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

by

Zishen Chen

A thesis
submitted to the University of Ottawa
in partial fulfillment of the
thesis requirement for the degree of
Master of Computer Science
in the
Ottawa-Carleton Institute for Computer Science

Declaration of Authorship

I hereby certify that this thesis is entirely my own original work except where otherwise indicated. I am aware of the University's regulations concerning plagiarism, including those concerning consequent disciplinary actions. Any use of the works of any other author, in any form, is properly acknowledged at their point of use.

Abstract

Monocular 6DoF pose tracking has many applications in augmented reality, robotics and other areas and because of the rise of deep learning new approaches such as category-level models are successful. The temporal information in sequential data is essential for both online and offline tasks, which can help boost the quality of predictions while encountering some unexpected influences like occlusions and vibration. In 2D object detection and tracking, substantial research has been done in leveraging temporal information to improve the performance of the model. Nevertheless, it is challenging to lift the temporal processing to 3D space because of the ambiguity of the visual data. In this thesis, we propose a method to calculate the temporal difference of points and pixels assuming that the K nearest points share similar features. The extracted features from the difference are learned to weigh the relevant points in the temporal sequence and aggregate them to provide support to the current frame's prediction. We propose a novel difference-based temporal module to incorporate both RGB and 3D points data in a temporal sequence. This module can be easily integrated with any category-level 6DoF pose tracking model which uses RGB and 3D points as input. We evaluate this module on two state-of-the-art category-level 6D pose tracking models and the result shows that it can increase the model's accuracy and robustness in complex scenarios.

Acknowledgements

I would like to give my greatest gratefulness to my supervisor, Professor Jochen Lang, who help me with my master's research. It is impossible for me to complete my thesis without his academic guidance and support. His kind and patient suggestions have led me to overcome several difficulties during my research. On the other hand, the variable comments from my colleagues in VIVA lab are indispensable, and give me a lot of inspiration.

Dedication

In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of the University of Ottawa's products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to the IEEE website to learn how to obtain a License from RightsLink.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

Object pose prediction is essential because the urge to make computers understand and model the motion of objects in 3D space. The object pose prediction problem has been substantially studied in the past decade. The application in Augmented Reality (AR) [4], autonomous driving [51, 125], SLAM (Simultaneous Localization and Mapping) [120] and robotics application [16, 44, 48] also reveals the importance of it. In these fields, the system needs to process the data given by sensors like LiDAR (point cloud) [45], RGBD camera (depth) [31].

Autonomous driving is one of the typical fields that require precise 3D and 2D information processing to predict the direction of other vehicles on the road will provide more flexibility to avoid subtle danger. In this case, the model is required to combine the

information from different sensors [8] including stereo camera and GPS to recognize and analyze the status of other vehicles and pedestrians status. Because in the scenario of autonomous driving, the vehicles can only rotate around the axis that is orthogonal to the ground, 3D object detection and tracking [81, 91, 123] and optical flow estimation [2, 85] to predict the 3D bounding box of it will be more efficient than pose detection and tracking. Unlike autonomous driving, in robotics manipulation and navigation, the target object has 6DoF (degree of freedom) pose, which is essential for the system to make instant reactions. Recognizing and grasping objects accurately and performing instant feedback in a noisy scene is always a challenging task to tackle in robotics-related problems. Thus, visual tracking plays an indispensable role in robotic applications. The two components, joint measurement [16, 36, 90] and object pose tracking [118] are the core algorithms of a robotic manipulation system. The system needs to analyze the pose of joints and target objects to give reasonable orders. AR is another field that also benefits from object pose estimation. In 1997, Ronald et al. [4] gave an explicit definition and comprehensive survey of AR. In their survey, a well-developed AR system needs to have three features. The combination of real and virtual, real-time interaction and 3D registration. Generally, an AR system consists of the alignment between the real and virtual world, which will achieve a high-level simulation to immerse users. In this point, VR (Virtual Reality) [92] has similar functions. In AR applications, its hardware is comprised of different sensors to collect real-world data and a display device to show users the results after processed by the system. With the development of the mobile phone, the AR device with a camera with a display screen becomes most people's preference. Therefore, it becomes a vision-based problem, and it becomes a basic research direction in AR. Basically, the problem contains the alignment

between CG (computer graphics) camera and real camera [70]. The CG camera and the real camera should have the same position and intrinsic parameters. This process is called compositing. The problem now becomes estimating the pose of the camera in the real world.

In this thesis, our method focuses on addressing the pose estimation challenges in AR applications. Monocular RGBD cameras that return the depth and RGB image of the scene are commonly used as sensors in AR applications because they can simply produce 3D information. In AR, the system is required to produce robust, accurate and instant pose predictions of the target object in video to enhance the human-computer Interaction. The robustness is one of the essential aspects to achieve during inference, where the quality of the input frame sequence is affected by a lot of factors as the cameras are mostly deployed on mobile devices. The motion of the camera is not steady. On the other hand, the target objects in AR are frequently manipulated by users. The pose and appearance of the objects will experience rapid change. Making stable and real-time pose predictions is the key to better align the CG camera and the real camera.

Among all the applications of AR, remote assistance in repair tasks has been a rising direction in the past decade [94]. Customers request repair services from manufacturers or repair shops when the products they purchased are broken. The procedure usually includes sending the products back to the factory, damage assessment, broken components replacement, reassembling and receiving the repaired items from the manufacturers, which will take time depending on the complexity of the products. Repair services are often not available and while manufacturers are obliged to exchange broken products under warrant terms, consumer products become unusable after even minor defects after the warranty

period. In addition, the practices of product exchange is not environmentally sustainable. Moreover, due to the outbreak of COVID-19 in 2020, the demand for repairing at home has been increasing, where the transport cost can be minimized. In this case, guidance that allows the customer to replace the broken components and reassemble the item is needed. Traditionally, instructional booklets, video clips and online audio guidance are common ways to provide customers with remote assistance. However, the limitations of these methods impede the interaction between customers and experts, where customers cannot get enough information from these channels. For example, the static images in the instructional booklets have a different pose or appearance from the real products, which will mislead the users. The experts who provide online audio assistance may have limited access to the broken product orally described by the users, which will produce incorrect guidance and cause cascading issues. Generally, these traditional methods are not interactive enough to let customers disassemble or assemble the items following experts' guidance. In this case, AR is one of the solutions for remote assistance in repair tasks, where the visual combination of virtual and real objects allows the customers to precisely execute the disassembly and assembly guidance. The experts can also recognize and track each component of the product [80, 106].

Compared to other AR applications like games, the hardware requirement for repair assistance can be simpler, where the assistance service can also be efficient and accurate without the immersive user experience provided by complex devices. One display screen and some sensors to receive visual input are enough. A mobile phone with AR system deployment can fit the demand. The repair assistance can be provided in an online or offline manner. In online service, the customers can book an appointment with the experts

to finish the repair together. The experts will monitor the repair process and give guidance to the users, where the position of the target component will be displayed as a virtual object in the users' device. The experts can modify the position of the virtual object to fit different conditions. In offline service, the customers can finish the repair by following the automated guidance pre-recorded in the system. In either of these two services, the AR system is required to know the pose of each component to highlight the part that the customer needs to pay attention to and the target position that the component needs to fit into. In this case, high-quality 6DoF object pose tracking is needed. Because the components after disassembly will be held in the hand of the customer, occlusion which may cause the system to lose track will occur. Providing stable 6DoF pose tracking during the repair process is the main challenge of the task. In this thesis, we propose a method to use temporal information to enhance the performance of 6DoF pose tracking in repair assistance using AR.

1.2 Definition

6DoF (6 Degree of Freedom) pose is comprised of a rotation $R \in SO(3)$ and a translation $T \in R^3$, which together indicate the status of an object in 3D space. R is an orthogonal matrix that represents the object's rotation around the x, y and z axis in 3D space. T is a vector that points to the object's center. These two variables can precisely indicate the object's status. They are both differentiable, which means they can be optimized using traditional non-linear optimization methods and deep learning.

1.3 Solution Approaches

In the past, the common way to deal with the 6DoF pose problems is through using optimization-based methods, integrating Lie Algebra [25] and non-linear optimization [89]. The majority of them [37, 86] rely on a hand-crafted feature. Thanks to the advance of deep learning techniques, modelling the problem into a deep learning network is possible. Zhaoxin et al. [26] provides a detailed review of the application of deep learning on pose prediction tasks. First, the term monocular means the input is either RGB or RGBD data captured by a single monocular camera. Correspondingly, there are other types of data captured by binocular [6, 62] and stereo [54, 121] cameras. Although stereo and binocular cameras provide more information regarding the scene, their high cost is an inevitable drawback, which makes monocular cameras still a preference for most people. Because the monocular camera can only provide RGB or RGBD data, efficient algorithm support is necessary.

The tasks can be categorized into two main types, category-level and instance-level. The difference between them is the availability of the 3D CAD model. For instance-level, the model is applied to predict the pose of an object whose shape, and size is exactly the same as the one during training. However, the category-level models aim at tackling more complicated scenarios, while the CAD model is not available during training and evaluation. It is trained on objects in the same category and is sensitive to all the objects despite different shapes and textures in the same category.

In recent years, researchers started to realize the advantage of multi-frame tracking over single-frame. Previously, pose tracking has usually been tackled by single-frame estima-

tion, which means the model will estimate the relative transformation from camera space to object space for the object being tracked in each frame of a video individually. According to the position of the camera and object, the transformation can be very large. Predicting such a large transformation can introduce a great amount of errors. On the other hand, it will also ignore inter-frame relations, because it does not take previous frames into account. Object tracking in 2D images has seen a lot of development in the last decade. Wenhan et al. [66] give a comprehensive review of it. The tracking of 3D objects, namely 6DoF pose tracking is rarely discussed. In a pose tracking problem, the model will predict the relative pose change between the current frame and the next frame. Traditionally, the pose of the objects can be recovered by using hand-crafted features [15, 37, 86] and solving an optimization problem. The optimization problem is to minimize the error of projection, which is not efficient and accurate enough as the texture and categories of objects are becoming more complicated. The increasing vision variations including illuminant change, motion blur, scene clusters and occlusions make the task more complex. With the incorporation of deep learning techniques, Wang et al. [100] propose a way to address the 6DoF category-level pose tracking problem. They follow the idea of [95], proposing a keypoint-based method to track the objects in the NOCS-REAL275 dataset [103]. 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 square optimization [3]. The biggest advantage of this model is that it does not require the manual annotation of precise keypoints neither does it need a CAD model of the object. The model will learn to generate matched keypoints in the preset 3D anchor box. After that, a lot of research into 6DoF pose tracking was conducted including another

state-of-the-art regression-based method Captra [115].

1.4 Motivation

However, in most of the application scenarios, pose tracking is used to process video sequences. The majority of current pose tracking methods only consider two frames, the current frame and the previous frame without paying attention to a temporal history of frames which contains rich temporal and spatial features that can improve the pose prediction. The rapidly changing objects' motion in the video will make the model difficult to produce promising results if we only take one frame into account. The object can be temporarily occluded, or blurred because of it. This will always cause the loss of features which makes models make false judgements. The motion pattern extracted from the history, in this case, will reduce errors. Although, history has been deeply explored in 2D computer vision [18, 19, 63, 93]. In 2D computer vision, the method used to process sequential data is commonly called temporal modelling and action recognition [104]. Attention [1, 9, 73] and difference [24] techniques have been deeply exploited in 2D. Although the temporal difference in 2D computer vision tasks is associated with optical flow [39] and it can support motion extraction, there is still a lack of a proper method in 3D to incorporate temporal information. In temporal 6DoF tracking tasks, the methods proposed by Wen et al. [110, 112, 113] store the historical frames as unordered references to achieve the maximum overlap to conduct the optimization of current pose and learn the shape of the target object to transform the category-level model to a pseudo-instance-level model, where the continuity and inter-relation across frames are not considered. TP-AE proposed

by Zheng et al. [128] uses the temporal frames to learn the trajectory of the object to estimate a prior pose of the current frame. However, the learning of the prior pose may not correctly reflect the expected position of the object while the rapid motion appears in the consecutive frames. In this case, embedding the whole stored history sequence into useful features to enhance the current prediction will be the better choice. Due to the different representations and high variance of data, how to exploit the relations of 3D points across frames becomes the most challenging problem.

1.5 Thesis Statements

In this thesis, we propose a novel way to aggregate history information into state-of-the-art 6D pose trackers. Our method is based on difference calculations between current frames and historical frames. Considering the input of the 6D tracker can be either 3D points [115] or RGBD [100], the proposed method aggregates pixels and points. In order to calculate the difference between points and pixels, we need to first find their correspondence. We use KNN (K-NearestNeighbor) to define that one point's feature is similar to its K neighbours with the closest Euclidean distance. Then the difference calculation will be applied to different point sets to accomplish cross-frame feature aggregation. All of the calculations between the feature of two pointsets are based on KNN. Like most of the 2D temporal model, we use a sliding window with a pre-defined size to dynamically store and update frames. The module will work on the history sequence and learn to produce weights for each frames based on the difference calculation method aforementioned. These weights represent the per-channel importance of each point's feature. Then we globally and locally

aggregate the historical features into current frame (see Chapter 3 for more details). The module we have developed is a plug-in module that can be easily used on any category-level 6D tracker that takes RGBD or 3D point cloud as input. The experiments show that models gain significant improvements from our difference-based history module.

1.6 Contributions

The summary of this thesis's contributions is as follows:

- We designed a difference-based temporal module that can solve the 6D pose tracking problem. This module aggregates features by calculating differences to enhance the prediction of the object pose in the current frame.
- We propose a new way to fuse and calculate the difference between two frames based on their distance.

We tested our methods in two state-of-the-art architectures: 6-Pack [100] and Captra [115] on NOCS-REAL275 [103]. and MoVi-E [30] datasets. The model with our module outperforms the corresponding baseline.

1.7 Thesis Organization

This section is a brief outline of this thesis's structure. This thesis has 6 chapters.

- Chapter 2 is the literature review. We introduce the basic definitions of 6DoF pose and how to minimize the pose error in optimization and deep learning. Then we discuss the instance-level and category-level 6DoF pose detection and tracking approaches from different aspects. To better describe our ideas of temporal difference in the later sections, we will cover some reviews of temporal networks, action recognition and motion detection models used in 2D and 3D. Our model fuses the RGB and points representation, so we also introduce feature fusion methods that learn to fuse features from different inputs. Furthermore, we describe the metrics and dataset that we used in our experiments.

- Chapter 3 is the detailed introduction to our difference-based temporal module. In this section, we start with our module's architecture, we explicitly describe each step of the data flow and calculations from end to end. We also explicitly describe how we integrate the history module into other networks.

- Chapter 4 covers the experimental setup details. We introduce the data augmentation, loss function and symmetry object handling methods. Then, we specify how we construct our training data and training strategy.

- Chapter 5. In this Chapter, we present the training results on the benchmarks. We give a comparison of our model and two state-of-the-art methods and conduct an ablation study including the comparison between baseline (6-pack and Captra) and the one integrated with our history model and the impact of sliding window's length to the performance.

- Chapter 6 is the conclusion and future work. We summarize the work that we have

done and analyze our model's advantages and disadvantages for our future research.

Chapter 2

Literature Review

In this section, we review the works that are related to our thesis including 3D detection, Instance-level and Category-level 6D pose detection, Instance-level and Category-level pose tracking, 2D and 3D temporal models and commonly used benchmarks.

To estimate the position of the objects in 3D scene, there are three options, which are 6DoF pose detection, 6DoF pose tracking and 3D bounding box detection. Although their purposes are inferring the object's pose directly (6DoF pose detection and tracking) or indirectly (3D bounding box detection), the scheme they are using is different. For 3D bounding box detection, the model does not cover the pose-related information, the aims at predicting the bounding boxes of the objects, which is predicting the edges and the corners. The 3D bounding can somehow reflect the pose of the object. However, the gap between the actual pose and the pose inferred from the 3D bounding box cannot be eliminated. The advantage of 3D bounding box detection is that it can efficiently handle

tasks that do not require precise pose estimation.

However, although 3D bounding box prediction can fit most of the demands that only require knowing the approximate pose of the objects, some tasks require accurate pose, the direct pose estimation is needed. The pose detection allows the model to directly process the pose-related parameters to have a better understanding of the object's geometry. Nevertheless, predicting the object's pose transformed from the camera coordinate system to the object coordinate system may suffer from errors with increasing distance from the camera and the object. In this case, predicting the relative pose change between the current frame and the previous frame given a video sequence will be a better choice. Hence, 6DoF pose tracking is proposed for real-time inference. Instead of estimating the pose of the object with respect to the camera, the 6D pose tracking models predict the relative pose change between two consecutive frames.

Based on 6D pose tracking models, most models only focus on two frames, the current and the previous frames. The lack of history information may lead to poor performance encountering occlusion and fast motion. Learning from historic frames allows the model to extract the features from the object that is occluded or blurred because of fast motion in the current but intact in historic frames to enhance the feature of the current frame. These methods are commonly named temporal models in previous 2D computer vision tasks. However, in 3D computer vision, the temporal model is rarely discussed. The combination of temporal models and 6D pose track is a potential direction from our point of view.

In the following sections, we will cover detailed reviews of 3D bounding box detection,

6D pose detection, 6D pose tracking, and the temporal model's application on both 2D and 3D.

2.1 Background

In this section, we will give a brief introduction to the least-square 3D points matching algorithm [3, 47] and the Umeyama algorithm [97].

2.1.1 Least-squares 3D points matching algorithm [3, 47]

Least-square 3D points matching is commonly used to align the two point sets with known correspondence. We mainly introduce the proof based on the method proposed by Arun et al. [3]. Given the two point sets p_i^t and p_i^s, the problem can be described as an optimization problem:

$$[R|t] = \underset{R|t}{argmin} \frac{1}{N} \sum_{i=1}^{N} \|p_i^t - Rp_i^s - t\|^2 \tag{2.1}$$

Non-linear optimization and SVD (Singular Value Decomposition) are commonly used to solve this problem. In this thesis, we mainly use SVD. In order to solve it, the rotation and translation are computed separately. For rotation, the centers of the two point sets (u^s, u^t) are calculated. The target formula of rotation R is $R^* = argmin$ $\frac{1}{N}\sum_{i=1}^{N} \|(p_i^t - u^t) - R(p_i^s - u^s)\|^2$. The formula can be simplified as $R^* = argmax \sum_{i=1}^{N}(p_i^s - u^s)R(p_i^t - u^t)^{\mathrm{T}}$. With the definition of the trace of a matrix, the formula can be further transformed to $R^* = argmax\ tr(RH)$, where $H = \sum_{i=1}^{N}(p_i^s - u^s)(p_i^t - u^t)^{\mathrm{T}}$.

For two matrices A and B, where $BB^{\mathrm{T}} = I$. We have

$$tr(BAA^T) = tr(A^T BA) \tag{2.2}$$

$$tr(A^T BA) = \sum_i A_i^T (BA_i) \tag{2.3}$$

where the A_i means the i^{th} column vector in matrix A. For Eq. 2.3, according to Cauchy-Schwarz inequality, we have

$$\sum_i A_i^T (BA_i) \leq \sum_i \sqrt{(A_i^T A_i)(A_i^T B^T BA_i)} \tag{2.4}$$

$$\sum_i A_i^T (BA_i) \leq \sum_i A_i^T A_i \tag{2.5}$$

$$tr(A^T (BA)) \leq tr(A^T A) \tag{2.6}$$

Then we have the conclusion $tr(AA^{\mathrm{T}}) \geq tr(BAA^{\mathrm{T}})$, $BB^{\mathrm{T}} = I$. According to this conclusion, we have $tr(RH) \geq tr(BRH)$, $BB^{\mathrm{T}} = I$. When RH can be written in the form of AA^{T}, $tr(RH)$ has the largest value. Hence, we calculate the SVD of H, we have $H = USV^{\mathrm{T}}$. When $R = VU^{\mathrm{T}}$ and $A = VS^{\frac{1}{2}}$, we have $VU^{\mathrm{T}} \cdot USV^{\mathrm{T}} = AA^{\mathrm{T}}$, which satisfies $AA^{\mathrm{T}} = RH$. After knowing the rotation, the translation can be computed through $u^t - Ru^s$.

2.1.2 Umeyama Algorithm [97]

Umeyama Algorithm is also a least-squares solution to 3D points alignment problem, which is similar to the method described above. The optimization target is

$$[R|t|s] = \underset{R|t|s}{argmin} \frac{1}{2} \sum_{i=1}^{N} \|p_i^t - s \cdot Rp_i^s - t\|^2 \tag{2.7}$$

where s is the scale, which is an extra variable that needs to be calculated. Also, in the previous section, the Umeyama algorithm decouples the problem into the estimation of R, t and s. Similarly, the centers of the two point sets (u^s, u^t) are calculated. Eq. 2.7 can be further written as

$$[R|t|s] = \underset{R|t|s}{argmin} \frac{1}{2} \sum_{i=1}^{N} \|p_i^t - s \cdot Rp_i^s - t - u^t + s \cdot Ru^s + u^t - s \cdot Ru^s\|^2 \tag{2.8}$$

$$[R|t|s] = \underset{R|t|s}{argmin} \frac{1}{2} (\sum_{i=1}^{N} \|p_i^t - u^t - s \cdot R(p_i^s - u^s)\|^2 + \|u^t - sRu^s - t\|^2) \tag{2.9}$$

Because R and s are only relevant to the first term in Eq. 2.9 and t can be computed by $t = u^t - sRu^s$, we can solve the R, s above. The target then becomes

$$[R|s] = \underset{R|s}{argmin} \frac{1}{2} (\sum_{i=1}^{N} \|p_i^t - u^t\|^2 - 2s(p_i^t - u^t)^T R(p_i^s - u^s) + s^2 \|R(p_i^s - u^s)\|^2) \tag{2.10}$$

We define the variance (σ^s, σ^t) and covariance matrix (D) of p^s and p^t as

$$\sigma^t = \frac{1}{N} \sum_{i=1}^{N} \|p_i^t - u^t\|^2 \tag{2.11}$$

$$\sigma^s = \frac{1}{N}\sum_{i=1}^{N} \|p_i^s - u^s\|^2 \tag{2.12}$$

$$D = \frac{1}{N}\sum_{i=1}^{N} (p_i^t - u^t)^T R(p_i^s - u^s) \tag{2.13}$$

R can be calculated by using Eq. 2.13 and the SVD described in Chapter 2.2.1. Then we need to calculate s and t. First, for scale s, Eq. 2.10 can be written into

$$[R|s] = \underset{R|s}{argmin}\, \sigma^t - 2sD + s^2\sigma^s \tag{2.14}$$

$$[R|s] = \underset{R|s}{argmin}(s\sqrt{\sigma^s} - \frac{D}{\sqrt{\sigma^s}})^2 + \frac{(\sigma^t\sigma^s - D^2)}{\sigma^s} \tag{2.15}$$

The first term $(s\sqrt{\sigma^s} - \frac{D}{\sqrt{\sigma^s}})^2$ is relevant to s, while the second term $\frac{(\sigma^t\sigma^s - D^2)}{\sigma^s}$ is not. For each R (D contains R), we can always find a s to make the first term equal to 0. For s, when the first term is equal to 0, Eq. 2.15 has the minimum value, hence, we solve

$$(s\sqrt{\sigma^s} - \frac{D}{\sqrt{\sigma^s}})^2 = 0 \tag{2.16}$$

$$s = \frac{D}{\sigma^s} \tag{2.17}$$

After knowing R and s, t can be computed as $u^t - s \cdot Ru^s$.

2.2 Instance-level Object Pose Detection

In this section, we will review the related works in instance-level pose detection. Although our thesis is a category-level pose detection method, instance-level methods have inspired us. We follow the categories given by Fan et al. [26].

2.2.1 Definition

In instance-level object pose detection, the CAD model M is one of the inputs during training and evaluation. Given an RGBD image I and the target object's CAD model M, the task can be defined as $\mathcal{T} = F([I, M], \Theta)$, where $\mathcal{T} \in SE(3)$ is the transformation comprised of rotation $R \in SO(3)$ and translation $T \in R^3$. Θ are the learnable parameters of the model F. The instance-level model works with exactly the same object during training and during prediction.

