RECOVERING 6D POSE OF RIGID OBJECT FROM POINT CLOUD AT THE LEVEL OF INSTANCE AND CATEGORY

Wei Chen

Supervised by Dr. Aleš Leonardis

A thesis submitted to
the University of Birmingham
for the degree of
DOCTOR OF PHILOSOPHY

Robotic and Computer Vision Group
School of Computer Science
College of Engineering and Physical Sciences
University of Birmingham
Augest 2021

ABSTRACT

Estimating the 3D orientation and 3D position, i.e. 6D pose, of rigid objects plays an essential role in computer vision tasks. This field has been made huge progress with the development of deep learning techniques. However, some challenges still need to be addressed, such as occlusion, viewpoint variation, and intra-class variation in category-level pose estimation. This thesis is philosophically built upon addressing the aforementioned problems in 3D space via point cloud representation. Via addressing these problems, there are mainly three findings of this thesis: point cloud representation in 3D space is more suitable for 6D object pose estimation; feature design is essential to pose estimation tasks; rotation representation has an important impact on pose estimation results. For the first finding, all the three proposed pipelines use RGB information for the 2D location of the target object and estimate the 6D pose of the object in the detected region with point cloud input. The experimental results show that this fashion focuses the network learning useful 3D information from the point cloud, which is useful to pose estimation tasks. As to the second finding, for different challenges, we design different features. For the occlusion challenge at the instance level, we propose to extract dense local features by regressing point-wise vectors for pose hypotheses generation and select the best pose candidate based on 3D geometry constraints by RANSAC. Via this fashion, the network can better utilize the local 3D information from the point cloud. However, due to a large number of hypotheses, this generation and verification strategy is time-consuming. Then to mitigate the time-consuming and viewpoint variation problem, we propose the embedding vector feature. With this newly designed feature, the proposed method can effectively extract the viewpoint information from the train-

ing dataset, which leads to the fast, over 20fps, 6D pose estimation of the target object. However, we still need a large amount of labelled data to train the model. To make the model less dependent on the labelled data, this thesis then addresses the category-level pose estimation problem. To handle the intra-class variation in the categorical 6D pose estimation task, we propose to use 3D graph convolution for category-level latent rotation feature learning. Finally, to fully decode the rotation information from the latent feature, we employ two decoders based on the newly designed rotation representation. With this new rotation representation and learned feature, the proposed method achieves state-of-the-art performance with almost real-time speed at the level of category.

DEDICATION

Dedicated to the research of human being.

ACKNOWLEDGMENTS

First, I would like to thanks my supervisor Prof. Aleš Leonardis for his careful guidance in my research. His guidance teaches me to be a good researcher and conduct research in an efficient way. Then I would like to thank my co-supervisor Dr. Hyung Jin Chang for his support to my research and life. I also want to thanks my colleges, friends, and family. It is their continuous help that gives me motivation when I was suffering setbacks in my Ph.D. study. Last, I want to thank my beloved wife for her endless support to my research activity, for her loving care when I was in a difficult time during my Ph.D. Her love penetrates time and space like gravity, from China to Britain.

Nature and nature's laws lay hid in the night. God said, Let Newton be! and all was light!

Alexander Pope

Contents

List of Figures

List of Tables

List of Notations

M_c	coordinates in camera space
M_o	coordinates in object space
$\mathbf{R}$	object rotation matrix
$\mathbf{T}$	object translation vector
$\mathcal{M}$	object 3D model
$\mathfrak{p}$	object point in scene
$\mathbf{p}^{key}$	keypoint
p	image pixel
N_O	number of object points
$\mathcal{P}$	point set
$\mathcal{L}$	loss function
$\mathcal{K}$	camera parameters

List of Acronyms

3DGC 3D Graph Convolution.

6D 6 Degree of Freedom.

ADD(-S) Average Diameter Distance (-Symmetry).

AI Artificial Intelligence.

BBX Bounding box.

CNN Convolutional Neural Network.

EVF embedding vector feature.

FS fast shape-based.

G2L global to local.

ICP Iterative Closest Point.

PnP Perspective-n-Point.

RANSAC Random Sample Consensus.

VDR vector-based decoupled rotation.

VSD Visible Surface Discrepancy.

Chapter One

Introduction

Computer vision plays an essential role in the development of Artificial Intelligence (AI) technology. Generally, the goal of computer vision is to extract useful information from digital inputs such as images and video. However, the machine is different from the human being. In the machine, the images and videos are represented as matrices with different values. Then the main challenge for computer vision tasks is to extract human-readable information such as object categories and locations, with this pure digital input.

Among the computer vision tasks, 6 Degree of Freedom (6D) object pose estimation is an important one, since it is affected by the major challenges of computer vision, such as dramatic object appearance changes subject to conditions like viewpoint variation, occlusion, and illumination changes. In this thesis, we focus on solving the challenges of 6D object pose estimation in real-world scenes at the level at instance and category.

1.1 Motivation

Estimating the 6D pose of the rigid object, which consists of 3D orientation and 3D position of the target object in camera-centric coordinate, plays a very important role in

the field of computer vision. In the last decades, it has attracted greater attention for its significance. In recent years, increasing ubiquity of the low-cost depth data collection equipment, such as Kinect [Zhang, 2012] and Intel® RealSense [Keselman et al., 2017], further boost the research of 6D object pose. In the following subsections, we will describe how the 6D object pose facilitates the related areas.

1.1.1 Robotics

Object pose estimation can be applied to robotics in many fashions, such as object grasping and navigation. For example, in many real world applications, such as in assembly lines or automated warehouses [Doumanoglou et al., 2014, Morrison et al., 2018, Tremblay et al., 2018, Zhu et al., 2014], the robots need to grasp specific objects. In these situations, the robot has to localize the object relative to itself based on orientation and translation, which means the pose of the object needs to be accurately estimated by the robot.

1.1.2 Augmented Reality

Augmented reality means overlaying real camera footage with virtual content such that the real and virtual world blend together [Brachmann, 2018, Burdea and Coiffet, 2003, Marchand et al., 2016, Marder-Eppstein, 2016]. For example, pose estimation can be used to align virtual objects in a real environment that enables engineers to examine a virtual object, such as a product concept, in detail in a natural fashion. The engineer would wear augmented reality glasses and move around the virtual object as if it were real. This offers a very natural mode of navigation and, potentially, interaction.

1.1.3 Autonomous Driving

As the name suggests, autonomous driving means that a car can be driven without human manipulation. It is an important technique to decrease the number of traffic accidents. Therefore, one of the essential tasks in this field is to effectively detect humans and other vehicles. With the help of the 3D object detection technique, which is similar to 6D pose estimation, the machine can better understand its surroundings and make the right choices.

1.2 Problem Formulation

To define the pose of an object in 3D space, usually in camera coordinate space, we need to define six degrees of freedom (shown in Figure 1.1). The translation that defines the location of the object in 3D space has three degrees of freedom. However, this is not enough, since the object might be arbitrarily rotated. Therefore, we need to define the remaining degrees of freedom for rotation to measure the relative orientation to the three principal axes of camera coordinate. Then we have (shown in Figure 1.1):

$$M_c = \mathbf{R}M_o + \mathbf{T}, \tag{1.1}$$

where M_c and M_o represent object coordinate in camera space and object space, respectively. **R** denotes the rotation and **T** denotes the translation.

6D pose can be estimated from different object forms, such as rigid object pose estimation and deformable object pose estimation. In this thesis, we focus on estimating the 6D pose of rigid objects. For rigid objects, we assume that the objects have no moving parts and deformation parameters. In this case, a rigid body transformation can represent object 6D pose adequately. For 6D pose estimation, we have instance-level and category-level tasks. For instance-level pose estimation, we assume that the same object instance is shared in the training set and testing set. Therefore, there is no

intra-class variation challenge in instance-level tasks. Given an RGB-D image I where an instance O_i of the target O appears, then our goal is to estimate the rotation **R** and translation **T** of the object instance O_i relative to the camera coordinate system.

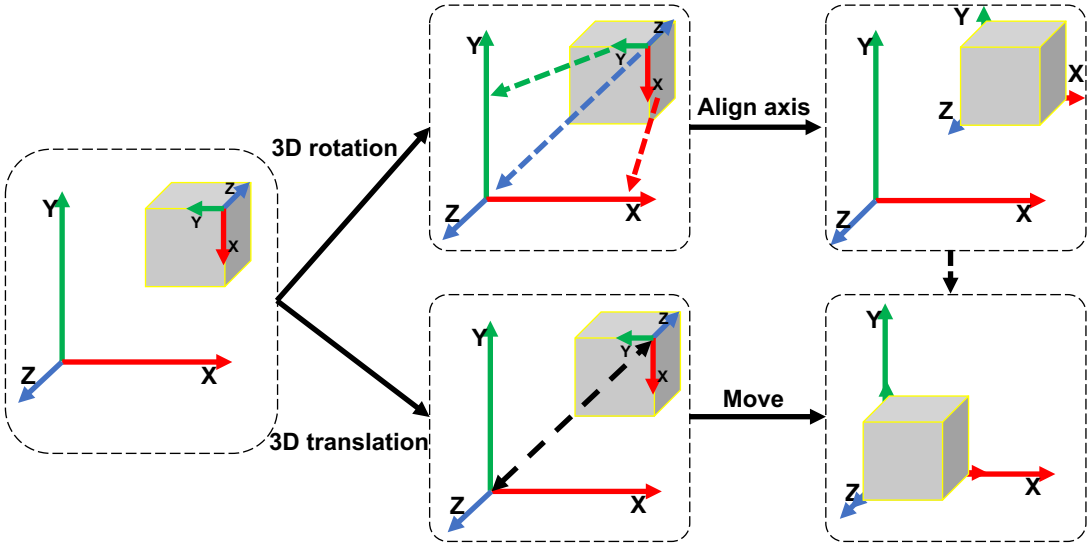


Figure 1.1: **Definition of 6D pose**. 6D pose consists of 3D rotation and 3D translation. With 3D rotation, we can align the local object coordinate system with the global camera coordinate; with 3D translation, we can know how far the object is away from the camera center.

In category-level tasks, we assume that the same object category is shared, while the object instances are different in the training set and testing set. This means we also need to estimate the shape variation of the object instance. To simplify the problem, we assume that the shape variation can be denoted by a 3D vector S. Then for category-level pose estimation, given an RGB-D image I where an instance O_i of the target O in category C appears, our goal is to estimate the rotation **R** and translation **T** of the object instance O_i relative to the camera coordinate system and also the shape variation S of O_i.

1.3 Challenges in Instance- and Category-Level Pose Estimation

Although great progress has been made on object 6D pose estimation, there still exist many challenges in this field. According to the level of the tasks, we categorize these challenges into instance-level and category-level.

1.3.1 Instance-Level Challenges

Illumination Changes:

Object appearance has a close relationship to the light source. Therefore, illumination changes can have a big impact on object appearance, which will pose a challenge on learning methods based on RGB information. Furthermore, in some cases, the light source may create shadows for original target objects. We humans can easily detach the target objects from these situations. However, the algorithms in the computer vision community do not have this ability, which means it is still challenging for people to design a method that is robust to illumination changes. To alleviate this issue, in this thesis, we build our methods based on the pipeline that uses RGB for 2D location (which is easier than 6D pose estimation) and uses point cloud for 6D pose estimation.

Occlusion:

Occlusion is one of the most common challenges in computer vision tasks. In the real-world 6D pose estimation task, it is also an inevitable problem. Occlusion occurs when the target object instance is partly or completely occluded by other object instances, which will lead to a performance decrease. There are general two methodologies to handle the partly occlusion in pose estimation tasks. On the one hand, one could try

to model the occlusion in the training stage with generated occluded training examples. On the other hand, one could design a part-feature-based approach that predicts the 6D pose of the target object instance from the observed parts. In this thesis, to handle the occlusion, we propose to extract dense local features from the point cloud by regressing point-wise vectors pointing to the keypoints. Then we use these oriented vectors to generate pose hypotheses and select the best pose candidate based on 3D geometry constraints by RANSAC. Via this fashion, the network can better utilize the local 3D information that is useful in occlusion scenarios.

Viewpoint Variation:

In testing scenes, the target objects can locate in any poses which distribute from 0° to 360° relative to X, Y, and Z axis of the camera coordinate. However, in the training stage, the number of training examples is limited, which can only cover part of the poses distribution. Therefore, the viewpoint variation of testing scenes poses a challenge for 6D object pose estimation. How to extract effective viewpoint information from limited training examples is still challenging. In this thesis, to effectively extract the viewpoint information from the point cloud, we propose to extract latent embedding vector features by encoding the point cloud to point-wise vectors. To fully decode the viewpoint information from the learned latent embedding vector features, we also design a rotation residual estimator to narrow the gap between the estimated rotation and the ground truth.

1.3.2 Category-Level Challenges

Please note, the challenges or problems existing in instance-level 6D object pose estimation tasks can also occur in category-level 6D object pose estimation since we can take instance-level tasks as special cases of category-level tasks. One main challenge in the

category-level 6D object pose estimation task is the intra-class variation that includes object shape variation and color variation. Although different object instances belong to the same category in category-level 6D object pose estimation, their shape and color variation can be seen in RGB channels and depth channel, respectively. Especially, the color variation, which poses a big challenge in category-level pose estimation. As shown in Figure 1.2, for the same category, bowl, the color property is very different among different object instances.

