] with 3DTRL for image classification task using CIFAR-10 [34], CIFAR-100 [34] and ImageNet-1K [15] datasets.

Training We use the training recipe of DeiT [59] for training our baseline Vision Transformer model on CIFAR and ImageNet datasets from *scratch*. We performed ablations to find an optimal location in ViTs where 3DTRL should be plugged-in (Section 4.6). Thus, in all our experiments we place 3DTRL after 4 Transformer layers, unless otherwise stated. The configuration of our DeiT-T, DeiT-S, and DeiT-B is identical to that mentioned in [59]. All our transformer models are trained for 50 and 300 epochs for CIFAR and ImageNet respectively. Further training details and hyper-parameters can be found in the supplementary material.

Results From Table 1, 3DTRL with all DeiT variants shows consistent performance improvement over the baseline on the CIFAR datasets, with only $\sim$2% computation overhead and $\sim$4% more parameters. Despite fewer training samples, 3DTRL significantly outperforms the baseline in CIFAR, showing the strong generalizability of 3DTRL when limited training data are available. We argue that multi-view data is not available in abundance, especially in domains with limited data (like in medical domain), thus 3DTRL with its ability to learn viewpoint-agnostic representations will be crucial in such domains. We also find the performance improvement on ImageNet is less than that on CIFAR. This is because ImageNet has limited viewpoints in both training and validation splits, thus reducing the significance of performing geometric aware transformations for learning view agnostic representations. In contrast, CIFAR samples present more diverse camera viewpoints, so it is a more suitable dataset for testing the quality of learned viewpoint-agnostic representations.

Table 1: Top 1 classification accuracy (%) on CIFAR-10 and 100, ImageNet-1K (IN-1K), viewpoint-perturbed IN-1K (IN-1K-*p*), and ObjectNet. We also report the number of parameters (#params) and computation (in MACs). Note that the MACs are reported w.r.t. IN-1K samples.

Method	#params	MACs	CIFAR-10	CIFAR-100	IN-1K	IN-1K-*p*	ObjectNet
DeiT-T	5.72M	1.08G	74.1	51.3	73.4	61.3	21.3
+3DTRL	5.95M	1.10G	**78.8** (+4.7)	**53.7** (+2.4)	**73.6** (+0.2)	**64.6** (+3.3)	**22.4** (+1.1)
DeiT-S	22.1M	4.24G	77.2	54.6	79.4	71.1	25.8
+3DTRL	23.0M	4.33G	**80.7** (+3.5)	**61.5** (+6.9)	**79.7** (+0.3)	**72.7** (+1.6)	**27.1** (+1.3)
DeiT-B	86.6M	16.7G	76.6	51.9	81.0	70.6	27.0
+3DTRL	90.1M	17.2G	**82.8** (+6.2)	**61.8** (+9.9)	**81.2** (+0.2)	**74.7** (+4.1)	**27.3** (+0.3)

Figure 2: **Top**: original IN-1K samples. **Bottom**: viewpoint-perturbed IN-1K samples.

Robustness on Data with Viewpoint Changes In order to emphasize the need of learning viewpoint-agnostic representations, we further test the models trained on IN-1K on two test datasets: ObjectNet [2] and ImageNet-1K-perturbed (IN-1K-*p*). ObjectNet [2] is a *test* set designed to introduce more rotation, viewpoint, and background variances in samples compared to ImageNet. Overall, ObjectNet is a very challenging dataset considering large variance in real-world distributions. IN-1K-*p* is a *viewpoint*-perturbed IN-1K validation set by applying random perspective transformations to images, constructed by our own. Example images are shown in Figure 2. We note that perspective transformation on these static images is not equivalent to real viewpoint changes. Nonetheless, it is a meaningful for a proof-of-concept experiment. Results in Table 1 show that models with 3DTRL consistently outperform their corresponding Transformer baselines in these two test sets, suggesting 3DTRL is trained to generalize across viewpoint changes.

4.2 Multi-view Video Alignment

Video alignment [18, 23, 41] is a task to learn a frame-to-frame mapping between video pairs with close semantic embeddings. In particular, we consider a *multi-view* setting that aligns videos captured from the same event but different viewpoints, which could further facilitate robot imitation learning from third-person views [4, 49, 51, 56]. Here, video pairs from the same event are temporally synchronized.

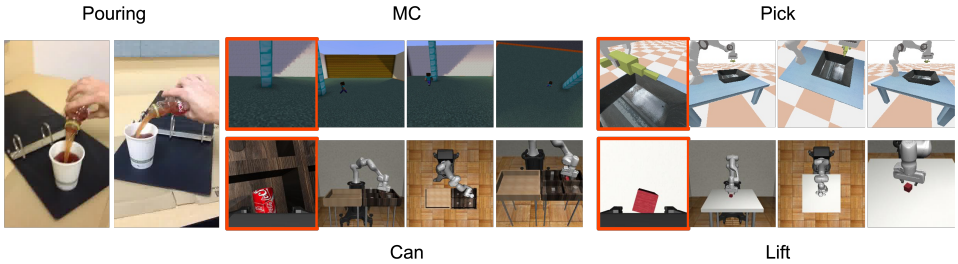

Figure 3: Examples video alignment datasets. Each dataset has synchronized videos of at least 2 viewpoints. All datasets except Pouring have one ego-centric view, highlighted in red boxes. More details are available in supplementary material.

Datasets We use 5 multi-view datasets from a wide range of environments: **Minecraft** (MC) – video game, **Pick**, **Can**, and **Lift** from robot simulators (PyBullet [9] and Robomimic [39]), and **Pouring** from real-world human actions [49]. Example video frames are provided in Figure 3. Each dataset contains synchronized videos from multiple cameras (viewpoints). There is one ego-centric camera per dataset except Pouring, which are continuously moving with the subject. These ego-centric videos make the alignment more challenging. Detail dataset statistics are available in supplementary.

Training We follow common video alignment methods [49] to train an encoder that outputs frame-wise embeddings. We still use DeiT [59] as a baseline model (DeiT+TCN) and apply 3DTRL to it (+3DTRL), similar to image classification. During training, we use the time-contrastive loss [49] to encourage temporally closed embeddings to be similar while temporally far-away embeddings to be apart. Then, we obtain alignments via nearest-neighbor such that an embedding u_i from video 1 is being paired to its nearest neighbor v_j in video 2 in the embedding space. And similarly u_j from video 2 is paired with its nearest neighbor v_k in video 1. We use ImageNet-1K pre-trained weights for experiments on Pouring, but we train from scratch for other datasets considering that simulation environments are out of real-world distribution.

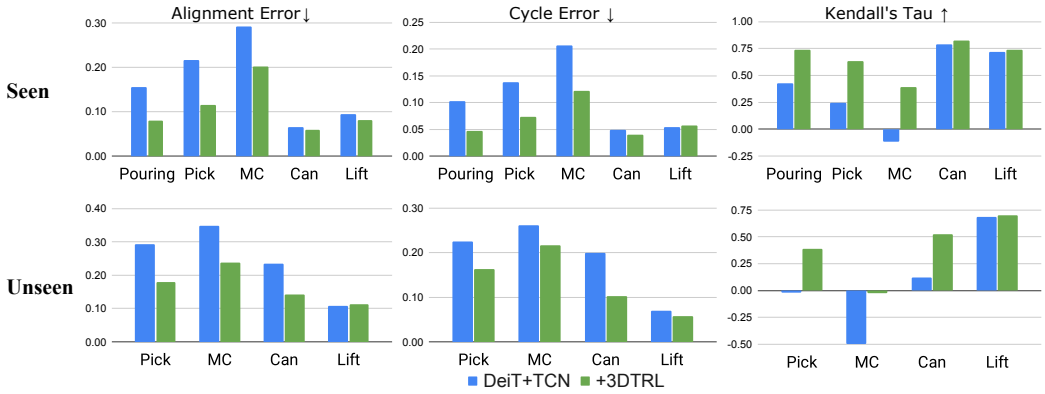

Figure 4: Results on video alignment in **Seen** and **Unseen** protocols. ↑ indicates a higher metric is better and ↓ is otherwise. Blue bars are for DeiT+TCN without 3DTRL and green bars are with 3DTRL. 3DTRL outperforms the baseline consistently in both settings. We note **Unseen** is not applicable to Pouring dataset because only two cameras are available in this dataset.