The transformation $\mathcal{T}$ usually refers to the transformation between two coordinate systems. In the scene, the coordinate system can be divided into object space, world space and camera space. Methods commonly predict the transformation between the camera and object space. Because the RGB data doesn't provide any three-dimensional information, most of the RGB methods [49, 119] solve ill-posed problems. Deep learning methods are commonly used to solve this problem and we will focus on these methods review.

2.2.2 Keypoints-based methods

Keypoints-based methods treat the object as keypoints, and use those keypoints to construct 3D-3D correspondence, either considering the keypoints as the vertices of the 3D bounding box or using keypoints to build correspondence between two point sets. PVNet3D proposed by Yisheng et al. [35] is a pioneer work in this field. PVNet3D takes the fusion of RGB and points features to generate a per-point offset with respect to the selected keypoints and the semantic segmentation of each point. In this scheme, a Hough voting strategy is applied to acquire the keypoints and center of the object. During the training, the keypoints are selected using FPS (farthest point sampling) [76] by giving the ground truth center and the mesh of the object. FPS algorithm is widely used in point sampling. Suppose that we need to sample N points from a source point set to the target point set. The strategy is then to iteratively update the target point set using the point in the source point set with the greatest distance from all the points in the target point set until the N points are selected to the target point set. In this case, the keypoints can be selected on the surface of the object and be representation of the object. After knowing the predicted keypoints of two point sets, the transformation can be derived by solving a least-squares problem [3].

This work inspires many keypoints-based deep 6D pose estimation tasks. Similarly, PointPoseNet [12] proposed by Wei et al. also adopts the idea of hough voting. PointPoseNet takes RGB and points as input and uses a 2D detection network to detect the object's 2D bounding box and category. Then, like PVNet3D, it predicts a vector for each point, which points to the keypoints and the model also does the segmentation for the

points. The predicted vector for each point is a norm, which only represents the direction. In their experiments, predicting unit vectors has a better result than actual displacement. Thus, the intersection of those directional vectors will be the hypothesis of keypoints. However, the non parallel lines in 3D may not have an intersection, unlike 2D. The author uses the midpoint of the shortest line segment to determine the intersection with a confidence value.

Wu et al. [117] propose a new method to estimate 3D keypoints. Instead of using vector voting in PointPoseNet [12] and PVNet3D [35], they incorporate radial voting. Radial voting uses a sphere with an estimated radial radius for each point. The keypoints are supposed to lie on the surface of the spheres. These spheres are then used to vote for incremented accumulator space on the surface of the spheres, the peak of which is the voted keypoint. Unlike previous methods, the model needs to predict a 3-dimensional vector for each point, which is not accurate for the sparse keypoints. Radial voting only needs to know the radius of the sphere, and render those spheres during inference to vote for keypoints.

2.2.3 Retrieval-based methods

In instance-level object pose detection, the CAD model of the object is available. Thus, we can use the CAD model to construct the virtual object with different poses. Given those samples, the task is becoming comparing the input RGBD data and the samples to find the most similar pair. Kehl et al. propose [50] a method to handle the pose detection problem through a CAE (Convolutional AutoEncoder) [72] or AE (AutoEncoder) [78].

They train the CAE on sampled local patches from the LineMOD dataset [37]. Each patch contains two pieces of the image, each of which is the augmentation of the same scene. The model will minimize the reconstruction error. The purpose of this pretraining is to get a low-dimensional feature descriptor. The pose can be derived by matching the descriptor output from CAE. The biggest drawback of this method is the model depends too much on the representation of the object and lacks occlusion and texture-less object handling.

MTTM (Multi-Task Template Matching) [82] proposed by Park et. al. retrieves the nearest sample with segmentation and transformation prediction. Basically, this model incorporates multiple tasks into the calculation of the similarity of two samples. The model uses Resnet-50 [33] to extract ROI (Region of Interest) and basic features from depth. These features will be then forwarded to three components for multi-task learning. The first task is the distance calculation, which expresses the similarity between the two samples. The mask prediction, task can help the model to exclude the points that belong to occlusions or backgrounds. The last task is the transformation prediction. Unlike the model proposed by Wadim et al. [50], the model predicts the transformation offset between the target and source sample. Because errors always exist, no matter how close the two samples are. However, predicting the transformation offset, in this case, can massively increase the accuracy. The transformation offset is small because the sample has been pre-processed by selecting the closest feature distance. Generally, although retrieval-based methods have robust and stable prediction results in most cases, they rely on the 3D CAD model to generate training templates and representative descriptors [5, 116].

2.3 Category-level Object Pose Detection

In this section, we will review the category-level pose detection method. Because 3D object detection is very important to pose detection and predicting 3D bounding boxes somehow reveals the pose information. 3D object detection is also in the range of our discussion.

2.3.1 Definition

Unlike in instance-level pose detection. The 3D CAD model is not available during either the training or testing phase in category-level object detection. Because the model doesn't have specific prior knowledge of the target object, the size of the object is unknown. Therefore, the model needs to predict 9DoF (9 Degrees of Freedom) of the object. The task can be formulated as $(\mathcal{T}, S) = F([I, M], \Theta)$, where $\mathcal{T} \in SE(3)$ is the transformation comprised of $R \in SO(3)$ and $T \in R^3$. $S \in R^3$ is the size of the object consisting of length, height and depth, ie., So the 9DoF is 3-dimensional rotation, 3-dimensional translation and 3-dimensional shape.

2.3.2 Category-level 3D Object Detection

As we discussed in Chapter 1, 3D detection techniques are significant for AR and autonomous driving. Unlike 2D detection [29,32,42], the predicted 3D bounding box contains the object's rotation information in addition to the location of the object in the image. Similar to 6D pose detection, 3D detection estimate the rotation and translation through a 3D bounding box. The difference between 3D detection and pose detection is that 3D

detection has more emphasis on translation accuracy, which suits those tasks that value more the location than the orientation. This property makes 3D detection models more efficient in inference and easier to train. Chen et al. [13] first propose a monocular 3D object detection method for autonomous driving. The method uses a ground plane, which is assumed to be orthogonal to the image plane, to generate 3D candidates. Then, the 3D candidates are projected to the image plane to calculate confidence scores by using different data including class segmentation, instance level segmentation, shape, contextual features and location priors.

Instead of directly regressing the orientation of the 3D object, many methods combine 2D bounding boxed in 3D detection. Mousavian et al. [77] use 2D bounding boxes as geometric constraints to produce a stable 3D detection. In most cases of autonomous driving, the objects are assumed to only have 3 DoF. Because the vehicle and pedestrians are assumed to only make movements along the axes that are parallel with the ground and rotations around the axis that is vertical to the ground. Therefore, for each 2D bounding box side, there is at least one projection of a 3D bounding box corner. Hence, these constraints are not enough to recover the 6DoF parameters, and the model regresses the bound box's dimension based on the observation of the invariance of the objects' size in the same category. Thus, as long as the object's category is known, the bounding box dimension can be easily recovered. As the 2D bounding box constraints bring a great advantage to 3D detection in autonomous driving. Li et al. propose GS3D to regress the 3D bounding box. This model disentangles the problem into bounding box and orientation regression problems. As we discussed previously, in autonomous driving, the 3D bound box of a vehicle fits tight in its 2D bounding box. Predicting the orientation of the object

can easily recover its 3D bounding box prior to guidance through its 2D bounding box. Then a post-processing refinement step is applied to get a more accurate 3D bounding box, including a surface extraction module, which assumes that the center of the top plane and bottom plane of the 3D bounding boxes are close to their 2D bounding box. This assumption is also the main factor that leads to the error in real-time inference.

To better solve this problem, Liu et al. [7] propose M3D-RPN, a region proposal network to generate 2D and 3D bounding boxes simultaneously based on Faster R-CNN [29] through using 3D anchors. It defines the 2D and 3D anchors with the size and orientation of the bounding box and uses a depth-aware convolution to extract features with 2D-3D constraints in the camera view.

Pseudo-LiDAR [107] proposed by Wang. et al. argue that the gap between using LiDAR (points) and depth (RGBD) data to train a 3D detection network can be shrunk by converting the depth data into 3D points by the camera's intrinsic matrix. Unlike, previous models, they are based on depth data and treat depth as one of the direct inputs of the model, Pseudo-LiDAR lifts the 2D pixels to 3D. Thus, the depth data can be treated as 3D data by off-the-shelf point cloud models, although the gap in performance between actual LiDAR and Pseudo-LiDAR cannot be eliminated. This research affects a lot of later 6D pose detection models because deriving depth data is always easier and cheaper than LiDAR data. Qian et al. [84] propose an end-to-end Pseudo-LiDar to integrate depth estimation and 3D detection, which is treated separately in previous methods. They point out that the two objectives are commonly aligned well, where the depth estimation will pay attention to the overall information among the image while the detection module needs to distinguish the target objects. Basically, they also point out that the existing

method [14, 52, 58] adopts a grid-based strategy, which discretizes the points into a fixed grid. The tensor will only record the occupation and densities. The advantage of this strategy is the 3D and 2D convolution can be directly applied to those tensors. However, it is difficultly differentiable. The author proposes CoR (Change of Representation modules) to facilitate back-propagation and to maintain the modularity and compatibility of pseudo-LiDAR.

Ma et al. [67] do a deep survey of pseudo-LiDAR and observe that the improvement of pseudo-LiDAR comes from the coordinate transformation, instead of data representation. Based on this theory, a CNN-based 3D detector, Patch-Net is proposed to verify their idea. Patch-Net is mostly similar to pseudo-LiDAR. The difference is that Patch-Net organizes the generated 3D points data as an image where the 3D points are divided into three patches and each patch is one of the x, y, and z values of the points coordinate. By doing so the model gets more promising results compared to the original pseudo-LiDAR and it also proves that the input representation matters in a pseudo-LiDAR network.

On the other hand, Ye [122] et al. propose DA-3Ddet to use domain adaption to solve the real-LiDAR and pseudo-LiDAR gap problem. The model contains three main components, a domain adaptor, foreground segmentation estimator and 3D bounding box predictor. The model consists of two branches. The two branches are identical in structure, the first branch is pre-trained using real LiDAR data and only forwarded during training. The second branch is the pseudo-LiDAR branch. The domain adaption is then performed by calculating an adaption loss between the extracted features from the two branches. Therefore, the pseudo-LiDAR branch will mimic the real-LiDAR's features.

In conclusion, the pseudo-LiDAR theory massively affects not only 3D detection models but 6D pose estimation models. Although, the domain shift problem still remains challenging.

2.3.3 Category-Level 6D pose estimation

As we discussed in the previous section. Although 3D bounding box detection is widely used in autonomous driving and robotics manipulation and achieves high accuracy in predicting the location, it is still an ill-posed problem in pose detection with respect to rotation prediction. In this section, we will discuss a more straightforward method, category-level 6DoF pose estimation. Instead of predicting the pose through its 3D bounding box, 6D pose estimation directly predicts the target object's transformation.

Because, unlike instance-level pose estimation, there's no CAD model available during training and testing. We need to predict 9DoF pose including the three-dimensional size of the object. The main problem in category-level pose detection is to make models to be category-sensitive. In autonomous driving, there's no need to predict the pitch and roll angle, which makes the problem more accessible and can primarily benefit from the 2D proposal. However, as the main application scenario of category-level pose detection is AR and robotic manipulation, the model needs to predict all the dimensions.

Sahin [87] et al. first define the main challenges of category-level 6D pose detection problems including intra-class variations, distribution shifts etc. The authors argue that the basic difference between instance-level and category-level is defecting the distribution of 3D points in training and testing samples. In instance-level detection, the distribution

of the points is the same as the points assigned to the COM (Center of Mass) from both training and testing samples and the model tends to predict the transformation between COM of the target object and the camera coordinate. The objects in category-level pose detection task have different COM and different distributions of points. And the same methods cannot be used to define the 6D pose of unseen objects. Sahin et al. redefine the 6D pose of category-level objects through SSC (Semantically Selected Centers). Typically, for one category, there is only one SSC. Given the skeleton structure for each instance and the distance between those nodes and the COM, the SSC can be computed by finding the repetitive nodes between all instances and applying interpolation to the repetitive nodes. Although the distribution of object coordinates in each instance under the same category is quite different based on COM, the SSC of them maintains consistency across different instances. This definition of SSC inspires many works in category-level 6D pose detection.w

Refinement-based methods

Refinement-based methods are early methods used in deep category-level 6D pose detection problems. They recover the transformation by a post-processing algorithm to align the two coordinates. Commonly, they rely on the Umeyama algorithm [97], ICP (Iterative Closest Point) [102] etc. The first theoretical deep learning refinement-based category-level method is proposed by Wang et al. [103]. In their work, the authors propose a novel way to represent the instance in object coordinate, which is called NOCS (Normalized Object Coordinate Space). NOCS is a unit cube that aligns with the object's center in 3D space. The origin of the NOCS is on one of the cube's corners. To transform the ordinal coordinates in object space to NOCS, one only needs to shift the origin of the object coordinate system to the

corner of the bounding box and scale the x, y, and z axis component values to a range of 0 to 1. In this case, all the coordinates of the object in NOCS are normalized. The purpose of constructing NOCS is to find a unified representation for instances of different sizes, shapes and textures in the same category. Nevertheless, a model that is built upon Mask R-CNN [32] is trained to predict the NOCS, mask and labels only from RGB images. The depth is then used to lift the image with the predicted NOCS and mask to 3D points to recover the 6D pose through the Umeyama algorithm [97]. Wang et al. also publish a benchmark NOCS-REAL275 for category-level 6D pose detection which is comprised of both synthetic data and real data. Their work is an important milestone in deep learning category-level 6D pose detection. The baseline for comparison in this thesis also builds on NOCS-REAL275 dataset.

Tian et al. [96] develop a model to solve the pose problem by using a category shape prior. The category prior is the global characteristics of a category across different instances. In order to extract the category-level shape, Tian et al. train an autoencoder to extract the low-dimension latent embeddings of each instance. Then, the mean of the embeddings will be passed to a decoder to get the mean shape category prior. The output category prior is the point cloud that represents the general shape of the category. The advantage of the shape prior is its corresponds to the category rather than the instance. Once we have the in the shape prior, the model takes the prior, RGB and points as input to infer a deformation field and correspondence matrix. The reconstructed point cloud is equal to the summation of the points in the deformation field and shape prior. The correspondence matrix is used to map each point in the input data and the reconstructed point cloud because the number of points of these two sets may be different. After knowing the

reconstructed points, depth and their correspondence, the 6D pose can be then estimated using the Umeyama algorithm [97].

Regression-based methods

Although the aforementioned refinement-based methods can improve the performance of category-level 6D pose detection. They still need a post-processing refinement step to obtain the final result, which is not efficient. Another solution is to directly regress the transformation, which is proposed by Chen et al [11]. This method learns CASS (Canonical Shape Space) to tackle the intra-class variations. The whole model is comprised of three parts: Canonical Shape Space learning, Pose-dependent Feature Extraction and Pose Estimation. Like the two methods mentioned in the previous section. The model needs to find a proper representation for the category. The authors propose CASS (Canonical Shape Space) in their work the task is separated into two parts, the pose estimation and size estimation. The authors separately learn pose-dependent features and size-dependent features, where the size-dependent features are learned through reconstructing a canonical point cloud, that is pose-normalized and has real-scale size. The size of the model can be derived from this canonical point cloud. Then the other part will directly regress the transformation matrix consisting of rotation and translation from the RGBD input.

CPS++ [69] proposed by Manhardt et al. incorporates self-supervised learning. Similarly, for a category-level method, class representation is needed. The authors use AtlasNet [98] to learn latent space descriptors from the input point cloud, which are then concatenated with points sampled from 2D images to conduct a shape reconstruction

through a decoder. Each category will learn an auto-encoder for class representation. Then, a class-specific predictor will regress the pose-related parameters. The class predictor from the auto-encoder of the first step also encodes the features to reconstruct its shape, which can give additional supervision to the pose prediction. However, the training of this model needs annotation, that are time-consuming to collect, and hence developing a self-supervised scheme will be a better choice. The authors proposed a self-supervised learning method to solve the problem by computing the chamfer distance between the predicted point cloud and the sensor point cloud.

Improving the supervision of 6D pose is commonly used in category-level detection. Lin et al. [59] propose a totally different idea. They propose an end-to-end pose predictor with a parallel structure consisting of explicit and implicit decoders, to improve the supervision of pose. The input image will first be processed by MaskRCNN [32] to extract the ROI and then the input is separated into two streams, which are points and RGB values. Given these two streams, the pose encoder is applied to learn the pose-sensitive features of them respectively. It fuses the features of the two streams by spherical convolutions [17]. The features output from the encoder are fed into an explicit decoder and an implicit decoder simultaneously, where the explicit decoder uses features to regress the pose parameters directly and the implicit decoder reconstructs a canonical point cloud instead. The two predicted results from explicit and implicit decoders are used to refine the final pose by forcing their pose consistency. In this case, the pose prediction can benefit from extra constraints.

Keypoints-based methods

Like instance-level methods, keypoints are also very important in category-level methods. MobilePose [40] is a lite model developed for AR in mobile phones. Unlike previously mentioned models, which take RGBD data as input. MobilePose only uses RGB to infer the 6D pose of objects. MobilePose uses MobileNetv2 [88] as the backbone to build the encoder. In MobilePose, the first step is to do instance segmentation to find the foreground with specific categories. The authors conduct segmentation by assigning anchor grids to the image and use the model to predict the Gaussian distribution for each object to find peaks. MobilePose add an intermediate layer to MobileNetv2 for shape supervision. The shape is comprised of 3 channel coordinates and 1 channel segmentation. However, a CAD model is required to render object coordinates that are assumed to be partially available in the training data. During training, the model will skip computing the loss if the CAD model is not available. After knowing the peak locations of the objects in the 2D image, the center and 2D bounding box vertices can be approximated. EPnP [55] is applied to recover the 3D bounding box.

2.3.4 Summary

The instance-level and category-level pose detection models focus on single-image pose estimation with respect to the camera space. Given a video, a 6D pose detection model will estimate the transformation from camera space to object space regardless of the continuity of the pose in the real scene. Compared to single-image pose detection, pose tracking that works on a frame sequence is appropriate in real scenarios and has the potential to

improve the estimation. Hence, pose tracking is proposed to tackle sequential data. Unlike 6D detection models, the tracking models predict the relative pose between two consecutive frames. Given the initial pose of the object in the first frame, the model can iteratively recover the poses in each frame. Additionally, the temporal information in the tracking model is significant In the next section, we will introduce instance-level and category-level tracking models.

2.4 Instance-level Object Pose Tracking

From this section on, we will discuss the monocular pose tracking problem, which is also the topic of our work. For the cases with and without CAD models, we divide the methods into instance-level pose tracking and category-level pose tracking, which is similar to pose detection except the prediction is based on the inter-frame transformation.

2.4.1 Definitions

Given a sequence of frames and the initial pose of the target object with its CAD model, Instance-level poses tracking requires the prediction of the 6DoF pose ($R \in SO(3)$ and $T \in R^3$) of the object in each frame. Because the CAD model is available in training and testing, the size of the object is known.

2.4.2 Regression-based methods

Like the regression-based methods, we discussed in the previous section on pose detection. In the pose tracking problem, the relative pose between two frames will be directly regressed. Garon et al. [28] propose Deep 6-DoF to formulate the instance-level tracking problem. Like many other tracking methods, the model takes the reference frame's object position and current frame as input to predict the actual pose of the object. In order to generate the observed position, the authors incorporate an augmented strategy, where the current frame's observed position is generated by applying random rendering and transformation to the predicted pose in the previous frame. The model takes the predicted position in the previous frame and the observed object in the current frame as input through a multi-head CNN network to regress the relative transformation between these two frames. The authors also prove the robustness of their tracking model encountering occlusions. However, this method does not consider the interaction between the object and its environment. Besides, this method lacks a specific occlusion solution. Marougkas et al. [71] propose a multi-attention framework to address background occlusions and clusters. The authors demolish the problem into foreground attention and occlusion with Fire layers [43] and skip-connections. The purpose of such a multi-attention architecture is to help the model learn to distinguish occlusions from the foreground. FCR-TrackNet proposed by Zhu et al. [130] incorporates multi-feature Fusion and Classification-Regression to predict the rotation information. Similarly, like other instance-level methods, it requires the 3D model of the target object to render the reference frame and estimate the relative pose represented by Lie Algebra between the current frame and the reference frame. The input

data (current frame and rendered previous frame) will be fed into low-level and high-level feature extraction modules to learn detailed and semantic information respectively. These features will be fused to regress the parameters of Lie Algebra representation. Additionally, a classification module is added as a sub-branch to the rotation regression. The authors proposed a novel rotation classification strategy, CRA (Classification label for Rotation Angles). The CRA classifies the rotation angle into different ranges and converts the discrete angle space into a smooth coding label by dividing the rotation angle into multiple areas. The CRA proposed in FCR-TrackNet takes the range of rotation angle into account to solve the boundary problem in rotation estimation.

2.4.3 Optimization-based methods

Wen et al. proposed $se(3)$-TrackNet [111], which estimates the relative pose given the RGBD image of the current frame and the synthetic image rendered using the optimal pose estimated from the previous frame and the CAD model of the object. The rendering of the previous frame and the RGBD of the current will be passed to two encoders respectively. The model will learn the residual pose transforms represented by Lie Algebra given a proper loss function. This method only relies on the synthetic data during training and can be easily transferred to real scenes. However, the fact that it requires the CAD model to render is its drawback in practice. PoseRBPF [21] utilizes Rao–Blackwellized Particle Filter and autoencoder network to estimate rotation and translation separately. The particle in PoseRBPF is represented by a translation hypothesis and rotation distribution conditioned on the translation hypothesis. The particles are fed to a motion model to determine the

RoI (Region of Interest) according to the translation. The autoencoder is then used to compute a feature embedding from the RoI. Then, the authors match the embeddings with the embeddings in the pre-computed codebook to derive the observation likelihood and calculate the weights of the particles. Finally, through resampling the particles using the weights, the translation and the probability of the rotation can be computed.

2.4.4 Refinement-based methods

Unlike regression-based methods, refinement-based methods add a post-procession step in the end to refine the predicted or prior pose. Manhardt et al. [68] propose an RGB-based model to tackle the tracking problem. The method proposed does not require depth or segmentation. The model is trained to infer the transformation updates between two patches with a 6D pose hypothesis. The goal of the network is to refine this hypothesis till it fits the real transformation. Specifically, given a CAD model, the authors render it with a random transformation to a random background to get a scene patch. Then a perturbation pose (hypothesis) is applied to the scene patch to get a render patch. Therefore, the hypothesis is no longer aligned well with the scene patch. The task is then the refinement of the hypothesis. In order to optimize the increment of rotation, the authors use quaternions to define a regression problem. Meanwhile, a contours-based loss is proposed to tackle the invariance to object's shape and the lack of depth information, which degenerates the 3D alignment problem into a 2D matching problem.

Li et al. [57] propose DeepIM, a 6D pose matching network. The authors argue that the direct regression of object pose may produce poor results. Matching between rendered

images and observed images has a better performance. The authors adopt a high-resolution zoom-in to the image and segmentation map before feeding it to the model. Given the observed image and its corresponding initial pose estimation. DeepIM uses FlowNetSimple [23] to iteratively refine the pose by matching the rendered image and the observed image. Because the objects are observed originally in the camera coordinate system. Scale and rotation-dependent translation variation change may appear, which requires the model to memorize each object's size when matching mismatched pairs. In this condition, untangled transformation representation is presented to tackle this problem. Basically, the prediction of rotation and translation needs to be separated. Therefore, the rotation estimation needs to be based on the object coordinate system and the axes are parallel to the camera's coordinate system. This solves the problems of object size memorization and dependent rotation estimation. Insufficient real training data is also a challenge of 6D pose tracking problems, and learning from synthetic data is the main tendency.

Additionally, Busam et al. reformulate the instance-level tracking problem into an action decision process by incrementally updating the initial pose. First, the training pair is acquired by randomly rendering the input object with different poses. The goal is to iteratively update one of the objects to be close to the other through action. The possible actions can be discretized into an action vector, which is composed of 6 positive actions, 6 negative actions and a stop sign. The problem now becomes finding an optimal action sequence. The authors use a CNN with an attention module and Hamming distance loss to learning the best action sequence. This action decision model provides a new direction for pose tracking. Instead of considering the transformation as an entity comprised of rotation and translation, the discretized action can also represent the pose refinement process.