To handle intra-class variation, in this thesis, we introduce three novelties: i) the 3D Graph Convolution (3DGC) is introduced for category-level latent rotation feature learning via observed points reconstruction; ii) the vector-based decoupled rotation (VDR) representation is designed for effective category-level rotation feature decoding; iii) an online augmentation technique is proposed for training data augmentation in 3D space. With these novelties, the proposed method achieves state-of-the-art performance with almost real-time speed at the level of category.

Figure 1.2: **Color variation**. Top row: three bowl instances randomly chosen from the NOCS-REAL dataset. Bottom row: three bowl instances randomly cropped from the internet image search results (using the keyword 'bowl'). The color is varied in the same category.

1.4 Methods and Contributions

To address the aforementioned challenges, many researchers have proposed many compelling methods. We first analyse the limitations of the existing methods and then provide a brief description of the proposed methods. Finally, we describe the connection and differences among the three proposed methods.

1.4.1 Proposed Method 1

In this part, we use the point cloud representation to estimate the 6D pose of rigid objects in 3D space. Previous 6D object pose estimation methods [Oberweger et al., 2018, Peng et al., 2018, Rad and Lepetit, 2017, Xiang et al., 2017] are mainly focusing on estimating pose from images (RGB or depth) using a 2D Convolutional Neural Network (CNN).

However, there are some limitations of this common fashion. First, when the target object is in the clutter, which is common in real-world scenes, it is usually difficult for CNN to do segmentation. The reason is that the background clutter always contains the same color information as the target object that will confuse the CNN, and pixels from distant objects can be near-by to each other, which also makes it more difficult to separate different objects. Second, since the depth image and RGB image are associated with camera intrinsic parameters, a learning model trained by these RGB or depth images needs to be applied in the scene that is captured by the same camera. This will limit the application of the trained model. The reason is that even with the same rotation and translation, the image observation can be very different under different camera parameters. We show some examples in Figure 1.3. Although we can transfer the images from different camera parameters to the camera parameter used in the training stage, we inevitably lose some pixel information. This is because the size of the image changed during the transfer and there exist some black holes in the depth image.

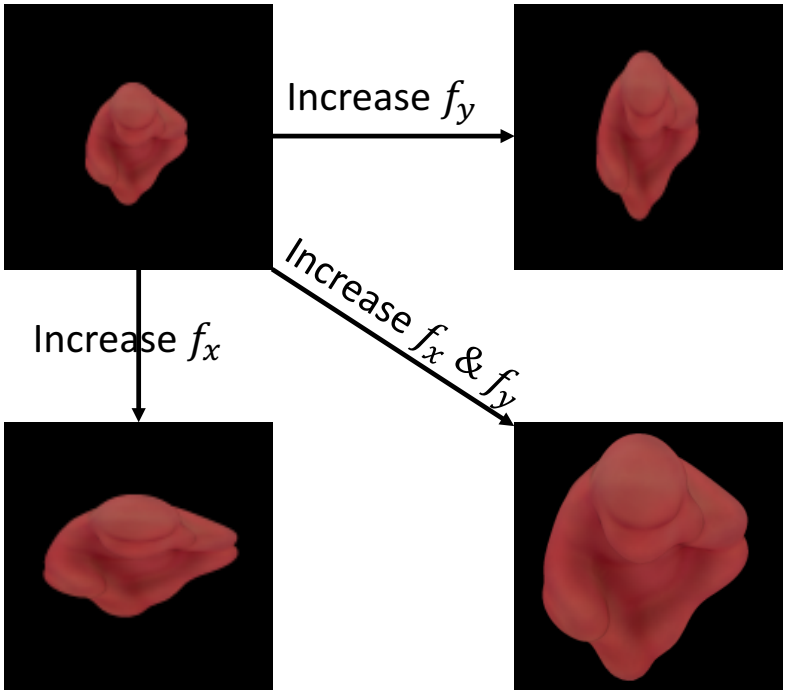


Figure 1.3: **Camera with different parameters**. Illustration of different intrinsic camera parameters with the same pose. f_x and f_y are the focal length. When increasing f_x, the object becomes wider, and increasing f_y, the object becomes longer.

To overcome these limitations, we propose to use the 3D point cloud to train our model. A point cloud is denoted as a set of 3D points, where each point contains the 3D coordinate plus extra information, such as normal, color, etc. There are several advantages of point cloud: First, 3D point clouds lay in 3D space, where we only need to care about the points that are close to the target points. This means the segmentation of different objects is easier than that in 2D images. Second, the point cloud is not associated with camera parameters, which means the model trained via point cloud will have better generalization performance.

Some works also employed point cloud to estimate 6D object pose [Qi et al., 2018, Wang et al., 2019]. However, they focus on using global features extracted from the point clouds derived from the depth image of the object, which is sensitive to the occlusion scene. Instead, we propose a novel framework that focuses on local 3D shape information from the point cloud. This makes our framework more robust to the occlu-

sion and clutter. The contributions of our proposed method are:

- We present a novel deep learning approach for 6D pose estimation which learns local 3D vector-field representation to account for object 3D local geometry information as well as localize 3D keypoints. Different from previous point cloud methods, our method focuses on local information that makes our method more robust to the occlusion scene.
- We propose a scoring mechanism that utilizes the geometry constraints from the observed depth image to select the best pose hypothesis from those are generated by predicted 3D unit vectors.

1.4.2 Proposed Method 2

The existing methods suffered from inefficient viewpoint feature extraction that leads to the fact that they need elaborate post-hoc refinement steps to better utilize the 3D viewpoint information in the depth image that will hinder the real-time performance of the algorithm.

There are mainly two types of refinement methods: Iterative Closest Point (ICP) [Chen and Medioni, 1992] algorithm and pose hypotheses generation with verification. However, there are two main issues of ICP. First, since the optimization procedure of ICP depends on the distance between local point pairs, it tends to fall into local minima in complicated scenes. Second, ICP is an iterative optimization method, which means it will take time to access the optimal value. Due to such iterative and local properties, the ICP algorithm cannot guarantee the convergence to a global minimum even with the compromise of increasing running time.

Some other methods [Brachmann et al., 2016, Chen et al., 2020, Li et al., 2018] relied on pose hypothesis generation and hypotheses verification to get a more accurate

pose from some pose candidates. However, this procedure is usually time-consuming, we need to check every hypothesis, and the verification procedure is mainly based on calculating the closest distance of each point. Although [Krull et al., 2017] proposed an efficient method to process a large number of hypotheses, their method still cannot meet the real-time requirement.

Some methods [Li et al., 2019, Peng et al., 2018, Rad et al., 2018] can run at fast speed with accurate pose estimation results without refinement, while they only take RGB images as input. RGB information is sensitive to illumination changes and background clutter.

To overcome these problems and better extract the 3D viewpoint information from depth images, we propose a novel feature learning method for efficient viewpoint information extraction in global to local (G2L) fashion. Via this novel feature extraction mechanism, the proposed pipeline runs in real-time on the benchmark dataset with state-of-the-art performance.

In summary, the contributions of this work are:

- We propose orientation-based point-wise embedding vector feature (EVF), which better utilize viewpoint information than the conventional global point features.
- We propose a rotation residual estimator to estimate the residual between predicted rotation and ground truth, which further improves the accuracy of rotation prediction.

1.4.3 Proposed Method 3

In this part, we focus on the category-level 6D object pose estimation. As described in Section 1.3.2, the major challenge in category-level 6D object pose estimation is intra-class variation including color variation and shape variation.

To mitigate the intra-class issue, [Sahin and Kim, 2018] used 3D skeleton structures to derive shape-invariant features from depth input via the random forest, however, due to inefficient feature extraction, a hypothesis generation/verification procedure is needed to access accurate pose, which will increase the running time. Most deep learning based methods employed a big synthetic dataset to learn the color variation and a big 3D model repository with 1085 3D models to handle the shape variation.

Although these deep learning based methods achieved state-of-the-art performance, there are still two issues with this strategy. First, for category-level pose estimation, the benefits of using the RGB features are questionable. Since different objects in the same category have very different colors (shown in Figure 1.2), the model trained on a limited training set cannot cover that kind of variation very well. For this issue, to alleviate the color variation, we merely use the RGB features for 2D detection, while using the shape features learned with point cloud cropped from depth image for category-level pose estimation.

Secondly, such a big synthetic dataset and 3D model repository need a large storage space to store, which will increase the hardware burden for the machine. To overcome this issue, we propose a 3DGC autoencoder [Lin et al., 2020] and a new VDR representation to effectively learn the category-level pose features via observed points reconstruction of different objects instead of uniform shape mapping. We further propose an online box-cage-based 3D data augmentation mechanism to narrow the shape gap between the training set and testing set. To summarize, the main contributions of this paper are as follows:

- We propose a pipeline, named fast shape-based (FS)-Net, with 3DGC autoencoder to reconstruct the observed points for latent orientation feature learning.
- Then we design a VDR representation to fully decode the orientation information. This decoupled mechanism allows us to naturally handle the circle symmetry object.

Table 1.1: **The connection and difference among the three proposed methods.** We show the connection and difference in four aspects: architecture choice, pose prediction fashion, object level, and running speed of the corresponding pipeline.

	Proposed Method 1	Proposed Method 2	Proposed Method 3
Architecture	YOLOv3+PointNet [Qi et al., 2017]		YOLOv3+PointNet+3DGConv [Lin et al., 2020]
Pose Predition	Rotation &Translation together	Rotation &Translation separately	Decoupled Rotation &Translation separately
Object Level	Instance Level		Category Level
Running Speed	Slow: < 5fps	Fast: >20fps	

- Based on the shape similarity of intra-class objects, we propose a novel box-cage-based 3D deformation mechanism to augment the training data.

1.4.4 Connection and Difference

The major connections and differences of the three proposed methods are listed in Table 1.1. From Table 1.1 and the aforementioned description, we can see that proposed methods 1 and 2 are some kind similar. However, there are several differences between them. First, the architectures are different. Method 1 estimated rotation and translation from pose hypothesis in one network branch, while method 2 estimated the rotation and translation in two different network branches. Second, method 1 directly utilised the point-wise vectors to generate the pose hypothesis, while method 2 treated the point-wise vectors as intermediate output and used a decoder to decode the rotation information from the features trained by the point-wise vectors. Third, the pose estimation part of method 2 is end-to-end trainable, while method 1 is not.

1.5 Publications

The work of this thesis has produced the following publications:

- **Wei Chen**, Jinming Duan, Hector Basevi, Hyung Jin Chang, and Aleš Leonardis.

PonitPoseNet: Point pose network for robust 6d object pose estimation. In The IEEE Winter Conference on Applications of Computer Vision (WACV), March 2020 (spotlight).

- **Wei Chen**, Xi Jia, Hyung Jin Chang, Jinming Duan, and Aleš Leonardis. G2L-net: Global to local network for real-time 6d pose estimation with embedding vector features. In IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2020.

- **Wei Chen**, Xi Jia, Hyung Jin Chang, Jinming Duan, Linlin, Shen and Aleš Leonardis. FS-Net: Fast Shape-based Network for Category-Level 6D Object Pose Estimation with Decoupled Rotation Mechanism. In IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2021 (oral, 4.2% accept rate).

These three publications correspond to the main content of Chapters 3, 4, and 5 of this thesis. The first publication, described in Chapter 3, is about using point-wise features for occlusion scenes in the 6D object pose estimation task. The second publication, which is presented in Chapter 4, describes how the proposed method achieves state-of-the-art performance in both accuracy and speed with designed effective viewpoint feature extraction mechanism, i.e. embedding vector feature extraction in global to local fashion. The third publication, described in Chapter 5, is about estimating 6D object pose in category-level tasks with the proposed new architecture and rotation representation. In addition, we add new experiments to show that the proposed VDR representation is more suitable for pose estimation tasks at both instance level and category level.

1.5.1 Author Contribution Statements

In the above publications, the first author completed all the experiment design, code writing, result analyse, and paper draft writing. The last author is the major supervisor of the first author. The order of other authors is decided based on their contribution to the paper representation. Thanks for their effort to make the content of the papers clearer to the readers.

1.6 Thesis Structure

The structure of this thesis is organized as follows. It starts from the overview of the related works in Chapter 2, which includes the review of relevant work of 6D object pose estimation methods, the popular datasets used in this field, and the metrics that are used to evaluate the 6D pose estimation methods. In Chapter 3, we describe our first work on 6D object pose estimation with point-wise vector features and Random Sample Consensus (RANSAC) voting mechanism. Chapter 4 presents our novel real-time 6D object pose estimation pipeline. In this pipeline, we estimate the 6D pose of the target object in a global to local fashion with the embedding vector features. Then in Chapter 5, we present how we use the proposed novelties: visible points reconstruction, VDR representation, and 3D online augmentation, to fast estimate category-level 6D pose with high accuracy. Finally, in Chapter 6, we summarize the thesis and analyse the potential future work of the proposed methods.

Chapter Two

Literature Review

With the development of this field, plenty of compelling works have been proposed about 6D object pose estimation, we report the basic information of some typical methods in the pose estimation community in Table 2.1.

Table 2.1: **Typical methods.** The 'Type' means how to get the final pose from the output of trained model.