Evaluation We evaluate the alignment by three metrics: Alignment Error [49], Cycle Error, and Kendall's Tau [33]. Let the alignment pairs between two videos be (u_i, v_j) and (v_j, u_k). In brief, Alignment Error measures the temporal mismatching $|i-j|$ of (u_i, v_j). Cycle Error is based on cycle-consistency [18, 62], where two pairs (u_i, v_j) and (v_j, u_k) are called *consistent* when $i = k$. Thus, Cycle Error measures the inconsistency based on distance metric $|i-k|$. Kendall's Tau (τ) measures ordering in pairs. Given a pair of embeddings from video 1 (u_i, u_j) and their corresponding nearest neighbors from video 2 (v_p, v_q), the indices tuple (i, j, p, q) is *concordant* when $i < j$ and $p < q$ or $i > j$ and $p > q$. Otherwise the tuple is *discordant*. Kendall's Tau computes the ratio of concordant pairs and discordant pairs over all pairs of frames. Let N be the number of frames in a video, then the formal notations of the three metrics are:

$$\text{Alignment Error} = \mathbb{E}_i \frac{|i-j|}{N}; \ \text{Cycle Error} = \mathbb{E}_i \frac{|i-k|}{N}; \ \tau = \frac{\#\text{ concordant pairs} - \#\text{ discordant pairs}}{N(N-1)/2}. \tag{6}$$

We establish two evaluation protocols: (a) **Seen** and (b) **Unseen**. In **Seen**, we train and test models on videos from all cameras. However, in **Unseen**, we hold out several cameras for test, which is a representative scenario for validating the effectiveness of 3DTRL. Detail of experimental settings are provided in supplementary.

Results Figure 4 illustrates the evaluation results of 2 viewpoint settings over 5 datasets, compared to the DeiT baseline. 3DTRL outperforms the baseline consistently across all datasets. In particular, 3DTRL improves the baseline by a large margin in Pouring and MC, corroborating that 3DTRL adapts to diverse unseen viewpoints. The improvements of 3DTRL on Pick and MC also suggests the strong generalizability when learning from smaller datasets. With enough data (Lift & Can), 3DTRL still outperforms but the gap is small. When evaluating in **Unseen** setting, both methods have performance drop. However, 3DTRL still outperforms in Pick, MC, and Can, which suggests the representations learned by 3DTRL are able to generalize over novel viewpoints. MC has the largest viewpoint diversity so it is hard to obtain reasonable align results in the unseen setting for both the models.

Table 2: Video alignment results compared with SOTA methods. Values are alignment errors.

Method	Backbone	Input	Pouring	Pick	MC
TCN [49]	CNN	1 frame	0.180	0.273	0.286
Disentanglement [51]	CNN	1 frame	-	0.155	0.233
mfTCN [17]	3DCNN	8 frames	0.143	-	-
mfTCN [17]	3DCNN	32 frames	0.088	-	-
DeiT [59]+TCN	Transformer	1 frame	0.155	0.216	0.292
+3DTRL	Transformer	1 frame	**0.080**	**0.116**	**0.202**

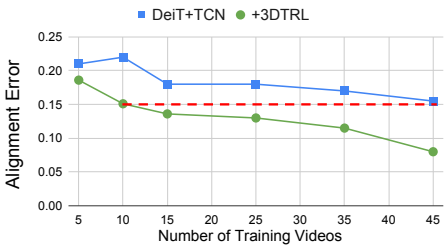

Figure 5: Alignment error w.r.t # training videos. Red dashed line indicates 3DTRL using 10 videos outperforms DeiT using 45 videos.

In Table 2, we further compare 3DTRL with previous methods on certain datasets. Note that Disentanglement [51] uses extra losses whereas others use time-contrastive loss only. We find that

3DTRL with only *single-frame* input is able to surpass the strong baselines set by models using extra losses [51] or multiple input frames [17]. We also vary the number of training videos in Pouring dataset, and results from Figure 5 show that 3DTRL benefits from more data. Meanwhile, 3DTRL can outperform the baseline while only using 22% of data the baseline used.

4.3 Quantitative Evaluations on Recovering 3D information

We perform quantitative evaluations on how well 3DTRL recovers 3D information. We emphasize once more that the 3D recovery in our approach is done without any supervision; it is optimized with respected to the final loss (e.g., object classification), without any access to the ground truth 3D information. We first discuss 3D estimations focusing on pseudo-depth estimation (Section 4.3.1), then we evaluate camera estimation (Section 4.3.2).

4.3.1 3D Estimation Evaluation

The key component of our 3D estimation is pseudo-depth estimation. In order to evaluate the 3D estimation capability, we compare the pseudo-depth map with ground truth depth map, using NeRF [40] dataset. We test the pseudo-depth with DeiT-T+3DTRL trained on IN-1K.

Metric: Depth Correlation. Since our estimated pseudo-depth (d') and ground truth (d) are in different scales, we measure their relative correspondence, i.e., correlation of two sets of data. We use Pearson's r: $r = \text{correlation}(d, d')$, where we regard two depth maps as two groups of data. Note that 3DTRL operates on visual Transformers, so the pseudo-depth map is at a very coarse scale (14×14). We also resize the given depth map to 14×14 to perform the evaluation. We report the average of r across all evaluation subsets.[1].

Results. Figure 6 shows the evaluation results. We find the estimated pseudo-depth highly correlates with the ground truth ($r \approx 0.7$). Compared to the random prediction baseline, the depth from 3DTRL shows much higher correlation to the ground truth. We also find that the model learns the estimated depth with higher correlation in the earlier stage of the training (e.g., 20 epochs) and then drops a bit in the later training epochs (~ 0.07 drop).

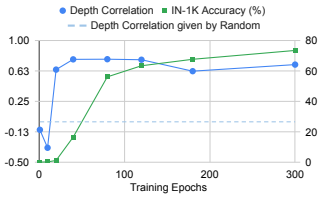

Figure 6: Depth correlation evaluation results. We find our estimation has a high ($\sim$0.7) correlation to the ground truth.

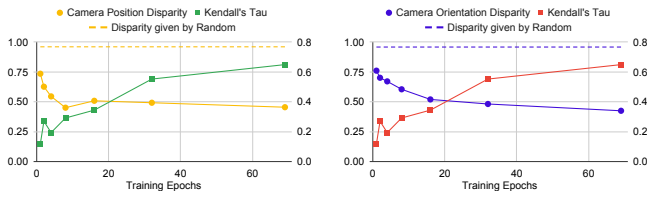

Figure 7: Quantitative evaluations on camera position (left) and camera orientation (right) respectively. Overall, we show the estimated cameras from 3DTRL have an acceptable mapping to the ground truth (both give < 0.5 disparity).

4.3.2 Camera Estimation Evaluation

In this evaluation, we mainly answer how well 3DTRL estimates camera position and orientation. To do this, we evaluate DeiT+3DTRL trained in previous video-alignment task (Section 4.2), using **Can** dataset. Recall that we train the model from scratch for the video alignment, without access to ground truth 3D information. We use first-person view videos for our evaluation — the camera moves together with the robot and our objective is to estimate its pose.

Metrics. We introduce two metrics for this evaluation.

Camera Position Disparity: For each video, we get the estimated camera positions $\{\mathbf{p}'\}$ and ground truth camera positions $\{\mathbf{p}\}$, which are 3-D vectors. Since estimated and ground truth cameras are in two different coordinate systems, we need to measure how estimated camera positions map to the ground truth in a scale, translation and rotation-invariant way. The existing metrics like AUROC [30, 48] are not applicable. Therefore, we use Procrustes analysis [21] and report the disparity metric. The disparity value ranges $[0, 1]$, where a lower value indicates that two sets are more similar. We take the average disparity of all videos.

[1]To do so, we convert r to Fisher's z, take the mean value across all subsets, and convert back to Pearson's r.

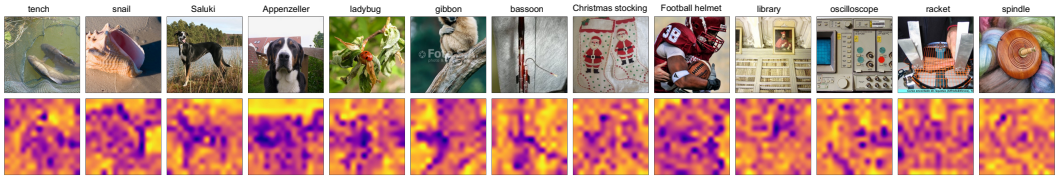

Figure 8: **Top**: IN-1K samples with corresponding class label. **Bottom**: Estimated pseudo-depth map, interpolated from 16×16 to 224×224 for better understanding. Depth increases from blueish to yellowish region. More examples are in Figure 14 in Appendix.

Camera Orientation Disparity: Each camera has its orientation, i.e. "looking-at" direction, described by a 3-D vector. We note the estimated camera orientations $\{\mathbf{o}'\}$ and ground truth $\{\mathbf{o}\}$. Similar as the camera position disparity mentioned above, we use the disparity given by Procrustes analysis [21] to measure how $\{\mathbf{o}'\}$ and $\{\mathbf{o}\}$ align. We report the mean disparity value from all the videos.

Results. Results are shown in Figure 7. Both position and orientation disparity of the final model are relatively small (< 0.5), showing that the camera estimation from 3DTRL is partially aligned with the ground truth. We also observe that the disparity decreases over the training epochs.