2.5 Category-level object pose tracking

Like category-level object detection, the biggest challenge of category-level pose tracking is to learn scale-invariant and shape-invariant features of the categories with different input instances. While the CAD model is not given, the model can only learn from the point cloud without specific prior knowledge, which makes the task more complicated. But it also can expand the category-level tracking model to more tasks than instance-level. This is the reason why we are interested in it. Our work is also built on the category-level object tracking method. In the following sections, we will introduce the related works of several state-of-the-art category-level tracking methods.

2.5.1 Definition

In category-level tracking problem. The CAD model is not available. Thus, the neural network is not able to get any prior of the target object's shape and size. Like category-level detection, learning the category-level features is the most important thing. The initial pose and the bounding box of the object are given, the size prediction is not strictly necessary except if it is a non-rigid object or non-articulated object.

2.5.2 Detection-based methods

In most scenes of autonomous driving is only one category (vehicle) and the orientation prediction is simply with only one degree of freedom. Detection-based methods that are inspired by 2D tracking models are often used. Hu et al. [41] propose a tracking method

for autonomous driving. The task here is defined as a multi-object tracking problem. The model aims at finding N trajectories for the objects detected in the image. Because the targets are all vehicles, predicting category-wise features becomes less important. In autonomous driving, the rotation can be represented by a single angle Θ. Therefore, the status of each vehicle can be represented by the object center, orientation and appearance features for re-identification. Because the GPS information is available, the ego view of the vehicles is known during training. The authors design a weighted bipartite matching based on affinity matrices between existing tracks and the new detection and matching with depth-ordering matching to solve the re-identification problem. LSTM (Long Short Term Memory) [38] is used to introduce temporal relations to each detection in sequential frames. Similarly, the 3D tracker proposed by Weng et al. [114] adopts the idea from deep multi-object tracking in 2D computer vision using a Kalman filter [109] and the Hungarian algorithm [53] for motion prediction and data association respectively. The Kalman filter is used to update the parameterized 3D bounding box from different frames. It also provides new metrics for 3D tracking evaluations.

2.5.3 Regression-based methods

On the other hand, in AR applications with diverse categories and the need of 6DoF transformation, Regression-based and keypoints-based methods are more appropriate. They are also the baselines that we evaluate in our experiments. So we will give a detailed introduction to these methods. Captra proposed by Weng et al. [115] takes both rigid and articulated objects into account and treats the tracking problem as a 9DoF regression. The

9DoF requires the model to predict the size of the object in each frame because the target objects include articulated object whose size may change during the tracking. Captra adopts the idea from NOCS [103] and proposes a pose-canonicalized space. Unlike NOCS, the pose-canonicalized space uses the object's pose in the previous frame to normalize the current frame's object. Suppose X_i are the points in frame i, R_i and T_i are the rotation and translation in frame i, respectively. S_i is the scale of the object that can be calculated by $S_i = \sqrt{W^2 + H^2 + L^2}$, where the (W, H, L) are the width, height, length of the 3D bounding box. S_i is the length of 3D bounding box's diagonal. Thus, the points in frame $i+1$ can be canonicalized to $Z_{i+1} = R_i^{-1}(X_{i+1} - T_i)/S_i$. The formula normalizes the current frame's points by applying the inverse transformation of the previous frame. Assuming the change of size between two consecutive frames is minor, the size of the object is approximated using the previous frame's size. The model takes Z_{i+1} as input to regress $\overline{R_{i+1}}, \overline{T_{i+1}}, \overline{S_{i+1}}$ to satisfy

$$R_{i+1} = R_i \overline{R_{i+1}} \tag{2.18}$$

$$T_{i+1} = S_i R_i \overline{T_{i+1}} + T_i \tag{2.19}$$

$$S_{i+1} = S_i \overline{S_{i+1}} \tag{2.20}$$

Noticeably, the output from the model are the relative transformations $\overline{R_{i+1}}, \overline{T_{i+1}}, \overline{S_{i+1}}$ between two frames. The difference is that Captra normalizes all the input points to a pose-canonicalized space using the previous frame. In this case, the transformation will be easier to regress. The model's architecture is shown in Fig. 2.1.

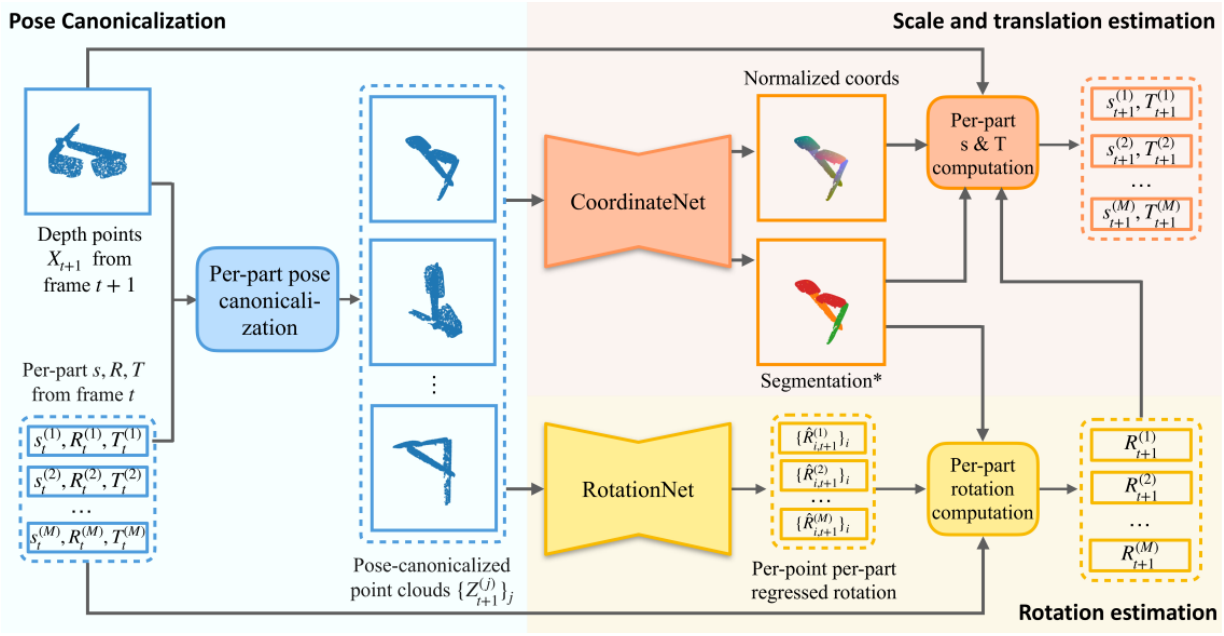

Figure 2.1: Captra architecture [115]. ©Copyright IEEE 2021.

The model is comprised of two sub-modules, namely CoordinateNet and RotationNet. The former is responsible for segmentation and normalized coordinates prediction. The normalized coordinates are similar to the pose-canonicalized space except the transformation (R_i, T_i, S_i) is replaced by $(R_{i+1}, T_{i+1}, S_{i+1})$. The purpose of predicting normalized coordinates is to use the Umeyama algorithm [97] to compute the translation along with RANSAC. The latter is built upon PointNet++ [83] for rotation prediction. Considering articulated objects, the input is their masked per-part points, pose, rotation and translation under pose-canonicalized space. Unlike other methods using Lie Algebra [25] or Euler angles [108], RotationNet uses a neural network continuous representation [129] for rotation.

The training and testing strategies are different. In training, the pose-canonicalized

coordinates are transformed using a random rotation, translation and scale instead of the real previous frame's pose because the majority of data in the dataset [103] that is used in Captra are synthetic and using random values can augment the data. Suppose (n_s, n_r, n_t) are Gaussian random noise. The pose after the perturbation is $\{S' = S(1 + n_s), R' = Rn_r, T' = T + n_t\}$. The (S', R', T') is ground truth in training phase. The previous frame's pose is only used in testing. Generally, Captra uses the previous frame's pose and size to normalize the current frame's points, which formulates the pose change between two frames in a normalized space to tackle the scale and pose variation.

2.5.4 Keypoints-based method

On the contrary, keypoints-based methods focus on representing the object as keypoints. CenterPoseTrack proposed by Lin et al. [61] is a single-stage model that takes monocular RGB video as input to predict the bounding cuboid and 6-DoF pose. The keypoints in this method are defined as the vertices of the 3D bounding box. The final pose is calculated using PnP (Perspective n Points). Given the 2D image of the current frame and the previous frame along with the center and keypoints heatmap of the previous frame, a neural network will be used to generate a heatmap to find the center and keypoints of the current object. The network is also trained to generate the offset vectors, which indicate the offset from the previous keypoint locations to current locations. After predicting the heatmaps, the Bayesian and Kalman filters are used to update the 2D keypoint locations as well as the relative cuboid dimensions. Finally, PnP is used to process the 2D keypoints to obtain the 6D pose. The advantage of this method is that it does not require 3D

data as input and directly predicts the distribution of keypoints in 2D image coordinates. Incorporating uncertainty estimation allows this method to work on previously unseen object instances. CatTrack proposed by Yu et al. [124] utilizes vision transformer [22] to remove the irrelevant information during the tracking process and emphasize useful features. Like CenterPoseTrack [61], the purpose of this method is to leverage the heatmap of keypoints from the previous frame and a 2D image of the current frame to produce the heatmap of the current frame. Then, using the PnP algorithm to recover the pose of the object. The model takes the heatmaps of the previous frame and the 2D image of the current frame as input and conducts visual self-attention on them to learn the heatmaps of object centroid and keypoints (the vertices of the 3D bounding box). One drawback of this method is when the objects are close to each other, the keypoint matching and grouping might fail. 6-pack [100] proposed by Wang. et al. address the tracking in the scenario of robotic manipulations. Due to the great workload of annotating data with handcrafted keypoints, in 6-pack, the network predicts the keypoints with respect to the input point cloud in an unsupervised manner based on an anchor-based attention module. 6-pack is a rigid-object-oriented 6D tracker. So the size of the object is assumed to be the same between frames. There is no need to predict the object's size. This model is also one of the baselines that we use for evaluation. Fig. 2.2 is the architecture of 6-pack.

First, the previous frame's pose or initial pose is used as an approximation of the current frame's pose by the model. The purpose is to find the ROI of the object. Given the bounding box of the previous frame, the model can crop out an enlarged 2D bounding box according to the projection of the 3D bounding box. The purpose of enlarging the 2D bounding box is to cover as much of the points as possible. Because the authors use a ran-

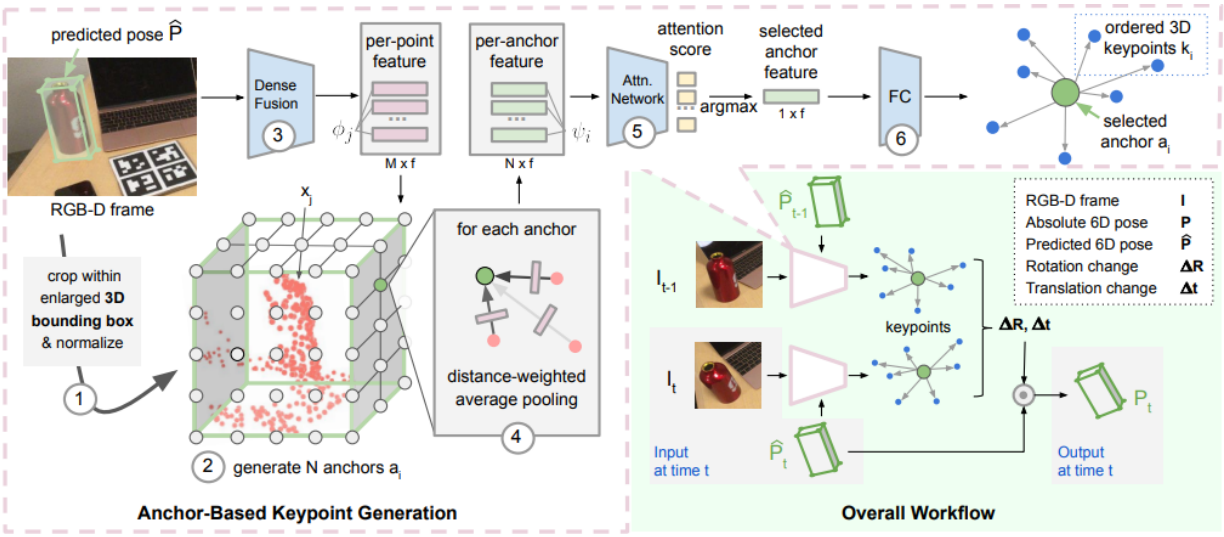

Figure 2.2: 6-Pack architecture [100]. ©Copyright IEEE 2020.

dom sampling strategy, an enlarged bounding box can increase the model's robustness. In addition, anchor points are set to the surface and inside of the 3D bounding box uniformly. Then, sampled points and their corresponding 2D pixels are fused using a per-point fusion method [101] to extract a per-point feature. In order to find the anchor that is the closest to the object's center, the anchors collect all the points' features weighted by their distance to the anchors. After that, the model is trained to find the anchor that is the closest to the object's center. The central anchor features will generate keypoints as the representation of the object. To let the model learn to generate the keypoints in an unsupervised manner, the authors use a multi-view consistency loss as the constraint.

$$L_{mvc} = \frac{1}{K} \sum \| k_i^t - [\Delta R_t^{gt} \mid \Delta t_t^{gt}] k_i^{t-1} \| \tag{2.21}$$

As this is the first category-level pose tracking method applied in robotic manipulation.

It provides a new idea for using keypoints, which provides inspiration for our works.

2.5.5 Summary

In this section, we have discussed instance-level and category-level pose tracking problems from different aspects including regression, indirect post-refinement and keypoints generation. There is one characteristic that is in common among these methods. They make predictions between only two frames to accomplish the tracking task. The task is however not about offline video processing, where the whole video is available during inference as e.g., video detection, or action recognition in 2D computer vision. In real-time processing, only the historic frames can be stored and used. Given the fact that the input is a real-time frame sequence, much valuable information is hidden in the history. The historic pose can be used to refine the pose of the current frame. In the next section, we will introduce the temporal models that inspire us in 2D and 3D.

2.6 Temporal Models

As we discussed in Sec. 2.4. Considering the drawbacks of 6D pose detection, pose tracking by relative pose prediction between two consecutive frames is more feasible in processing real-time video sequences. Thus, in video processing methods, historical information cannot be ignored. In 2D object detection, action detection, and video object detection. The temporal model has been deeply explored. However, the temporal model is challenging to migrate to 3D pose prediction problems because of the difference in data representation.

Category-level pose tracking still lacks a proper temporal method to process the history. This is why we focus on it. In this section, we will start with 2D computer vision to introduce the application of temporal information in both 2D and 3D.

2.6.1 2D Temporal Models

Wang et al. [104] propose TDN (Temporal Difference Networks) for action recognition in video, which incorporates short-term and long-term motion captures. Like what we propose in this thesis, TDN is based on a temporal difference operator. The network consists of three components including video sparse sampling, short-term motion and long-term motion. Considering the limit of GPU memory, the authors use a sparse video sampling method to organize the input data. For each video, a non-overlapping segment strategy is applied and a frame is randomly sampled for each segmentation. Then, the sampled frames are fed into a CNN to extract frame-wise features for a short-term process. The short-term difference is applied at the early stage. The temporal difference calculates the RGB difference between two adjacent frames and differences of the all the frames will be stacked to produce short-term motion. They use a low-resolution processing strategy. Because the authors observe that the difference has a high value only in the motion of salient regions, where high resolution is not necessary. In order to add constraints to the model and to maintain the sensitivity to motion, the authors adopt a long-term module to further process the features. If the short-term module concentrates the spatial feature between two frames, the long-term module aims at using a cross-segment structure to improve the feature representation. Fig. 2.3 is the architecture of TDN. The long-term structure is

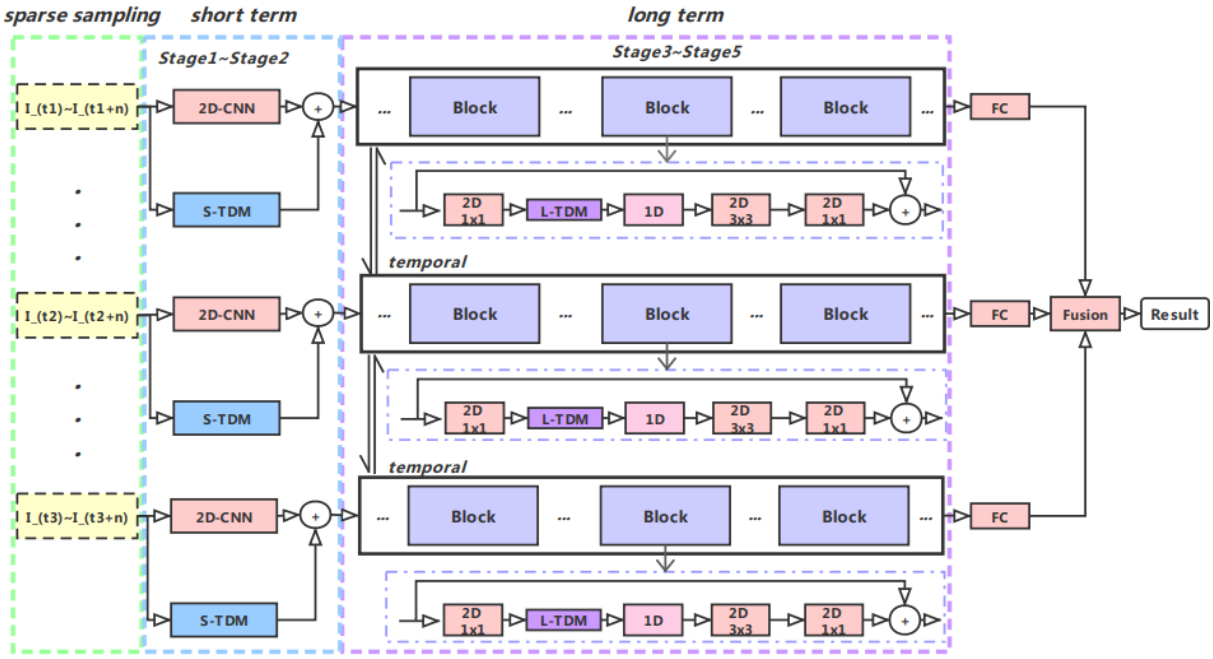

Figure 2.3: TDN architecture [104]. ©Copyright IEEE 2021.

comprised of multiple sub-modules, which correspond to one short-term module. The long-term structure formulates a multi-scale attention map on the different segments to enhance frame-level features. The design of our global attention map in this thesis is inspired by this idea. Please see Chapter 3 for more details.

In 2D video object detection, a single frame's features may be distorted encountering occlusion, motion blur, and illumination change. It is important to explore temporal information to enhance feature representation. Cui et al. [20] 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. The model will first sample a frame and its neighbouring frames in a video as the input. The feature relation module aims at learning a pixel-wise adaptive weight between two neigh-

bouring frames. The authors argue that the global weight used in existing methods makes the model incapable of suppressing the occlusion or other objects in the background. A pixel-wise weight is more suitable for the model to focus on the target object. The temporal relation module is a mini-network comprised of several convolution layers. The input is the concatenation of the frames. The output map of the mini-network is a 2D map that depicts the corresponding weight for each pixel according to their similarity. If the weighted similarity between the original frame and the one is greater than a threshold, the weight map will be set to 0. This step increases the model's efficiency by suppressing features of repetitive frames and makes the model concentrate on the feature difference. Nevertheless, the authors also adopt a feature adjustment module to maintain the feature consistency around in neighbouring frames. In the feature adjustment module, the model calculates a weighted summation between each frame and other frames in the neighbouring set using the weight map from the feature relation module. The feature blender is the last module to connect feature adjustment and feature relation, which is the weighted element-wise multiplication between the current frame and other adjusted frames in the neighbouring set. Fig. 2.4 is the overall architecture of TF-Blender.

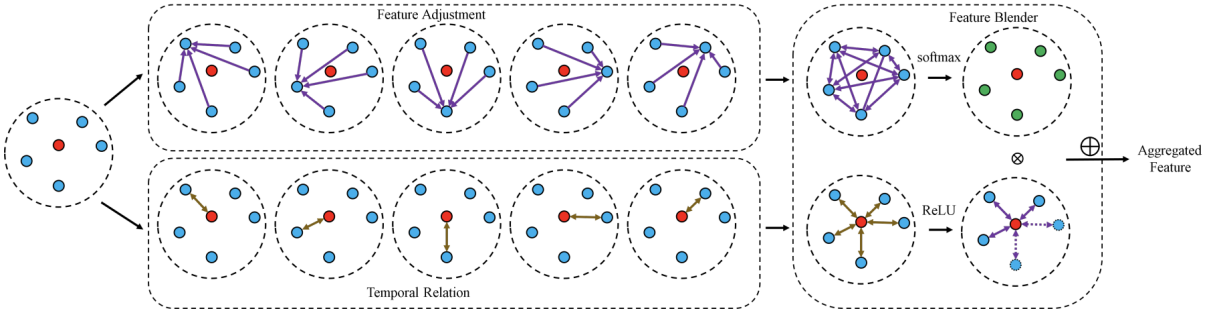

Figure 2.4: TF-Blender architecture [20]. ©Copyright IEEE 2021.

2.6.2 3D Temporal Models

It is more difficult for 3D models to leverage temporal information than for 2D models because the relation between different point clouds is hard to measure. Using a memory pool to store unordered frames is a common method to use 3D temporal information. TP-AE proposed by Zheng et al. [128] incorporates an auto-encoder and continuous temporal frames. The authors first use the history buffer to learn the prior pose of the current frame by extracting the trajectory in the history frames. Then, the prior pose will assist the downstream modules. The image and points will be sampled in an enlarged region and the prior pose is used to transform the sampled points. By doing so, an RGB-Cloud pair with the offset of the prior pose to the ground truth pose is generated. An auto-encoder-based strategy is used to address the problem of rotation estimation. The model will first learn the latent embeddings of rotations invariant to the translation by reconstructing the ground truth RGB-Cloud pairs, where the target objects are located in the center of the image. The embeddings of all rotations will be collected into a codebook for matching during inference. The output latent embeddings with the highest similarity score will be assigned with the corresponding rotation. Similarly, in translation prediction, the authors also build a codebook by reconstruction using an auto-encoder. The embeddings from rotation prediction are concatenated as one of the inputs for translation prediction because the authors observe that the rotation may provide some useful information for translation estimation. This method incorporates an auto-encoder to tackle the challenge of occlusions. However, the prior pose prediction using history frames lacks the consideration of the case that the pose of the object is in fast motion, where the error of prior pose estimation will be

enlarged. FoundationPose [113] proposes a novel setup for 6D temporal tracking using 3D models or model-free reference images combining the idea from Nerf [74]. In the absence of 3D models, the model uses a bunch of reference images to learn the shape of the object to effectively render images with sufficient quality for downstream modules. To this end, the authors propose an object-centric neural SDF (signed distance field) representation for object modeling. The object is represented using two functions, geometry function and appearance function. The geometry function intends to learn the signed distance from the 3D points and the appearance function takes the intermediate feature vector from the geometry module, a point normal, and a view direction to output the color. Then the object surface can be obtained by taking the zero-level set of the signed distance field. In addition to RGB rendering in Nerf [74], the model requires depth to do the pose estimation. The authors further perform marching cubes [65] to extract a mesh from the zero-level set of the SDF. After having the shape information from the 3D CAD models or the aforementioned neural field learning, the pose hypothesis can be generated through two steps, pose initialization and pose refinement. The model initializes rotation and translation by viewpoint sampling from an icosphere. Then, the model will refine the initial pose using the rendering of the object conditioned on the coarse pose and a crop of the input observation from the camera to refine the initial pose. Finally, a pose selection strategy will be used to select the optimal pose from those pose hypotheses. This method bridges the gap between model-based and model-free methods. It uses a neural implicit representation for effective novel view synthesis. Wen et al. [110] propose BundleTrack, a 6D pose tracking model that is generalizable to unseen objects without using any CAD model or prior knowledge of the object based on an object pose graph.

Unlike previously discussed 6-Pack [100] and Captra [115], it does not require training on each specific category and it can store previous poses to refine current frame's pose prediction. Fig. 2.5 is the architecture of BundleTrack.

