DTM: Difference-Based Temporal Module for Monocular Category-Level 6 DoF Object Pose Tracking

Zishen Chen, Jochen Lang
School of EECS, University of Ottawa
Ottawa, On, Canada
Email: {zchen314|jlang}@uottawa.ca

***Abstract*—We propose DTM, a novel difference-based temporal module to leverage historical information in category-level 6DoF pose tracking tasks. It can be easily integrated with various category-level 6DoF pose tracking models which use RGBD data as input. This module extracts temporal features through a KNN-based difference calculation strategy from both, pixels and 3D points. We evaluate this module on two pose estimation datasets, NOCS-REAL275 and MoVi-E by integrating our module with two state-of-the-art 6D pose tracking models. The result shows that DTM can significantly increase the accuracy and robustness of category-level 6DoF trackers.**

***Keywords*-6DoF Pose Tracking; Difference; Temporal Model;**

I. Introduction

Object pose prediction is essential in Augmented Reality (AR) [1], autonomous driving [2, 3], SLAM (Simultaneous Localization and Mapping) [4] and robotics application [5, 6]. We are motivated by AR applications where robust, accurate and instant pose predictions of a target object in video are needed to enhance human-computer interaction. Robustness is essential but challenging when the quality of the input frame sequence is low due to hand-held capture with mobile devices. The pose and appearance of target objects will experience rapid change. Making stable and real-time pose estimations is key for 6DoF pose tracking in AR applications which we address by using frame history.

In pose tracking, a model will predict the relative pose transformation between the current frame and the next frame. Traditionally, the pose of objects relative to the camera can be recovered by using hand-crafted features [7–9] and solving an optimization problem. As pose tracking is used in more complex environments, visual variations increase including illumination changes, motion blur, scene clusters and occlusions, Wang et al. [10] propose the incorporation of deep learning techniques into the 6DoF category-level pose tracking. They follow the idea of [11], proposing a keypoint-based method to track objects. They decomposed the 3D pose tracking problem into 3D keypoints detection and 6D pose estimation problem. By generating ordered keypoints, the model recovers the transformation through least squares optimization [12]. Due to the reliance on keypoints generation, the model requires clear views of the target object, while the capability to handle interference is limited and historical features are ignored. As a result, the robustness of the tracking model during interference is low.

The majority of current pose tracking methods only consider two frames, the current frame and the previous frame without paying attention to the temporal history. But the history of frames contains rich temporal and spatial features that can benefit the pose prediction. The object may be temporarily occluded, or blurred because of external interference which will cause feature loss. However, the motion pattern extracted from the history may reduce errors by bridging the temporal occlusions. History has been deeply explored in 2D computer vision [13–16]. In 2D computer vision, temporal modelling is commonly used to process sequential data inaction recognition [17]. Attention [18–20] and difference [21] techniques have been deeply exploited in 2D. However, there is still a need for an effective method in 3D to incorporate temporal information.

In this paper, we propose DTM, a novel method to aggregate history into state-of-the-art 6D pose trackers. DTM is based on difference calculations between current and historical frames. Considering the input of the 6D tracker can be either 3D points [22] or RGBD [10], we assume that one point's features are similar to the features of K closest spatial neighbours. We propose a difference calculations between various point sets to accomplish cross-frame feature aggregation based on KNN. Our module will learn to produce weights for each frame in the history sequence. These weights represent the per-channel importance of each point's feature. Then we globally and locally aggregate the historical features into the current frame. Our module is a plug-in module that can be easily used on any category-level 6D tracker that takes RGBD or 3D point clouds as input.

In this work, we propose DTM, a temporal module based on cross-frame difference calculation considering both pixels and points. The experiments on NOCS-REAL275 [23] and the challenging benchmark MoVi-E [24] show that baseline state-of-the-art 6DoF pose tracking networks [10, 22] gain significant improvements from DTM, particularly in the scenes with occlusions.

II. Related Works

In category-level pose estimation tasks, the 3D CAD model is not available during either training or testing. As a result, the model doesn't have specific prior knowledge of the target object. Sahin et al. [25] define the main challenges of category-level 6D pose estimation problems to deal with intra-class variations and distribution shifts. The first refinement-based category-level method using deep learning is proposed by Wang et al. [23]. They propose a novel way to represent the instance in object coordinates, which is called NOCS (Normalized Object Coordinate Space). NOCS is a unit cube that aligns with the object's center in 3D space. By normalizing the object in NOCS, the category-specific features can be easily extracted regardless of shape.

Instead of estimating the pose using a single frame, pose tracking methods infer the relative pose between two consecutive frames. Wang et al. [10] address the tracking problem using keypoints. Due to the difficulty of annotating keypoints in videos, 6-Pack predicts the keypoints with respect to the input point cloud in an unsupervised manner by minimizing the multi-view consistency error. 6-Pack takes RGBD data as input and fuses the features of pixels and points by conducting Densefusion [26]. Because 6-Pack is a salient method that uses both point and pixel features, we select it as one of the baseline models. Other methods [27, 28] define the vertices of the 3D bounding box as keypoints. Given the current and previous 2D image frames along with the center and keypoints heatmap of the previous frame, a neural network is used to generate a heatmap to find the center and keypoints of the object in the current frame. In order to investigate our model's performance on points-only methods, we also add DTM to CAPTRA [22], a regression-based tracker that takes both rigid and articulated objects into account and treats the tracking problem as a 9DoF regression problem. CAPTRA adopts the idea of NOCS [23] and proposes a pose-canonicalized space. The pose-canonicalized space uses the object's pose in the previous frame to normalize the current frame's object and regress the pose of the object in this canonicalized space directly.