4.4 Qualitative Evaluations

We visualize some qualitative results for an intuitive understanding of 3DTRL's effectiveness. The visualization contains estimated pseudo-depth maps which are from the key intermediate step of 3DTRL. We use ImageNet-1K trained DeiT-T+3DTRL and its validation samples for visualization. In Figure 8, we observe a fine separation of the object-of-interest and the background in most of the pseudo-depth maps. Thus, these predicted pseudo-depth maps are sufficient to recover the tokens corresponding to object-of-interest in 3D space. For a qualitative evaluation on camera pose, please refer to Figure 10 in the Appendix, where we show the images with the same/similar object pose (in different environments) result in similar estimated extrinsics regardless of the background. We believe our estimated extrinsics are doing object-centric canonicalization of images with respect to their object poses, in order to optimize the representations for the final losses.

4.5 3DTRL on More Transformer Architectures

3DTRL is designed to be a plug-and-play model for Transformers. We test 3DTRL with more Transformer architectures on CIFAR and two multi-view video alignment datasets. We use Swin [36] (Tiny) and TnT [24] (Small). Results are provided in Table 3. We find that 3DTRL generally improves the performance of two Transformer architectures in all the datasets. This confirms 3DTRL is applicable to different Transformer architectures. The relative small improvement over the Swin backbone is due to the strong inductive bias from its local window.

Table 3: Results of using 3DTRL in more Transformer architectures, on CIFAR and multi-view video alignment datasets. Reported numbers are accuracy for CIFAR and Kendall's tau for video alignment, both are the higher the better. We show 3DTRL generally improves the performance in all tasks.

Model	CIFAR-10	CIFAR-100	Pouring	Pick
Swin-T	50.11	21.53	0.584	0.623
+3DTRL	**50.29** (+0.18)	**21.55** (+0.02)	**0.683** (+0.099)	**0.640** (+0.017)
TnT-S	81.25	54.07	0.740	0.640
+3DTRL	**82.43** (+1.18)	**56.00** (+1.93)	**0.792** (+0.052)	**0.671** (+0.031)

4.6 Ablation Studies

We conduct our ablation studies on image models mostly using *CIFAR* for image classification and *Pick* for multi-view video alignment.

MLP vs. 3DTRL. 3DTRL is implemented by several MLPs with required geometric transforms in between. In this experiment, we replace 3DTRL with the similar number of fully-connected layers with residual connection, to have comparable parameters and computation as 3DTRL. Results are provided in Table 4. We find that MLP implementation is only comparable with the baseline performance despite the increase in parameters and computation. Thus, we confirm that the geometric transformations imposed on the token representations is the key to make 3DTRL effective.

Table 4: Ablation study results. For CIFAR, we test on models based on tiny (T), small (S) and base (B) backbones (DeiT) and report accuracy(%). For Pick we only test base model and report alignment error.

Method	CIFAR-10 (T/S/B)	CIFAR-100 (T/S/B)	Pick
DeiT	74.1/ 77.2 / 76.6	51.3 / 54.6 / 51.9	0.216
DeiT + MLP	74.2 / 77.2 / 76.5	47.9 / 54.7 / 53.4	0.130
DeiT + 3DTRL	78.8 / 80.7 / 82.8	**53.7** / 61.5 / **61.8**	**0.116**
Depth Estimation $\rightarrow$ xyz Estimation	76.7 / 78.2 / 77.4	48.3 / 54.1 / 52.6	0.134
Embedding $\rightarrow$ Concat.	**80.7** / **83.7** / **84.9**	53.4 / **61.8** / 60.2	0.133

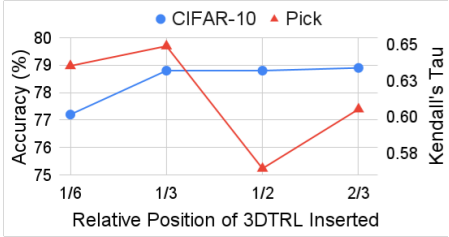

Figure 9: Results on inserting 3DTRL at different locations.

Token Coordinates Estimation. In this ablation, we show how estimating only depth compared to estimating a set of 3 coordinates xyz differs in 3DTRL. We find at Line 4 and 5 in Table 4 that depth estimation is better because it uses precise 2D image coordinates when recovering tokens in 3D space, whereas estimating xyz regresses for the 3D coordinates without any geometric constraints. Also, we find that estimating xyz hampers the performance in image classification task more than video alignment. This is because estimating 3D coordinates is harder when the training samples are in unconstrained scenarios. In contrast, video pairs in an alignment dataset share the same scene captured from different view angles, which facilitates the recovery of token in 3D space.

How to incorporate 3D positional information in Transformers? By default, we use Equation 5 to incorporate the 3D positional information in Transformer backbone through a learned positional embedding p^{3D}. In this experiment, we directly infuse the estimated 3D world coordinates p^{world} within the token representation by concatenating them across the channel axis. The $(m+3)$-d feature is then projected back to m-d by a MLP. We keep parameters and computation comparable to the default 3DTRL. We test this 3DTRL variant (Embedding $\rightarrow$ Concat.) and results are presented in Table 4. We find that the concatenation variant outperforms the default variant of 3DTRL on CIFAR-10, but comparable and worse results in CIFAR-100 and Pick. This observation substantiates the instability of using raw 3D coordinates. In comparison, the use of 3D positional embedding is generalizable to more challenging and diverse scenarios.

Where should we have 3DTRL? We vary the location of 3DTRL to empirically study the optimal location of 3DTRL in a 12-layer DeiT-T backbone. In Figure 9, we find that inserting 3DTRL at the earlier layers (after 1/3 of the network) yields the best performance consistently on both datasets.

4.7 3DTRL for Video Representation Learning

In this section, we illustrate how 3DTRL can be adapted for video models. Since we aim at learning viewpoint-agnostic representations, as a natural choice we validate the effectiveness of 3DTRL on video datasets with multi-camera setups and cross-view evaluation. Consequently, we conduct our experiments on two multi-view action recognition datasets: **Toyota Smarthome** [12] (Smarthome) and **NTU-RGB+D** [50] (NTU) for the task of action classification. For evaluation on Smarthome, we follow Cross-View 2 (CV2) and Cross-Subject (CS) protocols proposed in [12], whereas on NTU, we follow Cross-View (CV) protocol proposed in [50]. In cross-view protocols, the model is trained on a set of cameras and tested on a different set of cameras. Similarly for cross-subject protocol, the model is trained and tested on different set of subjects. Note that for Smarthome dataset, we report the action classification accuracy on two metrics (i) classification accuracy (*Acc*) and (ii) mean per-class accuracy (*mPA*). More details on these datasets are provided in the supplementary material.

Table 5: Results on action recognition on Smarthome and NTU. *Acc* is classification accuracy (%) and *mPA* is mean per-class accuracy. In methods using Kinetics-400 (K400) pre-training, TimeSformer backbone is always initialized with pre-trained weights, and 3DTRL w/, w/o K400 denotes 3DTRL is randomly initialized and is initialized from pre-trained weights respectively.

Method	Smarthome (CV2)		Smarthome (CS)		NTU (CV)
	Acc	*mPA*	*Acc*	*mPA*	*Acc*
TimeSformer [3]	59.4	27.5	75.7	56.1	86.4
+ 3DTRL	**62.9** (+3.5)	**34.0** (+6.5)	**76.1** (+0.4)	**57.0** (+0.9)	**87.9** (+1.5)
Kinetics-400 pre-trained					
TimeSformer [3]	69.3	37.5	77.2	57.7	87.7
+ 3DTRL w/o K400	69.5 (+0.2)	39.2 (+1.7)	77.5 (+0.3)	58.9 (+1.2)	**88.8** (+1.1)
+ 3DTRL w/ K400	**71.9** (+2.6)	**41.7** (+4.2)	**77.8** (+0.6)	**61.0** (+2.3)	88.6 (+0.9)

Network architecture & Training / Testing TimeSformer [3] is a straightforward extension of ViT [16] for videos which operates on spatio-temporal tokens from videos, so that 3DTRL can be easily deployed to TimeSformer as well. Similar to our previous experimental settings, we place 3DTRL after 4 Transformer blocks in TimeSformer. Please refer to Appendix I for detailed settings.

Results In Table 5, we present the action classification results on Smarthome and NTU datasets with 3DTRL plugged in TimeSformer.

3DTRL can easily take advantage of pre-trained weights because it does not change the relying backbone Transformer – just being added in between blocks. In Table 5, we present results for two fine-tuning scenarios: (a) 3DTRL w/o K400 and (b) 3DTRL w/ K400. For the first scenario (a), TimeSformer is initialized with K400 pre-training weights and leave the parameters in 3DTRL randomly initialized. Then in the fine-tuning stage, all the model parameters including those of 3DTRL is trained. In the second scenario (b), all parameters in TimeSformer and 3DTRL are pre-trained on K400 from scratch and fine-tuned on the respective datasets.

We find that all the variants of 3DTRL outperforms the baseline TimeSformer results. Our experiments show that although there is an improvement with 3DTRL compared to the baseline for different fine-tuning strategy, it is more significant when 3DTRL is pre-trained with K400. However, when large-scale training samples are available (NTU), 3DTRL does not require K400 pre-training. To sum up, 3DTRL can be seen as a crucial ingredient for learning viewpoint-agnostic video representations.