Methods	Type	Input	Dataset
[Hinterstoisser et al., 2012]	Template matching	RGBD	LINEMOD
[Oberweger et al., 2018]	Template matching	RGB	LINEMOD & OCCLUSION-LINEMOD & YCB-V
[Rad and Lepetit, 2017]	PnP	RGB	LINEMOD & TLESS & OCCLUSION-LINEMOD
[Peng et al., 2018]	PnP	RGB	LINEMOD & OCCLUSION-LINEMOD & YCB-V
[Chen et al., 2020]	Hypothesis/RANSAC	RGBD	LINEMOD & OCCLUSION-LINEMOD& YCB-V
[Xiang et al., 2017]	Regression	RGB	OCCLUSION-LINEMOD& YCB-V
[Brachmann et al., 2016]	Hypothesis/RANSAC	RGBD	LINEMOD & OCCLUSION-LINEMOD
[Wohlhart and Lepetit, 2015]	KNN algorigthm	RGBD	LINEMOD
[Kehl et al., 2016]	Voting	RGBD	LINEMOD
[Jafari et al., 2017]	Hypothesis/RANSAC	RGBD	LINEMOD & OCCLUSION-LINEMOD
[Wang et al., 2019]	Umeyama [Umeyama, 1991]	RGBD	NOCS-REAL & OCCLUSION-LINEMOD
[Chen et al., 2020]	Regression	RGBD	NOCS-REAL
[Tian et al., 2020]	Umeyama [Umeyama, 1991]	RGBD	NOCS-REAL
[Chen et al., 2021]	Kabsch [Kabsch, 1976]	RGBD	NOCS-REAL & LINEMOD

At the instance level, most methods are proposed to handle the existing challenges, such as occlusion, background clutter, and texture-less objects. However, it is

nontrivial to extend these methods to category-level pose estimation, we have to consider the intra-class issue in category-level tasks. To handle the new challenge, Wang et al. [Wang et al., 2019] proposed Normalized Object Coordinate Space (NOCS) that maps the various instance shapes to a unique shape. Then methods [Chen et al., 2020, Tian et al., 2020] focus on improving the reliability and scalability of NOCS. But these kinds of strategies are limited by the size of the training set.

In the following, we review the related works in the following aspects: 1) pose estimation methods: instance-level pose estimation approaches; 2) pose estimation methods: category-level pose estimation approaches; 3) post-hoc refinement procedure: how to access more accurate pose from initial pose; 4) pose estimation datasets: popular datasets used in this thesis; 5) pose estimation metric: popular metric used in pose estimation community.

In addition, since all the proposed pipelines are based on the detection results from the 2D detector, we provide a brief introduction for object detection methods.

2.1 Instance-Level Pose Estimation Approaches

In instance-level pose estimation, a known 3D object model is usually available for training and testing. Based on the 3D model, instance-level pose estimation can be roughly divided into three types: template matching-based, correspondences-based, and voting-based methods. Template matching methods [Hinterstoisser et al., 2012, Oberweger et al., 2018] aligned the template to the observed image or depth map via hand-crafted or deep learning feature descriptors. As they need the 3D object model to generate the template pool, their applications in category-level 6D pose estimation are limited. Correspondences-based methods trained their model to establish 2D-3D correspondences [Peng et al., 2018, Rad and Lepetit, 2017, Rad et al., 2018] or 3D-3D correspondences [Chen et al., 2020,]. Then they solved Perspective-n-Point (PnP)

[Gao et al., 2003] with 2D-3D or SVD problem with 3D-3D correspondences [Kabsch, 1976]. Some methods [Brachmann et al., 2016, Chen et al., 2020] also used these correspondences to generate voting candidates and then used RANSAC [Fischler and Bolles, 1981] algorithm for selecting the best candidate. However, the generation of canonical 3D keypoints is based on the known 3D object model that is not available when predicting the category-level pose.

2.1.1 RGB(D)-Based Pose Estimation Approaches

Template-Based Pose Estimation Approaches

In this kind of approach, the object pose is denoted by the features of a template image of the object in the template pool. During testing, if the extracted features of a test image match some template in the template pool, we can find the corresponding pose of the matched template. There are usually two steps. First, since the extracted features can not be the same as the features in the template pool, some template candidates will be chosen, then a coarse pose will be calculated from these matched templates. Second, a refinement procedure based on the ICP algorithm is adopted to access the final pose. However, it is easy to see that the size of the template pool limits the performance of the methods. To ensure performance, a large template pool is required that increases both inference time and storage burden.

Learning-Based Pose Estimation Approaches

CNNs have made huge success in image classification [Druzhkov and Kustikova, 2016, Rawat and Wang, 2017, Simonyan and Zisserman, 2014] and segmentation [He et al., 2017, Zaitoun and Aqel, 2015] tasks. Then some researchers focus on transferring the CNN techniques to 6D pose estimation tasks. For example, [Wohlhart and Lepetit,

2015] employed a CNN to extract feature descriptors for different objects with various poses in the training stage and accessed the pose via feature descriptors matching by the KNN algorithm in inference. In [Kehl et al., 2016], to handle occlusion, the authors deployed CNN-based auto-encoders to extract local features from randomly selected local patches. In testing, the final pose is accessed by local patch feature voting from learned features. [Jafari et al., 2017] estimated the 6D object pose in three steps. They first detected the target objects with the help of object instance segmentation algorithms, then they used a CNN-based auto encoder-decoder to estimate the object coordinate of each object instance, and finally, the pose is calculated by RANSAC algorithm.

The aforementioned methods are multi-step methods. Some other methods also learned to directly estimate 6D object pose or equivalent representations with CNN architectures.

In [Hu et al., 2019, Oberweger et al., 2018, Peng et al., 2018, Rad and Lepetit, 2017, Rad et al., 2018, Tekin et al., 2018], the authors trained a CNN-based model to regress the 2D keypoints, which are the 2D projection of 3D bounding box corners under corresponding poses. Then the final pose can be accessed by PnP algorithms. However, these methods do not utilize depth information which means they are sensitive to illumination changes and background clutter. When suffering illumination changes or background clutter, the features of a pixel or a patch would be ambiguous, which will hinder the performance.

When depth data is available, conventional approaches [Brachmann et al., 2014, Hinterstoisser et al., 2016, Tejani, 2014, Wohlhart and Lepetit, 2015] extract 3D features from the input RGB-D data and perform correspondence grouping and hypothesis verification. However, it is reported that the methods are not robust enough to image variations and background clutter [Shin and Balasingham, 2017, Yuan et al., 2016] or sensitive to occlusion. Recently, several deep learning-based methods [Balntas et al., 2017, Doumanoglou et al., 2016, Kehl et al., 2016, Li et al., 2018] fuse the depth input

as an additional channel to a CNN-based architecture. These approaches treated the depth channel as an additional channel, along with the RGB channels. While this approach is simple to implement, combining depth information with RGB information in this way cannot make full use of the geometric information in the data and makes it difficult to integrate information across viewpoints [Maturana and Scherer, 2015]. Instead, we convert depth maps to 3D point clouds and directly process the 3D point cloud by PointNets [Qi et al., 2017, 2018], which are shown can extract 3D geometric features efficiently.

2.1.2 Point Cloud-Based Pose Estimation Approaches

Pose Estimation Methods via Point Cloud

Recently, researchers [Qi et al., 2017, 2018] have found that process depth information in 3D space can achieve better performance via point cloud representation. A point cloud is denoted by a set of points, where each point is a vector with a 3D coordinate relative to a specific coordinate and some other channels such as color, normal, etc. Different from elements in 2D images or 3D volumetric grids, the points in the point cloud usually have no fixed order. However, the input of the conventional CNN is required to be ordered, which means point clouds cannot be directly processed by CNN. To address this, as a pioneer, PointNet [Qi et al., 2017] is the first network that can process point clouds directly. To handle the disorder of the points in the point cloud, PointNet utilizes a symmetric function, i.e. max pooling, to achieve permutation invariance for unordered point sets. PointNet takes a raw point cloud as input and learns to encode the high-level information and features from the input point cloud. The features are effective in point cloud shape classification and part segmentation [Qi et al., 2017,, Wang et al., 2018]. Many similar pose estimation works are proposed based on the PointNet. For example, some methods [Qi et al., 2018, Wang et al., 2019, Yang et al., 2018, Zhou

and Tuzel, 2018] directly performed 6D pose estimation or 3D object detection on 3D point cloud data. They use a PoinNet-like architecture to compute pose or access 3D detected results from the point cloud.

2.2 Category-Level Pose Estimation Approaches

Compared to instance-level pose estimation tasks, the major challenge of category-level pose estimation is the intra-class object variation, including shape and color variation. To alleviate the color variation, the previous methods employed an extra synthetic dataset to train the model. However, for a specific category, the possibility of color appearance is infinite. It is impossible to cover all the color possibilities in the training stage. For shape variation, [Wang et al., 2019] proposed to map the different objects in the same category to a normalized object coordinate space (NOCS) map. Then they used segmentation masks to calculate the observed points with known camera parameters. The 6D pose and size are calculated by the Umeyama algorithm [Umeyama, 1991] with the NOCS map and the observed points. [Tian et al., 2020] adopted similar method with [Wang et al., 2019], but both extra shape prior knowledge and dense-fusion features [Wang et al., 2019], instead of RGB features, are used. [Chen et al., 2020] estimated the 6D pose via the learning of a canonical shape space (CASS) with dense-fusion features [Wang et al., 2019]. The common feature of these methods is that they all use a uniform shape to handle the various shapes among different objects in one category. However, same as color variation, a large dataset is needed to cover different shapes in one category.

2.3 Post-Hoc Refinement Procedure

Here we assume that depth information is available. We focus on how different methods refine the final pose with given depth information.

2.3.1 Iterative Refinement

ICP algorithm is a widely used iterative algorithm for registering two point sets under the Euclidean metric. Many pose estimation methods [Hinterstoisser et al., 2016, Kehl et al., 2017, Xiang et al., 2017] use the ICP algorithm with the observed depth map and the estimated pose as the initial value to compute the final pose.

However, there are several limitations of the ICP algorithm. First, since it is an iterative method, the ICP algorithm is time-consuming, which will make the pose estimation algorithm slow for real-time applications. Second, the ICP algorithm needs a good initial pose. Without a good initial pose, ICP takes a longer time to converge. Furthermore, it is easy to output a locally optimal solution rather than a globally optimal solution. The reason is that the ICP algorithm only considers the distance between point and point, which is a local measurement. Third, ICP needs two different point clouds to compute pose between them. However, in the pose estimation task, the target point cloud is extracted from the depth image, which is different from the source point cloud derived from the object mesh model. Since we cannot see the points behind the object, the target point cloud only contains partial points of the object with some noise points. For source point cloud usually has no noise points and includes all points belonging to the object. It means that we need to choose a part of the points in the source point cloud for the ICP algorithm. The choosing mechanism will also increase the running time of the algorithm. Even we choose some of the points from the source point cloud, the property of the target point cloud and source point cloud for the ICP algorithm is different, which will make it more difficult to find a globally optimal solution.

2.3.2 Pose Hypotheses Verification

Some other methods [Brachmann et al., 2016, Li et al., 2018] relied on pose hypotheses generation and hypotheses verification to get a more accurate pose from an initial pose. This procedure always contains many steps: 1) generate some pose hypotheses; 2) use some geometry constraints to score the pose hypotheses; 3) choose the best pose hypothesis according to the score. For example, in [Li et al., 2018], Li et al. proposed a multi-view framework that refines the outputs of their single-view network. They first generated some pose hypotheses from their framework, then they employed the distance metric proposed by [Hinterstoisser et al., 2012] to measure the discrepancy between every two hypotheses. Finally, they chose a pose hypothesis that has the minimal average distance with other pose hypotheses. Although this procedure could improve the initial pose, it needs much time to find the best hypothesis.

2.4 Object Detection

In general, the main purpose of object detection tasks is to find the location of the target object(s). According to the detection fashion, the CNN-based detectors can be roughly divided into two categories: two-stage detectors and one-stage detectors. For two-stage detectors, in the first stage, it usually generates some object proposals, then in the second stage, the best object proposal is detected by the network. However, this kind of detector is computation redundancy in the second stage. To address this issue, one-stage detectors, such as YOLO detectors [Redmon and Farhadi, 2017, 2018] and SSD detector [Liu et al., 2016], are proposed for object detection. These one-stage detectors can detect target object(s) in real-time with high accuracy. In our work, we use the off-the-shelf detector YOLOv3 [Redmon and Farhadi, 2018] to detect the target object for 6D pose estimation tasks.

2.5 Pose Estimation Datasets

With the development of the pose estimation field, many datasets are created by researchers. Here we list some typical datasets: the LINEMDO dataset, YCB-V dataset, and NOCS-REAL dataset, which are used in our thesis work. The reason why we use these datasets is that these datasets contain many challenging scenarios such as textureless objects, occlusion, illumination changes, etc. Moreover, making use of these datasets enables comparisons with many other state-of-the-art methods. In addition, we also introduce some other datasets which can be further used for the proposed methods.

2.5.1 Instance-Level Pose Estimation Dataset

LINEMOD Dataset

There are 15 object instances in the LINEMOD dataset. The origin LINEMOD dataset contains a training set and testing set. In the training set, for each object instance, it provides synthetic images rendered by the object 3D model with various views distributed. Then in the testing set, for each object instance, the dataset provides 1100-1300 images with various poses, and pose ground truth of each image. In Figure 2.1, we show the testing image examples in the LINEMOD dataset. From Figure 2.1, we can see that the corresponding target object is in the center of the planar board, and the background objects are randomly selected for the rest of the object instances. In practice, to avoid the real-synthetic gap for deep learning architecture, people usually choose 15% testing set for training and the rest 85% for testing. It means there are only a few hundred images in training for each instance, which requires the model can extract useful viewpoint information from limited training examples.