6D pose detection estimates the pose between camera and object in each frame individually. Pose tracking is less error prone in processing real-time video sequences because it is more effective to predict the relative pose between two consecutive frames. However, tracking based only on the latest two frames and ignoring the history can make the model unstable with noisy inputs. Temporal modelling is commonly adopted in tasks that process sequential data. In 2D computer vision tasks, Wang et al. [17] propose TDN (Temporal Difference Networks) for action recognition in video, which incorporates short-term and long-term motion captures. Similar to our method, TDN is based on a temporal difference operator. Cui et al. [29] propose TF-Blender (Temporal Feature Blender) for video object detection. The proposed model considers the temporal feature integration problem as a combination of feature relation and adjustment. For category-level 6D pose tracking tasks, BundleTrack proposed by Wen et al. [30] stores unordered reference frames into a pose graph and conducts a maximum overlap optimization to refine the initial pose. By extending the idea of the pose graph, Wen et al. propose BundleSDF [31], a 6DoF tracking model combined with 3D reconstruction through learning a Neural Object Field. Wen et al. also present FoundationPose [32] to bridge category-level and instance-level methods where the stored frames in the memory are used to learn the Neural Object Field as in BundleSDF [31]. Based on the learned Neural Object Field or the ground truth CAD model, FoundationPose will generate several refined pose hypotheses and use a pose ranking network to find the hypothesis that aligns best with the input RGBD image. BundleTrack [30], BundleSDF [31] and FoundationPose [32] store the history as separate frames for 3D reconstruction or pose refinement, which is not a continuous sequence. The proposed method in this paper works on the temporal history which is orthogonal to their approach. For instance-level models, TP-AE [33] proposed by Zheng et al. uses a temporal sequence of frames to learn the trajectory of an object in order to estimate a prior pose based on the continuous history. The setup of the temporal sequence in TP-AE [33] is similar to DTM. Unlike our DTM, TP-AE is an instance-level method that requires a CAD object model.

III. Problem Definition

In a category-level 6DoF tracking model, the initial pose of the object is given with respect to the camera coordinate system in the first frame, $p_0 \in SE(3), p_0 = [R_0 \mid t_0]$ together with a video sequence $(I^0, I^1, I^2, ..., I^t)$. The problem can be depicted as tracking the continuous pose p_i of the object in each frame f_i given the pose in the previous frame p_{i-1}, which is the relative transformation between I^i and I^{i-1}, Δp_i. Thus, p_t can be computed through the accumulated multiplication of Δp, that is $p_t = \Delta p_t \cdot \Delta p_{t-1} \cdot \Delta p_{t-2} \cdot \Delta p_{t-3} \cdot \Delta p_{t-4} \cdot \cdot \cdot p_0$. Given the RGBD image I_i, the tracking task can be expressed as $p_t = F(I^t, I^{t-1}, I^{t-2}, I^{t-3}, \cdot \cdot \cdot, I^0, C^{t-1}, C^{t-2}, C^{t-3}, \cdot \cdot \cdot, C^0, p_0 \mid \theta)$, where θ is the tracking model's parameters and C^i is the corresponding point cloud of image I^i.

In order to estimate Δp between two frames in ordinal 6D tracking models, the model needs to take the current frame and previous frame as input, which is $\Delta p_t = H(I^t, C^t, I^{t-1}, C^{t-1} \mid \theta)$. For real-time tracking, we propose to use a sliding window (history sequence) to store and update past frames sequentially. The length of the sliding window $\mathcal{M}$ is L, where $\mathcal{M}^t = (I^{t-1}, C^{t-1}, I^{t-2}, C^{t-2}, I^{t-3}, C^{t-3} ... I^{t-L}, C^{t-L})$. Therefore, the tracking model with the temporal information can

be expressed as $\Delta p_t = H(I^t, C^t, \mathcal{M}^t \mid \theta)$. DTM can be integrated with off-the-shelf learning-based 6DoF pose tracking models in our design.

IV. Approach

DTM explores the relations between the current frame $[I^t|C^t]$ and the frames in the history sequence $\mathcal{M}^t$ as shown in Fig. 1 c). It outputs the refined features $f^{t'}$ of the current frame $[I^t|C^t]$.

In 2D video detection the RGB difference [34–36] and features difference [37–39] have been deeply explored. The difference in 2D can be directly calculated because pixels and features are spatially related. In 3D tasks, the challenge is to combine RGB values and points. We propose a bidirectional feature-wise difference measurement method based on Euclidean distance (Sec. IV-A). The purpose of the differences is to learn a feature-wise weight for the previous frames indicating the importance of each dimension of the feature. Based on the weights, the current frame's features are refined with the last frame's features. By doing so, the model can extract significant features from the history and suppresses those features that can affect the model's performance negatively.

The local difference module calculates the difference between every two consecutive frames in the history sequence $\mathcal{M}^t$ and learns feature-wise weights. The iterative update between two consecutive frames in the history sequence $\mathcal{M}^t$ will start from the oldest frame's feature f^{t-L}. Suppose the update function of local difference is $S_{local}()$, which satisfies

$$\overline{f^{t-L+i}} = S_{local}(f^{t-L+i}, \overline{f^{t-L+i-1}}) \tag{1}$$

where t is the index of current frame and i means the i^{th} frame in the history sequence. L is the length of the sequence. $\overline{f^i}$ are the features after refinement. The function S_{local} takes the refined features of frame $f^{t-L+i-1}$ and the features of frame f^{t-L+i} as input to generate the refined feature of frames $\overline{f^{t-L+i}}$. This formula will be iteratively applied to all the frames in the history sequence until it updates the current frame's features producing the refined features of the current frame $\overline{f^t}$. In the local difference module, the weight will accumulate from the first frame in the history sequence. However, it will diminish the effect of early frames in the history sequence. In order to solve this problem, we further design a global difference module that can benefit from the features of each frame equally.

The global difference module will calculate the difference between each frame in M^t and f^t and accumulate the weighted features together to update the current frame's feature. Suppose the global difference function is S_{global} that treats the relation between all the frames equally. Then we have

$$\widetilde{f^t} = S_{global}(\overline{f^{t-1}}, \overline{f^{t-2}} \cdots f^{t-L}) \tag{2}$$

Local difference and global difference module outputs will be combined to generate the final refined feature of the current frame (see Sec. IV-B and Sec. IV-C for more details).