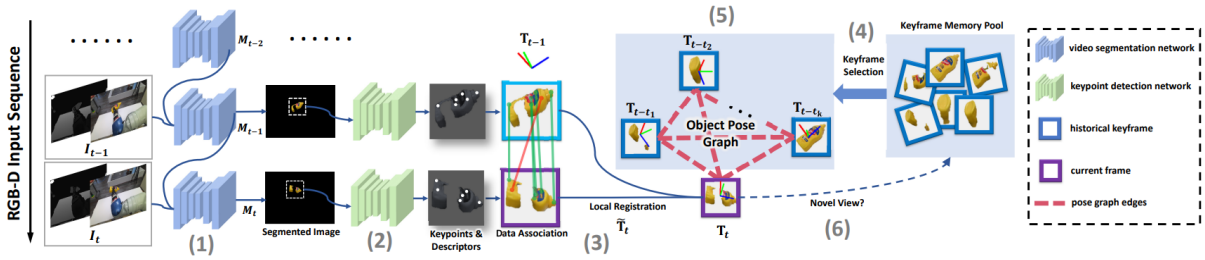

Figure 2.5: BundleTrack architecture [110]. ©Copyright IEEE 2021

The model is a recurrent structure. In each timestamp, the input is the currently observed RGBD image I_t and the previous frame's segmentation map M_{t-1}. Then the off-the-shelf segmentation network [126] will predict the corresponding segmentation. After cropping out the object in two frames, a keypoints detector will predict the keypoints and feature descriptors. The estimated keypoints are associated using RANSAC [27]. Based on these two keypoint sets, an initial coarse transformation can be calculated by solving a least square problem [3], which initializes the current node in the pose graph (step (5) in Fig. 2.5). The initialized pose T_t is refined using the selected keyframes in the memory pool. The strategy of frame selection is to find the set of frames that has the largest mutual viewing overlap, which can maintain the multi-view consistency. Supposed the size of the memory pool is N, the authors formulate the problem as a minimum H-subgraph of an

edge-weighted graph problem [99]:

$$\arg\min_{x} \sum_{i \in N} \sum_{j \in N} x_i x_j arccos \frac{tr(R_i^T R_j) - 1}{2} \tag{2.22}$$

where $x_i \in 0, 1, i \in N$ and R_i is the rotation matrix of keyframe i. The purpose of the optimization is to find a binary vector $x \in R^N$ that indicates the selection of frames in the memory pool.

After selecting the keyframes, the next step is to use those keyframes to optimize the pose of the current frame. As shown in Fig. 2.5, the authors propose an object pose graph to conduct the optimization. In the pose graph, each node is the pose of the corresponding frame. For each edge in the graph, two types of energies E_f and E_g are introduced. The E_f corresponds to the residuals between features and E_g corresponds to the geometric residuals measured by dense pixel-wise point-to-plane distance. The spatiotemporal consistency can be achieved by minimizing the total energy. The pose graph can be denoted as $G = V, E$. The optimization equation is as follows:

$$E = \sum_{i \in |V|} \sum_{j \in |V|} (\lambda_1 E_f(i, j) + \lambda_2 E_g(i, j)) \tag{2.23}$$

The strategy to update the keyframe memory pool is by comparing the geodesic distance between the currently determined pose and each existing keyframe in the memory pool. The pose is added to the pool if the distances are larger than a threshold. BundleTrack provides a novel direction for using optimization methods to align temporal information in pose tracking. The model does not require any prior knowledge of the category or instance.

2.6.3 Summary

In this section, we introduce the temporal methods in both 2D and 3D that provide ideas for our thesis. In 6DoF poses tracking problem, the TP-AE [128], FoundationPose [113], BundleTrack [110] and its improved version BundleSDF [112] proposed by Wen et al. are typical temporal models that are used in 6D pose tracking problem. However, the temporal information in most of their methods is unordered and acts like a bunch of reference frames, where the continuity of consecutive frames is ignored. The TP-AE [128] is the only one that uses continuous temporal information, which has a similar setting to our method. We think the learning-based temporal model in 2D computer vision can be generalized to 6D tracking tasks. Hence we adopt the idea of the temporal difference model in 2D combined with the bidirectional feature fusion between pixels and points to propose a difference-based model for 6DoF pose tracking.

2.7 Benchmarks

In this section, we will introduce some commonly used datasets in 6DoF pose detection and tracking, which include the two datasets that we used to evaluate our model.

2.7.1 NOCS-REAL275 [103]

The NOCS-REAL275 dataset is a category-level dataset used for pose detection and pose tracking. It consists of synthetic and real data. For synthetic data, 184 instances from ShapeNetCore [10] under 6 categories are rendered with 31 indoor backgrounds. For real

data, there are about 8000 RGBD frames from 18 real scenes (7 for training, 5 for validation, and 6 for testing). The real objects are classified into 3 instances for each category. NOCS-REAL275 dataset also contains the NOCS coordinate space for each object for training and testing. Fig. 2.6. The reason why we use NOCS-REAL275 dataset to evaluate our

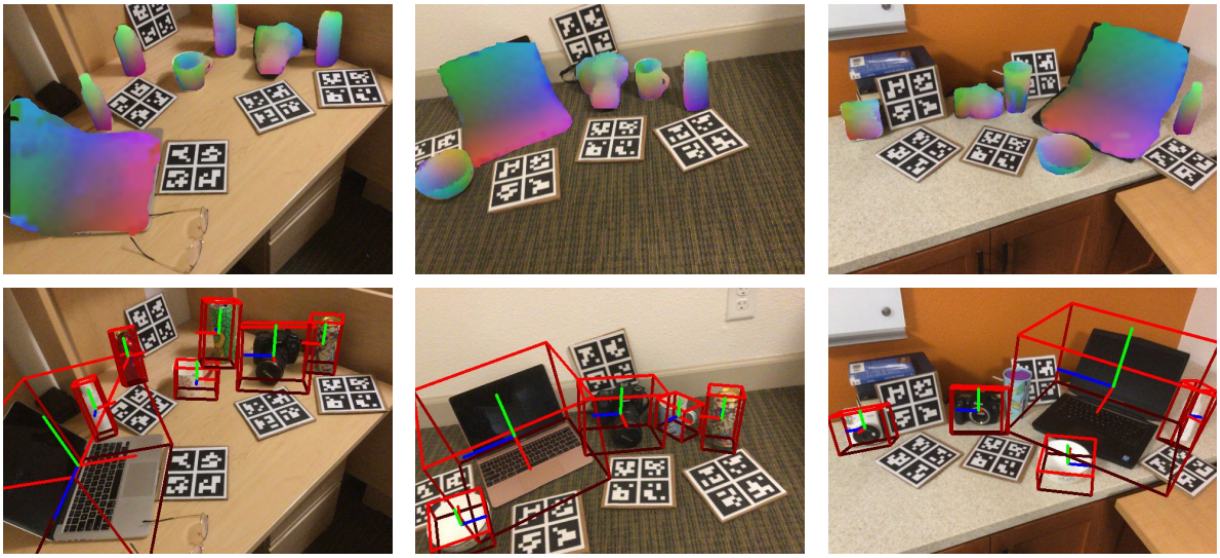

Figure 2.6: NOCS-REAL275 samples [103]. ©Copyright IEEE 2019

model is because it has a great number of instances and has them well organized into 6 categories, which is suitable for category-level 6D pose tacking. It contains real data, which consist of videos. The real data can be used to train and evaluate our temporal model.

2.7.2 MoVi [30]

MoVi is a synthetic dataset generated using Kubric [30]. MoVi contains 5 subsets (Movi-A - MoVi-E) with increasing complexity. The dataset consists of up to 24 categories, each category has up to 1000 videos. Each video consists of 24 frames with 12 FPS (Frame

per Second). The advantage of this dataset is that the pose of the object in each frame is continuous, which is suitable for temporal model training and it has a great amount of training data. Unlike other datasets, the object instances in MoVi are falling to the ground with a moving camera and occluded frequently. It makes the detection tasks more challenging than the static objects placed on the ground with a moving camera. The dataset provides the annotation of optical flow, segmentation, 6D pose, surface normals, depth and bounding boxes. Fig. 2.7 is an example of MoVi. The frames in this dataset are all organized as video, which is appropriate for our temporal model's training. Due to its sufficient training data, we can better analyze the performance of our model. Nevertheless, the objects in the scenes in the MoVi dataset have more complex motions compared to NOCS-REAL275, which allows us to evaluate the model's robustness when encountering occlusions and rapid pose change.

2.8 Summary

In this Chapter. We reviewed many related works in the field of pose prediction. We start with pose detection and 3D bounding box detection to introduce the basic theory and we classify the pose-related works into instance-level and category-level depending on the availability of the CAD model. Then, instance-level and category-level are further categorized into refinement-based, retrieval-based, keypoints-based and detection-based methods and we give details of some state-of-the-art methods. We analyze the limit of pose detection of a single frame in an real scene. Pose tracking that predicts the relative pose between the current frame and previous frames is applicable to tracking the pose

of the object in a real-time sequence. Captra [115] and 6-Pack [100] are two successful tracking models. Captra is a regression-based method that only takes points as input, while 6-Pack is a keypoints-based method that takes both RGB and points as inputs. Temporal processing is an inevitable topic in video-related tasks. Although many pose tracking methods have achieved state-of-the-art, they still suffer from the loss of important features caused by occlusion, motion blur etc. These problems can be solved by temporal processing, which has been widely applied in 2D computer vision. Inspired by the idea of temporal difference and bidirectional feature fusion we intend to transfer the difference-based model in 2D action recognition to 6D pose tracking. In the following sections, we will explicitly introduce our difference-based temporal model and evaluate its performance.

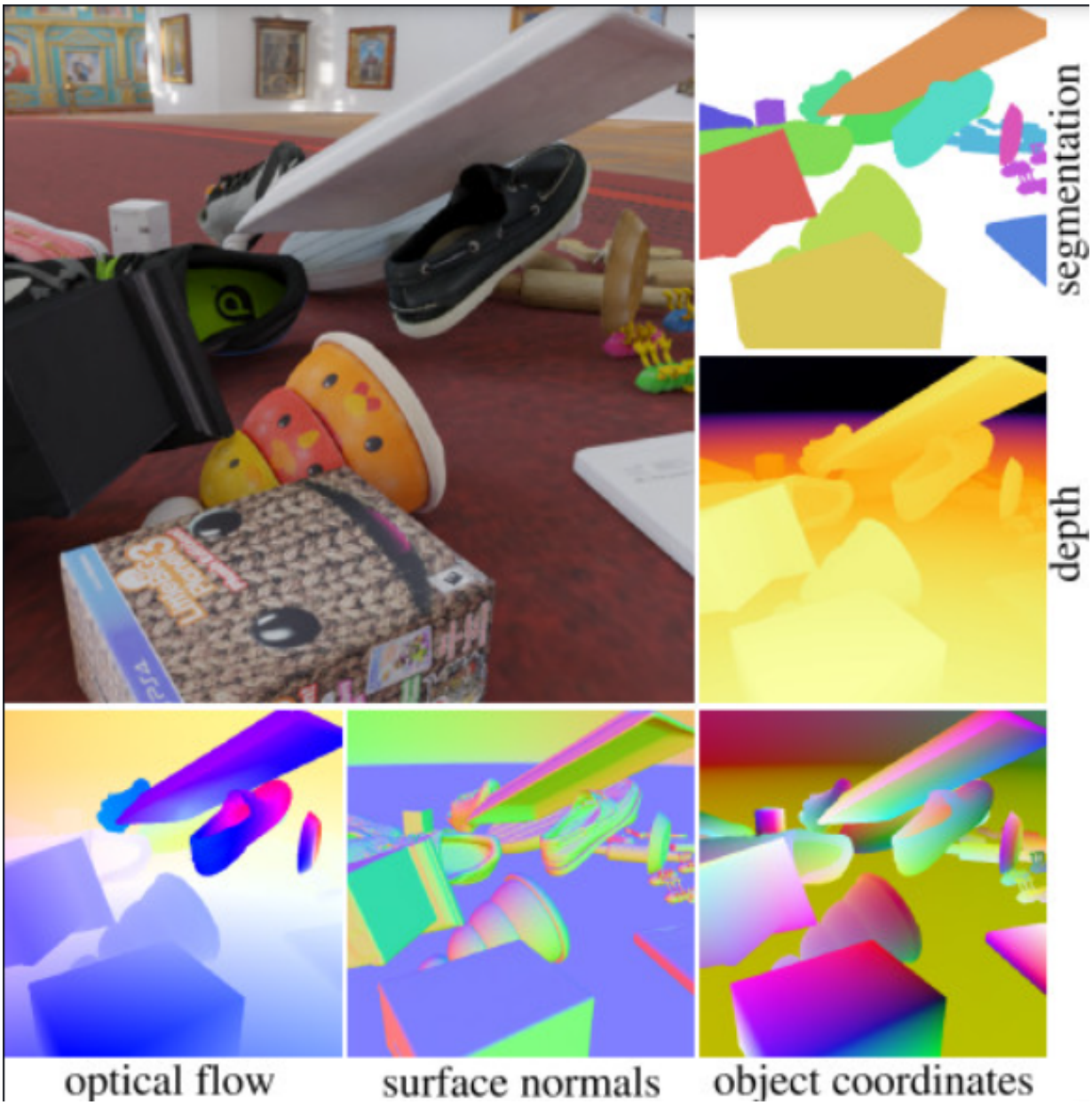

Figure 2.7: MoVi samples [30]. ©Copyright IEEE 2022.

Chapter 3

Approach

In this chapter, we will introduce our difference-based temporal model. First, we will give a definition of the temporal model in category-level 6DoF tracking. Second, we will go through the overall architecture of our model. Then, we analyze each component in our model and how to integrate our temporal model with the two backbones 6-Pack [100] and Captra [115].

3.1 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 previous frame pose 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 $p_t = \Delta p_t \cdot \Delta p_{t-1} \cdot \Delta p_{t-2} \cdot \Delta p_{t-3} \cdot \Delta p_{t-4} \cdots p_0$. Given the RGBD image I_i, the tracking task can be expressed as $p_t = F(I^t, I^{t-1}, I^{t-2}, I^{t-3}, \cdot \cdot \cdot, I^0, C^{t-1}, C^{t-2}, C^{t-3}, \cdots, C^0, p_0 \mid \theta)$, 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 $\Delta p_t = H(I^t, C^t, I^{t-1}, C^{t-1} \mid \theta)$. Considering temporal information, there are a lot of ways to incorporate it into the tracking model. As we discussed in Chapter 2, using an unordered memory pool [110,112] is one of the options. In this thesis, the model is required to run in a real-time manner, where the whole video is not available and past frames can be utilized. We propose to use a sliding window (history sequence) to store and update history frames sequentially. Supposed 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 $\Delta p_t = H(I^t, I^{t-1}, C^t, C^{t-1}, \mathcal{M}^t \mid \theta)$.

3.2 Overview

As discussed in Sec. 3.1. The direct pixel-wise or feature-wise difference calculation is not applicable for 3D points. Additionally, the differences of points, RGB values of pixels and their fusion need to be considered. The proposed difference-based temporal model explores the relations between the current frame I^t and the frames in the sliding window $\mathcal{M}^t$ as shown in Fig. 3.1. It outputs the refined features $\overline{f^t}$ of the current frame I^t.

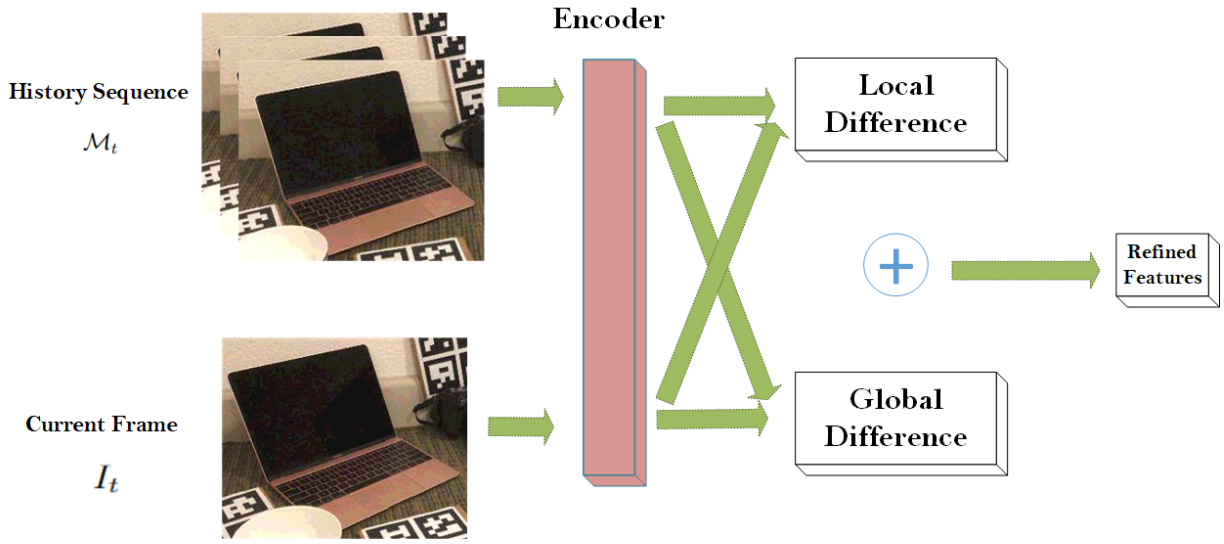

Figure 3.1: The overall structure of the difference-based temporal model.

The model takes both RGB pixel values and points as input and fuses them based on the method inspired by Ffb6d [34]. Ffb6d is a pose estimation model proposed by He et al. [34] that only considers individual frames. The authors propose a bidirectional fusion strategy in the encoder to fuse the feature of pixels and points and it produces promising results. We are inspired to fuse the difference of pixels and points, which will be introduced in the next section. Two sub-modules are applied to the fused features to calculate the difference. In 2D video detection or motion detection models, the difference calculation has been deeply explored by RGB difference [79, 105, 127] and difference of features [46, 56, 64]. The difference in 2D can be directly calculated because the pixels and features are spatio-related. In 3D tasks, the situation becomes challenging because the combination of RGB values and points needs to be taken into account. In this case, we propose a bidirectional feature-wise difference measurement method based on Euclidean distance, which will be

discussed in Chapter 3.3. The purpose of the differences is to learn a feature-wise weight for the previous frames, that indicates the importance of each dimension of the features. After that, the current frame's features are refined using the weight and the last frame's features. By doing so, the model can extract useful features from the history and suppress those features that can affect the model's performance negatively.

For the local difference module, the difference is calculated between every two consecutive frames in $\mathcal{M}^t$. The learned feature-wise weight will be applied to the previous frame's features and the weighted features added to the current frame's features in an iterative manner for all the frames in $\mathcal{M}^t$. The iterative update will start from the latest frame's feature (f^{t-L}) in the history sequence of frame t. Suppose the update function of local difference is $S_{local}()$, which satisfies

$$\overline{f^{t-L+i}} = S_{local}(f^{t-L+i}, \overline{f^{t-L+i-1}}) \tag{3.1}$$

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. $\overline{f^i}$ is the feature after refinement. The function S_{local} takes the refined feature of frame $f^{t-L+i-1}$ and the feature of frame f^{t-L+i} as input to generate the refined feature of frame $\overline{f^{t-L+i}}$. This formula will be iteratively applied to all the frames in the history sequence until it updates the current frame's feature and gets the refined feature of the current frame $\overline{f^t}$

The local difference module iteratively updates the frame sequence. However, this may cause the model to forget early information in the sliding window, that is the feature f^{t-L}. In the local difference module, based on Eq. 3.1, we can represent current frame's feature

using all the frames in the history sequence. By expanding Eq. 3.1, we have:

$$\overline{f^t} = S_{local}(f^t, S_{local}(f^{t-1}, S_{local}(\cdots S_{local}(f^{t-L+1}, f_{t-L}))))) \tag{3.2}$$

The update function S_{local} is to compute a weight and multiply the previous frame with this weight. Then the current frame is updated with this weighted feature from the previous frame (See Chapter 3.5 for more details). By doing so, the model can learn from the continuity of the sequence. However, the weight will accumulate from the first frame in the history sequence, which diminishes the effect of early frame 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.

A global module is necessary to align the current frame's features with all the frames in the sliding window regardless of their position in the window. 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

$$\widetilde{f}^t = S_{global}(f^{t-1}, f^{t-2} \cdots f^{t-L}) \tag{3.3}$$

Local difference and global difference module outputs will be combined to generate the final refined feature of the current frame. Please see Chapter 3.5 and Chapter 3.6 for more details.

3.3 KNN-based Difference Calculation

The difference in pixels can be calculated by direct subtraction because of the known correspondence between pixels of the two images. In 3D computer vision, the point sampling in each frame is different. What we have are unordered point sets, where direct subtraction is not available. In this case, we propose to calculate the difference based on KNN. In 6-Pack [100], the features fusion is through Densefusion [101]. The features extracted from points and pixels are directly concatenated correspondingly. In Densefusion, the features of points and pixels are extracted using encoders and concatenated feature-wise depending on their correspondence to generate the fused features. Although dense fusion is enough for fusing the features of points and pixels according to their correspondence, it ignores spatial relations of the neighbouring points and pixels. Inspired by Ffb6d [34], we propose a KNN-based bidirectional difference calculation and fusion method shown in Fig. 3.2.

Pseudo-LiDAR [107] is back-projecting the pixels using the intrinsic matrix of the camera to obtain points in camera coordinates. Thus, the correspondence between each pixel in the image and the 3D point is known. 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.

The image I^a and point cloud C^a are the RGB image and point cloud from scene a. We propose an extension of the method of Ffb6d [34] to calculate the difference between points and pixels respectively and fuse them in a bidirectional manner. Given the point sets and RGB pixels 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.

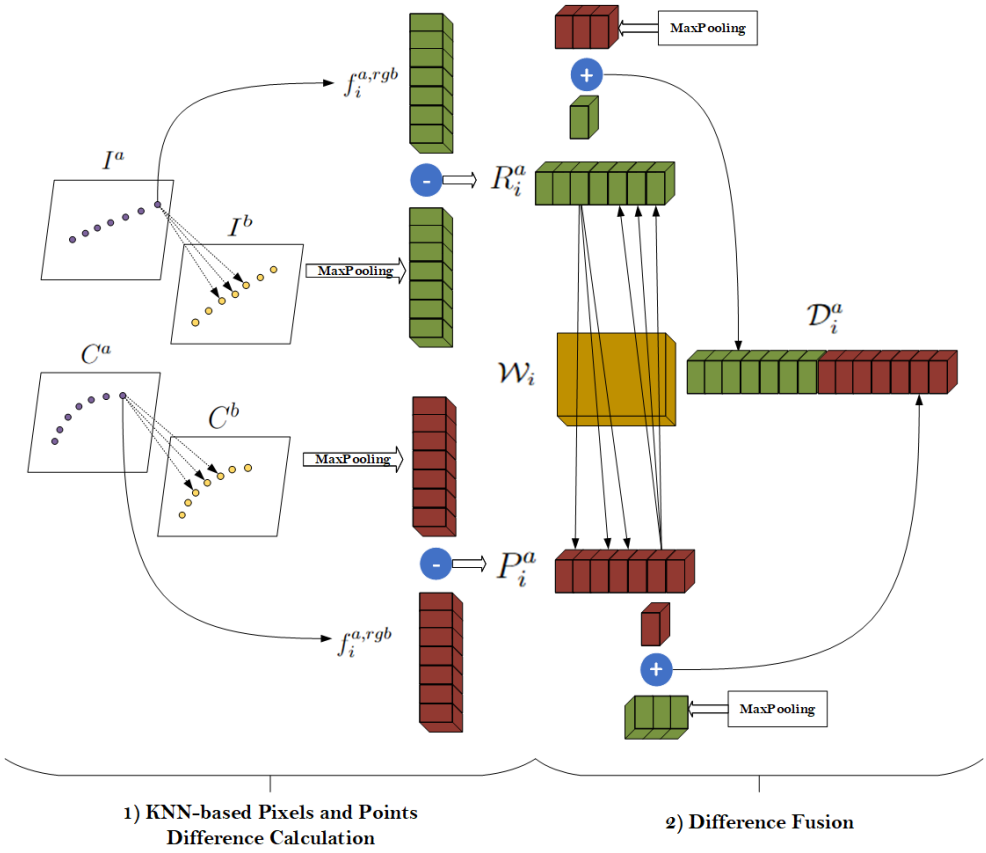

Figure 3.2: KNN-based difference calculation and fusion.