Although the LINEMOD dataset is widely used in the instance-level pose estimation community, there is one issue of this dataset, which is that one cannot train their

Figure 2.1: **Image examples in LINEMOD dataset**. We randomly choose some image examples from the LINEMOD dataset to show how it looks like. The left image is for object 'Ape' in the dataset, the middle is for 'Bench Vise', and the right is for 'Phone'.

models directly on this dataset. The reason is that every image in this dataset contains some markers that are used to compute the ground truth. If directly train the model on these images, we cannot prevent the model learn pose information from the markers.

YCB-VIDEO Dataset

YCB-VIDEO dataset is proposed by [Xiang et al., 2017]. Different from LINEMOD, the authors did not label the pose of each frame by the markers. They only manually label the pose of the first frame in one scene, then infer the poses of the rest by tracking camera trajectory. This means the relative position of different objects in the frame is fixed (shown in Figure 2.2). From Figure 2.2, we can also see that as the poses change, some target object(s) is partially occluded by other objects, which poses occlusion challenge for pose estimation algorithms. In Table 2.2, we provide the detailed statistics of YCB-VIDEO dataset.

2.5.2 Other datasets

Here we also list some other datasets that can be used to test the 6D pose estimation methods further. For more datasets please refer to this challenge [Hodan et al., 2018].

- T-LESS [Hodan et al., 2017] contains thirty indoor textureless industry objects.

Table 2.2: **Detailed statistics of YCB-VIDEO.** 'Min Object' represents the minimal number of target objects instances appeared in the image. 'Max Object' means the maximal number. 'Mean Object' denotes the mean number of the total dataset.

Parameter	Value
Number of Objects	21
Number of Videos(Scene)	92
Number of Videos(Testing)	12
Min Object	3
Max Object	9
Mean Object	4.47
Number of Images	133827
Image Size	640×480

Figure 2.2: **Image examples in YCB-V dataset**. We choose three image examples from the 'sequence_01' in the training set. These three images are chosen from the beginning, middle, and end of the sequence.

Many of the objects are symmetrical and similar to each other.

- IC-BIN [Doumanoglou et al., 2016] is designed for the bin-picking scenario that contains heavy occlusion and background clutter in the scene.

- TUD-L [Hodan et al., 2018] contains three real objects with significant illumination changes, which poses a great challenge to the 2D detection part.

Figure 2.3: **Categorical dataset**. The categories in NOCS-REAL dataset.

2.5.3 Category-Level Pose Estimation Dataset

NOCS-REAL Dataset

This dataset consists of 18 different real scenes with 6 different categories. Among these scenes, the training set has 7 scenes, the validation set has 5 scenes, and the rest are for testing. In each category, there are three unique instances. The six categories are $bottle$, $bowl$, $camera$, can, $laptop$, and mug, which shown in Figure 2.3. From Figure 2.3 we can see that some categories in this dataset are symmetric, such as $bowl$ and can, which pose challenges when estimating pose on this dataset.

2.5.4 Point Could Labelling

However, the aforementioned datasets do not provide the label for each point. To train the proposed network in a supervised way, we propose an automatic method to generate the label of each point for the point cloud. First, for the 3D mesh model of each object, we transform it into the camera space using the corresponding ground truth pose matrix. We use the implementation in the package of [Hodaň et al., 2016] for this process. Second, for each point on the corresponding point cloud in the target region, we compute its shortest distance to the transformed mesh model. If the distance is smaller than a small threshold $\epsilon = 8mm$ we label the point as 1, otherwise 0. Figure 2.4 further illustrates the labelling process.

2.6 Evaluation Metrics

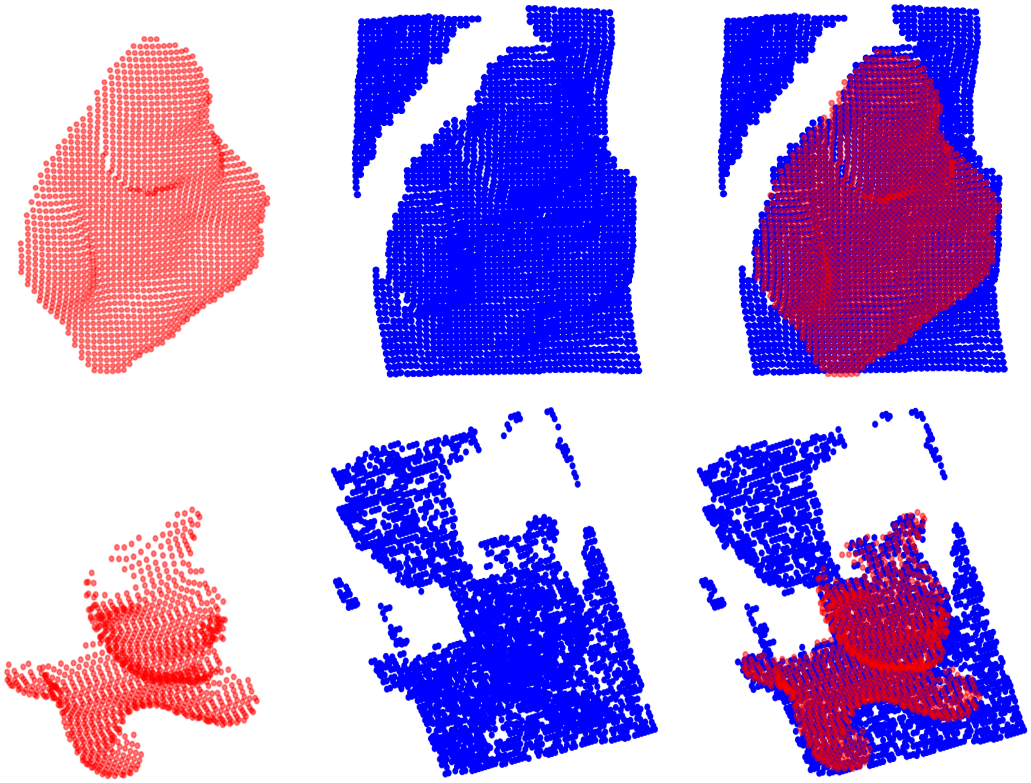
There are several different metrics for 6D pose estimation evaluation, we describe them in detail.

2.6.1 ADD(-S) Metric

Average Diameter Distance (-Symmetry) (ADD(-S)) metric is design by [Hinterstoisser et al., 2012], to estimate the transformed object 3D model from ground truth pose and estimated pose, respectively:

$$\frac{1}{|\mathcal{M}|} \sum_{\mathbf{p} \in \mathcal{M}} \|(\mathbf{R}\mathbf{p} + \mathbf{T}) - (\widetilde{\mathbf{R}}\mathbf{p} + \widetilde{\mathbf{T}})\|, \tag{2.1}$$

where $|\mathcal{M}|$ is the number of points in the object model. $\mathbf{p}$ represents the point in the object 3D model. $\mathbf{R}$ and $\mathbf{T}$ are the ground truth pose, and $\widetilde{\mathbf{R}}$ and $\widetilde{\mathbf{T}}$ are the estimated pose. In this metric, the mean distance between the two transformed point sets is computed. When the average distance is less than 10% of the 3D object model diameter, we



(a) (b) (c)

Figure 2.4: **Point cloud labelling**. (a) The mesh model of the object 'Ape' (top) and 'Cat' (bottom) in LINEMOD dataset; (b) the point cloud derived from the depth images in the target region; (c) the transformed mesh model is overlapped on the point cloud. We can label each point cloud according to the distance between the points on the point cloud and the corresponding transformed mesh model.

consider that estimated 6D pose as correct. For symmetric objects, we employ ADD-S metric [Hinterstoisser et al., 2012], where the average distance is calculated using the shortest point distance:

$$\frac{1}{|\mathcal{M}|} \sum_{\mathbf{p}_1 \in \mathcal{M}} \min_{\mathbf{p}_2 \in \mathcal{M}} \|(\mathbf{R}\mathbf{p}_1 + \mathbf{T}) - (\widetilde{\mathbf{R}}\mathbf{p}_2 + \widetilde{\mathbf{T}})\|. \tag{2.2}$$

2.6.2 Visible Surface Discrepancy

To evaluate the estimated 6D pose with depth map visibility, [Hodan et al., 2018] proposed Visible Surface Discrepancy (VSD) metric. The basic step of this metric is: first,

the object 3D model $\mathcal{M}$ is rendered by predicted pose and ground truth pose. Then we can get two distance maps $\hat{S}$ and $\bar{S}$ with the known camera matrix. The visibility masks $\hat{V}$ and $\bar{V}$ are calculated by comparing $\hat{S}$ and $\bar{S}$ with depth map S_I of the test image I. Finally, the error is computed by:

$$e_{VSD}\left(\hat{S},\bar{S},S_I,\hat{V},\bar{V},\tau\right) = \underset{p\in\hat{V}\cup\bar{V}}{\text{avg}} \begin{cases} 0 & \text{if } p\in\hat{V}\cap\bar{V}\wedge|\hat{S}(p)-\bar{S}(p)|<\tau \\ 1 & \text{otherwise}, \end{cases}, \quad (2.3)$$

where τ is the misalignment tolerance, p denotes the pixel. Equation 2.3 shows that pose error e_{VSD} is computed from the visible part of the transformed model surface and therefore the ambiguous poses (of symmetric objects) are treated as equivalent.

2.6.3 Rotation and Translation Error

This metric is used to measure the accuracy of estimated rotation and translation. For rotation measurement, we use geodesic distance to calculate the error between the predicted rotation and ground truth, and for translation, we use square error to measure.

For rotation geodesic distance, assume that we have predicted rotation $\widetilde{\mathbf{R}} \in \mathbb{R}^{3\times3}$ and ground truth $\mathbf{R} \in \mathbb{R}^{3\times3}$, then for geodesic distance $\mathcal{E}_{geo}$, we have:

$$\mathcal{E}_{geo} = \arccos\left(\left(\mathcal{T}(\mathcal{R})-1\right)/2\right), \quad (2.4)$$

where $\mathcal{R} = \mathbf{R}^{-1}\widetilde{\mathbf{R}}$, $\mathcal{T}(\mathcal{R})$ means the trace of $\mathcal{R}$.

For translation error, we use Euclidean distance: $\left\|\mathbf{T}-\widetilde{\mathbf{T}}\right\|_2$, where $\mathbf{T}$ and $\widetilde{\mathbf{T}}$ are translation ground truth and prediction, respectively. n° m **cm** represents the rotation error less than n° and the translation error less than m **cm** is accepted (see Table 5.4).

2.6.4 Intersection-over-Union

Intersection-over-Union (IoU) accuracy is used to measure 3D object detection methods via different overlap thresholds. Here we first transform the 3D bounding box of the

object by the predicted pose and ground truth respectively, then calculate the overlap ratio between intersection volume and union volume of these two transformed bounding boxes. IoU_X means the overlap ratio larger than $X\%$ is accepted.

Chapter Three

Point-Wise Voting For Robust 6D Object Pose Estimation

3.1 Overview

With the popularity of 3D sensors, such as Kinect Camera [Zhang, 2012], the amount of the available 3D data (such as depth image and point clouds) has tremendously increased. However, which deep neural network architectures are suitable for pose estimation or 3D object detection from the 3D data remains an open problem.

Some existing works converted the 3D data to volumetric grids by quantization [Maturana and Scherer, 2015, Song and Xiao, 2016, Wu et al., 2015], then applied convolutional neural networks to proceed with such volumetric data. However, this data representation transformation misses the fine geometric details of the object and introduces quantization artifacts that can obscure natural invariances of the data. Some other methods [Kehl et al., 2016, Li et al., 2018] processed the depth image as an additional image channel along with the RGB channels. However, this approach did not make full use of the geometric information in the data which makes it difficult to integrate information across viewpoints [Maturana and Scherer, 2015].

To better utilize 3D information in 3D data for 6D object pose estimation, in this chapter, we propose a novel pipeline for 6D object pose estimation from RGB-D images. Inspired by [Peng et al., 2018, Qi et al., 2018], we estimate 6D object pose via multi-stage. First, we locate the object by using a 2D CNN detector to access the point cloud of the object in the detected region. Then we use PointNet [Qi et al., 2017] to segment object points from background points as well as estimate object pose based on the segmented point cloud.

Compared to the methods proposed in [Qi et al., 2018], we have two improvements. Firstly, instead of directly regressing the global point features to 3D keypoints, we regress the point-wise features to unit vectors that point to the 3D keypoints. This improvement makes our method more robust to occlusion. Secondly, we propose a new simply scoring method that utilizes the geometry constraints of the object point cloud to score the different poses computed from keypoint hypotheses. Then we choose the pose hypothesis with the highest score as the final pose. Our method shares features of PVNet [Peng et al., 2018]: both methods use unit vector regression to estimate pose, however, our method takes point cloud as input, and instead of using 2D keypoints, we use 3D keypoints, then we use our proposed scoring mechanism to access the final pose which is different to the optimization-based method in [Peng et al., 2018].

The output of our method is a 3D vector-field representation for 3D keypoints localization that guides the network to learn local geometry features of the object point cloud. This means even when the object is occluded partially, the proposed method can still accurately estimate its pose by using only the remaining visible parts of the object.

In summary, the key contributions of this chapter are as follows:

- We present a novel deep learning approach, named PointPoseNet, that regresses point-wise features to unit vectors pointing to the 3D keypoints for 6D pose estimation. Our network learns 3D vector-field representation to account for object

3D local geometry information and 3D keypoints localization.

- We propose a new scoring mechanism that utilizes the geometry constraints between ground truth and transformed point cloud to select the best pose hypothesis among those that are generated by predicted 3D unit vectors.