A. KNN-based Difference Calculation

We calculate the difference between the unordered point-sets of the frames based on KNN. Inspired by Ffb6d [40], we propose a KNN-based bidirectional difference calculation and fusion method shown in c) of Fig. 1. Our extension of the method of Ffb6d [40] fuses the image I and point cloud C in a bidirectional manner.

The correspondence between each pixel in the image and the 3D point is known because the points are calculated by back-projecting the pixels using the intrinsic matrix of the camera [41]. We depict this correspondence as a mapping $\mathcal{X}()$, that is $C_i = \mathcal{X}(I_i)$, where C_i is the i^{th} point in the point cloud C and I_i is the i^{th} pixel of the corresponding RGB image.

Given the point sets C^a, C^b and RGB pixels I^a, I^b of two scenes a and b, the Euclidean distances of points between scene a and b are calculated as $\|\mathcal{X}(I_i^a) - \mathcal{X}(I_j^b)\|$ and $\|C_i^a - C_j^b\|$ assuming a static camera. The KNN-based pixels difference of K nearest point neighbours G_i^a of I_i^a selected in the scene b can be described as

$$G_i^a = f_i^{a,rgb} - MaxPool(\mathcal{W}_i \cdot \mathcal{K}(\mathcal{X}(I_i^a), f^{b,rgb})) \tag{3}$$

where $f_i^{a,rgb}$ are the features of i^{th} pixel in scene a, $f^{b,rgb}$ are the features of the pixels in scene b. The KNN (K nearest neighbors) function $\mathcal{K}()$ takes a point C_i^a and the feature map as inputs, and returns the K selected features for point C_i^a in scene a to the pixel in scene b according to the back-projection $\mathcal{X}()$. Additionally, we also apply a distance dependent weight ($\mathcal{W}_{ij}$) on the selected features, which can be expressed as

$$\mathcal{W}_{ij} = SoftMax(-\|\mathcal{X}(I_i^a) - C_j^b\|) \tag{4}$$

For simplicity, we use $\mathcal{W}_i$ to denote a set $\{\mathcal{W}_{i0}, \mathcal{W}_{i1}, ..., \mathcal{W}_{iN}\}$, where N is the total number of points. Assuming the points with closer distance share more similar features, the weight $\mathcal{W}_i$ ensures the model assigns high priority to close points. The reason that we use maxpooling is to emphasize the largest difference in each dimension of a feature. Considering the point C_i^a in the scene a has feature $f_i^{a,points}$ and the features of points in scene b are $f^{b,points}$, then the KNN-based points difference P_i^b of the i^{th} point in scene a can be described as

$$P_i^a = f_i^{a,points} - MaxPool(\mathcal{W}_i \cdot \mathcal{K}(C_i^a, f^{b,points})). \tag{5}$$

After deriving the pixels difference G_i^a and points difference P_i^a of the i^{th} point in scene a, we now consider fusing the two differences.

The bidirectional differences fusion of the i^{th} point in scene a is as follows

$$U_i^a = G_i^a \oplus MaxPool(\mathcal{W}_i \cdot \mathcal{K}(C_i^a, P^a)), \ and \tag{6}$$