According to the computed distance, given one point in scene a, we can find its K nearest neighbors and given one pixel, we can also find its K nearest pixel neighbors in scene b. The correspondence is then used in the feature map to conduct feature difference calculation and fusion. The KNN-based difference of K nearest point neighbours R_i^a of I_i^a selected in the scene b can be described as

$$R_i^a = f_i^{a,rgb} - MaxPool(\mathcal{W}_i \cdot \mathcal{K}(\mathcal{X}(I_i^a), f^{b,rgb})) \tag{3.4}$$

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 and $\mathcal{K}()$ is the KNN (K nearest neighbors) function, which 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 (after back-projecting the points, we can know the correspondence) $\mathcal{X}()$. Additionally, we also apply a weight ($\mathcal{W}_i$) depending on the distance to weigh the selected features, which can be expressed as

$$\mathcal{W}_{ij} = SoftMax(-\|\mathcal{X}(I_i^a) - C_j^b\|) \tag{3.5}$$

Assuming the points with closer distance share more similar features, the weight $\mathcal{W}_i$ ensures the model assigns high priority to closer points. $MaxPool()$ is the feature-wise maxpooling. The reason that we use maxpooling is to emphasize the largest difference in each dimension of a feature. if given point in the scene a, C_j^a has feature $f_j^{a,points}$. The features of points in scene b are $f^{b,points}$ The points KNN-based points difference P_j^b of the j^{th} point in scene a can be described as

$$P_i^a = f_i^{a,points} - MaxPool(\mathcal{W}_i \cdot \mathcal{K}(C_i^a, f^{b,points})) \tag{3.6}$$

After deriving the RGB and points difference representations (R_i^a and P_i^a) of the i^{th} point in scene a, we now consider fusing the two differences.

3.4 Difference Fusion

In order to generate the fused difference feature of point i given its point difference P_i^a and pixel difference R_i^a, we follow the bidirectional fusion from Ffb6d [34]. As shown in 2) of Fig. 3.2. The difference fusion is also based on KNN. The bidirectional differences fusion of the i^{th} point in scene a is shown as follows

$$U_i^a = R_i^a \oplus MaxPool(\mathcal{W}_i \cdot \mathcal{K}(C_i^a, P^a)),\ and \tag{3.7}$$

$$O_i^a = P_i^a \oplus MaxPool(\mathcal{W}_i \cdot \mathcal{K}(C_i^a, R^a)) \tag{3.8}$$

where $\oplus$ is the concatenation along the feature dimension. U_i^a is the feature-wise concatenation between the pixel difference (R_i^a) and the K nearest neighbors points difference from the points difference features (P^a) calculated in the previous section. The distance-based weight and maxpooling are also applied to the selected features. Correspondingly, the concatenation along the feature dimension between point difference P_i^a and the K closest pixels differences R^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.

3.5 Local Difference

After knowing the difference calculation strategy considering both 2D pixels and 3D points, we will discuss the local difference module in this section. 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 can be expressed in Fig. 3.3 as follows

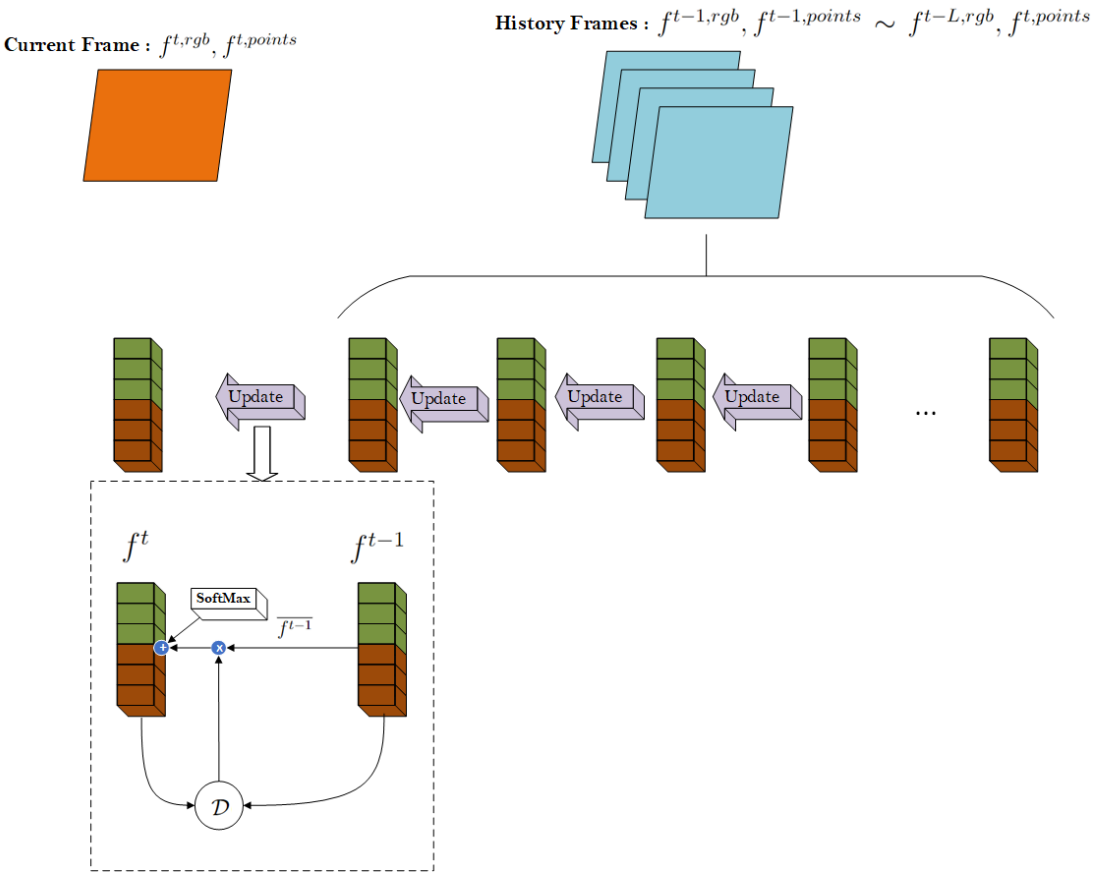

Figure 3.3: Local Difference Update Process.

where $\mathcal{D}^{t-1,t}$ is the difference calculation in Chapter 3.4. The update process starts from the earliest frame I^{t-L} in $\mathcal{M}$. For two consecutive frames (I^t, I^{t-1}), we use a relation module B to learn the feature-wise weight from $\mathcal{D}^{t-1,t}$ in Chapter 3.4, 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 (Multi-Layer Perceptron) with three layers. The update between the features of two consecutive frames can be expressed as:

$$\overline{f_i^t} = SoftMax(f_i^t + AvgPool(\mathcal{K}(C_i^t, B(\mathcal{D}_i^{t-1,t}) \otimes \overline{f_i^{t-1}}))) \tag{3.9}$$

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. As we discussed in Chapter 3.2. The update process will be done across all the frames in $\mathcal{M}$ and finally accumulated to the current frame, I^t. For the first frame in $\mathcal{M}$, $\overline{f^{t-L}} = f^{t-L}$. In Eq. 3.9, inspired by TF-blender [20], we also use the SoftMax function to normalize the refined features. According to Eq. 3.2 and Eq. 3.9, 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.

In the local difference module, each frame only concentrates on the consecutive frame. A strong connection is added between the current frame and the previous frame. However, according to Eq. 3.2, an issue arises. The frame may forget those features of frames that are early in $\mathcal{M}$. Their features are weakened. Thus, we need as well a global difference to treat all the features in $\mathcal{M}$ evenly.

3.6 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. Fig. 3.4 is the structure of the global difference module. Given the feature of the current frame f^t and

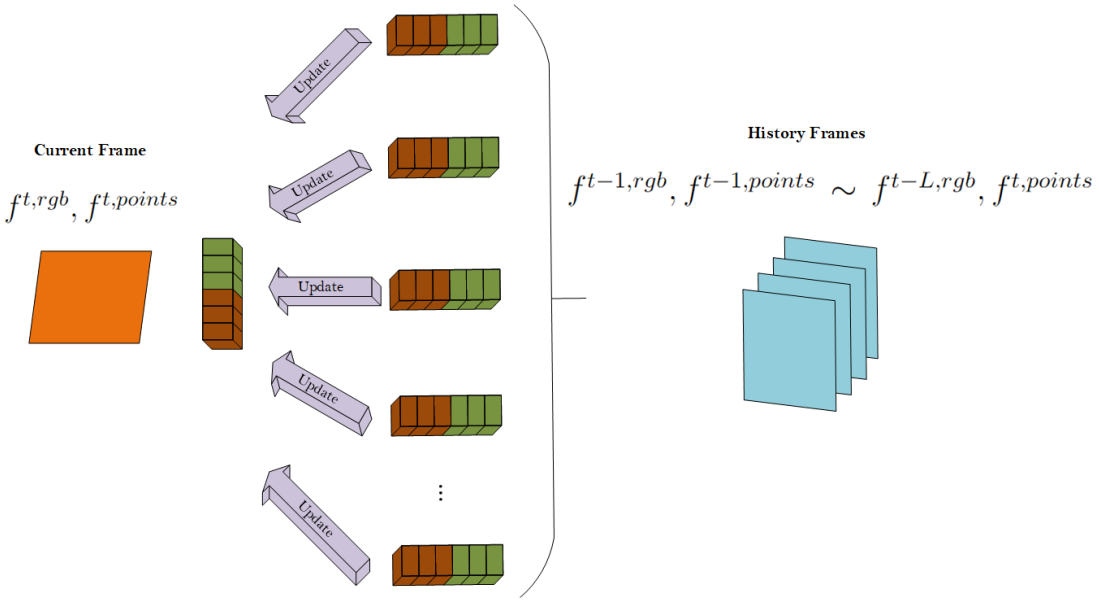

Figure 3.4: Global Difference Update Process.

the features of frames in the history sequence, f^{t-j}, $j \leq L$. The updated current frame can be expressed as

$$\widetilde{f}^t = \sum_{j=0}^{L} AvgPool(\mathcal{K}(C^t, B(\mathcal{D}^{t-j,t}) \otimes \overline{f^{t-j}})) + f^t \tag{3.10}$$

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 Chapter 3.5.

As shown in Fig. 3.1, 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

$$f^{t'} = Mlp(\widetilde{f}^t \oplus \overline{f^t} \oplus f^t) \tag{3.11}$$

We will use the temporally refined feature $f^{t'}$ of f^t for pose prediction. Our model can be applied to any category-level pose tracker or estimator by plugging it into the layer right after the feature encoder. Concretely, we use two models for evaluation, 6-Pack [100] and Captra [115], one of which requires both RGB image and points as inputs and the other only takes 3D points as input. Minor adjustment is needed for the case when the input is 3D points only. We will cover more details about it in Chapter 4. For 6-Pack, the refined feature $f^{t'}$ will be used for keypoints generation. For Captra, the feature will be used for NOCS prediction, segmentation and rotation regression.

3.7 Integration with other Learning-based 6DoF Pose Tracking Models

Our difference-based temporal module can be integrated into other off-the-shelf category-level 6DoF pose tracking networks to refine the current feature by using historical information. Normally, 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. Our temporal module can be added as a mid-layer after the encoder to refine the extracted features from the encoder with the history features. Fig. 3.5 shows the pipeline of integrating our difference-based temporal module with category-level 6DoF pose tracking methods.

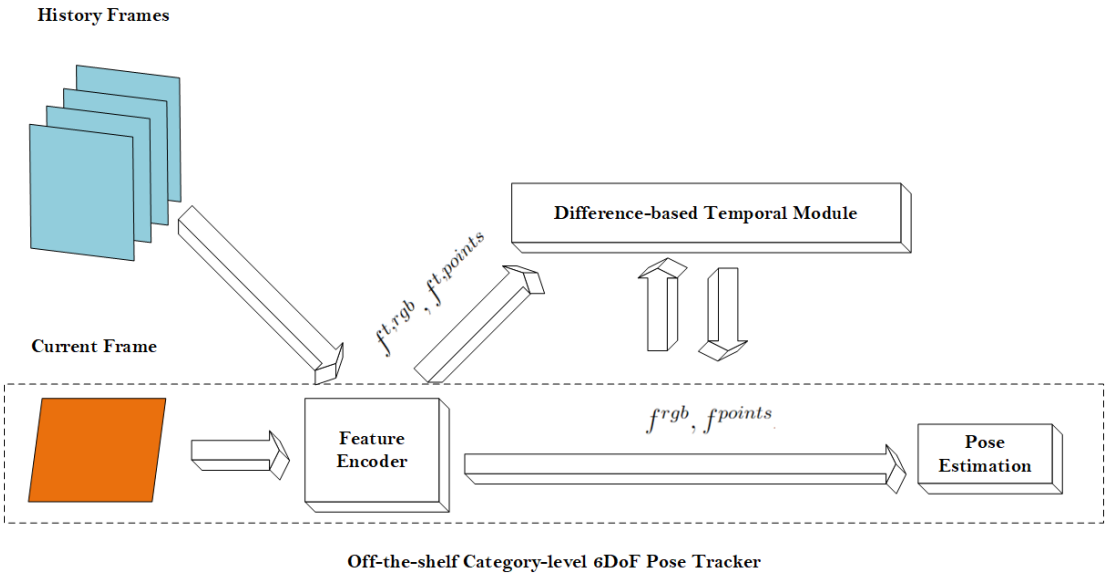

Figure 3.5: The integration between our difference-based temporal module and other category-level 6DoF pose tracking methods.

3.8 History Sequence Updates

Another critical problem in our temporal model is how to manage the history sequence $\mathcal{M}$. Unlike 2D images, the coordinate system plays an important role in the system. 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. Under this scheme, the history sequence can work with the temporal model in a real-time manner.

However, the coordinates of 3D points are meaningless if they are not in the same coordinate system. The model will encounter this problem when trying to align history frames with the current frame. Using the camera coordinate system for all the frames is one of the solutions to it. As the camera and the object are moving in the world coordinates, the object coordinates and camera coordinates can be aligned through the transformation between the camera and the object. A solution to this is to use object space rather than camera space. The relative pose can be regressed through transforming the current frame back to previous frame's object space and letting the model regress the relative pose according to the offset of points. Using the object coordinate system can better normalize the points in different frames at the same scale, which is beneficial for multi-frame consistency and pose regression. In both 6-Pack and Captra, the points are transformed to object space for keypoints generation and rotation regression. However, it is meaningless to transform the points of history frames to their corresponding object space. Because the points of the same object in their object space are all the same. In this case, there are no relations that can be explored.

We propose to align the pose of frames in the history sequence with the current frame, which is 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. Suppose that the $R_t \in SO(3)$ and T_t is the 3×3 rotation matrix and 1×3 translation of frame t from camera

space to object space, which satisfies:

$$C^t_{obj} = (C^t_{cam} - T_t) \cdot R_t \tag{3.12}$$

where the C^t_{obj} are the object coordinates system and C^t_{cam} are the camera space coordinates of scene at time t. The current frame's points C^t_{cam} has its transformation T_t and the points in other frames of $\mathcal{M}$, have their transformation T_{t-i}. For a single point Y_{obj} (1×3 coordinates) in C_{obj}, its camera space coordinate is $Y_{cam} = R \cdot Y_{obj} + T$. Instead of a single point, we use a pointset, which is a $n \times 3$ matrix, where n is the number of points. Thus the rotation matrix needs to be moved to the right side of the pointset and transposed. Because the transpose of the orthogonal matrix is equivalent to its inverse. Then we have $C_{cam} = C_{obj} \cdot R^{-1} + t$. Eq. 3.12 can be derived by moving the C_{obj} to the left side of the equation. Before feeding the history sequence and current frame to the temporal model. All the points are transformed as

$$C^{t-i'}_{obj} = (C^{t-i}_{cam} - T_{t-1}) \cdot R_{t-1}, \quad 0 \leq i \leq L \tag{3.13}$$

where $C^{t-i'}_{obj}$ are the transformed points of frame $t-i$ in the object coordinates. In the 6D pose tracking task, the goal of the model is to predict relative pose $[\Delta R|\Delta T]$. After knowing this relative pose and the transformation between camera coordinate system to object coordinate system in the previous frame, we can compute its new transformation with respect to the camera in the current frame, which is $R_t = R_{t-1} \cdot \Delta R$ and $T_t = T_{t-1} + \Delta T \cdot R^T_{t-1}$. After knowing the transformation between the object coordinate and camera coordinate in the current frame, Eq. 3.13 can be further used to process the next frame and

history frames to let them be in the same coordinate system. In this case, all the points including history frames and current frames are transformed using the transformation of frame $t-1$ and all the poses are continuous and the relations of different frames can be inferred subsequently.

3.9 Adaption to Points-Only Networks

Based on previously discussed concepts. The model works well with the models that take both pixels and points as inputs. However, some of 6D tracking networks only rely on 3D points. Captra [115] that we evaluated in this thesis is a points-only model, thus a simple adaption is needed.

For the models that only rely on points data like Captra, we remove the difference calculation and fusion in Chapter 3.3 and Chapter 3.4 for pixels. In this case, the difference representation of i^{th} point in scene a only contains P_i^a. Then we have $\mathcal{D}_i^{a,b} = P_i^a$ when the model only takes points as input.

3.10 Summary

In this chapter, we explicitly discuss the proposed difference-based temporal model. For the difference calculation, we propose a bidirectional fusion method to combine RGB pixels' and 3D points' differences across scenes. The difference is conducted between the two frames through a KNN-based difference between pixels' features and points' features. The

overall difference can be represented by their bidirectional fusion. The relation between two frames is learned to generate a feature-wise weight map through a relation module.

Based on the difference calculation, the two different modules are proposed. The local difference focuses on the relation between consecutive frames in the history sequence and the current frame. In order to enhance the early features in the history sequence, the global difference is to treat all the frames in the history sequence equally. The refined features of the current frame can be generated by combining these two types of differences.

We also argue that the object space coordinates can facilitate the points normalization and pose regression. Points in previous frames need to be in the same coordinate system before feeding them into the temporal model.

In the next chapter, we will discuss the details of implementation and the data processing step for our experiments.

Chapter 4

Setup

This Chapter will cover the details of our model's implementation, data augmentation, training strategies and hype parameters.

4.1 Dataset

In this thesis, we investigate two category-level datasets. NOCS-REAL275 [103] is the commonly used dataset for 6DoF pose estimation. However, most of data in NOCS-REAL275 have synthetic objects picked from ShapeNetCore [10]. The objects under 6 categories are then placed in random virtual scenes with virtual light sources to mimic real indoor scenes. The synthetic data consists of 1085 individual object instances, which enrich the diversity of objects and improve the model's generalization. These instances compose 300K synthetic images for training and evaluation. The NOCS-REAL275 dataset has 3 asymmetric categories (Laptop, Mug and Camera)and 3 symmetric categories (Can,

	NOCS-REAL275 (Real Data Only)						MoVi-E		
	Bottle	Bowel	Camera	Can	Laptop	Mug	Shoe	Bottle, Can and Cup	Bag
Training	3647	2203	2900	4367	2375	3203	169,824	68976	73824
Testing	2757	2370	2669	2729	2754	2837	23400	17496	7416

Table 4.1: The number of frames of each category split into training and testing sets.

Bottle, and Bowl). In addition to the synthetic scenes, NOCS-REAL275 also contains continuous real data comprised of 18 real scenes (7 for training, 5 for validation, and 6 for testing). Our baseline models, 6-Pack and Captra, use both synthetic data and real data in training. However, due to the randomization of the object's pose and scenes, there's 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 all using only the real data of NOCS-REAL275.

Due to the limited training data in the real data part of NOCS-REAL275, we use another challenging benchmark, MoVi-E [30]. The MoVi dataset is a synthetic dataset series with increasing complexity from A to F. The MoVi-E dataset has 16 categories, among which the "Shoe" category has the greatest number of training samples. Although it is a synthetic dataset, its scene rendering is based on videos. Although the length of the videos is short (24 frames per video), in each video, the pose of the object is continuous. In this case, our temporal model can fully use the dataset. Considering the time limit of training, we select two asymmetric categories (Shoe and Bag) and one symmetric category consisting of Bottle, Can and Cup. The MoVi dataset classifies Bottle, Can and Cup as a common category. Table 4.1 is the number of frames for training and testing of each category in the two datasets.

4.1.1 Data Augmentation

Background Replacement

We follow the data augmentation strategy in 6-Pack [100]. The data augmentation strategies adopted in our experiments can be separated into background replacement and transformation perturbation. For the NOCS-REAL275 dataset, we follow 6-Pack to randomly change the background of each image by picking an RGB image from the COCO [60] dataset and randomly changing the background depth with the depth of other frames. By doing this, extra noise are added to the sampled points in the ROI (Region of Interest) in addition to the target object. Fig. 4.1 shows the background replacement data augmentation examples. where the first row shows the original images and the second row is images

Figure 4.1: The examples of background replacement.

with different backgrounds. We can use the segmentation map of the foreground object to crop out the pixels with their depth. Then we do the same thing to the background RGBD images and compose the foreground and background RGBD as the new training sample.

However, using the original depth to compose may cause inconsistency. The foreground object may lie behind the background object. Thus, we maintain the original distance between the background and the foreground object when we compose a new RGBD image. In order to do so, we keep the original depth of the foreground and calculate the shortest distance between foreground and background. Suppose the depth map of foreground is d^{obj}, the depth map of original background is d^{back} and the depth map of the new background is d^{new}. Then adjusted depth map d^{new} is $d^{new'} = d^{new} - min(d^{new}) + min(d^{back})$, where $min()$ and $max()$ are the two functions for finding the minimum and maximum depth value among the depth map. During the training of 6-Pack, the model takes two images as input, the background of these two frames as well as the frames in the history sequence will be all random.

Pose Perturbation

In addition to randomizing the scenes. A training strategy based on transformation perturbation is also applied. There is no need to use the actual previous frame in the training, while the model takes two frames (the current frame and the previous frame) as input, both of these frames are in their object space, that is the coordinate system with a origin in the centroid of the object. Because, during real-time inference, the frame will be transformed to the previous frame's object space to predict the current frame's pose with respect to the previous frame. Thus, we can use a random transformation to represent the relative transformation. The frame in the training is $C_{obj}^{t'} = ((C_{cam}^{t} - t_t) \cdot r_t) \cdot r_{random}^{-1} + t_{random}$, where the $[r_{random} \mid t_{random}]$ is the random transformation, which is different from frame to frame and $[r_t \mid t_t]$ is the transformation from camera space to objects. Thus, as discussed in Sec.

3, all the frames in the history sequence are $C_{obj}^{t-i'} = ((C_{cam}^{t-i} - t_t) \cdot r_t) \cdot r_{random}^{-1} + t_{random}, i \leq L$. The $[r_{random} \mid t_{random}]$ is the same random transformation as the current frame. After the inference of the current frame's pose, the history frame's pose will be back-transformed to camera space. In this case, the coordinate systems of all the frames are guaranteed to be the same. In testing, the frames in the history sequence are not transformed to current frame's coordinate system because the transformation to it is unknown. Instead, we transform all the frames including history frames and the current frame to the object space of C^{t-1}.

4.1.2 Symmetry Handling

One of the challenging problems in 6D pose tracking is symmetric handling. Symmetric objects may suffer from rotation invariance. The different rotations may result in the same shape. The rotation's degree of freedom will degrade to two. We follow the symmetry handling methods of 6-Pack and Captra and use them in both MoVi-E and NOCS-REAL275 dataset.

6-Pack

For 6-Pack, its loss function (see Chapter 4.4 for more details) cannot handle symmetric objects with their Cartesian coordinates. In 6-Pack, it transforms a symmetric object into a new coordinate system. Supposed the symmetry axis of the object is l, around which the object's rotation is ambiguous, for any keypoint $k_i : (x_i, y_i, z_i)$ that belongs to the symmetric object, the transformation from the Cartesian coordinate system to the

rotation-invariant coordinate system can be represented as (d, h, θ), where the d is the distance to the symmetry axis l, and h is the distance between the projection of k_i onto the symmetry axis l and the original Cartesian origin. θ is the relative angle between the two radial vectors between the keypoints and l. As the generated keypoints are ordered in 6-Pack, the angles θ between two consecutive keypoints can be used to represent their relative information. Fig. 4.2 shows the coordinate systems. On the left side of Fig. 4.2 are the original coordinates and on the right side is the bird's-eye view coordinates after the conversion. The axis marked as green is the symmetry axis l. The bird's-eye view coordinate system is then used in the loss function of a symmetric object.