3.2 Related Works

The related work of this part has been described in Section 2, in this chapter, we also make use of PointNet-like architecture but with 3D vector-field representation and 3D geometry constraint. We also show that the proposed method achieves better performance than state-of-the-art methods with this new representation and constraint.

3.3 Point-Wise Voting Pipeline

The overview of our proposed pipeline is shown in Figure 3.1. Given an RGB-D image, we first use a state-of-the-art 2D detector, YOLOv3[Redmon and Farhadi, 2018], to locate the object with a bounding box and output the object label. Then, we transform the corresponding depth region to the point cloud with the known camera matrix. However, the point cloud derived from this region contains both target points and background points. To access the points only belonging to the object and predict the unit vectors pointing to the keypoints (see Section 3.3.4 for details), given the point cloud transformed from the depth image in the target region, our network performs two related tasks. First, the object points are segmented from the points in the detected region. Second, the network predicts point-wise unit vectors pointing to the keypoints.

Then, given the oriented vectors to a certain object keypoint from all points belonging to the object, we generate hypotheses of 3D locations for that keypoint. With

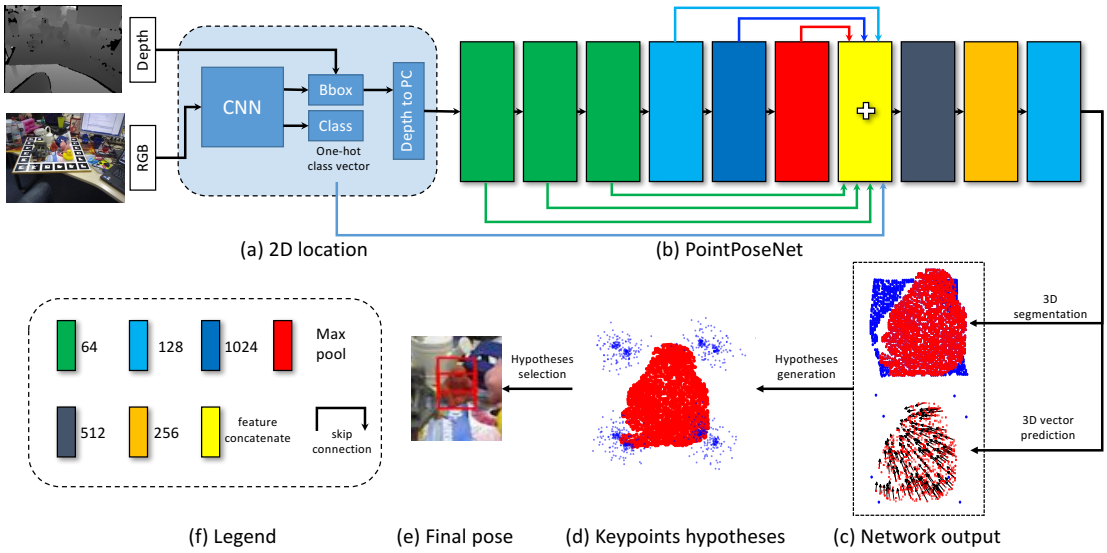


Figure 3.1: **Overview of the proposed pipeline.** (a) Given an RGB image, we use CNN to detect the bounding box of the target object, and the object label that is used as one-hot features for PointPoseNet. (b) Given the point clouds in the target region, we use the proposed PointPoseNet to do 3D segmentation and vector prediction. (c) Top: 3D mask for target points; bottom: Point-wise unit vectors pointing to the keypoint. (d) 3D keypoints hypotheses generated from the unit vectors. (e) Final pose after hypotheses selection. (f) Legend of this figure. The number is the output channel of the corresponding layer. Hollow " $+$ " represents feature concatenation.

these keypoint hypotheses and keypoints in the canonical frame, we can compute corresponding pose hypotheses, then we choose the best pose hypothesis as the final pose by our proposed scoring mechanism.

3.3.1 2D Object Region Detection

In this step, we derive the point cloud that represents the object of interest in the depth image. To achieve this, we train a 2D CNN detector, YOLOv3 [Redmon and Farhadi, 2018], to localize the object in the RGB image with a bounding box and output the object label that is used as one-hot class vector for better point cloud instance segmen-

tation in PointPoseNet. The bounding box is then applied to extract the corresponding region in the depth image. By inverting the image formation process, the localized depth region can be converted to a 3D point cloud in the camera space with known camera parameters. However, the point cloud still contains some points that are not belonging to the object. In the next step, we will describe how we segment the object points from background points and estimate the 6D pose of the object.

3.3.2 Point-Wise Vector Generation

In contrast to directly regressing the 3D bounding box for the object [Qi et al., 2018], the proposed network is trained to predict point-wise directional vectors that enforce the network to focus on the local features of the object. Different to [Peng et al., 2018, Xiang et al., 2017] that predict dense vectors pointing to 2D keypoints, our network predicts dense vectors pointing to 3D keypoints. There are two advantages of our method: i) our method enables to locate invisible keypoint(s) from other visible parts of the object point cloud in 3D space, and ii) previous methods [Qi et al., 2017, 2018] have shown that learning in 3D space can better exploit the geometric and topological structure of 3D space that useful to pose estimation.

More concretely, for a object point $\mathfrak{p}$, our network outputs its semantic label with the unit vectors $\mathbf{v}_k(\mathfrak{p})$ that denotes the direction from the point $\mathfrak{p}$ to k^{th} 3D keypoint $\mathbf{p}_{k\rho}^{key}$ transferred by the pose ρ. $\mathbf{v}_k(\mathfrak{p})$ is defined as:

$$\mathbf{v}_k(\mathfrak{p}) = \frac{\mathbf{p}_{k\rho}^{key} - \mathfrak{p}}{\|\mathbf{p}_{k\rho}^{key} - \mathfrak{p}\|_2}. \tag{3.1}$$

Another possible way is to estimate the displacement vector $\mathbf{p}_{k\rho}^{key} - \mathfrak{p}$, however, in experiments (see Table 3.2) we show that regressing to unit vector can achieve better results (98.4%) than predicting absolute displacement (96.3%). [Xiang et al., 2017] also suggested that scale-invariant unit vectors are easier to train.

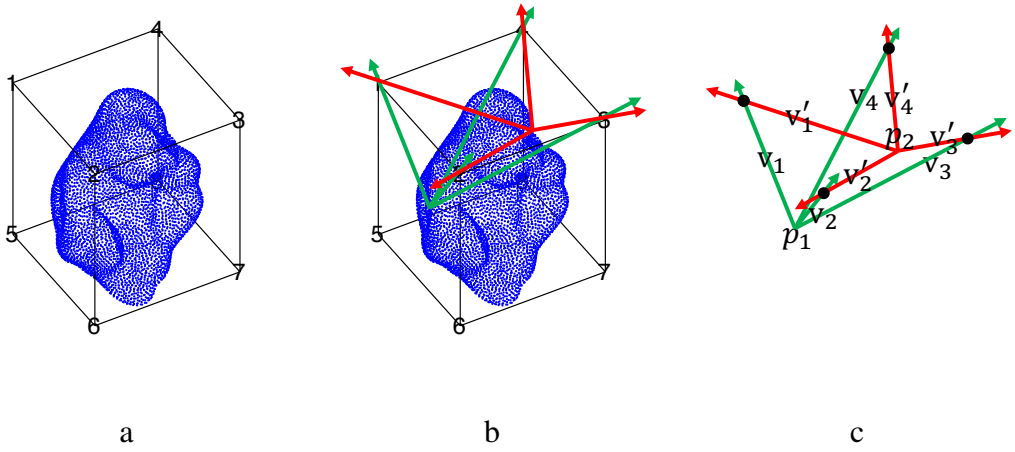
Given semantic labels and unit vectors, we generate 3D keypoint hypotheses with voting weight. First, we find the points of the target object by using semantic labels. Then, we randomly sample N point pairs and use the intersection of the straight lines that have their vectors as direction vector as N hypotheses for keypoints (see Figure 3.2 (b) and (c) for the illustration on how to derive the keypoints from two points). However, different from 2D cases in [Peng et al., 2018, Xiang et al., 2017] where two nonparallel lines always have an intersection, two nonparallel lines in 3D can have no intersection (shown in Figure 3.3). To address this problem, in this paper, we use the midpoint of the shortest line segment of two lines where the two vectors lie, as the intersection point(see Figure 3.3 for the illustration). Finally, we need a criterion to weigh each hypothesis. Intuitively, a reliable hypothesis should satisfy these requirements: (1) it should coincide with different predicted directions; (2) the distance between two lines that generate this hypothesis should be close. Based on these two requirements, we calculate the voting weight $w_{k,i}$ of a keypoint hypothesis $\mathbf{h}_{k,i}$ as:

$$w_{k,i} = \sum_{\mathfrak{p} \in O} \mathbb{I}\left(\mathbb{I}(d_i^k \leq \epsilon) \frac{(\mathbf{h}_{k,i} - \mathfrak{p})^T}{\|\mathbf{h}_{k,i} - \mathfrak{p}\|} \mathbf{v}_k(\mathfrak{p}) \geq \theta \right), \tag{3.2}$$

where $\mathbb{I}$ represents the indicator function, θ is a threshold, and $\mathfrak{p} \in O$ means that the point $\mathfrak{p}$ belongs to the object O. $\mathbf{v}_k(\mathfrak{p})$ is the k^{th} predicted vector of point $\mathfrak{p}$. d_i^k is the distance between two straight lines where the vectors are located, this distance is used to measure the confidence of the intersection (see Figure 3.3 for details). ϵ is a distance threshold.

3.3.3 Voting with Geometry Constraint

Assuming that we have sample N keypoint hypotheses, by using the keypoints in the canonical frame and Kabsch algorithm [Kabsch, 1976] we can compute N pose hypotheses via 3D-3D correspondences. However, not all these pose hypotheses are good for our task. We need to find the best pose hypothesis from these hypotheses (see Figure 3.4 for illustration). According to the definition of the pose, the mesh model transformed



a b c

Figure 3.2: **Keypoints selection and generation.** (a) The corners on the 3D bounding box of the object are selected as the keypoints; (b) two arbitrary points (p_1 and p_2) are selected, and for each point, the network predicts four directional vectors (for a better visualization the top 4 corners are selected as the keypoints); (c) for each vector pair ($\mathbf{v}_1$ and $\mathbf{v}_1'$ for example), an intersection point can be located, which is defined as a keypoint hypothesis. In this example, four keypoints are generated.

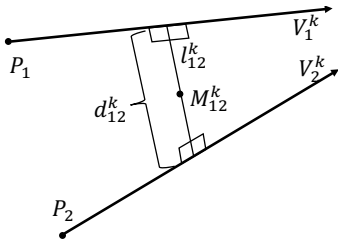


Figure 3.3: **Find intersection.** l_{12}^k is the shortest line segment between line V_1^k and V_2^k. M_{12}^k is the midpoint of l_{12}^k. d_{12}^k is the length of l_{12}^k.

by the good pose hypothesis should match the point cloud of the object in the test scene very well. The question is how to measure this match. Inspired by the method proposed in [Aldoma et al., 2012], here we use interior point count to measure this match: if a point in the test scene is close enough to the transformed mesh model, we call this point an interior point. Therefore, in this task we aim at finding the pose hypothesis that can

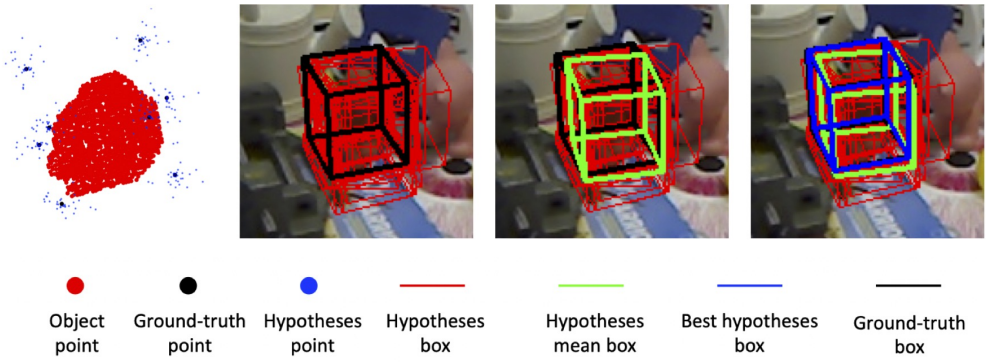


Figure 3.4: **Hypotheses selection.** We use the 3D bounding box transformed by the corresponding pose to represent the pose. From left to right: (1) Generated keypoints hypotheses and ground truth keypoints. (2) Pose hypotheses from keypoint hypotheses and the ground truth pose. (3) The mean pose (green box) of these pose hypotheses, which does not match the ground truth very well. Please note, we first use each keypoint hypothesis to get a pose hypothesis, and then average these pose hypotheses to get the mean pose. Finally, we use this mean pose to transform the 3D bounding box of the object to the scene. (4) Pose selected (blue box) by our scoring mechanism, which matches the ground truth well.

maximize the interior point count:

$$H^* = \arg\max_{H \in \mathcal{H}} \sum_{\mathfrak{p} \in O} \mathbb{I}_{dist(\mathfrak{p},\, \mathcal{M}_H) < \tau}, \tag{3.3}$$

where $\mathfrak{p}$ denotes the points belonging to the object in the camera space. $\mathcal{M}_H$ is the mesh model that transformed into the camera space via the pose hypothesis H. $\mathcal{H}$ denotes all the pose hypotheses. $\mathbb{I}$ is the indicator function, which is 1 if the statement is true. $dist(\mathfrak{p}, \mathcal{M}_H)$ stands for the shortest Euclidean distance between the object point $\mathfrak{p}$ and the transformed mesh model $\mathcal{M}_H$. τ is a positive threshold. The formulation can be efficiently optimized by the pre-emptive RANSAC [Shotton et al., 2013]. Compared to the verification mechanism in [Aldoma et al., 2012], our mechanism does not need to calculate the normal and neighboring points of different points, which reduces the computation burden.