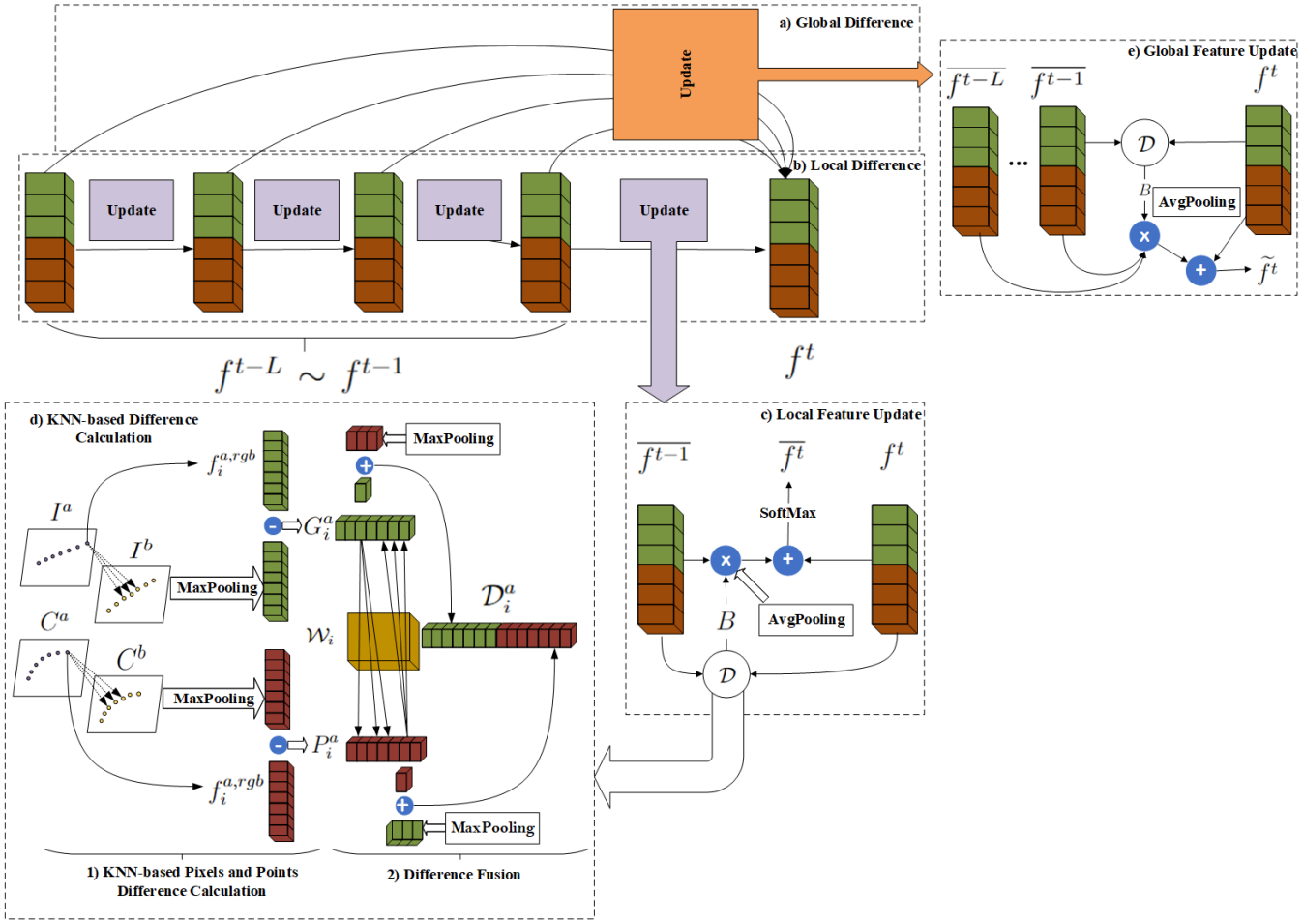

Figure 1. DTM consists of two main components, **a) Global Difference** and **b) Local Difference**. Given the features of the current frame f^t and the history sequence $f^{t-L} \sim f^{t-1}$, the feature update process (**c) Local Feature Update** and **e) Global Feature Update**) relies on KNN-based difference calculation (**d) KNN-based Difference Calculation**) to compute the difference between two frames considering their RGB images and point clouds. A relation module B will be applied to this difference to learn feature-wise weights to map the feature of the previous frame.

$$O_i^a = P_i^a \oplus MaxPool(\mathcal{W}_i \cdot \mathcal{K}(C_i^a, G^a)) \tag{7}$$

where $\oplus$ is the concatenation along the feature dimension. U_i^a is the feature-wise concatenation between G_i^a and the K nearest neighbors points difference from P^a calculated in the previous section. The distance-based weight and maxpooling are also applied to the selected features. The concatenation along the feature dimension between point difference P_i^a and the K closest pixels differences G_i^a is also calculated. Then, U_i^a and O_i^a are concatenated to generate the difference representation of i^{th} point in scene a, that is $\mathcal{D}_i^{a,b} = U_i^a \oplus O_i^a$, where $\mathcal{D}_i^{a,b}$ is the difference representation of point i in scene a considering both pixels and points differences between scenes a and b.

B. Local Difference

Given the pixel features $f^{a,rgb}$ and points features $f^{a,points}$ of frame a with the features for the history frames in the history sequence $f^{j,rgb}$, $f^{j,points}$, $j \in \mathcal{M}^t$, the procedure of local difference calculation is shown in Fig. 1 b) and c), where $\mathcal{D}$ is the difference calculation in Sec. IV-A. The update process starts from the earliest frame I^{t-L} in $\mathcal{M}$. For two consecutive frames $([I^t|C^t], [I^{t-1}|C^{t-1}])$, we use a relation module B to learn the feature-wise weight from $\mathcal{D}^{t-1,t}$ in Sec. IV-A, which is the difference representation of scene $t-1$ fusing both pixels and points differences between $t-1$ and t. The relation module is constructed by an MLP with three layers. The update between the features of two consecutive frames can be expressed as

$$\overline{f_i^t} = SoftMax(f_i^t + AvgPool(\mathcal{K}(C_i^t, B(\mathcal{D}_i^{t-1,t}) \otimes \overline{f_i^{t-1}}))) \tag{8}$$

where $\otimes$ is the feature-wise multiplication. $\overline{f_i^t}$ is the refined feature of f_i^t obtained through the local difference module. The update process will be done across all the frames in $\mathcal{M}$ and finally accumulated to the current frame. For the first frame in $\mathcal{M}$, $\overline{f^{t-L}} = f^{t-L}$. In Eq. 8, we also use the SoftMax function to normalize the refined features. According to

Eq. 8, the refined features of the local difference module will accumulate the weighted features from previous frames in the history sequence. Not applying normalization will result in large values.

C. Global Difference

Similar to the local difference, the global difference module also needs to conduct an update process between two frames. But in the global difference, the update is directly between the features of each frame in $\mathcal{M}$ and the features of the current frame. The process is shown in a) and c) of Fig. 1. Given the feature of the current frame f^t and the features of frames in the history sequence, f^{t-j}, $j \leq L$. The updated current frame can be expressed as

$$\widetilde{f^t} = \sum_{j=0}^{L} AvgPool(\mathcal{K}(C^t, B(\mathcal{D}^{t-j,t}) \otimes \overline{f^{t-j}})) + f^t \quad (9)$$

where $\mathcal{D}^{t-j,t}$ is the learned difference representation between each frame in the history sequence and the current frame. $\overline{f^{t-i}}$ is the refined feature from the local difference module in Sec. IV-B .

As shown in Fig. 1, the features of the local difference $\overline{f^t}$ and global difference $\widetilde{f^t}$ will be concatenated and fed to an MLP for the output of the final refined feature. The final aggregation is

$$f^{t'} = Mlp(\widetilde{f^t} \oplus \overline{f^t} \oplus f^t) \quad (10)$$

The refined feature $f^{t'}$ of f^t will be used for downstream pose prediction.

D. History Sequence Updates

Another critical task in DTM is the management of the history sequence $\mathcal{M}$. We follow a first-in-first-out strategy. The current refined frame will be added to the history sequence once the pose prediction of the current frame is completed and the oldest frame will be deleted from the sequence. The transformation between camera space to object space of the added frame will be recorded. We align the pose of frames in the history sequence with the current frame by transforming all of them using the latest frame's pose in the history sequence to the object space, where the object space is a fixed reference frame. Under this scheme, the history sequence can work with the temporal model in a real-time manner.