5 Related Work

There has been a remarkable progress in visual understanding with the shift from the use of CNNs [6, 26, 35, 58] to visual Transformers [16]. Transformers have shown substantial improvements over CNNs in image [7, 16, 24, 31, 36, 59, 65] analysis and video [1, 3, 37, 44, 47] understanding tasks due to its flexibility in learning global relations among visual tokens. Studies also combine CNNs with Transformer architectures to leverage the pros in both the structures [10, 22, 38, 52, 53]. In addition, Transformer has been shown to be effective in learning 3D representation [68]. However, these advancements in architecture-types have not addressed the issue of learning viewpoint-agnostic representation. Viewpoint-agnostic representation learning is drawing increasing attention in the vision community due to its wide range of downstream applications like 3D object-detection [46],video alignment [8, 18, 19], action recognition [54, 55], pose estimation [25, 57], robot learning [4, 27, 29, 49, 51, 56], and other tasks.

There is a broad line of work towards directly utilizing 3D information like depth [25], pose [13, 14], and point clouds [43, 45], or in some cases deriving 3D structure from paired 2D inputs [61]. However, methods rely on the availability of multi-modal data which is hard to acquire are not scalable.

Consequently, other studies have focused on learning 3D perception of the input visual signal in order to generalize the learned representation to novel viewpoints. This is done by imposing explicit geometric transform operations in CNNs [5, 28, 42, 46, 64], without the requirement of any 3D supervision. In contrast to these existing works, our Transformer-based 3DTRL imposes geometric transformations on visual tokens to recover their representation in a 3D space. To the best of our knowledge, 3DTRL is the first of its kind to learn a 3D positional embedding associated with the visual tokens for viewpoint-agnostic representation learning in different image and video tasks.

6 Conclusion

In this work, we have presented 3DTRL, a plug-and-play module for visual Transformer that leverages 3D geometric information to learn viewpoint-agnostic representations. Within 3DTRL, by pseudo-depth estimation and learned camera parameters, it manages to recover positional information of tokens in a 3D space. Through our extensive experiment, we confirm 3DTRL is generally effective in a variety of visual understanding tasks including image classification, multi-view video alignment, and cross-view action recognition, by adding minimum parameters and computation overhead.

Acknowledgment

We thank insightful discussions with members of Robotics Lab at Stony Brook. This work is 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 is also supported by the National Science Foundation (IIS-2104404 and CNS-2104416).

References

[1] A. Arnab, M. Dehghani, G. Heigold, C. Sun, M. Lučić, and C. Schmid. Vivit: A video vision transformer. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 6836–6846, 2021.

[2] A. Barbu, D. Mayo, J. Alverio, W. Luo, C. Wang, D. Gutfreund, J. Tenenbaum, and B. Katz. Objectnet: A large-scale bias-controlled dataset for pushing the limits of object recognition models. *Advances in neural information processing systems*, 32, 2019.

[3] G. Bertasius, H. Wang, and L. Torresani. Is space-time attention all you need for video understanding? In *Proceedings of the International Conference on Machine Learning (ICML)*, July 2021.

[4] R. Burgert, J. Shang, X. Li, and M. Ryoo. Neural neural textures make sim2real consistent. *arXiv preprint arXiv:2206.13500*, 2022.

[5] A.-Q. Cao and R. de Charette. Monoscene: Monocular 3d semantic scene completion. *arXiv preprint arXiv:2112.00726*, 2021.

[6] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *The British Machine Vision Conference (BMVC)*, 2014.

[7] C.-F. R. Chen et al. Crossvit: Cross-attention multi-scale vision transformer for image classification. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021.

[8] M. Chen, F. Wei, C. Li, and D. Cai. Frame-wise action representations for long videos via sequence contrastive learning. *arXiv preprint arXiv:2203.14957*, 2022.

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

[10] Z. Dai, H. Liu, Q. V. Le, and M. Tan. Coatnet: Marrying convolution and attention for all data sizes. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[11] S. Das and M. S. Ryoo. Viewclr: Learning self-supervised video representation for unseen viewpoints. *Arxiv*, 2112.03905, 2021.

[12] S. Das, R. Dai, M. Koperski, L. Minciullo, L. Garattoni, F. Bremond, and G. Francesca. Toyota smarthome: Real-world activities of daily living. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2019.

[13] S. Das, S. Sharma, R. Dai, F. Bremond, and M. Thonnat. Vpn: Learning video-pose embedding for activities of daily living. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 72–90. Springer, 2020.

[14] S. Das, R. Dai, D. Yang, and F. Bremond. Vpn++: Rethinking video-pose embeddings for understanding activities of daily living. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[15] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255, 2009.

[16] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

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

[18] D. Dwibedi, Y. Aytar, J. Tompson, P. Sermanet, and A. Zisserman. Temporal cycle-consistency learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1801–1810, 2019.

[19] J. Gao, M. Chen, and C. Xu. Fine-grained temporal contrastive learning for weakly-supervised temporal action localization. *arXiv preprint arXiv:2203.16800*, 2022.

[20] J. J. Georgia Gkioxari, Jitendra Malik. Mesh r-cnn. *Proceedings of the International Conference on Computer Vision (ICCV)*, 2019.

[21] J. C. Gower. Generalized procrustes analysis. *Psychometrika*, 40(1):33–51, 1975.

[22] B. Graham, A. El-Nouby, H. Touvron, P. Stock, A. Joulin, H. Jégou, and M. Douze. Levit: a vision transformer in convnet's clothing for faster inference. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 12259–12269, 2021.

[23] I. Hadji, K. G. Derpanis, and A. D. Jepson. Representation learning via global temporal alignment and cycle-consistency. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11068–11077, 2021.

[24] K. Han et al. Transformer in transformer. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[25] A. Haque, B. Peng, Z. Luo, A. Alahi, S. Yeung, and L. Fei-Fei. Towards viewpoint invariant 3d human pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 160–177. Springer, 2016.

[26] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[27] K. Hsu, M. J. Kim, R. Rafailov, J. Wu, and C. Finn. Vision-based manipulators need to also see from their hands. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2022.

[28] M. Jaderberg, K. Simonyan, A. Zisserman, et al. Spatial transformer networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 28, 2015.

[29] R. Jangir, N. Hansen, S. Ghosal, M. Jain, and X. Wang. Look closer: Bridging egocentric and third-person views with transformers for robotic manipulation. *IEEE Robotics and Automation Letters (RA-L)*, 2022.

[30] L. Jin, S. Qian, A. Owens, and D. F. Fouhey. Planar surface reconstruction from sparse views. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12991–13000, 2021.

[31] K. Kahatapitiya and M. S. Ryoo. Swat: Spatial structure within and among tokens. *arXiv preprint arXiv:2111.13677*, 2021.

[32] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017.

[33] M. G. Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938.

[34] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[35] Y. LeCun, Y. Bengio, et al. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995, 1995.

[36] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021.

[37] Z. Liu, J. Ning, Y. Cao, Y. Wei, Z. Zhang, S. Lin, and H. Hu. Video swin transformer. *arXiv preprint arXiv:2106.13230*, 2021.

[38] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie. A convnet for the 2020s. *arXiv preprint arXiv:2201.03545*, 2022.

[39] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *arXiv preprint arXiv:2108.03298*, 2021.

[40] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 405–421. Springer, 2020.

[41] I. Misra, C. L. Zitnick, and M. Hebert. Shuffle and learn: Unsupervised learning using temporal order verification. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 527–544. Springer, 2016.

[42] A. Piergiovanni and M. S. Ryoo. Recognizing actions in videos from unseen viewpoints. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4124–4132, June 2021.

[43] A. Piergiovanni, V. Casser, M. S. Ryoo, and A. Angelova. 4d-net for learned multi-modal alignment. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 15435–15445, 2021.

[44] K. Ranasinghe, M. Naseer, S. Khan, F. S. Khan, and M. Ryoo. Self-supervised video transformer. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[45] D. Robert, B. Vallet, and L. Landrieu. Learning multi-view aggregation in the wild for large-scale 3d semantic segmentation. *arXiv preprint arXiv:2204.07548*, 2022.

[46] D. Rukhovich, A. Vorontsova, and A. Konushin. Imvoxelnet: Image to voxels projection for monocular and multi-view general-purpose 3d object detection. In *IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 2397–2406, 2022.

[47] M. S. Ryoo, A. Piergiovanni, A. Arnab, M. Dehghani, and A. Angelova. Tokenlearner: What can 8 learned tokens do for images and videos? *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[48] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich. Superglue: Learning feature matching with graph neural networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4938–4947, 2020.

[49] 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 *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1134–1141. IEEE, 2018.

[50] A. Shahroudy, J. Liu, T.-T. Ng, and G. Wang. Ntu rgb+d: A large scale dataset for 3d human activity analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[51] J. Shang and M. S. Ryoo. Self-supervised disentangled representation learning for third-person imitation learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 214–221, 2021. doi: 10.1109/IROS51168.2021.9636363.

[52] J. Shang, K. Kahatapitiya, X. Li, and M. S. Ryoo. Starformer: Transformer with state-action-reward representations. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2022.