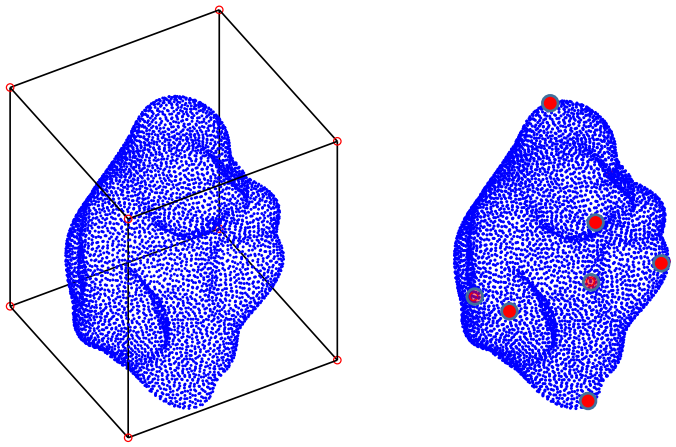


Figure 3.5: **Keypoint selection schemes**. The left image is a 3D object point cloud and its 3D bounding box; the right image is the keypoint selected by the FPS algorithm. The keypoints are shown in red color.

3.3.4 Keypoint Selection

To train our network, we need to define some keypoints. The keypoints are defined based on the 3D object model (see Figure 3.5). Two aspects need to be decided for the keypoints: number and location. A simple way is to use the 8 corners of the 3D bounding box of the object as the keypoints. This fashion is widely adopted by many CNN-based methods in 2D cases [Oberweger et al., 2018, Rad and Lepetit, 2017, Rad et al., 2018, Tekin et al., 2018]. Since the 3D bounding box corners are distributed well in the 3D space, which should be easier for the network to regress, we also use this keypoint definition in our experiments. Another way is adopted in [Peng et al., 2018] that used the farthest point sampling (FPS) algorithm to sample the keypoints. Figure 3.5 shows an example of different keypoint selection schemes. In Section 3.4.2, we show how the number and position of the keypoints influence the pose estimation results.

3.4 Experiments

In this section, we describe how we train our network and report the experimental results on the LINEMOD and YCB-V datasets.

3.4.1 Training Details

First we fine-tune the YOLOv3 [Redmon and Farhadi, 2018] that pre-trained on the ImageNet [Deng et al., 2009] to localize the 2D region of the interest. To prevent overfitting, we adopt data augmentation, which includes random rotation, resizing, and cropping as well as adding synthetic images to the training set. We use the default training parameters in the YOLOv3.

Then we train our proposed network. We use PointNet [Qi et al., 2017] as our backbone network but remove the transformer networks proposed in [Qi et al., 2017] to preserve viewpoint information. Another major difference is that we add an input one-hot class vector to provide semantic information. The architecture details are shown in Figure 3.1.

Our network performs two related tasks: point cloud segmentation and unit vector prediction. Therefore, the loss function of our network consists of two different loss functions. For point cloud segmentation, we use cross-entropy as the loss function. For learning unit vectors, according to experimental results, the loss function is defined as the mean square error between the predicted and ground truth directional vectors:

$$\mathcal{L} = \frac{1}{KN_O} \sum_{k=1}^{K} \sum_{\mathfrak{p} \in O} \left\| \widetilde{v}_k(\mathfrak{p}) - v_k(\mathfrak{p}) \right\|_2 , \tag{3.4}$$

where K is the number of keypoints. $\widetilde{v}_k(\mathfrak{p})$ and $v_k(\mathfrak{p})$ are the predicted vector and the ground truth vector of object point $\mathfrak{p}$, respectively. N_O denotes the number of object points.

We use Adam [Kingma and Ba, 2014] to optimize the proposed network. We set the initial learning rate as 0.001 and halve it every 60 epochs. The maximum epoch is 300.

However, the original datasets do not provide the label for each point. Please refer Chapter 2, Section 2.4 for the labelling details.

3.4.2 Ablation Studies

We conduct ablation studies to compare different keypoints selection schemes and the numbers of keypoints on the LINEMOD dataset. Table 3.1 summarizes the results of ablation studies.

In Section 3.3.4, we discussed the keypoints selection schemes. Here we compare the pose estimation results based on different keypoint sets. “Bounding box (BBX)-8-S" means using the 8 bounding box corners and “FPS-8-S" represents the 8 points are selected by the FPS algorithm. “S" represents accessing the final pose by our scoring mechanism. Comparing “BBX-8-S" and “FPS-8-S" in Table 3.1, the results show that “BBX-8-S" can get better accuracy than “FPS-8-S". Furthermore, to explore the influence of the keypoint number on pose estimation, we train our network to regress to different numbers of keypoints that are selected by the FPS algorithm. Comparison among columns “FPS-4-S", “FPS-8-S", and “FPS-12-S" shows that selection 8 keypoints can achieve better results. For efficiency and simplicity, we use “BBX-8-S" in all other experiments.

We also compared different mechanisms to access the final pose from pose hypotheses that are generated from 8 bounding box corners. We refer to the mechanism which uses the mean value of all pose hypotheses without scoring mechanism as “MEA" and the mechanism using a similar optimization method as [Peng et al., 2018] to compute the final pose from pose hypotheses as “OPT". Compared “BBX-8-S", “BBX-8-

Table 3.1: **Ablation studies on different parameters for pose estimation on LINEMOD dataset.** The metric we used to measure performance is ADD(-S) metric where 'Glue' and 'Egg Box' are considered as symmetric objects. **BBX-8** means using the 8 corners of the 3D bounding box as keypoints. **FPS-K** means that we use K keypoints generated by the FPS algorithm. The last two columns show using different mechanisms to access the final pose from pose hypotheses. **MEA** means using the mean value of all hypotheses without pose sampling and selection. **OPT** means using similar optimization method as [Peng et al., 2018] to compute final pose from pose hypotheses.

Method	BBX-8-S	FPS-8-S	FPS-4-S	FPS-12-S	BBX-8-MEA	BBX-8-OPT
Ape	**97.9%**	96.5%	90.2%	97.8%	96.8%	96.5%
Bench Vise	**99.6%**	96.7%	92.1%	95.6%	94.0%	97.3%
Camera	98.5%	**98.9%**	94.4%	97.3%	96.2%	96.3%
Can	**99.4%**	98.7%	93.8%	97.4%	97.2%	97.6%
Cat	**99.3%**	99.0%	97.4%	98.7%	97.7%	98.6%
Driller	**97.5%**	96.5%	95.3%	96.4%	96.4%	96.3%
Duck	96.1%	**99.6%**	98.9%	92.8%	90.9%	90.5%
Egg Box	97.9%	97.8%	99.0%	97.8%	96.0%	96.7%
Glue	**100%**	98.9%	97.9%	98.5%	96.8%	97.5%
Hole Puncher	97.8%	**99.4%**	93.8%	98.1%	98.1%	97.9%
Iron	99.4%	99.0%	**99.5%**	97.4%	95.3%	97.2%
Lamp	99.1%	**99.6%**	97.7%	97.9%	98.1%	98.8%
Phone	98.9%	98.9%	**99.9%**	99.7%	99.1%	99.2%
Ave	**98.4%**	98.3%	94.6%	97.3%	94.2%	96.4%

MEA", and "BBX-8-OPT" in Table 3.1, the results show that our pose sampling and scoring mechanism achieve better results.

In Table 3.2, we show that predicting unit vector can achieve better results than absolute displacement regression.

Table 3.2: **Ablation studies on unit vector and displacement regression.** We use ADD(-S) metric to measure the performance of different methods.

Method	Unit Vector	Displacement
Ape	**97.9**%	95.6%
Bench Vise	**99.6**%	99.3%
Camera	98.5%	**98.7**%
Can	**99.4**%	96.7%
Cat	**99.3**%	96.4%
Driller	**97.5**%	94.6%
Duck	**96.1**%	93.0%
Egg Box	**97.9**%	95.7%
Glue	**100%**	99.7%
Hole Puncher	97.8%	**98.9**%
Iron	**99.4**%	96.8%
Lamp	99.1%	**99.4**%
Phone	**98.9**%	98.5%
Average	**98.4**%	96.3%

3.4.3 Comparison with the State-of-the-Arts

In this section, we compare our method with the state-of-the-art pose estimation methods on three popular datasets: LINEMOD [Hinterstoisser et al., 2012], Occlusion LINEMOD [Brachmann et al., 2014, Hinterstoisser et al., 2012], and YCB-Video dataset [Xiang et al., 2017]. We empirically set the number of hypotheses as 10k for LINEMOD and Occlusion LINEMOD dataset, and set the number as 20k for the YCB-Video dataset. Both visual and quantitative results are provided.

Results on LINEMOD dataset. In Table 3.3, we summarize the pose estimation results from the original papers on the LINEMOD dataset. We compare our method with

Table 3.3: **6D pose estimation accuracy on the LINEMOD dataset.** We use ADD metric to evaluate the methods. For symmetric objects 'Egg Box' and 'Glue', we use ADD-S metric. For refinement, 'ICP' means ICP refinement, and 'HV' means hypothesis generation/verification refinement.

Method	PVNet [Peng et al., 2018]	DeepIM [Li et al., 2018]	Frustum-P [Qi et al., 2018]	Hinterstoisser [Hinterstoisser et al., 2016]	DenseFusion [Wang et al., 2019]	Ours
Refinement	×	✓(ICP)	×	✓(ICP)	✓(ICP)	✓(HV)
Ape	43.6%	77.0%	85.5%	**98.5%**	92.3%	97.9%
Bench Vise	99.9%	97.5%	93.2%	99.0 %	93.2%	99.6%
Camera	86.9%	93.5%	90.0%	**99.3%**	94.4%	98.5%
Can	95.5%	96.5%	91.4%	98.7%	93.1%	**99.4%**
Cat	79.3%	82.1%	96.5%	**99.9%**	96.5%	99.3%
Driller	96.4%	95.0%	96.8%	93.4%	87.0%	**97.5%**
Duck	52.6%	77.7%	82.9%	**98.2%**	92.3%	96.1%
Egg Box	99.2%	97.1%	**99.9%**	98.8%	99.8%	97.9%
Glue	95.7%	99.4%	99.2%	75.4%	**100%**	**100%**
Hole Puncher	81.9%	52.8%	92.2%	**98.1%**	92.1%	97.8%
Iron	98.9%	98.3%	93.7%	98.3%	97.0%	**99.4%**
Lamp	**99.3%**	97.5%	98.2%	96.0%	95.3%	99.1%
Phone	92.4%	87.7%	94.2%	98.6%	92.8%	**98.9%**
Average	86.3%	88.6%	93.4%	96.3 %	94.3 %	**98.4 %**

state-of-the-art RGB and RGB-D methods. In Table 3.3, the second and third column are RGB methods. The rest are RGB-D methods. From this table, we can see that the best RGB-D methods can outperform about 10% of the best RGB methods. We use Frustum-PointNets [Qi et al., 2018] as our baseline. We re-implement Frustum-PointNets to regress the 3D bounding box corners of the objects. From Table 3.3, we can see that our method outperforms its 2D counterpart PVNet [Peng et al., 2018], the baseline, and other state-of-the-art methods, which shows that our method can better utilize 3D information from depth images.