E. Integration with other Learning-based 6DoF Pose Tracking Models

DTM can be integrated in various off-the-shelf category-level 6DoF pose tracking networks to refine the current features by using temporal information. Typically, learning-based pose tracking or estimation networks contain an encoder to extract RGB and points features. Then, the downstream components will use these features to estimate the pose. DTM can be added as a mid-layer after the encoder to refine the extracted features from the encoder with the history features. We use 6-Pack [10] and CAPTRA [22] as baseline models.

V. Experiments

A. Datasets

We evaluate DTM in two datasets, NOCS-REAL275 [23] and MoVi-E [24]. NOCS-REAL275 [23] is commonly used for 6DoF pose estimation. It has two parts, synthetic and real data. The synthetic data has no continuity across more than 2 frames and hence it is not suitable for the training of our temporal model. Thus, our training and evaluation are only using the real data of NOCS-REAL275. This is also why the presented evaluation results of the baseline methods in Table I are worse than those in the respective papers.

The MoVi dataset [24] is a synthetic dataset series with increasing complexity from A to F. Although the length of the videos is short (24 frames per video), in each video, the pose of the object is continuous. Hence, DTM can fully use the dataset. We select two asymmetric categories (Shoe and Bag) and one symmetric category (Bottle, Can and Cup). The MoVi dataset classifies Bottle, Can and Cup as a single category.

B. Metrics

We follow NOCS [23] and use 5 metrics to evaluate the models.

- $5°5cm$: The percentage of predicted relative pose with rotation error less or equal to $5°$ and translation error less or equal to 5cm.
- $10°10cm$: The percentage of predicted relative pose with rotation error less or equal to $10°$ and translation error less or equal to 10cm.
- mIoU (Mean Intersection over Union): The average percentage of the 3D intersection between the predicted 3D bounding box and the 3D bounding box.
- Rotation Error (R_{error}) : The average error of rotation in degree.
- Translation Error (T_{error}) : The average error of translation in centimetres.

C. Results

Evaluation Results on the real data of NOCS-REAL275 dataset: Table I list the results on the testing set of the NOCS-REAL275 dataset [23]. For $10°10cm$, our model with a 6-Pack backbone outperforms 6-Pack by itself in all categories except the laptop. Especially, the percentage of rotation error and translation below $10°$ and 10 centimetres is twice the percentage of 6-Pack in the bottle category. Although our translation error is slightly higher than 6-Pack (0.85), the rotation error (21.43) is half of 6-Pack (44.36). Based on these metrics, DTM refines the features to generate a more robust trajectory with less

Table I
COMPARISON ON THE REAL DATA PART OF NOCS-REAL275 DATASET.

		$5°5cm\uparrow$	$10°10cm\uparrow$	mIoU $\uparrow$	$R_{error}\downarrow$	$T_{error}\downarrow$	$S_{error}\downarrow$
6-Pack [10]	Bottle	4.5	7.81	**32.30**	54.76	**15.72**	×
	Bowl	**11.79**	14.60	36.99	53.30	**4.43**	×
	Camera	2.19	5.80	**37.18**	53.30	7.40	×
	Can	9.55	17.0	44.11	28.26	**8.82**	×
	Laptop	27.48	**68.67**	**68.05**	**8.59**	3.29	×
	Mug	2.80	7.8	54.60	67.95	6.84	×
	Overall	9.71	20.28	45.53	44.36	**7.75**	×
6-Pack w Ours	Bottle	**6.60**	**14.2**	31.79	**13.24**	18.39	×
	Bowl	6.9	**16.45**	**50.14**	**24.46**	7.26	×
	Camera	**4.8**	**9.45**	35.25	**30.75**	**6.44**	×
	Can	**10.0**	**28.74**	**48.94**	**12.44**	14.89	×
	Laptop	**36.80**	60.45	65.88	9.90	**2.25**	×
	Mug	**4.27**	**10.35**	**60.68**	**37.80**	**2.78**	×
	Overall	**11.56**	**23.27**	**48.78**	**21.43**	8.6	×
CAPTRA [22]	Bottle	36.13	**47.90**	**28.28**	22.07	**4.9**	**1.45**
	Bowl	**29.8**	**38.64**	**26.87**	31.35	5.3	1.55
	Camera	0.2	1.08	**3.19**	**72.38**	5.4	2.09
	Can	**32.47**	**33.35**	**19.32**	**38.54**	**33.00**	**90.39**
	Laptop	**67.64**	87.33	59.95	12.7	0.5	0.05
	Mug	1.37	20.01	34.35	77.86	1.7	1.10
	Overall	27.82	38.05	28.66	42.48	**8.46**	**16.10**
CAPTRA w Ours	Bottle	**39.26**	39.23	22.82	**16.81**	7.1	2.05
	Bowl	27.54	33.77	24.67	**24.79**	**2.9**	**1.17**
	Camera	**0.41**	**2.53**	9.52	79.44	**3.9**	**1.77**
	Can	30.91	31.98	18.70	53.98	77.28	211.47
	Laptop	60.46	**94.49**	**65.39**	**4.58**	**0.2**	**0.01**
	Mug	**31.61**	**62.15**	**34.57**	**14.18**	**1.5**	**0.10**
	Overall	**31.69**	**44.02**	**29.27**	**32.29**	15.48	36.09

rotation and translation error. DTM's average mIoU over all categories is higher than 6-Pack by itself, because 6-Pack uses the ground truth 3D bounding box as the size of objects instead of regressing it.

We also use CAPTRA as the backbone to test our model's performance. The $5°5cm$ and $10°10cm$ are better than 6-Pack due to the smaller rotation error and translation error. The overall mIoU is worse than 6-Pack because 6-Pack uses the ground truth 3D bounding box as the scale of the object in each frame while CAPTRA predicts the scale in each frame. The mIoU of CAPTRA will be affected by the scale error. The overall results of our model are better than CAPTRA by itself except the translation error and scale error. The reason why the four metrics (R_{error}, T_error, $5°5cm$ and $10°10cm$) are high in one category is that this category has many accurate estimations as well as many erroneous estimations, which means the distribution of R_{error} and T_error has a high variance. These two metrics are mainly affected by the performance in the category of Can. The Can category has high translation and scale errors in both our model and the baseline CAPTRA. However, the translation error and scale error are much higher than the baseline, where the translation error is 77.28 and the scale error is 211.47. For the Mug category, our model's $5°5cm$ (31.61) is much better than CAPTRA (1.37). 1.37% for $5°5cm$ means CAPTRA cannot maintain a low rotation and translation error, while our model can handle it well. Also, in the Mug category, the $10°10cm$ (62.15) is three times larger than that of the baseline CAPTRA (20.01).