[53] J. Shang, X. Li, K. Kahatapitiya, Y.-C. Lee, and M. S. Ryoo. Starformer: Transformer with state-action-reward representations for robot learning. *IEEE transactions on pattern analysis and machine intelligence*, 2022.

[54] G. A. Sigurdsson, A. Gupta, C. Schmid, A. Farhadi, and K. Alahari. Actor and observer: Joint modeling of first and third-person videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7396–7404, 2018.

[55] G. A. Sigurdsson, A. Gupta, C. Schmid, A. Farhadi, and K. Alahari. Charades-ego: A large-scale dataset of paired third and first person videos. *arXiv preprint arXiv:1804.09626*, 2018.

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

[57] J. J. Sun, J. Zhao, L.-C. Chen, F. Schroff, H. Adam, and T. Liu. View-invariant probabilistic embedding for human pose. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 53–70. Springer, 2020.

[58] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2818–2826, 2016. doi: 10.1109/CVPR.2016.308.

[59] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jegou. Training data-efficient image transformers: Distillation through attention. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 139, pages 10347–10357, July 2021.

[60] A. Vaswani et al. Attention is all you need. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[61] S. Vijayanarasimhan, S. Ricco, C. Schmid, R. Sukthankar, and K. Fragkiadaki. Sfm-net: Learning of structure and motion from video. *arXiv preprint arXiv:1704.07804*, 2017.

[62] X. Wang, A. Jabri, and A. A. Efros. Learning correspondence from the cycle-consistency of time. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2566–2576, 2019.

[63] S. Xiang. Eliminating topological errors in neural network rotation estimation using self-selecting ensembles. *ACM Transactions on Graphics (TOG)*, 40(4):1–21, 2021.

[64] X. Yan, J. Yang, E. Yumer, Y. Guo, and H. Lee. Perspective transformer nets: Learning single-view 3d object reconstruction without 3d supervision. *Advances in Neural Information Processing Systems (NeurIPS)*, 29, 2016.

[65] L. Yuan, Y. Chen, T. Wang, W. Yu, Y. Shi, Z.-H. Jiang, F. E. Tay, J. Feng, and S. Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 558–567, 2021.

[66] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 6023–6032, 2019.

[67] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz. mixup: Beyond empirical risk minimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

[68] H. Zhao, L. Jiang, J. Jia, P. H. Torr, and V. Koltun. Point transformer. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 16259–16268, October 2021.

[69] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li. On the continuity of rotation representations in neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5745–5753, 2019.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyper-parameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Qualitative Evaluation on Camera Estimation

We collected images of the same object with various object poses and backgrounds, and visualized their camera estimation in Figure 10. This was done with DeiT-T+3DTRL trained on ImageNet. We observe 3DTRL estimates similar camera poses (clustered in the red-dashed circle) for similar object poses regardless of different backgrounds. When given different object poses, 3DTRL estimates cameras in scattered position and orientations. We notice that there is an outlier whose estimation is also introduced in this cluster, owing to the model invariance to horizontal flipping which is used as an augmentation during training. This visualization suggests that 3DTRL, when trained for the object classification task, might be performing object-centric canonicalization of the input images.

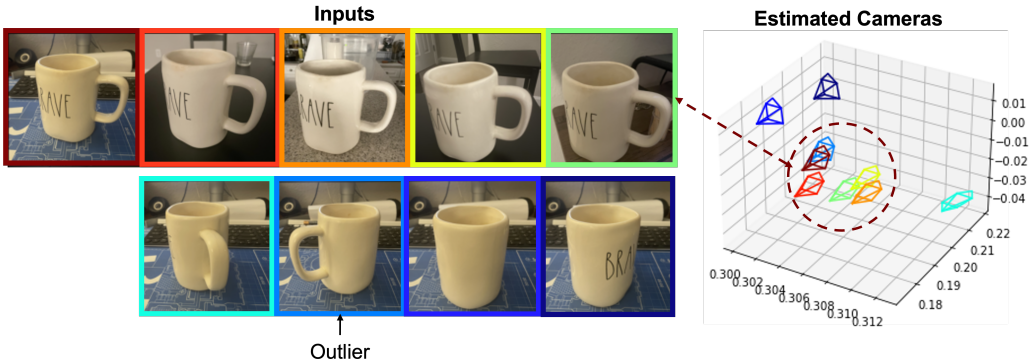

Figure 10: Qualitative experiment on how 3DTRL reacts to similar and different poses of the same object. Photos are taken by a regular smartphone. We use DeiT-T+3DTRL trained on ImageNet. The estimated cameras from the similar poses (first row) are clustered as shown in the red-dashed circle, while the other cameras from different poses (second row) are apart. We notice an outlier whose estimation is also introduced in this cluster, owing to the model invariance to horizontal flipping which is used as an augmentation during training.

We also present more qualitative visualizations of estimated camera positions (Figure 11) and estimated 3D world locations of image patches (Figure 12). We find that the estimations approximately reflect the ground truth or human perception which the model has no access to during training. These estimations are not necessarily required to be perfectly aligned with ground truth, but the results show that they are reasonable and sufficient for providing 3D information.

B Experiment on Perspective Augmentation

In this experiment, we investigate whether perspective augmentation applied to input images will help the model to learn viewpoint-agnostic representations. We test with DeiT+3DTRL on multi-view video alignment task. The training procedure is the same for all variants. Results are shown in Table 6. From the results we show that naively adding perspective augmentation does not improve the viewpoint-agnostic representation learning. Instead, it harms the performance compared to the DeiT baseline, since the perspective augmentation is overly artificial compared to the real-world viewpoint changes. Such augmentation does not contribute to viewpoint-agnostic representation learning.

Table 6: Comparison between 3DTRL and perspective augmentation on training data. Overall, perspective augmentation shows a negative effect on all the tasks, because the perspective augmentation on image is not the real viewpoint change.

Model	Pouring	Pick	MC	Can	Lift
DeiT	0.426	0.244	-0.115	0.789	0.716
DeiT + Perspective Augmentation	0.200	-0.249	-0.419	0.342	0.486
DeiT + 3DTRL	**0.740**	**0.635**	**0.392**	**0.824**	**0.739**

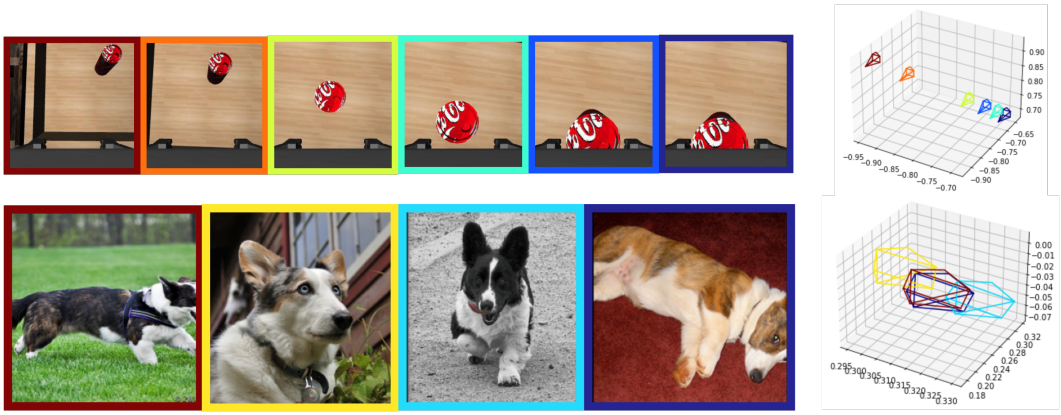

Figure 11: Visualization of image samples (left, with colored boundaries) and estimated camera positions in a 3D space (right). The color of the boundary on each image corresponds to the estimated camera from that image. **Top**: Images are from a video clip captured by an egocentric (eye-in-hand) camera on a robot arm in Can environment, ordered by timestep from left to right. The estimated camera positions approximately reflects the motion of the robot arm, which is moving towards right and down. **Bottom**: Samples from ImageNet-1K. The estimated camera pose of the second image (yellow boundary) is somehow at a head-up view, and the rest are at a top-down view. These estimated cameras are approximately aligned with human perception.

C Discussion on Pseudo-depth Estimation

Most of images from ImageNet have a simple scene (background), so it's easier for the pseudo-depth estimation to focus on objects. In examples shown in Figure 13, we show that the pseudo-depth is also estimated for other foreground objects apart from the primary class object.

D More Ablation Studies

D.1 How many 3DTRLs should be used?

We explore the possibility of using multiple 3DTRLs in a Transformer backbone. This usage is essentially estimating 3D information from different levels of features, and injecting such information back into the backbone. We test several DeiT-T variants using multiple 3DTRLs on CIFAR-10 and results are shown in Table 7. We find that using multiple 3DTRLs further increases the performance in general compared to using only one 3DTRL. This shows the extra capacity from multiple 3DTRLs benefits representation learning. Specifically, we demonstrate that inserting 3DTRLs at layer 4, 6, and 8 yields the best result among all the strategies we explore. This experiment empirically shows multiple 3DTRLs potentially benefit the model.