Results on Occlusion LINEMOD dataset. Same as other state-of-the-art methods, we train our model on LINEMOD dataset and test it on the Occlusion LINEMOD. We sum-

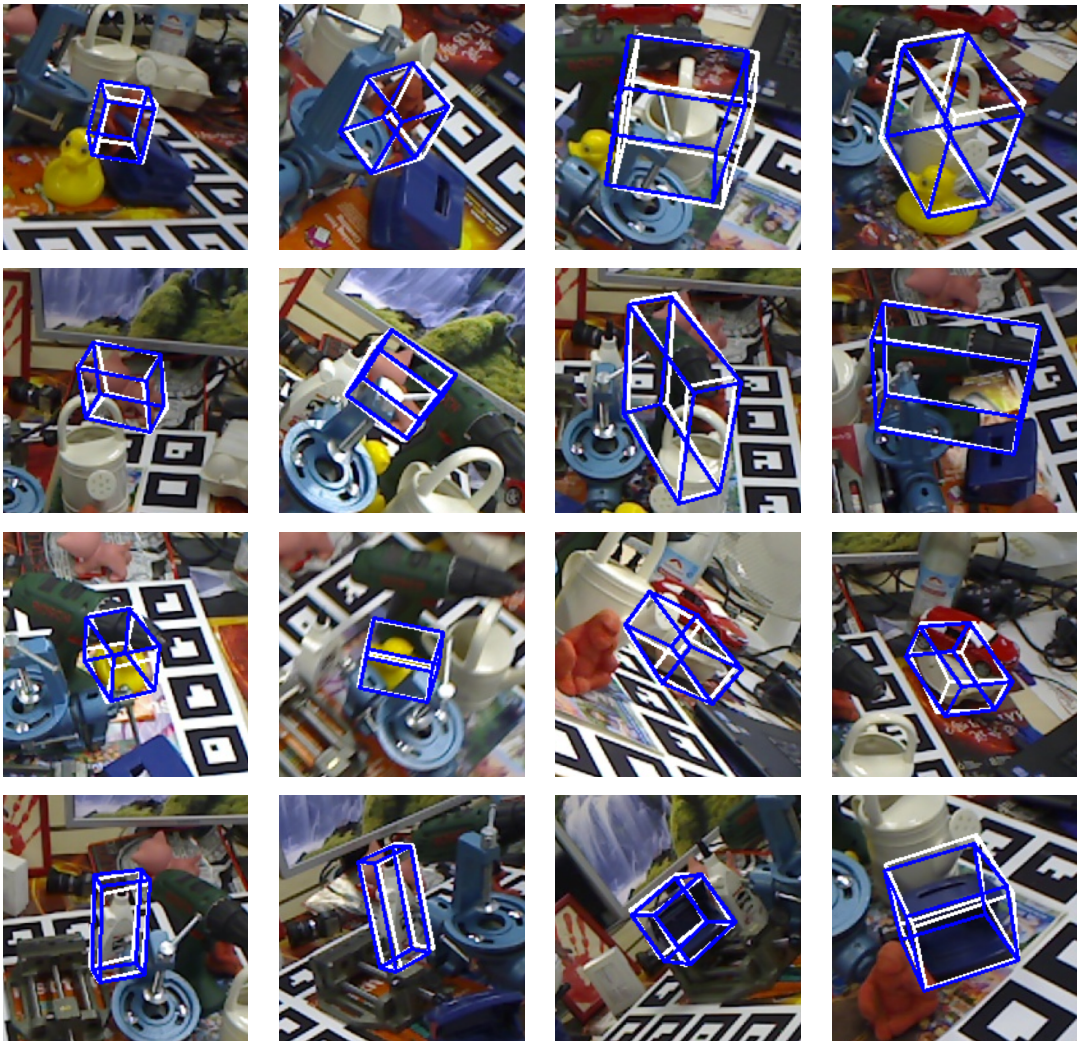


Figure 3.6: **Visualizing pose estimation results**. White 3D bounding boxes are the ground truth, while blue 3D bounding boxes represent our results. For each object in the Occlusion LINEMOD dataset, we show two predicted results.

marize the experimental results from the original papers into Table 3.4. We compare our method with state-of-the-art methods. Overall, our method outperforms other methods. Some qualitative results are shown in Figure 3.6. The improvement of our method is the most obvious on the Cat and $Glue$. For the cat, its ears and tail are useful local geometry information for pose estimation. For the $Glue$, its bulge on its shoulder is useful local geometry information for pose estimation (see Figure 3.7). Our proposed method can better utilize this kind of 3D local geometry information via 3D vector-field representation, making it more robust to the occlusions. However, from Table 3.4 we can see

Table 3.4: **Pose estimation on occlusion dataset.** 6D pose estimation accuracy on the Occlusion LINEMOD dataset in terms of the ADD(-S) metric, where 'Egg Box' and 'Glue' are considered as symmetric objects. For refinement, 'ICP' means ICP refinement and 'HV' means hypothesis generation/verification refinement.

Method	PoseCNN [Xiang et al., 2017]+ICP	Michel [Michel et al., 2017]	Hinterstoisser [Hinterstoisser et al., 2016]	Krull [Krull et al., 2015]	Ours
Refinement	✓(ICP)	✓(ICP)	✓(ICP)	✓(ICP)	✓(HV)
Ape	76.2%	80.7%	**81.4%**	68.0%	80.2 %
Can	87.4%	88.5%	**94.7%**	87.9%	90.1%
Cat	52.2%	57.8%	55.2%	50.6%	**61.2%**
Driller	90.3%	94.7%	86.0%	91.2%	**94.8%**
Duck	77.7%	74.4%	**79.7%**	64.7%	77.6 %
Egg Box	72.2%	47.6%	65.5%	41.5%	**72.9%**
Glue	76.7%	73.8%	52.1%	65.3%	**77.5%**
Hole Puncher	91.4%	**96.3%**	95.5%	92.9%	81.8%
Average	78.0%	76.7%	76.3%	70.3%	**79.5%**

Figure 3.7: **Visualizing LINEMO objects**. Left: Cat; middle: Glue; right: Hole Puncher.

that the performance of our method is particularly low on object $Hole\ Puncher$. From Figure 3.7, we can see that the $Hole\ Puncher$ object has large flat surfaces. Our proposed method extracts local 3D geometry information from the object. However, when only the flat surfaces are visible, there are no unique local features to extract, then in keypoint hypotheses generation step, our method generates some non-focused keypoint hypotheses, especially when nonflat parts are occluded by other objects, which makes it difficult to access a good pose.

Table 3.5: **6D Pose estimation accuracy on the YCB-V dataset.** We use ADD-S AUC metric to evaluate the methods. For refinement, 'ICP' means ICP refinement and 'HV' means hypothesis generation/verification refinement.

Method(RGB+DEPTH)	PoseCNN + ICP [Xiang et al., 2017]	MCN [Li et al., 2018]	DenseFusion [Wang et al., 2019]	Ours
Refinement	✓(ICP)	✓(HV)	✓(HV)	✓(HV)
002_master_chef_can	95.8%	96.2%	**96.4**%	95.2%
003_cracker_box	91.8%	90.9 %	**95.5**%	89.1%
004_sugar_box	**98.2**%	95.3%	97.5%	96.0%
005_tomato_soup_can	94.5%	**97.5**%	94.6%	91.9%
006_mustard_bottle	**98.4**%	97.0%	97.2%	96.3%
007_tuna_fish_can	**97.1**%	95.1%	96.6%	97.0%
008_pudding_box	**97.9**%	94.5%	96.5%	96.5%
009_gelatin_box	**98.8**%	96.0%	98.1%	97.8%
010_potted_meat_can	92.8%	**96.7**%	91.3%	86.2%
011_banana	**96.9**%	94.4%	92.1%	94.3%
019_pitcher_base	**97.8**%	96.2%	97.1%	93.0%
021_bleach_cleanser	**96.8**%	95.4%	95.8%	93.5%
024_bowl	78.3%	82.0%	**88.2**%	82.8%
025_mug	95.1%	96.8%	97.1%	**97.2**%
035_power_drill	**98.0**%	93.1%	96.0%	91.1%
036_wood_block	90.5%	**93.6**%	89.7%	86.1%
037_scissors	92.2%	94.2%	**95.2**%	93.0%
040_large_marker	97.2%	95.4%	**97.5**%	97.0%
051_large_clamp	75.4%	93.3%	72.9%	**96.1**%
052_extra_large_clamp	65.3%	90.9%	69.8%	**93.6**%
061_foam_brick	**97.1**%	95.9%	92.5%	92.9%
Average	93.0%	**94.3** %	93.1 %	93.2 %

Results on YCB-Video dataset. In Table 3.5, we compared our method with other state-of-the-art methods [Li et al., 2018, Wang et al., 2019, Xiang et al., 2017]. We use ADD-S AUC metrics. Our method achieves comparable results with the state-of-the-art methods and only falls behind MCN [Li et al., 2018]. One possible reason is that

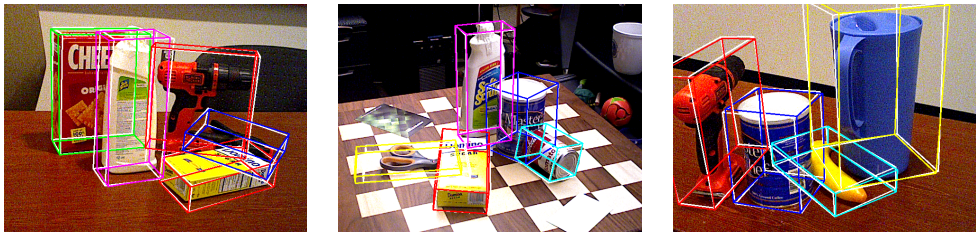


Figure 3.8: **Visualizing pose estimation results on YCB-Video dataset**. White 3D bounding boxes are the ground truth and colorful 3D bounding boxes represent our results.

some objects in the YCB-Video dataset contain much useful texture (see Figure 3.8), since we only use RGB image to locate the object, our method cannot utilize this kind of information in the pose computing process. Another possible reason is that in the YCB-Video dataset, some objects that contain large flat surfaces are the symmetrical-box type, our method cannot efficiently handle objects in this type (see Section 3.4.5).

3.4.4 Running Time Analyse

In this section, we first analyse the time complexity of the proposed pipeline and then report the running speed in our experimental setting. Assuming that there are in object 3D mesh model, N_O points in the segmented point cloud, and the number of hypotheses is N. For every point in the segmented point cloud, we need to calculate its distance to the $|\mathcal{M}|$ points transformed by N hypothesis and find the closest distance, then the time complexity can be defined by:

$$\mathcal{T} = \mathcal{O}(N(N_O\,|\mathcal{M}|^2)) \tag{3.5}$$

In our experiments, we set $M = 1000$, and $N = 10K$, given a 480×640 image with its corresponding depth image, our method runs at about 3 fps (with 10K hypotheses) on a desktop with an Intel i7-4930K 3.4GHz CPU and a GTX 1080 Ti GPU. Specifically, the 2D detector takes 10 ms for object location, and pose estimation

for each bounding box takes a total of 230 ms ∼530 ms of which 30 ms for vector prediction and 3D segmentation by PointPoseNet, and 200 ms ∼500 ms for hypotheses generation and selection. In Table 3.6, we report the running speed of the proposed method with different numbers of hypotheses and the corresponding performance on the Occlusion LINEMOD dataset.

Table 3.6: **Running time on the Occlusion LINEMOD dataset.** The first column is the number of the pose hypotheses. The last column is the average accuracy of all objects in the Occlusion LINEMOD dataset.

Number	Running Speed (fps)	Hole Puncher	Average
1000	15	17.3%	25.4%
2000	9	45.8%	58.4%
5000	5	66.2%	73.4%
10000	3	81.8%	79.5%
20000	1.3	89.1%	80.4%

3.4.5 Failure Cases

Although our method achieves state-of-the-art performance, we also observe failure cases of our work. Our method has difficulty with objects with large flat surfaces, especially when they are occluded. One way to alleviate this problem is to increase the number of pose hypotheses, however, this can dramatically increase the running time (see Table 3.6 for details). In Figure 3.9, we show failure examples on Occlusion LINEMOD dataset.

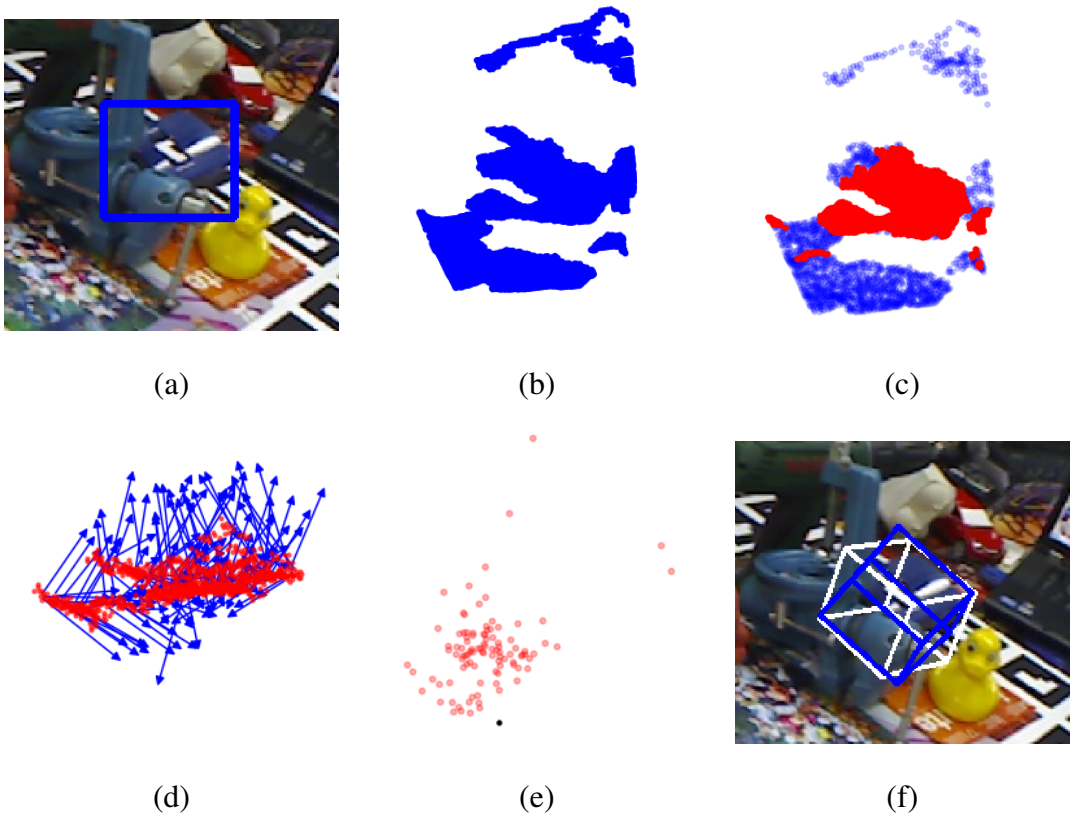


Figure 3.9: **Failure cases**. We show the output of each step of one failure case on object 'Hole Puncher': a) object location, b) point cloud in the target region, c) point cloud segmentation, d) vector prediction, e) keypoint hypotheses. (the black point is the ground truth), f) pose estimation result. We can see that due to non-focused keypoint hypotheses, the pose estimation result is not very good.