In summary, DTM mostly improves the performance of baseline models in the real data of NOCS-275. However, the real data in NOCS-275 has a limited scope of the scenes and the number of sample is low. Hence, we utilize MoVi-E to work with more challenging scenarios with multiple moving objects, more occlusions due to clutter and more varied and dynamic motion patterns.

Table II
COMPARISON WITH THE BASELINE METHODS ON MOVI-E.

		$5°5cm\uparrow$	$10°10cm\uparrow$	mIoU $\uparrow$	$R_{error}\downarrow$	$T_{error}\downarrow$	$S_{error}\downarrow$
6-Pack [10]	Shoe	12.02	25.37	55.74	26	9.18	×
	Bottle, Can and Cup	21.94	37.69	55.85	19.84	5.66	×
	Bag	20.29	36.21	60.93	24.64	5.86	×
	Overall	18.08	33.09	57.32	23.49	6.89	×
6-Pack w Ours	Shoe	**43.18**	**63.25**	**65.09**	**11.94**	**0.51**	×
	Bottle, Can and Cup	**33.33**	**53.18**	**64.20**	**17.75**	**5.49**	×
	Bag	**35.59**	**53.39**	**69.14**	**17.23**	**4.5**	×
	Overall	**37.36**	**56.6**	**66.14**	**13.87**	**3.5**	×
CAPTRA [22]	Shoe	23.89	52.30	**35.12**	20.11	**1.29**	1.92
	Bottle, Can and Cup	17.23	47.70	**30.06**	**23.55**	**2.16**	1.75
	Bag	10.77	28.97	**28.11**	**27.52**	**2.23**	**2.91**
	Overall	17.29	42.99	**31.09**	**23.72**	**1.8**	**2.1**
CAPTRA w Ours	Shoe	**27.8**	**58.44**	28.30	**19.80**	1.89	**1.84**
	Bottle, Can and Cup	**26.12**	**49.32**	29.45	29.94	2.9	**1.3**
	Bag	**19.13**	**45**	26.56	28.25	2.5	3.9
	Overall	**24.35**	**50.92**	28.10	25.99	2.4	2.3

Evaluation Results on the MoVi-E dataset: Table II contains the results of our model compared with the baseline methods on the testing set of MoVi-E. DTM outperforms the baseline in all metrics in MoVi-E. There is a large gap in the metric of $5°5cm$ and $10°10cm$. For the shoe category, the $5°5cm$ and $10°10cm$ (43.18 and 63.25) are almost three times the percentage of the baseline (12.02 and 25.37). Specially, the translation error of the shoe category is 0.51 while the baseline is 9.18, which is about 10cm error less than the baseline. For the symmetric category, Bottle, Can and Cup, the metrics are a bit worse than the other two categories but the $5°5cm$, $10°10cm$ and $mIoU$ are about 10 percentage higher than the baseline.

For the baseline CAPTRA, our model's overall $5°5cm$ and $10°10cm$ are better than the baseline. For the symmetric category (Bottle, Can and Cup) and the bag categories, the $10°10cm$ is 45%, which is 21 percentage points higher than the baseline CAPTRA. Although the mIoU, R_{error}, T_{error} and S_{error} are worse than the baseline CAPTRA, their gaps are minor, where the largest one is the $mIoU$ of the Shoe category. Our model's mIoU (28.30%) is about 7 percentage points lower than the baseline model. Other metrics are almost the same but slightly worse, which are about 2 to 3 percentage points lower than the baseline CAPTRA.

Fig. 2 shows the examples of DTM integrated with the baseline CAPTRA and 6-Pack. The green bounding box is the ground truth and the red bounding box is the predicted bounding box. The visualized results include two scenes where the target object is partially visible. DTM helps the model to align the bounding box better with the ground truth, i.e., DTM is more robust when the vision of the object is limited.

The processing time of TDM is 94ms in CAPTRA and 75ms in 6-Pack with a length 4 history sequence. The longer run-time in CAPTRA is due to the larger number of points sampled in CAPTRA than in 6-Pack.

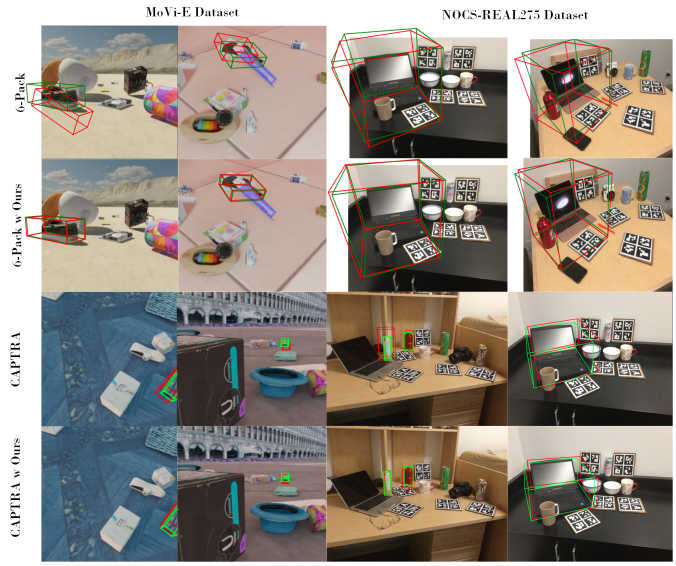

Figure 2. Qualitative results of DTM and the two baseline methods.