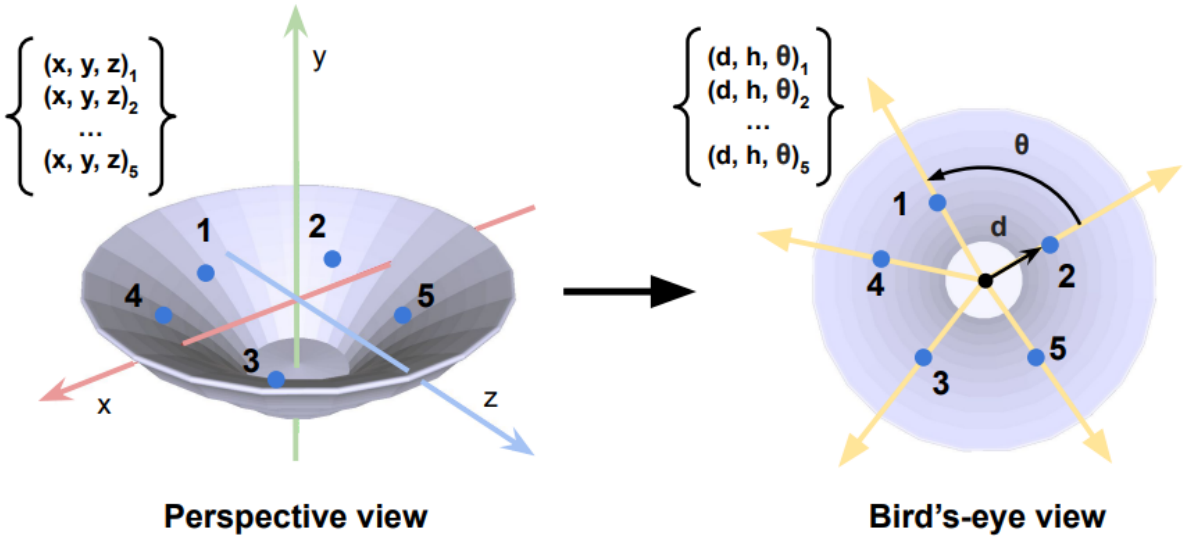

Figure 4.2: Symmetry-invariant Coordinate System [100]. ©Copyright IEEE 2020.

Captra

Unlike 6-Pack, Captra uses a 2D Umeyama algorithm [97] to estimate the rotation of symmetric objects. This method is more generic and does not require the keypoints to represent relative angles. As discussed in Chapter 2, Captra's rotation prediction and translation prediction are separate. The predicted rotation from RotationNet will be used with the predicted NOCS coordinate to recover scale and translation. We have $\widetilde{U}^{i+1} = \widetilde{s}^{i+1} \cdot R(l,\theta) \cdot \widetilde{Y}^{i+1} + \widetilde{T}^{i+1}\widetilde{R}^{i+1}$ and $\widetilde{U}^{i+1} = \widetilde{C}_{cam}^{i+1}\widetilde{R}^{i+1}$, where $\widetilde{R}^{i+1}$ is the predicted rotation from RotationNet and $\widetilde{s}^{i+1}, \widetilde{T}^{i+1}$ are the scale and translation we want to predict. $\widetilde{Y}^{i+1}$ is the NOCS space coordiates and $\widetilde{C}_{cam}^{i+1}$ is the input camera space coordinates. However, due to the ambiguity of symmetric objects, there's an arbitrary rotation $R(l,\theta)$ around the symmetry axis l. Translation and scale cannot be computed without knowing $R(l,\theta)$. We follow the method from Captra. Suppose the symmetry axis is z axis, all the points will be projected to the $x-y$ plane. In this case, $R(l,\theta)$ is considered as a 2D rotation. Thus, the $R(l,\theta)$ can be derived by solving a 2D version of the Umeyama algorithm [97].

The loss functions that the two baselines used on asymmetric and symmetric objects are slightly different. We will discuss that in Sec. 4.4.1.

4.2 Model Integration

In this section, we will specify how we integrate our difference-based temporal model with the two baselines. As mentioned in Sec. 3, our temporal model's purpose is to refine the current frame's feature through a history of frames. So our model is supposed to connect

to each model's encoder.

6-Pack is a keypoint generator that generates ordered keypoints for each frame. The encoder in 6-Pack is Densefusion [101], which fuses the feature of each pixel with its corresponding point. Our model intends to refine these features. The shared-weight encoder is applied to extract basic features for the current frame and the history frames. Our model extracts the bidirectional features mentioned in Chapter 3 to learn the inter-frame relations, which are used to update the basic features provided by Densefusion. During the training phase, we maintain two history sequences, one for the previous frame and the other one for the current frame. During training, the two selected frames have no order. The only constraint is that they are from the same scene. Thus, while the two history sequences may be overlapping, they may also be distinct. Captra uses the two submodules RotationNet and CoordinateNet which are built upon PointNet++ [83]. Our model refines the output features from PointNet++. Because Captra only takes points as input, the pixels-to-points correspondence is not feasible. In this case, there is no bidirectional fusion between pixels and points. We directly use the output features from PointNet++ and calculate the KNN-based difference between two point sets to learn the weights for feature updates. Because Captra has two sub-modules and the temporal model can be applied separately to each of them, we can apply our temporal to both or to one or the other. Based on our experiments, the performance of the model depends on to which submodules we apply the temporal model to. We have done multiple experiments to find out the best combination, which will be discussed in Chapter 5.

4.3 Metrics

We follow NOCS [103] 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): We follow the way of mIoU calculation in 6-Pack [100]. The average percentage of the 3D intersection between the predicted 3D bounding box and the 3D bounding box. In order to calculate the IoU of 3D bounding box. Suppose the vertices of the two bounding boxes a, b are bb_a, bb_b. $max()$ and $min()$ are the two functions that can find the maximum and minimum among x, y and z components in a pointset. The function $proc()$ is to compute the multiplication of the x, y and z components of a vector. The intersection of a, b can be represented as $In = prod(max(min(bb_a), min(bb_b) - min(max(bb_a), max(bb_b))$. This formula approximates the intersection between two bounding boxes as a cuboid. The union can be expressed as $Un = prod(max(bb_a) - min(bb_a)) + prod(max(bb_b) - min(bb_b)) - In$. Finally, the IoU is $IoU = \frac{In}{Un}$.
- Rotation Error (R_{error}) : The average error of rotation in degree. Suppose the ground truth rotation matrix is R_g and the predicted rotation matrix is R_p. The E_{rot} can be expressed as: $arccos(\frac{tr(R_p \cdot R_g^T) - 1}{2})$, where $tr()$ is the trace of matrix.

- Translation Error (T_{error}) : The average error of translation in centimetres. Given the ground truth translation (T_g) and predicted translation (T_p). The translation error can be computed as $\|T_g - T_p\|$.

4.4 Training

4.4.1 Loss Functions in 6-Pack [100]

The loss function is the key to unsupervised keypoints generation. The loss function is mainly comprised of 4 parts, multi-view consistency loss, separate loss, pose estimation loss and centroid loss.

Pose Estimation Loss:

This is the loss for relative transformation between generated keypoints k, which is as follows:

$$L_{trans} = \|(\overline{k}^t - \overline{k}^{t-1})\| \tag{4.1}$$

$$L_{rot} = 2arcsin(\frac{1}{2\sqrt{2}}\|\Delta\widetilde{R}_t - \Delta R_t\|) \tag{4.2}$$

where the $\Delta\widetilde{R}_t$ is the predicted relative rotation matrix and ΔR_t is the ground truth rotation matrix between the object in current and previous frames. The norm in Eq. 4.2 is applied to the summation of values in the two matrices. These two losses can force the change of the keypoints position to be close to the ground truth.

Multi-view Consistency Loss [95]:

The objective of the multi-view consistency loss is to align the keypoints in two different frames. The multi-view consistency loss is as follows:

$$L_{mvc} = \frac{1}{K} \sum_{i} \|k_i^t - \Delta T_t^{gt} k_i^{t-1}\| \tag{4.3}$$

where the K is the number of keypoints, $T_t^{gt} \in SE(3)$ is the ground truth transformation between the two frames. However, the multi-view consistency loss causes the model to degenerate because the keypoints may be generated at the same location. To solve this problem, a separate loss is needed.

Separate Loss:

The loss is to separate each keypoint to guarantee the generated keypoints will not end up in the same place.

$$L_{sep} = -\frac{1}{K} \sum_{i} \sum_{j} \|k_i^t - k_j^t\| - \frac{1}{K} \sum_{i} \sum_{j} \|k_i^{t-1} - k_j^{t-1}\| \tag{4.4}$$

The two terms in the loss function are the average distance between every two keypoints in the current and previous frames.

Centroid Loss:

This loss aims at forcing the centroid of generated keypoints to the centoid of the object, which can suppress the noise from the points sampling and will also prevent the keypoints

to cluster. The loss is as follows:

$$L_{cen} = \sum_i \|k_i^t\| + \sum_j \|k_j^{t-1}\| \tag{4.5}$$

where k_i^t is the i^{th} generated keypoint in frame t and k_j^{t-1} is the j^{th} keypoint in frame $t-1$. Finally, the overall loss will be the summation of the mentioned 4 losses. In 6-Pack's original paper, the authors also adopt a close-to-surface loss, which forces the predicted keypoints to be as close to the surface of the object as possible. This loss further requires the mess of the object. However, we argue that using the mesh assumes the priori knowledge of the instance's shape, which is not consistent with category-level task. We have evaluated the model trained with mesh and without mesh. The result shows no difference. Hence, we remove the separate loss.

For symmetric objects, there are adaptions to the multi-view consistency loss and rotation loss. As discussed in Sec. 4.1.2, suppose the conversion from the Cartesian coordinate to the symmetry-invariant coordinate is ρ. Then we have the new mult-view consistency loss and rotation loss:

$$L_{mvc}^{sym} = \frac{1}{K} \sum_i \|\rho(k_i^t) - \rho(\Delta T_t^{gt} k_i^{t-1})\| \tag{4.6}$$

$$L_{rot}^{sym} = arcos(\frac{\Delta s_{gt}^t \cdot \Delta \widetilde{s}^t}{\|s_{gt}^t\| \cdot \|\widetilde{s}^t\|}) \tag{4.7}$$

where the Δs_{gt}^t is the ground truth angle change of symmetry axis and $\widetilde{s}^t$ is the predicted change of symmetry axis. The two losses for symmetric objects decouple the rotation loss into the change of symmetry-invariant coordinate and angular change of symmetry axis.

4.4.2 Loss functions in Captra [115]

In Captra, the rotation is predicted through a PointNet++ [83] based network. The Euclidean mean [75] is used to conduct the rotation regression. The CoordinateNet is used to predict the NOCS space coordinates $\widetilde{Y}^{t+1}$ in frame $t+1$ given the original point cloud $\widetilde{C}^{t+1}$. Suppose $\widetilde{w}^{t+1} = \widetilde{Y}^{t+1}\widetilde{R}^{t+1}$, where the $\widetilde{R}^{t+1}$ is the predicted rotation. The rotation matrix is predicted through direct regression using MSE (Mean Square Error). We have $\widetilde{C}^{t+1} = \widetilde{s}^{j}_{t+1}\widetilde{w}^{t+1} + \widetilde{T}^{t+1}$. The $\widetilde{s}^{t+1}$ and $\widetilde{T}^{t+1}$ can be computed by:

$$\widetilde{s}^{t+1} = \frac{\sum_i \widetilde{w}^{t+1\top}_i \widetilde{C}^{t+1}_i}{\sum_i \widetilde{w}^{t+1\top}_i \widetilde{w}^{t+1}_i} \tag{4.8}$$

$$\widetilde{T}^{t+1} = avg(\widetilde{C}^{t+1} - \widetilde{s}^{t+1}\widetilde{W}^{t+1}) \tag{4.9}$$

For symmetric objects. Let $\widetilde{U}^{t+1} = \widetilde{C}^{t+1}\widetilde{R}^{t+1}$, then we have $\widetilde{U}^{t+1} = \widetilde{s}^{t+1}R(l,\theta)\widetilde{Y}^{j}_{t+1} + \widetilde{R}^{t+1}\widetilde{T}^{t+1}$, where the $R(l,\theta)$ is a rotation around the symmetry axis l. It means that there may be an extra rotation $R(l,\theta)$ between the NOCS space $\widetilde{Y}^{t+1}$ and $\widetilde{U}^{t+1}$. To use Eq. 4.8 and Eq. 4.9 to compute the s and T, we need to know the rotation $R(l,\theta)$. As discussed in Chapter. 4.1.2, the $R(l,\theta)$ is recovered using a 2D version Umeyama algorithm [97], where the points are projected to the plane that is orthogonal to the symmetry axis l. After knowing $R(l,\theta)$, the s and T can be computed using similar methods like Eq. 4.8 and Eq. 4.9.

4.5 Summary

In this Chapter, we have introduced the details of our implementation. We first describe the dataset information that we used in the experiments including the data type (symmetric or asymmetric object) and the data augmentation method we used. During training, we do not use two consecutive frames. Instead, we add a random transformation to each single frame to let the model learn the random pose change. Then we also introduce the symmetry handling methods and loss functions of the two baselines that we compare with. In the next Chapter, we will look into the evaluation results and their analysis.

Chapter 5

Experiments and Results

In this section, we will introduce the experiments we have done to evaluate the model's performance compared with the two baselines and the evaluation results.

First, in Chapter 5.1, we present an overall comparison between our model and the two baselines. The five metrics mentioned in Sec. 4.3 will be used to evaluate the results of the models under each category. For Captra, we use an extra metric E_{scale} to measure the error of the predicted scale. Because Captra was originally designed for 9DoF pose tracking and the size of the bounding box is required to predict. The size is reflected through a scale factor. The scale has been introduced in Chapter 4, which is a factor that scales the point cloud of the object from its NOCS space to its original object coordinate system. In this thesis, we only consider non-articulated objects. 6-Pack does not conduct size prediction as it uses the initial 3D bounding of the object in the first frame with the predicted pose to represent the status of the objects in each frame. Hence, it does not require the scale

of the object.

Secondly, we will do some ablation studies to evaluate our model in different aspects of our model including experiments regarding the relations between the length of the history sequence and the model's performance, and the effect of using normalization in the local difference module.

5.1 Overall Results

This section will mainly discuss the performance of our history model applied with 6-Pack in the real data of NOCS-REAL275 and MoVi datasets. Our history model is trained with a history sequence that has a length of four. Table 5.1 is the comparison with 6-Pack trained with the real data from NOCS-REAL275.

Our model is trained using a history sequence. For the metrics in the first row of the table, the up-arrow means the greater the better and the down-arrow means the smaller the better. Generally, The overall performance of our model surpasses the 6-Pack except for the translation error. Both of the models perform the best for the laptop category. $5°5cm$ is an essential metric that can reveal the model's robustness to keep the rotation error and translation error under $5°$ and 5 centimetres. Our history module Our model also obtains a 36.80% $5°5cm$, which is about 10 percentage points higher than 6-Pack. We observed that most of the rotation error and translation error are settled in $10°$ and 10 *centimetres* respectively. It is difficult to investigate the model's performance only through $5°5cm$. This is the reason why we also incorporate $10°10cm$ as well. For $10°10cm$, our

Table 5.1: Comparison with 6-Pack [100] in trained with the real data part in NOCS-REAL275 dataset.

		$5°5cm \uparrow$	$10°10cm \uparrow$	mIoU ↑	$R_{error} \downarrow$	$T_{error} \downarrow$
6-Pack [100]	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**
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

model outperforms 6-pack 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, our model refines the features to generate a more robust trajectory with less rotation and translation error. Because 6-Pack uses ground truth 3D bounding box as the size of objects instead of regressing it. Our history module's average mIoU over all categories is higher than 6-Pack, which also reflects the lower rotation and translation error we have. We also evaluate our model in another larger video dataset MoVi-E. Table 5.2 is the result of our model compared with 6-Pack using MoVi-E, a more challenging dataset with frequent occlusions and clutter. Our model outperforms the baseline in all metrics in MoVi-E. There

Table 5.2: Comparison with 6-Pack [100] in MoVi-E dataset.

		$5°5cm \uparrow$	$10°10cm \uparrow$	mIoU $\uparrow$	$R_{error} \downarrow$	$T_{error} \downarrow$
6-Pack	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
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**

is a huge 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.

Fig. 5.1 and Fig. 5.2 show some examples comparing 6-Pack and our module in the MoVi-E dataset when encountering occlusions and clutter. The green bounding box is the ground truth and the red bounding box is the predicted bounding box. Fig. 5.1 includes two scenes where the target object is partially visible. The target shoe in the first and second rows is partially out of the camera's range. For 6-Pack, the translation prediction can still allow the 3D bounding box to follow the target object. However, the rotation error becomes larger as the visible part of the object decreases. For our history module, the result is more robust when the vision of the object is limited. In the second scene (third row and fourth row of Fig. 5.1), The target shoe is occluded by a purple box. When

the occlusion appears, the rotation and translation error starts to increase. The error of the baseline model is higher than our model.

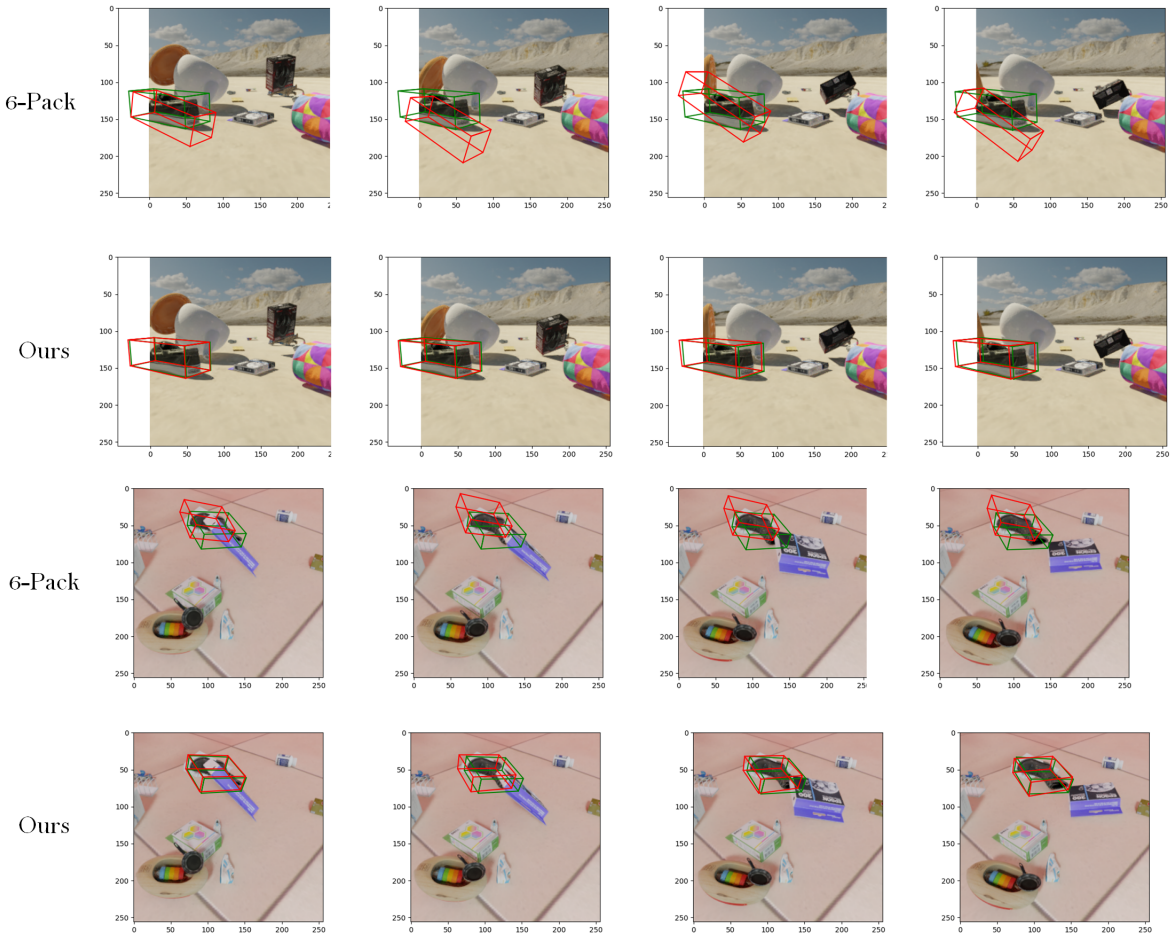

Figure 5.1: Examples evaluated in MoVi-E dataset encountering occlusions, where the red box is the prediction and the green box is the ground truth.

In Fig. 5.2, the object is placed in a cluttered scene. In this first and second row, the 6-Pack model is distracted by the white box that is near the target shoe, while our model's predicted 3D bounding box sticks with the object. In the third and fourth rows, the target

shoe falls to a cluttered scene, where the change of the object's pose is rapid. Both of the models show translation and rotation errors. But our model's predicted 3D bounding box still follows the object while the baseline mistakes the bowl near the shoe as the target.

Figure 5.2: Examples evaluated in MoVi-E dataset with cluttering scene, where the red box is the prediction and the green box is the ground truth.

Fig. 5.3 shows some examples of our model compared with 6-Pack in NOCS-REAL275

dataset. Because the real video data in NOCS-REAL275 dataset is insufficient for training, the overall performance of the models is worse than the models trained on MoVi-E dataset. In Fig. 5.3, we show two scenes where the target object (laptop) is occluded by a cup and a bowl respectively. The rotation error of the baseline in the first and third rows is becoming large from left to right, while our model's prediction results in the second and fourth row are more stable.

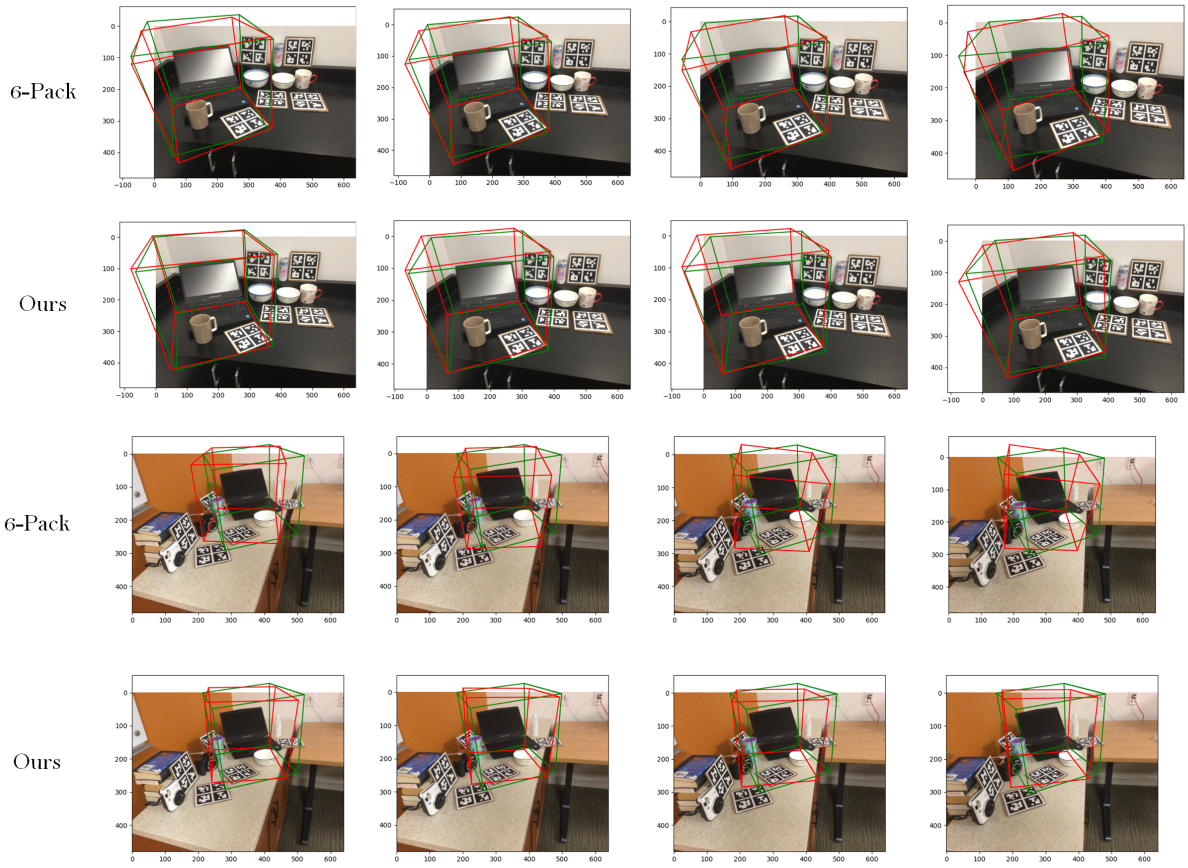

Figure 5.3: Examples evaluated in NOCS-REAL275 dataset compared with 6-Pack [100], where the red box is the prediction and the green box is the ground truth.