Table 7: CIFAR-10 Performance when using multiple 3DTRLs based on DeiT-T.

3DTRL Location(s)	N/A (DeiT baseline)	4	4, 6, 8	4, 4, 4	2, 4, 6, 8
CIFAR-10	74.1	78.8	**79.5**	79.3	79.1

D.2 Regularization effect of 3DTRL

Mixup [67] and CutMix [66] are commonly used image augmentation methods in training image models, which mixes two images to create an augmented image for training input. Such technique along with other augmentations provides diverse image samples so that the image model is regularized and avoids overfitting. We hypothesise that Mixup & CutMix could potentially damage structures in original image, which may cause inaccurate depth estimation in 3DTRL, thus hamper the training procedure. Therefore, we conduct an experiment on ImageNet-1K of disabling Mixup & CutMix. We train baseline DeiT and DeiT+3DTRL from scratch, and compare the results in both original

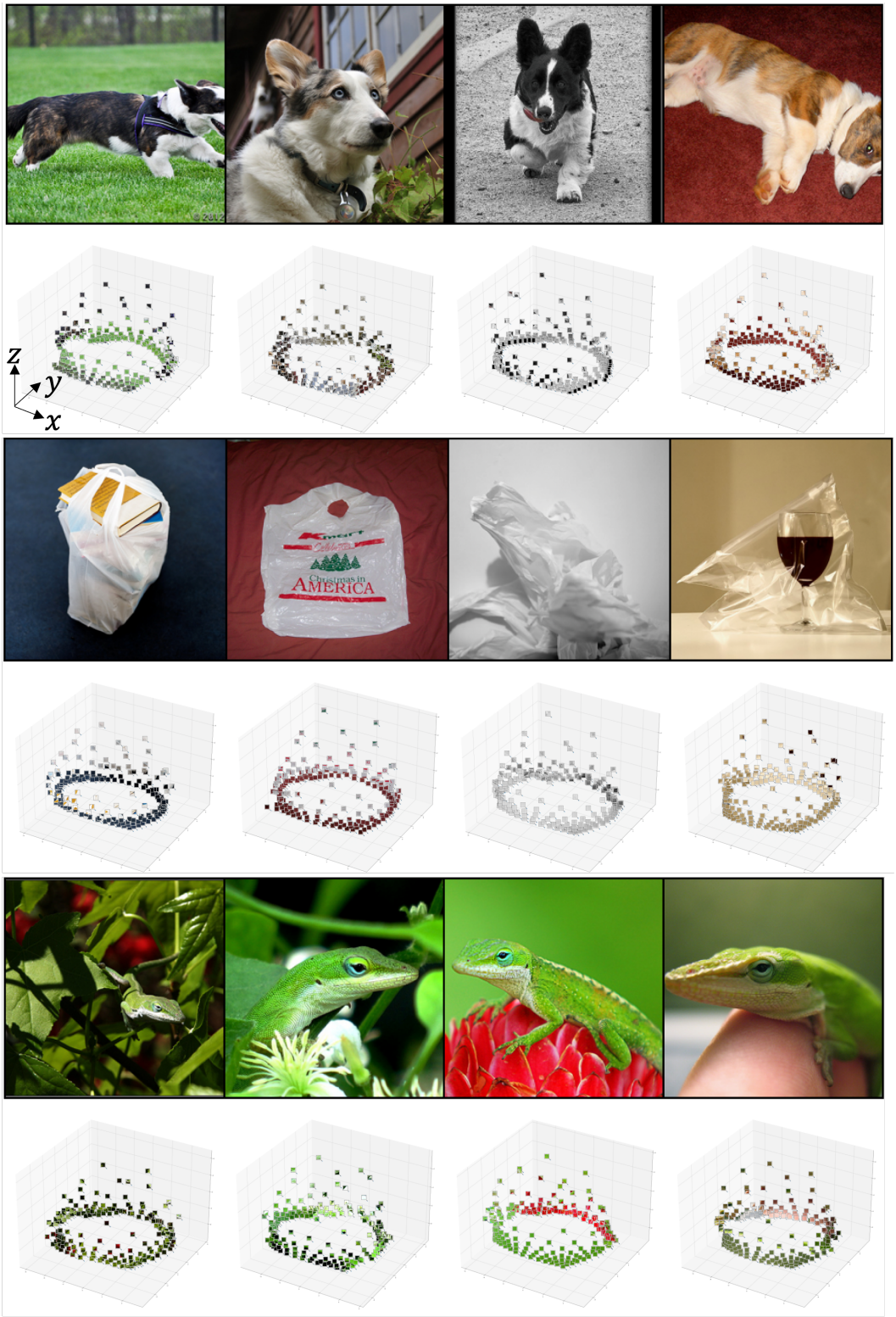

Figure 12: Visualization of image patches at their estimated 3D world locations. The center of the xy-plane (horizontal plane, at the bottom) is the origin of the 3D space. The vertical axis is z-axis. The patches corresponds to object-of-interest usually have larger z values (corresponding to larger pseudo-depth values), which are localized "farther" from the origin. Most of the background patches have smaller z values that are at the "closer" to the xy-plane.

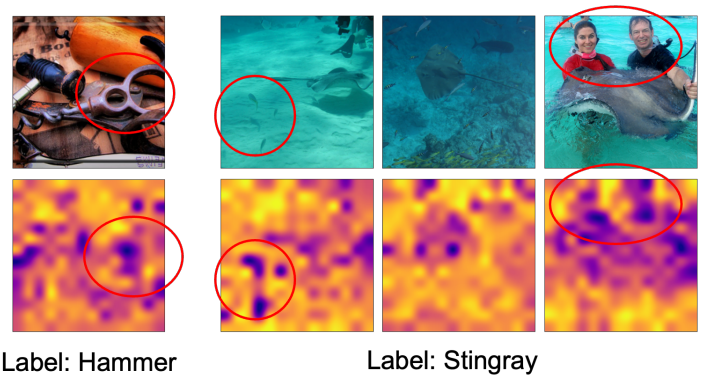

Figure 13: Examples of pseudo-depth estimation, applied to images with multiple types of objects. We use red circles to highlight the locations of non-class objects. In the hammer input, depth is also estimated on the other tool. In the first sample of stingray, three other fishes in the "foreground" corresponds to a lower depth value. In the last sample of stingray, two humans are predicted with a relative lower depth value, in between the stingray and the background.

validation set and perturbed set. In Table 8, we find that both baseline and our method increases validation scores after disabling Mixup & CutMix, and 3DTRL still outperforms the baseline by 0.2%. However, when tested on view-perturbed set, the baseline model shows a great performance drop (-6.8%) which is much larger than 3DTRL (-1.6%).

Table 8: ImageNet-1K and ImageNet-1K-Perturbed results when Mixup & Cutmix are disabled.

Model	ImageNet-1K	ImageNet-1K-Perturbed
DeiT-T	73.4	61.3
DeiT-T + Mixup & CutMix Disabled	74.2	54.5
DeiT-T+3DTRL	73.6	64.6
DeiT-T+3DTRL + Mixup & CutMix Disabled	74.4	63.0

D.3 Camera parameter estimation in video model

We implemented and evaluated two different strategies for the camera parameter estimation $g(\cdot)$ in 3DTRL for videos. These two strategies are: (a) Divided-Temporal (DT) and (b) Joint-Temporal (JT) estimation introduced below. In (a) DT strategy, we estimate one set of camera parameters $[\,\mathbf{R}|\mathbf{t}\,]_t$ per input frame (S_t) in a dissociated manner, and thus estimate a total of T camera matrices for the entire video. In (b) JT strategy, we estimate only one camera from all frames $S = \{S_1, \ldots, S_T\}$. The camera is shared across all spatial-temporal tokens associated with the latent 3D space. The underlying hypothesis is that the camera pose and location do not change during the video clip. JT could be helpful to properly constrain the model for scenarios where camera movement is not required, but it is not generalizable to scenarios where the subject of interest quite often moves within the field-of-view.

By default, we used DT strategy in all experiments presented in the main paper. In Table 9, we show a comparison of DT and JT strategies in different scenarios. Note that 3DTRL implemented with JT strategy under-performs the baseline TimeSformer on Smarthome (in most of the CV2 experiments) dataset. We find that the JT strategy adoption for video models is particularly effective when there is a large availability of training data, for example on NTU dataset. However, these results with JT strategy are inconsistent across different datasets and also less substantial w.r.t. the results with our default DT strategy. This shows the requirement of estimating camera matrix per frame rather than a global camera matrix for video representation tasks.

Table 9: Comparison of DT and JT strategies in 3DTRL for action recognition task.