D. Ablation Study

Length of History Sequence (Sliding Window): The length of the history sequence may affect the essential features gathered from history frames. We evaluate DTM with the backbone of 6-Pack in the MoVi-E dataset shown in Fig. 3

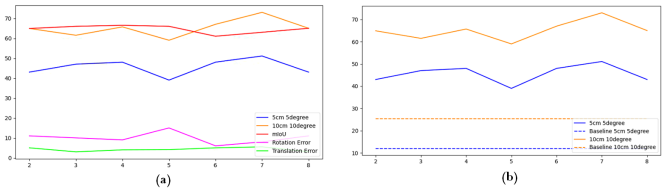

Figure 3. The performance of DTM with increasing length of history sequence using the shoe as the target object in MoVi-E with the 6-Pack baseline. (a) are all metrics from the MoVi-E dataset and (b) shows comparisons between $5°5cm$ and $10°10cm$ to compare DTM with the baseline 6-Pack (dashed line).

For (a) of Fig. 3, when the length of the history sequence is 6 and 7, the $5°5cm$ and $10°10cm$ reach a peak. However, the variation of performance with the window length indicates that the optimal length depends on the motion sequence in the video sequences. We also studied the influence of the window length on the performance of our model in the NOCS-REAL275 which were minor due to the scenes in NOCS-REAL275 dataset being more static and less challenging than MoVi-E. Due to the GPU limit, we are not able to investigate very long history sequences.

Normalization in Local Difference Module: We also evaluate the performance of our model with and without the softmax normalization in the local difference module.

Table III
DTM TRAINED ON LAPTOP CATEGORY WITH AND WITHOUT NORMALIZATION. (6-PACK AS BASELINE, TEMPORAL HISTORY LENGTH IS 4 FRAMES).

	$5°5cm\uparrow$	$10degree°10cm\uparrow$	mIoU $\uparrow$	$R_{error}\downarrow$	$T_{error}\downarrow$
Ours	**36.80**	**60.45**	**65.88**	**9.9**	**2.25**
Ours w/o Normalization	20.75	36.68	36.92	19.88	3.7

The results in Table III show the normalization step in the local difference module can help to improve the performance by normalizing the difference features iteratively gathered from the first frame in the history sequence to the current frame.

VI. Conclusion

We presented DTM, a difference-based temporal module for category-level 6DoF pose tracking models in AR applications. DTM refines the features of points and pixels by leveraging temporal differences. The difference is calculated based on KNN between the pixels and points of two frames. Our module acts as a mid-layer between the feature encoder and other downstream components. We integrate our module with two state-of-the-art category 6DoF trackers and evaluate their performance on two challenging benchmarks. DTM increases the accuracy and robustness over the baseline models. In the future, we want to research the integration of DTM also in pose tracking methods based on posegraphs.

Acknowledgment

J. Lang acknowledges the support of the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

[1] R. T. Azuma, "A survey of augmented reality," *Presence: teleoperators & virtual environments*, vol. 6, no. 4, pp. 355–385, 1997.

[2] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A survey of autonomous driving: Common practices and emerging technologies," *IEEE access*, vol. 8, pp. 58 443–58 469, 2020.

[3] N. Kothari, M. Gupta, L. Vachhani, and H. Arya, "Pose estimation for an autonomous vehicle using monocular vision," in *ICC*. IEEE, 2017, pp. 424–431.

[4] Z. Xiao, X. Wang, J. Wang, and Z. Wu, "Monocular orb slam based on initialization by marker pose estimation," in *ICIA*. IEEE, 2017, pp. 678–682.

[5] C. G. Cifuentes, J. Issac, M. Wüthrich, S. Schaal, and J. Bohg, "Probabilistic articulated real-time tracking for robot manipulation," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 577–584, 2016.

[6] D. Kappler, F. Meier, J. Issac, J. Mainprice, C. G. Cifuentes, M. Wüthrich, V. Berenz, S. Schaal, N. Ratliff, and J. Bohg, "Real-time perception meets reactive motion generation," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1864–1871, 2018.

[7] J. J. Rodrigues, J.-S. Kim, M. Furukawa, J. Xavier, P. Aguiar, and T. Kanade, "6d pose estimation of textureless shiny objects using random ferns for bin-picking," in *IROS*. IEEE, 2012, pp. 3334–3341.

[8] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab, "Model based training, detection

and pose estimation of texture-less 3d objects in heavily cluttered scenes," in *ACCV*. Springer, 2013, pp. 548–562.

[9] C. Choi and H. I. Christensen, "Real-time 3d model-based tracking using edge and keypoint features for robotic manipulation," in *ICRA*. IEEE, 2010, pp. 4048–4055.

[10] C. Wang, R. Martín-Martín, D. Xu, J. Lv, C. Lu, L. Fei-Fei, S. Savarese, and Y. Zhu, "6-pack: Category-level 6d pose tracker with anchor-based keypoints," in *ICRA*. IEEE, 2020, pp. 10 059–10 066.

[11] S. Suwajanakorn, N. Snavely, J. J. Tompson, and M. Norouzi, "Discovery of latent 3d keypoints via end-to-end geometric reasoning," *Advances in neural information processing systems*, vol. 31, 2018.

[12] K. S. Arun, T. S. Huang, and S. D. Blostein, "Least-squares fitting of two 3-d point sets," *IEEE Transactions on pattern analysis and machine intelligence*, no. 5, pp. 698–700, 1987.

[13] D. Cores Costa, V. M. Brea Sánchez, and M. F. Mucientes Molina, "Short-term anchor linking and long-term self-guided attention for video object detection," 2021.

[14] D. C. Costa, "Spatio-temporal convolutional neural networks for video object detection," Ph.D. dissertation, Universidade de Santiago de Compostela, 2022.

[15] M. Shvets, W. Liu, and A. C. Berg, "Leveraging long-range temporal relationships between proposals for video object detection," in *ICCV*, 2019, pp. 9756–9764.