Table 5.3: Comparison with Captra [115] in the real data of NOCS-REAL275 dataset.

		$5°5cm \uparrow$	$10°10cm \uparrow$	mIoU ↑	$R_{error} \downarrow$	$T_{error} \downarrow$	$S_{error} \downarrow$
Captra [115]	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**
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

We also use Captra as the backbone to test our model's performance. Table 5.3 shows the overall results of our history model compared with the baseline Captra.

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 scale error.

The overall results of our model are better than Captra except the translation error and scale error. These two metrics are mainly affected by the performance in the category of Can. The Can category has high translation errors and scale errors in both our model and 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).

We also evaluate our model with baseline Captra in MoVi-E dataset, where the training set is much larger than NOCS-REAL275 dataset. Table 5.4 shows the overall comparative results. 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 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. 5.4 are examples of our model's performance on NOCS and MoVi-E datasets. We select bottle from NOCS-REAL275 dataset and bag from MoVi-E dataset. We identify bottle as symmetric object and bag as asymmetric object. Unlike 6-Pack, Captra's 3D bounding box is not the ground truth bounding box, Captra predicts a scale in each frame, which causes the size of the predicted 3D bounding box to vary from different frames for the same instance. In Fig. 5.4, the first scene (NOCS-REAL275 dataset) displays the two symmetric bottles placed on the table, both our model and the baseline performs well in handling rotation and translation of symmetric object, where the orientation and the position of the predicted bounding box align well with the ground truth. However, the scale prediction is unstable in both of the methods. Our model better fits the ground truth compared to Captra. In the second scene, the target object is heavily occluded, the

scenes like this are frequent in MoVi-E dataset, which makes the task challenging. The bounding box of Captra in the second scene becomes slim while our model's bounding box gets enlarged.

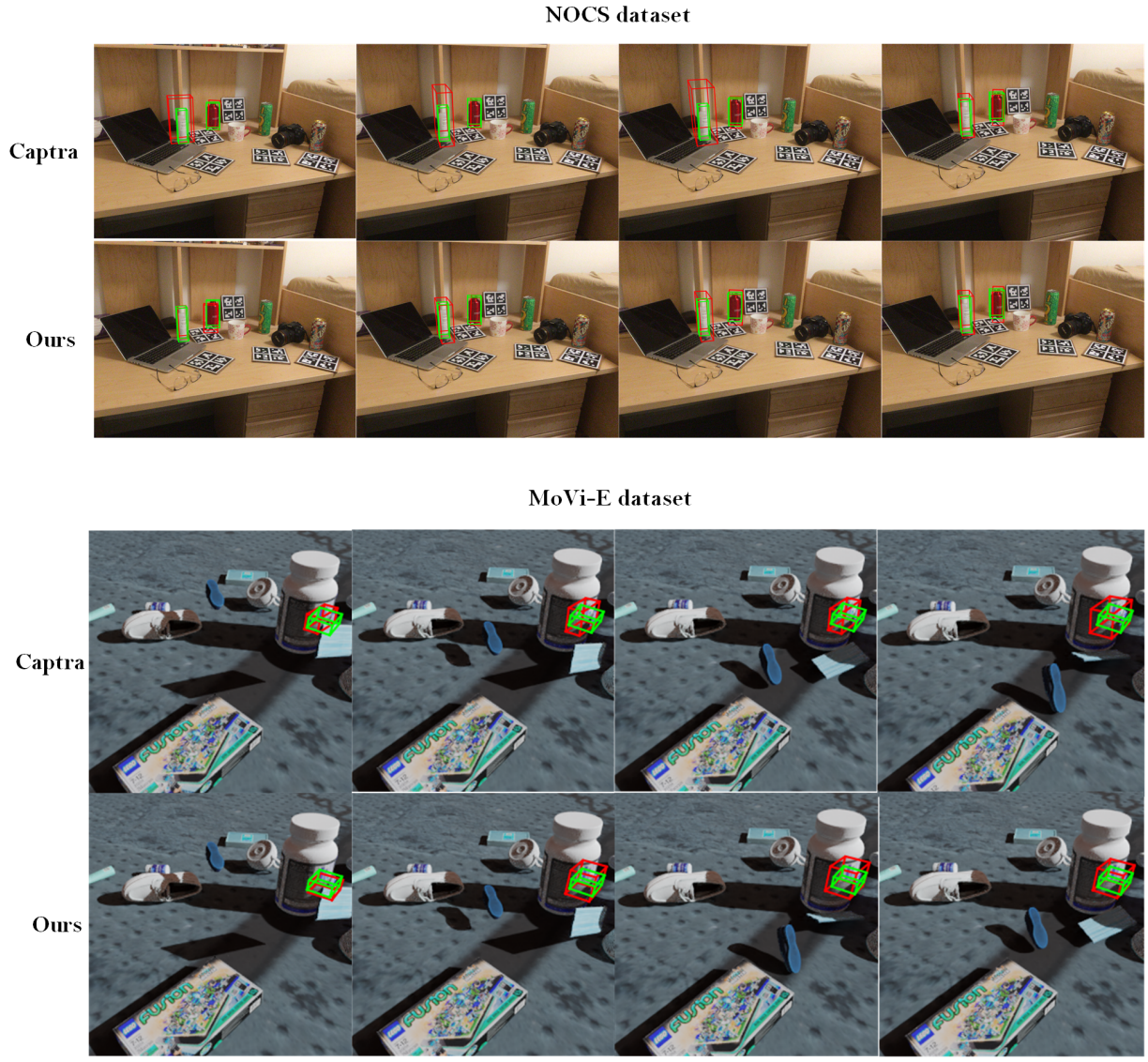

Figure 5.4: Examples evaluated in NOCS-REAL275 and MoVi-E dataset compared with Captra [115], where the red box is the prediction and the green box is the ground truth.

Table 5.4: Comparison with Captra [115] in MoVi-E dataset.

		$5°5cm \uparrow$	$10°10cm \uparrow$	mIoU ↑	$R_{error} \downarrow$	$T_{error} \downarrow$	$S_{error} \downarrow$
Captra	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**
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

5.1.1 Feature Visualization

In this Chapter, we visualize the difference features in the local difference module. Our local difference module focuses on the difference between consecutive frames and learns a feature-wise weight. We measure the feature-wise difference using cosine similarity. According to the structure of the local difference module, the cosine similarity will be applied between the current frame and its reference frame's features from the history sequence. In our design, the difference features are used to enhance the features of the target object, when another object appears in the points' sampling zone. The cosine distance of the difference features of the new object will relatively increase for both pixels and points difference. Fig. 5.5 shows the visualization results of difference features in our temporal model.

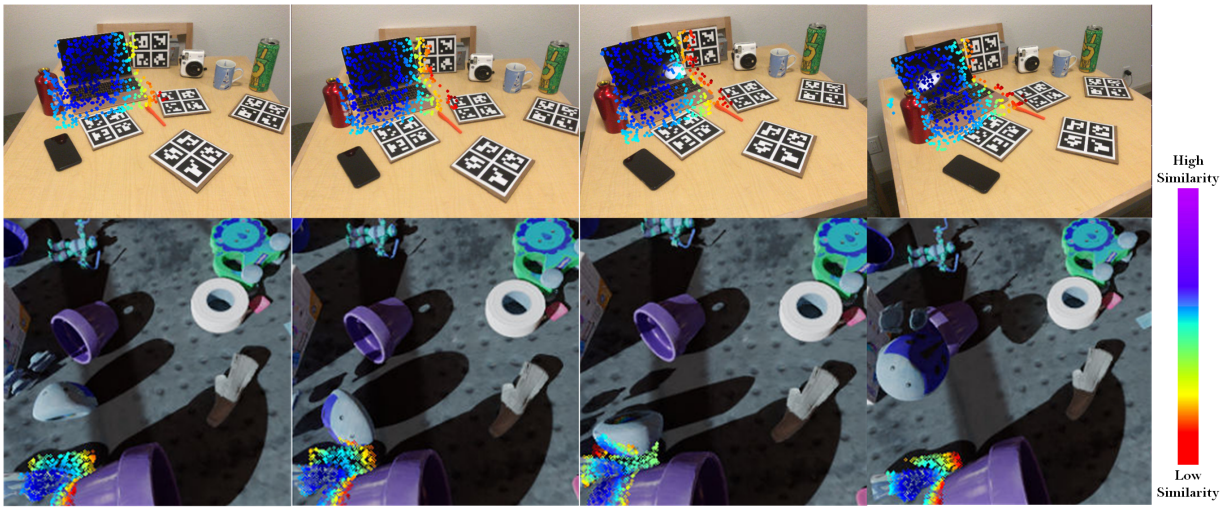

Figure 5.5: The visualization of difference features of the sampled points measured by cosine distance. For a feature point, the darker the color (blue or purple), the higher the cosine similarity with respect to its reference frame (previous frame). Because the change between two consecutive frames is minor, we display the results over two frames.

We visualize Eq. 3.6 by replacing the subtraction with cosine similarity, which is the cosine similarity between $f_i^{a,points}$ and $MaxPool(\mathcal{W}_i \cdot \mathcal{K}(C_i^a, f^{b,points}))$. The first row is the laptop category in NOCS-REAL275 dataset and the second row is the shoe category in the MoVi-E dataset. We visualize the features in different colors, where the color map is shown on the right side. The darker blue points mean the cosine similarity is high compared to its reference frame, while red and yellow means an obvious feature change appears for these points. The input points are subsampled. For the first row, in the sampling region, the area that is far from the centroid of the instance has a lower cosine similarity, where the boundary of the object may encounter frequent changes of point features. When the occlusion (bottle) enters the sampling area, the feature difference increases, which also means the feature difference of the bottle increases. For the second row, the same

phenomenon can be observed in cluttered scenes. These difference features can make our model sensitive to the change of features.

5.2 Ablation Study

In the ablation study, we will analyze the two components of our model with different parameters. First, we will discuss the performance of our model with different lengths of history sequences (sliding window). The history sequence is the structure that stores and updates the history information. The length of it will determine the features learned from differences gathered through the local difference module and global difference module. We would like to see what is the relation between the length of the history sequence and our model's performance. Second, we examine the necessity of feature normalization in the local difference module. In Chapter 3, we have introduced the normalization using SoftMax function for the local difference module to prevent the feature accumulation from producing a large value. We will do an experiment to evaluate the performance of our model with or without SoftMax normalization.

5.2.1 Length of History Sequence (Sliding Window)

The length of the history sequence is one of the important properties of our model. The length of the history sequence may affect the essential features gathered from history frames. Because the length one history sequence is meaningless, we start from length two. Due to the GPU limit, we are not able to investigate very long history sequence.

We evaluate our model with the backbone of 6-Pack in the MoVi-E dataset shown in Fig. 5.6. It is obvious that the $5°5cm$ and $10°10cm$ are the two metrics most affected by the length. For (a) of Fig. 5.6, when the length of the history sequence is 6 and 7, the $5°5cm$ and $10°10cm$ reach the peak. However, when the length is 5, the performance is worse than the other cases and the performance also gets worse when the length of the history sequence increases to 8 compared to the length of 7 and 6. For NOCS-REAL275 dataset, the gap between different lengths is smaller than MoVi-E. It is because the scenes in NOCS-REAL275 dataset are more stable and less challenging than MoVi-E. Occlusions and fast motion rarely appear. In NOCS-REAL275 dataset, the peaks are observed when the length of the history sequence is 3 and 7. We cannot directly observe the relation between the performance of our model. The significant impact from the length of the history sequence can be found in more complex scenes (MoVi-E dataset).

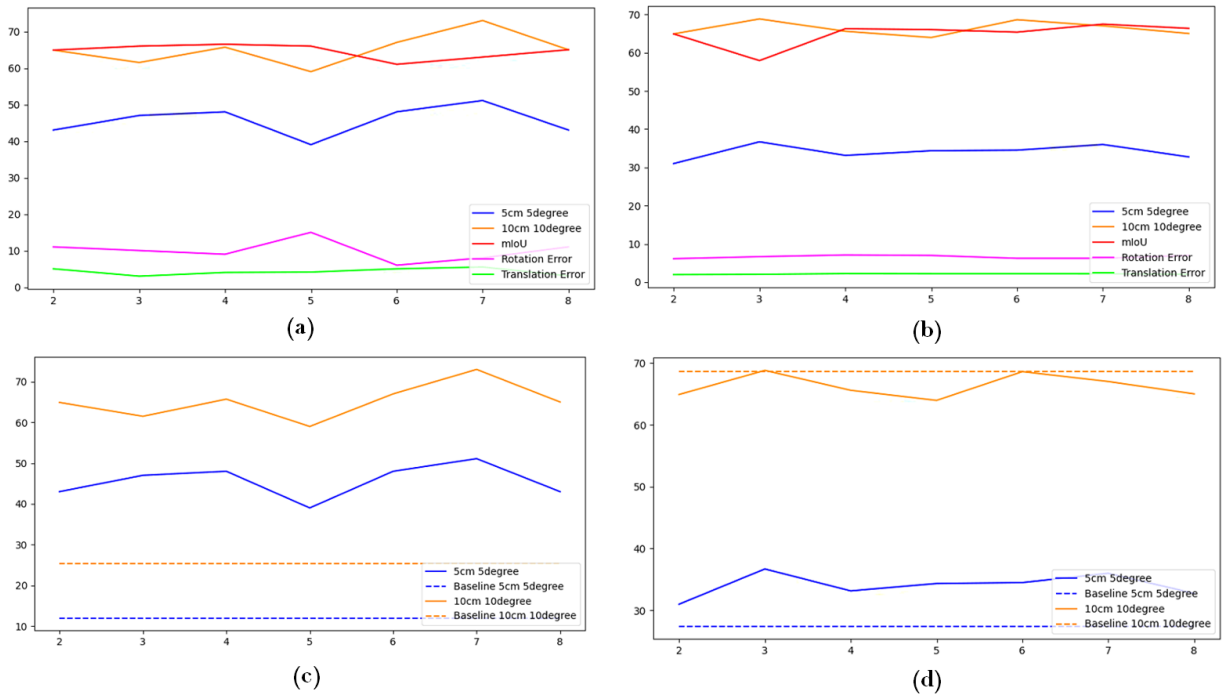

Figure 5.6: The performance of our temporal model with increasing length of history sequence using laptop and shoe as the target object. (a) are the results from MoVi-E dataset (Shoe) and (b) are the results from NOCS-REAL275 dataset (laptop). For (c) and (d), we pick $5°5cm$ and $10°10cm$ to compare with the baseline 6-pack. The dashed lines are the metrics of the baseline 6-Pack without our history module. (c) are the results from MoVi-E dataset and (d) are the results from NOCS-REAL275 dataset.

5.2.2 Normalization in Local Difference Module

In this Chapter, we evaluate the performance of our model with and without the softmax normalization in the local difference module.

The results in Table 5.5 show the normalization step in the local difference module can help to improve the performance by normalizing the difference features iteratively gathered

Table 5.5: The comparison of the history module trained on laptop category with and without normalization. The model is using 6-Pack as backbone applied with our temporal model trained with a length 4 history sequence.

	$5°5cm \uparrow$	$10\text{degree}°10cm \uparrow$	mIoU ↑	$R_{error} \downarrow$	$T_{error} \downarrow$
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

from the first frame in the history sequence to the current frame.

5.3 Summary

In this Chapter, we present the experimental results of our history module compared with two baselines, Captra and 6-Pack in MoVi-E and NOCS-REAL275 datasets. We show the results in 6 and 3 categories in NOCS-REAL275 and MoVi-E datasets, respectively, measured by 5 metrics discussed in Chapter 4. We also provide some examples to illustrate the performance of our model in several challenging scenes where the object is occluded or in fast motion. Our model is more robust and has less prediction error when encountering these issues compared to the two baselines. We also visualize the difference feature learned from the model through cosine similarity. The sampled points' difference feature should be relatively high for the target object comparing to the outliers based on the difference features observed in the history sequence. If the point does not belong to the target object, e.g. due to occlusions, which does not appear in the history, the feature difference will increase. Based on this, the robustness of our model can improve by suppressing the influence of other points and focusing on the target object. The difference feature is generated by fusing both pixels and point differences. The history difference in both the

global difference module and the local difference module can be perceived by the model to learn to refine the missing features of the current frame blocked by an occlusion.

Then we conduct an ablation study of two important components of our history model. One is the length of the history sequence and the other one is the normalization using softmax in the local difference module. For the length, we evaluate our model using history sequence with a length of 2 to 8. For the NOCS-REAL275 dataset, the different length does not make too much difference while the scenes in the NOCS-REAL275 dataset are more static than MoVi-E dataset where the objects is placed on the ground and the camera is the only thing that is moving. However, in MoVi-E, the model gains significant improvement when the length of the history sequence increases to 6 and 7. There is a huge gap between our normalized history model and the one without normalization. This means the normalization makes the model easier to learn the difference feature and prevents large values when the history sequence is long. In the next Chapter, we will make a conclusion for this thesis.

Chapter 6

Conclusion

In this Chapter, we conclude our work in this thesis. In Chapter 6.1, we will summarize the content of our thesis in each Chapter. In Chapter 6.2, we list the main contribution of our work. In Chapter 6.3, we will analyze some potential future works.

6.1 Summary

In the first Chapter, we discuss the definition and motivation for 6DoF pose detection and tracking in this thesis. We start from different aspects including AR (Augmented Reality), autonomous driving, SLAM (Simultaneous Localization and Mapping) and robotics applications to amplify the importance of 6DoF pose prediction in these fields. Then, we talk about the categorization of 6D pose estimation problems and the common methods to solve them. We specify what the remaining problems these methods face and where the idea for our thesis originates from.

In Chapter 2, we review the related works of category-level, instance-level 6DoF pose estimation and tracking. We first give a brief background of some commonly used points alignment methods and concepts, which will be further used in other methods we introduce in later Chapters. Then, we classify these methods into keypoint-based, retrieval-baesd, refinement-based and regression-based methods according to the scheme they follow to conduct the pose prediction or tracking as well as some widely used benchmarks. Our two baselines 6-Pack [100] and Captra [115] belong to category-level tracking methods. We analyze the pros and cons of these methods and found that they are lacking temporal processing in pose estimation tasks. We also cover the introduction of essential methods that inspire our work in both 2D and 3D.

In Chapter 3, we introduce our difference-based temporal model which consists of local and global difference modules. Unlike 2D temporal models, due to the random sampling strategy in each frame, the correspondence across frames cannot be built. In this case, we incorporate KNN to approximate the corresponding point in the reference frame as the MaxPooling value of its K nearest neighbors in the reference frame. We learn the pixels and points difference and fuse them in a bidirectional manner to generate a feature-wised weight to conduct a weighted summation of history frames. The global difference module will gather the weighted features from the history sequence on average, while the local difference module updates the features in history sequence iteratively. The feature of the current frame, global difference and local difference will be concatenated to generate the final refined feature representation of current frame. We also emphasize the importance of transforming the history point clouds into the same coordinate system, which makes the difference calculation meaningful.

Chapter 4 is the basic setup of our experiments. We show the number of data we used to train and test for the two datasets (NOCS-REAL275 and MoVi-E). For NOCS-REAL275 dataset, we only use the real data. The data augmentation strategy is also introduced. We randomly change the background of the frame to increase the number of noisy points. We introduce the symmetry handling strategy and loss functions used by the two baselines (6-Pack and Captra).

In Chapter 5, we analyze the experimental results of our model compared with 6-Pack and Captra in NOCS-REAL275 and MoVi-E. Examples, metrics and difference feature visualization are shown to present the performance of our model. Our model outperforms the two baselines in most of the metrics and has a better overall result. The examples show our model is more robust to occlusions and cluttered scenes where the rotation and translation error are smaller than the two baselines in the NOCS-REAL275 and the MoVi-E datasets. Then, we present an ablation study about the length of the history and the normalization in the local difference module. In complex scenes (MoVi-E) the model can be impacted by the length of the history sequence.

6.2 Contribution

The contributions of this thesis are as follows:

- We propose a difference calculation that can be applied on random sampled points in two consecutive frames based on KNN (K Nearest Neighbor).

- We propose two modules to collect difference features from history sequence. One is the global difference, which sums all the difference features in the history sequence and the other is the local difference, which iteratively updates the features in the history sequence.
- The proposed difference-based modules can be applied to any 6DoF pose tracker that takes video sequence as input to refine the features from points and pixels to further learn different tasks. This method also consider the combination of pixels features and points features.
- Based on our experimental results, our model is applied to the two baseline models (Captra and 6-Pack) and outperforms the respective baselines.

6.3 Limitation and Future Works

Although the proposed difference-based temporal model applied on the two 6D pose tracking networks outperforms the baselines, there are many drawbacks needed to be improved.

Static History

The first limitation is the history sequence. In our sliding window design, the history sequence will update after processing a new frame, that is adding that frame into the history sequence and removing the earliest frame in the sequence if the number of frames is greater than the length of the sequence. One of the drawbacks of this first-in-first-out strategy are proentially some "bad" instances in the sequence, which is unsuitable for

difference extraction to enhance the current feature. This may occur, for example, if the object is heavily occluded and only a corner is visible. An other case is if two frames are too similar, the difference is useless. Ideally, we would use a dynamic strategy to select frames to store. For similar frames, one option is using some feature distance calculation methods to measure the similarity like the cosine similarity used in Chapter 5.

Category Generalization

The learning-based category-level 6DoF pose tracking models are mostly category-specific, which means the model corresponds to the specific category. The model needs to be retrained if there is an instance of a new category. In our thesis, for the 6 categories in the NOCS-REAL275 dataset and 3 categories in MoVi-E dataset, we train 9 models in total to get the final evaluation results which is time consuming and inefficient. During inference, if we want to track enough categories, we need to load that amount of models, which will occupy considerable resources. The method to tackle this problem is to train one model that can be generalized to all the categories. Because in category-level tasks, the prior knowledge is unknown, how to learn the category-specific features is important. This field is not well explored in category-level 6D pose tracking and considering how to incorporate it into the temporal model is a potential direction.

Memory Limit

As discussed in Chapter 5, we cannot store too many frames due to the limited memory of GPU. Because our model calculates the difference of pixels and points according to their

corresponding 3D coordinates. So the temporal module needs to store the point clouds and their images, which occupies a lot of memory. One of the options can be learning location-invariant features. These features would be generated for the two consecutive frames. They can be used for direct difference calculation without computing the distance although the sampled points are random in each frame. One may also be able to change the point sampling strategy to sample the corresponding points between two frames.

References

[1] Mohamed H Abdelpakey, Mohamed S Shehata, and Mostafa M Mohamed. Denssiam: End-to-end densely-siamese network with self-attention model for object tracking. In *Advances in Visual Computing: 13th International Symposium, ISVC 2018, Las Vegas, NV, USA, November 19–21, 2018, Proceedings 13*, pages 463–473. Springer, 2018.

[2] Anshuman Agarwal, Shivam Gupta, and Dushyant Kumar Singh. Review of optical flow technique for moving object detection. In *2016 2nd international conference on contemporary computing and informatics (IC3I)*, pages 409–413. IEEE, 2016.

[3] K Somani Arun, Thomas S Huang, and Steven D Blostein. Least-squares fitting of two 3-d point sets. *IEEE Transactions on pattern analysis and machine intelligence*, (5):698–700, 1987.

[4] Ronald T Azuma. A survey of augmented reality. *Presence: teleoperators & virtual environments*, 6(4):355–385, 1997.