Method	Strategy	**Smarthome** (CV2)		**Smarthome** (CS)		**NTU** (CV)
		Acc	*mPA*	*Acc*	*mPA*	*Acc*
TimeSformer [3]	-	59.4	27.5	75.7	56.1	86.4
+ **3DTRL**	DT	**62.9** (+3.5)	**34.0** (+6.5)	76.1 (+0.4)	57.0 (+0.9)	**87.9** (+1.5)
+ 3DTRL	JT	58.6 (-0.8)	30.9 (+3.4)	**76.2** (+0.5)	**57.2** (+1.1)	**87.9** (+1.5)
			Kinetics-400 pre-trained			
TimeSformer [3]	-	69.3	37.5	77.2	57.7	87.7
+ 3DTRL w/o K400	DT	69.5 (+0.2)	39.2 (+1.7)	77.5 (+0.3)	58.9 (+1.2)	**88.8** (+1.1)
+ 3DTRL w/ K400	DT	**71.9** (+2.6)	**41.7** (+4.2)	**77.8** (+0.6)	**61.0** (+2.3)	88.6 (+0.9)
+ 3DTRL w/o K400	JT	66.6 (-2.7)	35.0 (-2.5)	77.0 (-0.2)	58.6 (+0.9)	88.6 (+0.9)
+ 3DTRL w/ K400	JT	68.2 (-0.9)	37.1 (-0.4)	77.0 (-0.2)	59.9 (+2.2)	87.7 (+0.0)

E Limitations

We find that 3DTRL suffers when estimating small objects in the scene, or estimating objects in a complex scene, due to the coarse scale (in 16x16 image patches) from the backbone Transformer. One possible solution is to decrease the patch size or enlarge the input size in the backbone Transformer, but in practice it is computationally infeasible as Attention complexity grows quadratically. Similar problem occurs when 3DTRL is applied on Transformers having hierarchical architectures like Swin Transformer [36], where we find our improvement is minor compared to DeiT. In hierarchical architectures, image patches are merged after one stage so the resolution of the pseudo-depth map decreases quadratically. To solve this issue, we recommend to place 3DTRL at the location before any patch merging in such hierarchical Transformers.

F Implementation Details

3DTRL is easily inserted in Transformers. The components of 3DTRL are implemented by several MLPs and required geometric transformations in between. We keep the hidden dimension size in MLPs the same as the embedding dimensionality of Transformer backbone, Tiny=192, Small=384, Base=768 in specific. We provide PyTorch-style pseudo-code about inserting 3DTRL in Transformer (Algorithm 1) and about details of 3DTRL (Algorithm 2). We use image Transformer for example and omit operations on *CLS* token for simplicity. Full implementation including video model is provided in supplementary files.

Algorithm 1: PyTorch-style pseudo-code for using 3DTRL in Transformer

```
# Use 3DTRL with Transformer backbone
Class Transformer_with_3DTRL:
   def __init__(self, config):
      # Initialize a Transformer backbone and 3DTRL
      self.backbone = Transformer(config)
      self.3dtrl = 3DTRL(config)
      # Before which Transformer layer we insert 3DTRL
      self.3dtrl_location = config.3dtrl_location

   def forward(self, tokens):
      for i, block in enumerate(self.backbone.blocks):
         # Tokens go through 3DTRL at desired insert location
         if i == self.3dtrl_location:
            tokens = self.3dtrl(tokens)
         # Tokens go through backbone layers
         tokens = block(tokens)
      return tokens
```

Algorithm 2: PyTorch-style pseudo-code for 3DTRL

```
Class 3DTRL:
   # Make a 3DTRL
   def __init__(self, config):
      # 2D coordinates on image plane
      self.u, self.v = make_2d_coordinates()
      # Depth estimator
      self.depth_estimator = nn.Sequential(
         nn.Linear(config.embed_dim, config.embed_dim),
         nn.ReLU(),
         nn.Linear(config.embed_dim, 1))

      # Camera parameter estimator, including a stem and two heads
      self.camera_estimator_stem = nn.Sequential(
         nn.Linear(config.embed_dim, config.embed_dim),
         nn.ReLU(),
         nn.Linear(config.embed_dim, config.embed_dim),
         nn.ReLU(),
         nn.Linear(config.embed_dim, 32),
         nn.ReLU(),
         nn.Linear(32, 32))
      # Heads for rotation and translation matrices.
      self.rotation_head = nn.Linear(32, 3)
      self.translation_head = nn.Linear(32, 3)

      # 3D positional embedding layer
      self.3d_pos_embedding = nn.Sequential(
         nn.Linear(3, config.embed_dim),
         nn.ReLU(),
         nn.Linear(config.embed_dim, config.embed_dim))

   def forward(self, tokens):
      # Depth estimation
      depth = self.depth_estimator(tokens)
      camera_centered_coords = uvd_to_xyz(self.u, self.v, depth)

      # Camera estimation
      interm_rep = self.camera_estimator_stem(tokens)
      rot, trans = self.rotation_head(interm_rep),
       self.translation_head(interm_rep)
      rot = make_rotation_matrix(rot)

      # Transformation from camera-centered to world space
      world_coords = transform(camera_centered_coords, rot, trans)

      # Convert world coordinates to 3D positional embeddings
      3d_pos_embed = self.3d_pos_embedding(world_coords)

      # Generate output tokens
      return tokens + 3d_pos_embed
```

G Settings for Image Classification

Datasets For the task of image classification, we provide a thorough evaluation on three popular image datasets: CIFAR-10 [34], CIFAR-100 [34], and ImageNet [15]. CIFAR-10/100 consists of 50k training and 10k test images, and ImageNet has 1.3M training and 50k validation images.

Training Configurations We follow the configurations introduced in DeiT [59]. We provide a copy of configurations here in Table 10 (CIFAR) and Table 11 (ImageNet-1K) for reference. We use 4 NVIDIA Tesla V100s to train models with Tiny, Small and Base backbones on ImageNet-1K for $\sim$22 hours, $\sim$3 days and $\sim$5 days respectively.

Table 10: CIFAR Training Settings

Input Size	32×32
Patch Size	2×2
Batch Size	128
Optimizer	AdamW
Optimizer Epsilon	1.0e-06
Momentum	β_1, β_2 = 0.9, 0.999
layer-wise lr decay	0.75
Weight Decay	0.05
Gradient Clip	None
Learning Rate Schedule	Cosine
Learning Rate	1e-3
Warmup LR	1.0e-6
Min LR	1e-6
Epochs	50
Warmup Epochs	5
Decay Rate	0.988
drop path	0.1
Exponential Moving Average (EMA)	True
EMA Decay	0.9999
Random Resize & Crop Scale and Ratio	(0.08, 1.0), (0.67, 1.5)
Random Flip	Horizontal 0.5; Vertical 0.0
Color Jittering	None
Auto-agumentation	rand-m15-n2-mstd1.0-inc1
Mixup	True
Cutmix	True
Mixup, Cutmix Probability	0.8, 1.0
Mixup Mode	Batch
Label Smoothing	0.1

H Settings for Video Alignment

Datasets We provide the statistics of 5 datasets used for video alignment in Table 12. In general, datasets with fewer training videos, more/diverse viewpoints, and longer videos are harder for alignment. We will also provide the copy of used/converted dataset upon publish.

Training Configurations The training setting for video alignment is listed in Table 13. The setting is the same for all datasets and all methods for fair comparison. GPU hours required for training vary across datasets, depending on the size of datasets and early stopping (convergence). Approximately we use 24 hours in total to fully train on all 5 datasets using an NVIDIA RTX A5000.

Table 11: ImageNet-1K Training Settings [59]

Input Size	224×224
Crop Ratio	0.9
Batch Size	512
Optimizer	AdamW
Optimizer Epsilon	1.0e-06
Momentum	0.9
Weight Decay	0.3
Gradient Clip	1.0
Learning Rate Schedule	Cosine
Learning Rate	1.5e-3
Warmup LR	1.0e-6
Min LR	1.0e-5
Epochs	300
Decay Epochs	1.0
Warmup Epochs	15
Cooldown Epochs	10
Patience Epochs	10
Decay Rate	0.988
Exponential Moving Average (EMA)	True
EMA Decay	0.99992
Random Resize & Crop Scale and Ratio	(0.08, 1.0), (0.67, 1.5)
Random Flip	Horizontal 0.5; Vertical 0.0
Color Jittering	0.4
Auto-agumentation	rand-m15-n2-mstd1.0-inc1
Mixup	True
Cutmix	True
Mixup, Cutmix Probability	0.5, 0.5
Mixup Mode	Batch
Label Smoothing	0.1

Table 12: Statistics of multi-view datasets used for video alignment.

Dataset	**# Training/Validation/Test Videos**	**# Viewpoints**	**Average Frames/Video**
Pouring	45 / 10 / 14	2	266
MC	4 / 2 / 2	9	66
Pick	10 / 5 / 5	10	60
Can	200 / 50 / 50	5	38
Lift	200 / 50 / 50	5	20

Table 13: Training Settings for Video Alignment

Positive Window of TCN Loss	3 frames in MC, Pick, Pouring; 2 frames in Can and Lift
Learning Rate	1e-6
Batch Size	1
Optimizer	Adam
Gradient Clip	10.0
Early Stopping	10 epochs
Random Seed	42
Augmentations	No

I Settings for Video Representation Learning

Datasets Our dataset choices are based on multi-camera setups in order to provide cross-view evaluation. Therefore, we evaluate the effectiveness of 3DTRLon two multi-view datasets Toyota Smarthome [12] and NTU-RGB+D [50]. We also use Kinetics-400 [32] for pre-training the video backbone before plugging-in 3DTRL.