[16] M. Liu, M. Zhu, M. White, Y. Li, and D. Kalenichenko, "Looking fast and slow: Memory-guided mobile video object detection," *arXiv preprint arXiv:1903.10172*, 2019.

[17] L. Wang, Z. Tong, B. Ji, and G. Wu, "Tdn: Temporal difference networks for efficient action recognition," in *Proceedings of the CVPR*, 2021, pp. 1895–1904.

[18] M. H. Abdelpakey, M. S. Shehata, and M. M. Mohamed, "Denssiam: End-to-end densely-siamese network with self-attention model for object tracking," in *ISVC*. Springer, 2018, pp. 463–473.

[19] P. Cavanagh and G. A. Alvarez, "Tracking multiple targets with multifocal attention," *Trends in cognitive sciences*, vol. 9, no. 7, pp. 349–354, 2005.

[20] H. S. Meyerhoff, F. Papenmeier, and M. Huff, "Studying visual attention using the multiple object tracking paradigm: A tutorial review," *Attention, Perception, & Psychophysics*, vol. 79, pp. 1255–1274, 2017.

[21] B. Du, Y. Sun, S. Cai, C. Wu, and Q. Du, "Object tracking in satellite videos by fusing the kernel correlation filter and the three-frame-difference algorithm," *IEEE Geoscience and Remote Sensing Letters*, vol. 15, no. 2, pp. 168–172, 2017.

[22] Y. Weng, H. Wang, Q. Zhou, Y. Qin, Y. Duan, Q. Fan, B. Chen, H. Su, and L. J. Guibas, "Captra: Category-level pose tracking for rigid and articulated objects from point clouds," in *ICCV*, 2021, pp. 13 209–13 218.

[23] H. Wang, S. Sridhar, J. Huang, J. Valentin, S. Song, and L. J. Guibas, "Normalized object coordinate space for category-level 6d object pose and size estimation," in *Proceedings of the CVPR*, 2019, pp. 2642–2651.

[24] K. Greff, F. Belletti, L. Beyer, C. Doersch, Y. Du, D. Duckworth, D. J. Fleet, D. Gnanapragasam, F. Golemo, C. Herrmann *et al.*, "Kubric: A scalable dataset generator," in *CVPR*, 2022, pp. 3749–3761.

[25] C. Sahin and T.-K. Kim, "Category-level 6d object pose recovery in depth images," in *ECCV Workshops*, 2018, pp. 0–0.

[26] C. Wang, D. Xu, Y. Zhu, R. Martín-Martín, C. Lu, L. Fei-Fei, and S. Savarese, "Densefusion: 6d object pose estimation by iterative dense fusion," in *CVPR*, 2019, pp. 3343–3352.

[27] Y. Lin, J. Tremblay, S. Tyree, P. A. Vela, and S. Birchfield, "Keypoint-based category-level object pose tracking from an rgb sequence with uncertainty estimation," in *ICRA*. IEEE, 2022, pp. 1258–1264.

[28] S. Yu, D.-H. Zhai, Y. Xia, D. Li, and S. Zhao, "Cattrack: Single-stage category-level 6d object pose tracking via convolution and vision transformer," *IEEE Transactions on Multimedia*, 2023.

[29] Y. Cui, L. Yan, Z. Cao, and D. Liu, "Tf-blender: Temporal feature blender for video object detection," in *ICCV*, 2021, pp. 8138–8147.

[30] B. Wen and K. Bekris, "Bundletrack: 6d pose tracking for novel objects without instance or category-level 3d models," in *IROS*. IEEE, 2021, pp. 8067–8074.

[31] B. Wen, J. Tremblay, V. Blukis, S. Tyree, T. Müller, A. Evans, D. Fox, J. Kautz, and S. Birchfield, "Bundlesdf: Neural 6-dof tracking and 3d reconstruction of unknown objects," in *CVPR*, 2023, pp. 606–617.

[32] B. Wen, W. Yang, J. Kautz, and S. Birchfield, "Foundationpose: Unified 6d pose estimation and tracking of novel objects," *arXiv preprint arXiv:2312.08344*, 2023.

[33] L. Zheng, A. Leonardis, T. H. E. Tse, N. Horanyi, H. Chen, W. Zhang, and H. J. Chang, "Tp-ae: Temporally primed 6d object pose tracking with auto-encoders," in *ICRA*. IEEE, 2022, pp. 10 616–10 623.

[34] L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool, "Temporal segment networks: Towards good practices for deep action recognition," in *ECCV*. Springer, 2016, pp. 20–36.

[35] Y. Zhao, Y. Xiong, and D. Lin, "Recognize actions by disentangling components of dynamics," in *CVPR*, 2018, pp. 6566–6575.

[36] J. Y.-H. Ng and L. S. Davis, "Temporal difference networks for video action recognition," in *WACV*. IEEE, 2018, pp. 1587–1596.

[37] Z. Liu, D. Luo, Y. Wang, L. Wang, Y. Tai, C. Wang, J. Li, F. Huang, and T. Lu, "Teinet: Towards an efficient architecture for video recognition," in *AAAI*, vol. 34, no. 07, 2020, pp. 11 669–11 676.

[38] B. Jiang, M. Wang, W. Gan, W. Wu, and J. Yan, "Stm: Spatiotemporal and motion encoding for action recognition," in *ICCV*, 2019, pp. 2000–2009.

[39] Y. Li, B. Ji, X. Shi, J. Zhang, B. Kang, and L. Wang, "Tea: Temporal excitation and aggregation for action recognition," in *CVPR*, 2020, pp. 909–918.

[40] Y. He, H. Huang, H. Fan, Q. Chen, and J. Sun, "Ffb6d: A full flow bidirectional fusion network for 6d pose estimation," in *CVPR*, 2021, pp. 3003–3013.

[41] Y. Wang, W.-L. Chao, D. Garg, B. Hariharan, M. Campbell, and K. Q. Weinberger, "Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving," in *CVPR*, 2019, pp. 8445–8453.