[5] Vassileios Balntas, Andreas Doumanoglou, Caner Sahin, Juil Sock, Rigas Kouskouridas, and Tae-Kyun Kim. Pose guided rgbd feature learning for 3d object pose estimation. In *Proceedings of the IEEE international conference on computer vision*, pages 3856–3864, 2017.

[6] Alexandre Bernardino and José Santos-Victor. Binocular tracking: integrating perception and control. *IEEE Transactions on Robotics and Automation*, 15(6):1080–1094, 1999.

[7] Garrick Brazil and Xiaoming Liu. M3d-rpn: Monocular 3d region proposal network for object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9287–9296, 2019.

[8] Sean Campbell, Niall O'Mahony, Lenka Krpalcova, Daniel Riordan, Joseph Walsh, Aidan Murphy, and Conor Ryan. Sensor technology in autonomous vehicles: A review. In *2018 29th Irish Signals and Systems Conference (ISSC)*, pages 1–4. IEEE, 2018.

[9] Patrick Cavanagh and George A Alvarez. Tracking multiple targets with multifocal attention. *Trends in cognitive sciences*, 9(7):349–354, 2005.

[10] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[11] Dengsheng Chen, Jun Li, Zheng Wang, and Kai Xu. Learning canonical shape space for category-level 6d object pose and size estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11973–11982, 2020.

[12] Wei Chen, Jinming Duan, Hector Basevi, Hyung Jin Chang, and Ales Leonardis. Pointposenet: Point pose network for robust 6d object pose estimation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2824–2833, 2020.

[13] Xiaozhi Chen, Kaustav Kundu, Ziyu Zhang, Huimin Ma, Sanja Fidler, and Raquel Urtasun. Monocular 3d object detection for autonomous driving. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2147–2156, 2016.

[14] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1907–1915, 2017.

[15] Changhyun Choi and Henrik I Christensen. Real-time 3d model-based tracking using edge and keypoint features for robotic manipulation. In *2010 IEEE International Conference on Robotics and Automation*, pages 4048–4055. IEEE, 2010.

[16] Cristina Garcia Cifuentes, Jan Issac, Manuel Wüthrich, Stefan Schaal, and Jeannette Bohg. Probabilistic articulated real-time tracking for robot manipulation. *IEEE Robotics and Automation Letters*, 2(2):577–584, 2016.

[17] Taco S Cohen, Mario Geiger, Jonas Köhler, and Max Welling. Spherical cnns. *arXiv preprint arXiv:1801.10130*, 2018.

[18] Daniel Cores Costa, Víctor Manuel Brea Sánchez, and Manuel Felipe Mucientes Molina. Short-term anchor linking and long-term self-guided attention for video object detection. 2021.

[19] Daniel Cores Costa. *Spatio-temporal convolutional neural networks for video object detection*. PhD thesis, Universidade de Santiago de Compostela, 2022.

[20] Yiming Cui, Liqi Yan, Zhiwen Cao, and Dongfang Liu. Tf-blender: Temporal feature blender for video object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8138–8147, 2021.

[21] Xinke Deng, Arsalan Mousavian, Yu Xiang, Fei Xia, Timothy Bretl, and Dieter Fox. Poserbpf: A rao–blackwellized particle filter for 6-d object pose tracking. *IEEE Transactions on Robotics*, 37(5):1328–1342, 2021.

[22] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[23] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2758–2766, 2015.

[24] Bo Du, Yujia Sun, Shihan Cai, Chen Wu, and Qian Du. Object tracking in satellite videos by fusing the kernel correlation filter and the three-frame-difference algorithm. *IEEE Geoscience and Remote Sensing Letters*, 15(2):168–172, 2017.

[25] Karin Erdmann and Mark J Wildon. *Introduction to Lie algebras*, volume 122. Springer, 2006.

[26] Zhaoxin Fan, Yazhi Zhu, Yulin He, Qi Sun, Hongyan Liu, and Jun He. Deep learning on monocular object pose detection and tracking: A comprehensive overview. *ACM Computing Surveys*, 55(4):1–40, 2022.

[27] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[28] Mathieu Garon and Jean-François Lalonde. Deep 6-dof tracking. *IEEE transactions on visualization and computer graphics*, 23(11):2410–2418, 2017.

[29] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[30] Klaus Greff, Francois Belletti, Lucas Beyer, Carl Doersch, Yilun Du, Daniel Duckworth, David J Fleet, Dan Gnanapragasam, Florian Golemo, Charles Herrmann, Thomas Kipf, Abhijit Kundu, Dmitry Lagun, Issam Laradji, Hsueh-Ti (Derek) Liu, Henning Meyer, Yishu Miao, Derek Nowrouzezahrai, Cengiz Oztireli, Etienne Pot, Noha Radwan, Daniel Rebain, Sara Sabour, Mehdi S. M. Sajjadi, Matan Sela, Vincent Sitzmann, Austin Stone, Deqing Sun, Suhani Vora, Ziyu Wang, Tianhao Wu,

Kwang Moo Yi, Fangcheng Zhong, and Andrea Tagliasacchi. Kubric: a scalable dataset generator. 2022.

[31] Hussein Haggag, Mohammed Hossny, Despina Filippidis, Douglas Creighton, Saeid Nahavandi, and Vinod Puri. Measuring depth accuracy in rgbd cameras. In *2013, 7th International Conference on Signal Processing and Communication Systems (ICSPCS)*, pages 1–7. IEEE, 2013.

[32] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[33] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[34] Yisheng He, Haibin Huang, Haoqiang Fan, Qifeng Chen, and Jian Sun. Ffb6d: A full flow bidirectional fusion network for 6d pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3003–3013, 2021.

[35] Yisheng He, Wei Sun, Haibin Huang, Jianran Liu, Haoqiang Fan, and Jian Sun. Pvn3d: A deep point-wise 3d keypoints voting network for 6dof pose estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11632–11641, 2020.

[36] Paul Hebert, Nicolas Hudson, Jeremy Ma, Thomas Howard, Thomas Fuchs, Max Bajracharya, and Joel Burdick. Combined shape, appearance and silhouette for simultaneous manipulator and object tracking. In *2012 IEEE International Conference on Robotics and Automation*, pages 2405–2412. IEEE, 2012.

[37] Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Stefan Holzer, Gary Bradski, Kurt Konolige, and Nassir Navab. Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes. In *Computer Vision–ACCV 2012: 11th Asian Conference on Computer Vision, Daejeon, Korea, November 5-9, 2012, Revised Selected Papers, Part I 11*, pages 548–562. Springer, 2013.

[38] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[39] Berthold KP Horn and Brian G Schunck. Determining optical flow. *Artificial intelligence*, 17(1-3):185–203, 1981.

[40] Tingbo Hou, Adel Ahmadyan, Liangkai Zhang, Jianing Wei, and Matthias Grundmann. Mobilepose: Real-time pose estimation for unseen objects with weak shape supervision. *arXiv preprint arXiv:2003.03522*, 2020.

[41] Hou-Ning Hu, Qi-Zhi Cai, Dequan Wang, Ji Lin, Min Sun, Philipp Krahenbuhl, Trevor Darrell, and Fisher Yu. Joint monocular 3d vehicle detection and tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5390–5399, 2019.

[42] Rachel Huang, Jonathan Pedoeem, and Cuixian Chen. Yolo-lite: a real-time object detection algorithm optimized for non-gpu computers. In *2018 IEEE international conference on big data (big data)*, pages 2503–2510. IEEE, 2018.

[43] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[44] Jamshed Iqbal, R Ul Islam, Hamza Khan, et al. Modeling and analysis of a 6 dof robotic arm manipulator. *Canadian Journal on Electrical and Electronics Engineering*, 3(6):300–306, 2012.

[45] Michel Jaboyedoff, Thierry Oppikofer, Antonio Abellán, Marc-Henri Derron, Alex Loye, Richard Metzger, and Andrea Pedrazzini. Use of lidar in landslide investigations: a review. *Natural hazards*, 61:5–28, 2012.

[46] Boyuan Jiang, MengMeng Wang, Weihao Gan, Wei Wu, and Junjie Yan. Stm: Spatiotemporal and motion encoding for action recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2000–2009, 2019.

[47] Wolfgang Kabsch. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, 32(5):922–923, 1976.

[48] Daniel Kappler, Franziska Meier, Jan Issac, Jim Mainprice, Cristina Garcia Cifuentes, Manuel Wüthrich, Vincent Berenz, Stefan Schaal, Nathan Ratliff, and Jean-

nette Bohg. Real-time perception meets reactive motion generation. *IEEE Robotics and Automation Letters*, 3(3):1864–1871, 2018.

[49] Wadim Kehl, Fabian Manhardt, Federico Tombari, Slobodan Ilic, and Nassir Navab. Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again. In *Proceedings of the IEEE international conference on computer vision*, pages 1521–1529, 2017.

[50] Wadim Kehl, Fausto Milletari, Federico Tombari, Slobodan Ilic, and Nassir Navab. Deep learning of local rgb-d patches for 3d object detection and 6d pose estimation. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part III 14*, pages 205–220. Springer, 2016.

[51] Nikunj Kothari, Misha Gupta, Leena Vachhani, and Hemendra Arya. Pose estimation for an autonomous vehicle using monocular vision. In *2017 Indian Control Conference (ICC)*, pages 424–431. IEEE, 2017.

[52] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven L Waslander. Joint 3d proposal generation and object detection from view aggregation. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–8. IEEE, 2018.

[53] Harold W Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics (NRL)*, 52(1):7–21, 2005.

[54] DooHyun Lee and InSo Kweon. A novel stereo camera system by a biprism. *IEEE Transactions on Robotics and Automation*, 16(5):528–541, 2000.

[55] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. Ep n p: An accurate o (n) solution to the p n p problem. *International journal of computer vision*, 81:155–166, 2009.

[56] Yan Li, Bin Ji, Xintian Shi, Jianguo Zhang, Bin Kang, and Limin Wang. Tea: Temporal excitation and aggregation for action recognition. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 909–918, 2020.

[57] Yi Li, Gu Wang, Xiangyang Ji, Yu Xiang, and Dieter Fox. Deepim: Deep iterative matching for 6d pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 683–698, 2018.

[58] Ming Liang, Bin Yang, Shenlong Wang, and Raquel Urtasun. Deep continuous fusion for multi-sensor 3d object detection. In *Proceedings of the European conference on computer vision (ECCV)*, pages 641–656, 2018.

[59] Jiehong Lin, Zewei Wei, Zhihao Li, Songcen Xu, Kui Jia, and Yuanqing Li. Dualposenet: Category-level 6d object pose and size estimation using dual pose network with refined learning of pose consistency. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3560–3569, 2021.

[60] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects

in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014.

[61] Yunzhi Lin, Jonathan Tremblay, Stephen Tyree, Patricio A Vela, and Stan Birchfield. Keypoint-based category-level object pose tracking from an rgb sequence with uncertainty estimation. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 1258–1264. IEEE, 2022.

[62] Lizheng Liu, Jianjun Cui, Yuxiang Huan, Zhuo Zou, Xiaoming Hu, and Lirong Zheng. A design of smart unmanned vending machine for new retail based on binocular camera and machine vision. *IEEE Consumer Electronics Magazine*, 11(4):21–31, 2021.

[63] Mason Liu, Menglong Zhu, Marie White, Yinxiao Li, and Dmitry Kalenichenko. Looking fast and slow: Memory-guided mobile video object detection. *arXiv preprint arXiv:1903.10172*, 2019.

[64] Zhaoyang Liu, Donghao Luo, Yabiao Wang, Limin Wang, Ying Tai, Chengjie Wang, Jilin Li, Feiyue Huang, and Tong Lu. Teinet: Towards an efficient architecture for video recognition. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 11669–11676, 2020.

[65] William E Lorensen and Harvey E Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Seminal graphics: pioneering efforts that shaped the field*, pages 347–353. 1998.

[66] Wenhan Luo, Junliang Xing, Anton Milan, Xiaoqin Zhang, Wei Liu, and Tae-Kyun Kim. Multiple object tracking: A literature review. *Artificial intelligence*, 293:103448, 2021.

[67] Xinzhu Ma, Shinan Liu, Zhiyi Xia, Hongwen Zhang, Xingyu Zeng, and Wanli Ouyang. Rethinking pseudo-lidar representation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIII 16*, pages 311–327. Springer, 2020.

[68] Fabian Manhardt, Wadim Kehl, Nassir Navab, and Federico Tombari. Deep model-based 6d pose refinement in rgb. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 800–815, 2018.

[69] Fabian Manhardt, Gu Wang, Benjamin Busam, Manuel Nickel, Sven Meier, Luca Minciullo, Xiangyang Ji, and Nassir Navab. Cps++: Improving class-level 6d pose and shape estimation from monocular images with self-supervised learning. *arXiv preprint arXiv:2003.05848*, 2020.

[70] Eric Marchand, Hideaki Uchiyama, and Fabien Spindler. Pose estimation for augmented reality: a hands-on survey. *IEEE transactions on visualization and computer graphics*, 22(12):2633–2651, 2015.

[71] Isidoros Marougkas, Petros Koutras, Nikos Kardaris, Georgios Retsinas, Georgia Chalvatzaki, and Petros Maragos. How to track your dragon: A multi-attentional framework for real-time rgb-d 6-dof object pose tracking. In *European Conference on Computer Vision*, pages 682–699. Springer, 2020.

[72] Jonathan Masci, Ueli Meier, Dan Cireşan, and Jürgen Schmidhuber. Stacked convolutional auto-encoders for hierarchical feature extraction. In *Artificial Neural Networks and Machine Learning–ICANN 2011: 21st International Conference on Artificial Neural Networks, Espoo, Finland, June 14-17, 2011, Proceedings, Part I 21*, pages 52–59. Springer, 2011.

[73] Hauke S Meyerhoff, Frank Papenmeier, and Markus Huff. Studying visual attention using the multiple object tracking paradigm: A tutorial review. *Attention, Perception, & Psychophysics*, 79:1255–1274, 2017.

[74] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

[75] Maher Moakher. Means and averaging in the group of rotations. *SIAM journal on matrix analysis and applications*, 24(1):1–16, 2002.

[76] Carsten Moenning and Neil A Dodgson. Fast marching farthest point sampling for point clouds and implicit surfaces. Technical report, University of Cambridge, Computer Laboratory, 2003.

[77] Arsalan Mousavian, Dragomir Anguelov, John Flynn, and Jana Kosecka. 3d bounding box estimation using deep learning and geometry. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 7074–7082, 2017.

[78] Andrew Ng et al. Sparse autoencoder. *CS294A Lecture notes*, 72(2011):1–19, 2011.

[79] Joe Yue-Hei Ng and Larry S Davis. Temporal difference networks for video action recognition. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1587–1596. IEEE, 2018.

[80] Riccardo Palmarini, John Ahmet Erkoyuncu, Rajkumar Roy, and Hosein Torabmostaedi. A systematic review of augmented reality applications in maintenance. *Robotics and Computer-Integrated Manufacturing*, 49:215–228, 2018.

[81] Xuran Pan, Zhuofan Xia, Shiji Song, Li Erran Li, and Gao Huang. 3d object detection with pointformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7463–7472, 2021.

[82] Kiru Park, Timothy Patten, Johann Prankl, and Markus Vincze. Multi-task template matching for object detection, segmentation and pose estimation using depth images. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 7207–7213. IEEE, 2019.

[83] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017.

[84] Rui Qian, Divyansh Garg, Yan Wang, Yurong You, Serge Belongie, Bharath Hariharan, Mark Campbell, Kilian Q Weinberger, and Wei-Lun Chao. End-to-end pseudo-lidar for image-based 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5881–5890, 2020.

[85] Tamas Regert, Benoit Tremblais, and Laurent David. Parallelized 3d optical flow method for fluid mechanics applications. In *Fifth international symposium on 3D data processing, visualization and transmission*, page 20, 2010.

[86] José Jeronimo Rodrigues, Jun-Sik Kim, Makoto Furukawa, Joao Xavier, Pedro Aguiar, and Takeo Kanade. 6d pose estimation of textureless shiny objects using random ferns for bin-picking. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3334–3341. IEEE, 2012.

[87] Caner Sahin and Tae-Kyun Kim. Category-level 6d object pose recovery in depth images. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0, 2018.

[88] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[89] LE Scales. *Introduction to non-linear optimization*. Springer-Verlag, 1985.

[90] Tanner Schmidt, Katharina Hertkorn, Richard Newcombe, Zoltan Marton, Michael Suppa, and Dieter Fox. Depth-based tracking with physical constraints for robot manipulation. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 119–126. IEEE, 2015.

[91] Henry Schneiderman and Takeo Kanade. A statistical method for 3d object detection applied to faces and cars. In *Proceedings IEEE Conference on Computer Vision and*

Pattern Recognition. CVPR 2000 (Cat. No. PR00662), volume 1, pages 746–751. IEEE, 2000.

[92] William R Sherman and Alan B Craig. Understanding virtual reality. *San Francisco, CA: Morgan Kauffman*, 2003.

[93] Mykhailo Shvets, Wei Liu, and Alexander C Berg. Leveraging long-range temporal relationships between proposals for video object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9756–9764, 2019.

[94] Lu Sun. *Remote Assistance for Repair Tasks Using Augmented Reality*. PhD thesis, Université d'Ottawa/University of Ottawa, 2020.

[95] Supasorn Suwajanakorn, Noah Snavely, Jonathan J Tompson, and Mohammad Norouzi. Discovery of latent 3d keypoints via end-to-end geometric reasoning. *Advances in neural information processing systems*, 31, 2018.

[96] Meng Tian, Marcelo H Ang, and Gim Hee Lee. Shape prior deformation for categorical 6d object pose and size estimation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXI 16*, pages 530–546. Springer, 2020.

[97] Shinji Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 13(04):376–380, 1991.

[98] Maria Vakalopoulou, Guillaume Chassagnon, Norbert Bus, Rafael Marini, Evangelia I Zacharaki, M-P Revel, and Nikos Paragios. Atlasnet: Multi-atlas non-

linear deep networks for medical image segmentation. In *Medical Image Computing and Computer Assisted Intervention–MICCAI 2018: 21st International Conference, Granada, Spain, September 16-20, 2018, Proceedings, Part IV 11*, pages 658–666. Springer, 2018.

[99] Virginia Vassilevska, Ryan Williams, and Raphael Yuster. Finding the smallest h-subgraph in real weighted graphs and related problems. In *Automata, Languages and Programming: 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part I 33*, pages 262–273. Springer, 2006.

[100] Chen Wang, Roberto Martín-Martín, Danfei Xu, Jun Lv, Cewu Lu, Li Fei-Fei, Silvio Savarese, and Yuke Zhu. 6-pack: Category-level 6d pose tracker with anchor-based keypoints. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10059–10066. IEEE, 2020.

[101] Chen Wang, Danfei Xu, Yuke Zhu, Roberto Martín-Martín, Cewu Lu, Li Fei-Fei, and Silvio Savarese. Densefusion: 6d object pose estimation by iterative dense fusion. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3343–3352, 2019.

[102] Fang Wang and Zijian Zhao. A survey of iterative closest point algorithm. In *2017 Chinese Automation Congress (CAC)*, pages 4395–4399. IEEE, 2017.

[103] He Wang, Srinath Sridhar, Jingwei Huang, Julien Valentin, Shuran Song, and Leonidas J Guibas. Normalized object coordinate space for category-level 6d object

pose and size estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2642–2651, 2019.

[104] Limin Wang, Zhan Tong, Bin Ji, and Gangshan Wu. Tdn: Temporal difference networks for efficient action recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1895–1904, 2021.

[105] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *European conference on computer vision*, pages 20–36. Springer, 2016.

[106] Xiangyu Wang, Soh K Ong, and Andrew YC Nee. A comprehensive survey of augmented reality assembly research. *Advances in Manufacturing*, 4:1–22, 2016.

[107] Yan Wang, Wei-Lun Chao, Divyansh Garg, Bharath Hariharan, Mark Campbell, and Kilian Q Weinberger. Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8445–8453, 2019.

[108] Eric W Weisstein. Euler angles. *https://mathworld. wolfram. com/*, 2009.

[109] Gregory F Welch. Kalman filter. *Computer Vision: A Reference Guide*, pages 1–3, 2020.

[110] Bowen Wen and Kostas Bekris. Bundletrack: 6d pose tracking for novel objects without instance or category-level 3d models. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8067–8074. IEEE, 2021.

[111] Bowen Wen, Chaitanya Mitash, Baozhang Ren, and Kostas E Bekris. se (3)-tracknet: Data-driven 6d pose tracking by calibrating image residuals in synthetic domains. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10367–10373. IEEE, 2020.

[112] Bowen Wen, Jonathan Tremblay, Valts Blukis, Stephen Tyree, Thomas Müller, Alex Evans, Dieter Fox, Jan Kautz, and Stan Birchfield. Bundlesdf: Neural 6-dof tracking and 3d reconstruction of unknown objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 606–617, 2023.

[113] Bowen Wen, Wei Yang, Jan Kautz, and Stan Birchfield. Foundationpose: Unified 6d pose estimation and tracking of novel objects. *arXiv preprint arXiv:2312.08344*, 2023.

[114] Xinshuo Weng, Jianren Wang, David Held, and Kris Kitani. 3d multi-object tracking: A baseline and new evaluation metrics. in 2020 ieee. In *RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10359–10366.

[115] Yijia Weng, He Wang, Qiang Zhou, Yuzhe Qin, Yueqi Duan, Qingnan Fan, Baoquan Chen, Hao Su, and Leonidas J Guibas. Captra: Category-level pose tracking for rigid and articulated objects from point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13209–13218, 2021.

[116] Paul Wohlhart and Vincent Lepetit. Learning descriptors for object recognition and 3d pose estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3109–3118, 2015.

[117] Yangzheng Wu, Mohsen Zand, Ali Etemad, and Michael Greenspan. Vote from the center: 6 dof pose estimation in rgb-d images by radial keypoint voting. In *European Conference on Computer Vision*, pages 335–352. Springer, 2022.

[118] Manuel Wüthrich, Peter Pastor, Mrinal Kalakrishnan, Jeannette Bohg, and Stefan Schaal. Probabilistic object tracking using a range camera. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3195–3202. IEEE, 2013.

[119] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *arXiv preprint arXiv:1711.00199*, 2017.

[120] Zaihong Xiao, Xin Wang, Jun Wang, and Zhong Wu. Monocular orb slam based on initialization by marker pose estimation. In *2017 IEEE International Conference on Information and Automation (ICIA)*, pages 678–682. IEEE, 2017.

[121] Ruigang Yang and Zhengyou Zhang. Model-based head pose tracking with stereovision. In *Proceedings of Fifth IEEE International Conference on Automatic Face Gesture Recognition*, pages 255–260. IEEE, 2002.

[122] Xiaoqing Ye, Liang Du, Yifeng Shi, Yingying Li, Xiao Tan, Jianfeng Feng, Errui Ding, and Shilei Wen. Monocular 3d object detection via feature domain adaptation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part IX 16*, pages 17–34. Springer, 2020.

[123] Tianwei Yin, Xingyi Zhou, and Philipp Krahenbuhl. Center-based 3d object detection and tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11784–11793, 2021.

[124] Sheng Yu, Di-Hua Zhai, Yuanqing Xia, Dong Li, and Shiqi Zhao. Cattrack: Single-stage category-level 6d object pose tracking via convolution and vision transformer. *IEEE Transactions on Multimedia*, 2023.

[125] Ekim Yurtsever, Jacob Lambert, Alexander Carballo, and Kazuya Takeda. A survey of autonomous driving: Common practices and emerging technologies. *IEEE access*, 8:58443–58469, 2020.

[126] Yizhuo Zhang, Zhirong Wu, Houwen Peng, and Stephen Lin. A transductive approach for video object segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6949–6958, 2020.

[127] Yue Zhao, Yuanjun Xiong, and Dahua Lin. Recognize actions by disentangling components of dynamics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6566–6575, 2018.

[128] Linfang Zheng, Aleš Leonardis, Tze Ho Elden Tse, Nora Horanyi, Hua Chen, Wei Zhang, and Hyung Jin Chang. Tp-ae: Temporally primed 6d object pose tracking with auto-encoders. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 10616–10623. IEEE, 2022.

[129] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao 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.

[130] Wenjun Zhu, Haida Feng, Yang Yi, and Mengyi Zhang. Fcr-tracknet: Towards high-performance 6d pose tracking with multi-level features fusion and joint classification-regression. *Image and Vision Computing*, 135:104698, 2023.