Toyota-Smarthome (Smarthome) is a recent ADL dataset recorded in an apartment where 18 older subjects carry out tasks of daily living during a day. The dataset contains 16.1k video clips, 7 different camera views and 31 complex activities performed in a natural way without strong prior instructions. For evaluation on this dataset, we follow cross-subject (CS) and cross-view (CV_2) protocols proposed in [12]. We ignore protocol CV_1 due to limited training samples.

NTU RGB+D (NTU) is acquired with a Kinect v2 camera and consists of 56880 video samples with 60 activity classes. The activities were performed by 40 subjects and recorded from 80 viewpoints. For each frame, the dataset provides RGB, depth and a 25-joint skeleton of each subject in the frame. For evaluation, we follow the two protocols proposed in [50]: cross-subject (CS) and cross-view (CV).

Kinetics-400 (K400) is a large-scale dataset with 240k training, 20k validation and 35k testing videos in 400 human action categories. However, this dataset do not posses the viewpoint challenges, we are addressing in this paper. So, we use this dataset only for pre-training purpose as used by earlier studies.

Training Configurations We use clips of size $8 \times 224 \times 224 \times 3$, with frames sampled at a rate of 1/32. We use a ViT-B encoder with patch size 16×16. The training setting for action recognition on both datasets follow the configurations provided in [3]. We train all the video models on 4 RTX 8000 GPUs with a batch size of 4 per GPU for 15 epochs. A gradient accumulation is performed to have an effective batch size of 64. Similar to [3], we train our video models with SGD optimiser with 0.9 momentum and $1e-4$ weight decay. During inference, we sample one and 10 temporal clips from the entire video on NTU and Smarthome datasets respectively. We use 3 spatial crops (top-left, center, bottom-right) from each temporal clip and obtain the final prediction by averaging the scores for all the crops.

J More Pseudo-depth Estimation Visualization

Figure 14 gives examples of more pseudo-depth maps.

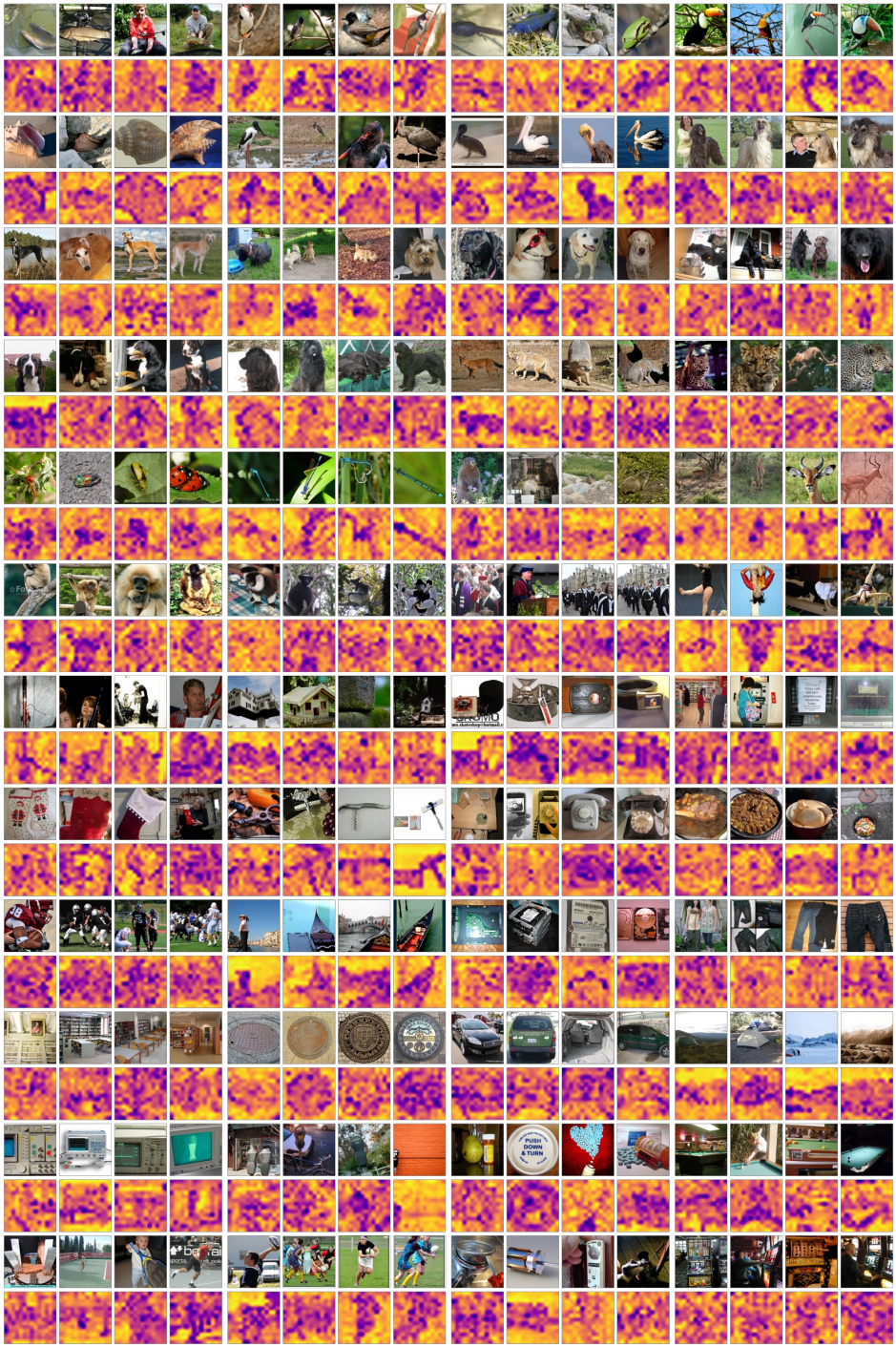

Figure 14: More examples of pseudo-depth maps.

K Pseudo-depth Map Visualization over Training Epochs

Figure 15 gives examples of pseudo-depth estimation over Training Epochs. We note that the results are from training 3DTRL with IN-1K. We find that the estimation varies significantly from epoch 10 to epoch 40 (higher foreground-background correctness, less missing parts of objects), but changes only a bit from epoch 40 to epoch 200 and finally to epoch 300 (mostly scales). This observation is also coherent with our quantitative evaluation in Section 4.3.1. Thus, the pseudo-depth estimation learns promptly, however the model convergence takes longer time since we are optimizing for a downstream task (eg. classification).

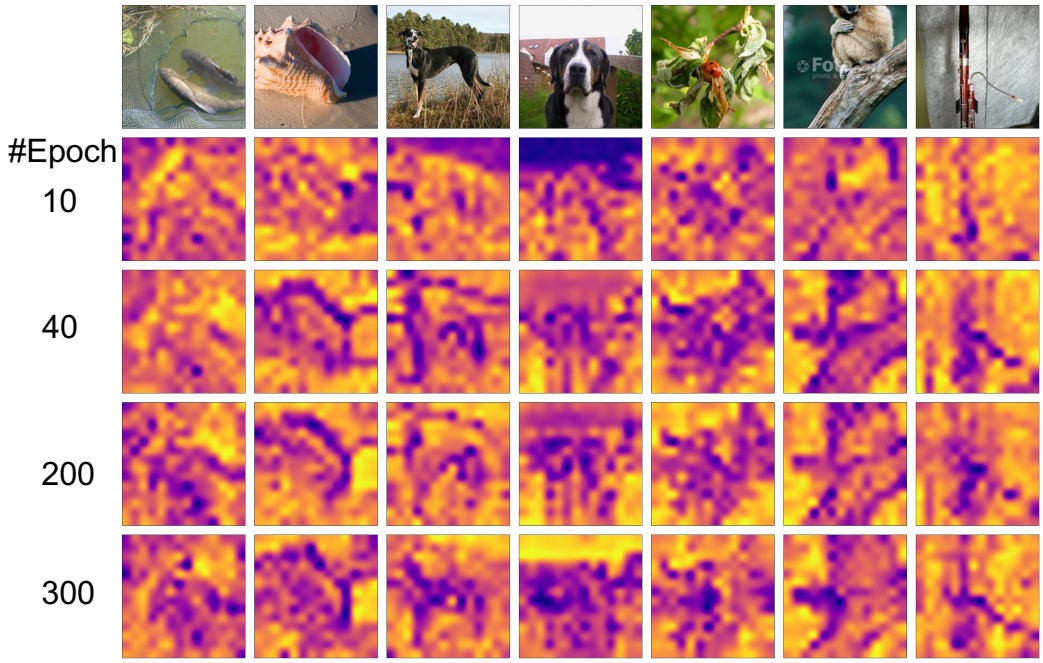

Figure 15: Pseudo-depth maps over training epochs.