3.5 Conclusion

In this chapter, we introduce a novel deep learning pipeline that mainly consists of two parts for 6D object pose estimation in 3D point clouds. In the first part, we train a 2D detector to fast locate the object region. In the second part, we train the proposed network to regress point-wise unit vectors that point to the pre-defined 3D keypoints, i.e. the corners of the 3D bounding box. Then these vectors are used to generate different pose hypotheses for best pose selection. The main work of this thesis is focusing on improving the ability of the second part, i.e. pose estimation part.

In this chapter, we can get state-of-the-art performance from the pose estimation part, however, as reported in Section 3.4.4, the running speed of the proposed pipeline is very slow that is far away from the requirements of real-world application. To overcome this issue, in the next chapter, we will describe how to design a real-time framework based on the pipeline described in this chapter, and in chapter 5, we further extend the proposed work to category-level pose estimation tasks.

Chapter Four

Embedding Vector Features for Real-Time 6D Object Pose Estimation

4.1 Overview

In this chapter, we focus on improving the real-time performance of 6D object pose estimation, which plays an essential role in augmented reality [Marchand et al., 2016, Marder-Eppstein, 2016], and robotic manipulation [Tremblay et al., 2018, Zhu et al., 2014]. As the low-cost depth data collection equipment is becoming more and more popular, researchers pay more attention to RGB-D information, many compelling methods [Chen et al., 2020, Kehl et al., 2016, Li et al., 2018, Xiang et al., 2017] have been proposed. These methods can achieve higher performance on benchmark datasets than RGB-based methods, however, they are usually computation-intensive and thus cannot meet the real-time requirement.

The reason why they are computation-intensive is that they cannot extract viewpoint information from depth information effectively, thereby they need an extra step to further process depth information that includes, post-refinement or hypothesis generation/verification. Although one can use the method proposed by [Krull et al., 2017] to process a large number of hypotheses efficiently, their methods still cannot meet the

real-time requirement.

To overcome this computation-intensive issue in RGB-D methods. First, we analyse what causes the issue. The depth image contains translation and rotation information of the object relative to the camera coordinate. Previous methods processed both information together. However, translation and rotation information tangle with each other. It is hard to directly access accurate 6D pose from depth. Therefore, these methods need the post-refinement or hypothesis generation/verification mechanism to further extract 6D pose information, which leads to computation-intensive. To address the issue, we choose to decouple the translation with rotation from the input data. Furthermore, we find that compared to translation estimation, rotation estimation is difficult. To better extract viewpoint information from depth data, we propose to use the novel embedding vector features.

Specifically, our network is built on [Qi et al., 2018] with three major novelties: i) instead of locating the object point cloud by a frustum, we locate the object point cloud by a 3D sphere, which can limit the 3D search range in a more compact space (see Section 4.3.1 for details), ii) to estimate the rotation more accurate, we propose the point-wise embedding vector features to effectively capture the viewpoint information from the point cloud transformed from the depth image, and iii) we estimate the rotation residual between predicted rotation and the ground truth. The rotation residual estimator further boosts the pose estimation accuracy.

We evaluate our method on three 6D object pose estimation datasets, i.e. LINEMOD [Hinterstoisser et al., 2012], YCB-Video [Xiang et al., 2017] and TUD-Light [Hodan et al., 2018] datasets. Experimental results show that the proposed method achieves state-of-the-art or comparable performance in terms of both accuracy and speed on these datasets.

In summary, the contributions of this chapter are as follows:

- We propose a novel real-time framework to estimate 6D object pose from RGB-D data in a global to local (G2L) way. Due to efficient feature extraction, the framework runs at over 20fps on a GTX 1080 Ti GPU, which is fast enough for many applications.

- To better utilize viewpoint information, we propose an orientation-based point-wise embedding vector features (EVF) mechanism.

- We propose a rotation residual estimator to estimate the residual between predicted rotation and ground truth, which further improves the accuracy of rotation prediction.

4.2 Related Works

Most of the related works have been described in Chapter 2, in the following, we analyse the works related to the proposed G2L pipeline.

4.2.1 Pose Estimation at One-stage

Given an image, previous methods aim to estimate the 6D pose, which includes 3D position and orientation, at once. Conventional methods [Hinterstoisser et al., 2012, Lepetit et al., 2005, Lowe, 1999] calculated 6D object pose by matching RGB features between templates (or 3D object model) and test image. They treated the 6D pose estimation as a classification task. However, these methods mainly utilized handcrafted features that are not robust to background clutter or image variations [Peng et al., 2018, Shin and Balasingham, 2017, Yuan et al., 2016]. To overcome this issue, many methods began to use deep learning techniques that have been made huge progress in the image field, in their algorithms. For example, Rad et. al. [Hu et al., 2019, Oberweger et al., 2018, Peng et al., 2018, Rad and Lepetit, 2017, Rad et al., 2018, Tekin et al., 2018]

employed CNNs to predict 2D keypoints (projected by the 3D bounding box of the object 3D model). The estimated 2D keypoints contain both rotation information and translation information, then these methods calculated the 6D pose from the estimated 2D keypoints with pre-defined 3D keypoints.

4.2.2 Pose Estimation with Decoupling

Different from the aforementioned methods, some other methods [Li et al., 2018, 2019, Xiang et al., 2017] estimated the two terms, i.e. rotation term and translation term, of 6D pose separately. [Kehl et al., 2017] used one network to output the rotation and translation. They discretized the rotation and translation space into classifiable bins. Then they estimated the rotation and translation by a CNN-based classifier. [Li et al., 2018, 2019, Xiang et al., 2017] employed different network branches for rotation and translation. In [Li et al., 2019] the network estimated rotation and translation via 2D-3D correspondences. In [Xiang et al., 2017] the rotation is estimated via quaternion regression, and in [Li et al., 2018] the rotation and translation estimation are regarded as a classification task. When depth information is available, [Li et al., 2018] added depth as the fourth channel for the network. Our proposed method also estimates the rotation and translation with different branches, but with the 3D point could input. It has been shown that treating depth as a 2D image cannot fully take advantage of 3D geometric information [Maturana and Scherer, 2015]. Therefore, our method can better extract viewpoint information from the given depth data.

4.2.3 Iterative Pose Refinement

As mentioned in Section 2.3.1, to achieve better 6D pose estimation results, the ICP algorithm is employed after the initial poses are predicted. However, ICP suffers from the issue of local minima due to its calculation procedure. To overcome this issue, some

methods [Li et al., 2018, Rad and Lepetit, 2017, Zakharov et al., 2019] refined the initial pose based on image observation. They first map the 3D model of the target object with the initially estimated pose, then they compared the rendered image with real-world observation. They usually trained another network that takes the rendered 3D model and real-world observation as input and outputs the final 6D pose.

In contrast, we propose a new rotation residual network to boost the rotation estimation. With the new proposed residual network, we can get the rotation and predicted rotation residual parallelly, which saves the running time compared to iterative pose refinement. In addition, our method can be easily extended to an iterative version for further refinement, and the iterative version of our method does not need an extra module or network (See Section 4.4.5 for details).

4.3 Global to Local Real-Time Pipeline

In this section, we will describe our framework in detail. In Figure 4.1, we show the architecture of our G2L-Net that estimates the 6D object pose in three steps: global localization, translation localization, and rotation localization.

In the global localization step, we try to complete these tasks: recognize and locate the target object. In the translation localization step, we further access the more precise position of the object based on the first step. Based on the first two steps, we finally estimate rotation by the proposed point-wise embedding vector features and rotation residual estimator. Since for a single step, the network can easily achieve good performance, when combining these three steps, our proposed method achieve state-of-the-art performance without the post-refinement component that is widely used in 6D pose methods [Kehl et al., 2017, Rad and Lepetit, 2017, Xiang et al., 2017].

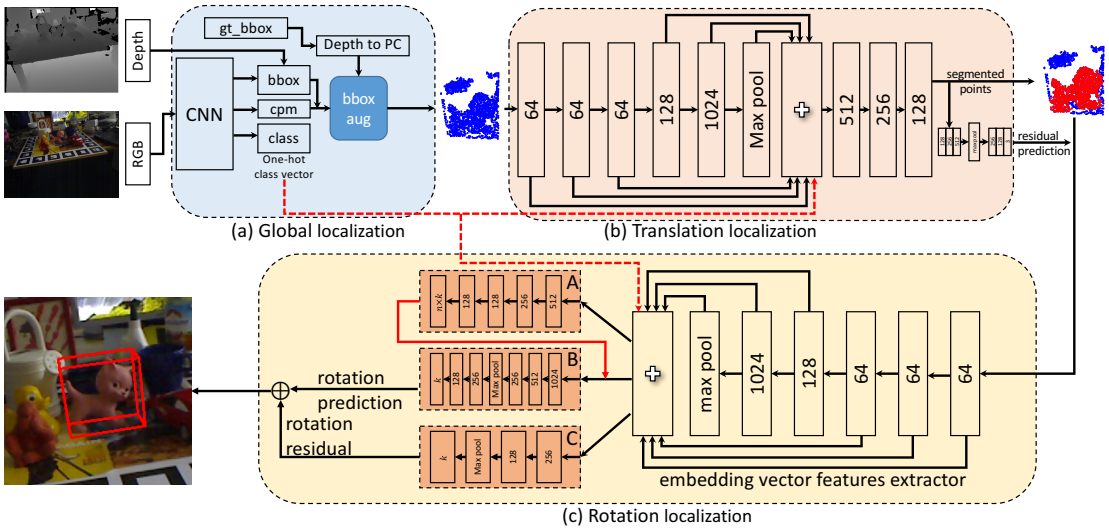


Figure 4.1: **Pipeline of the proposed G2L-Net.** (a) For the RGB image, we use a 2D detector to detect the bounding box (bbox) of the target object and the object label, which is used as one-hot features for the following networks. Also, we additionally choose the maximum probability location in the class probability map (cpm) as the sphere center (we transfer this 2D location to 3D with known camera parameters and corresponding depth value), which is used to further reduce the 3D search space. (b) Given the point clouds in the object sphere, we use translation localization networks to perform 3D segmentation and translation residual prediction. Then we use the 3D segmentation mask and the predicted translation to transfer the object point cloud into a local coordinate. (c) In the rotation localization network, we first train the embedding vector feature extractor and the top decoder (block A) to predict point-wise unit vectors pointing to the keypoints for embedding vector feature extraction. Then we feed the extracted features to decoders: the middle decoder (block B) directly outputs the rotation prediction, and the bottom decoder (block C) outputs the residual between predicted rotation and ground truth. k is the dimension of the output vector. Hollow "$+$" denotes feature concatenation.

4.3.1 Global Localization

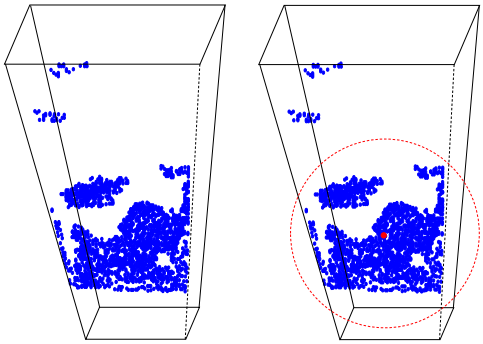
In this step, we aim to limit the 3D search space of the object to a more tight space in the global scene. There are three dimensions, x, y, z, to be limited. With the help of a 2D detector, we can fast locate the x, y position of the target object: inspired by [Qi et al., 2018] we train a state-of-the-art 2D CNN detector, YOLOv3 [Redmon and Farhadi, 2018], to detect the object bounding box in RGB image and output object label. However, in [Qi et al., 2018], they only use the 2D bounding box to generate frustum proposals which can only reduce the 3D search space of two axes ($\mathcal{X}$,$\mathcal{Y}$). The third axis ($\mathcal{Z}$) still has infinite possibilities.

To overcome this issue, we propose to employ a 3D sphere to further reduce the 3D search space in the third axis (z) (see Figure 4.2 for details). The center of the 3D sphere $[X, Y, Z]$ is transferred from the 2D location $[x, y]$ that has the maximum value in the class probability map with known camera matrix $\mathcal{K}$ and corresponding depth value z:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = s * \left(\left(\begin{bmatrix} \frac{1}{f_x} & 0 & 0 \\ 0 & \frac{1}{f_y} & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \left(\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} - \begin{bmatrix} u_x \\ u_y \\ 0 \end{bmatrix} \right) \right) \right), \tag{4.1}$$

where s is a scale factor that has the same value as z, f_x, f_y, u_x and u_z are parameters in $\mathcal{K}$.

The radius of this 3D sphere is the n times diameter of the detected object (set as 1 in our experiments). We only choose points in the intersection region of this compact 3D sphere and the frustum, which makes the learning task easier for the following steps. Another possible way is to cut out a smaller frustum around the center point $[X, Y, Z]$, which is similar to the proposed 3D sphere.



(a) (b)

Figure 4.2: **Global 3D sphere.** In the global localization step, we locate the object point clouds by bounding box as well as a 3D sphere. (a) Locate the object point cloud by bounding box. In this case, it can only locate the object in two-dimensional space, some points can still be far away from the object on the third axis. (b) Locate the object point cloud by the intersection region of the bounding box and the 3D sphere. All points lay in a more compact space.

4.3.2 3D Point Cloud Segmentation and Translation Estimation

Although the extracted point cloud from the global localization step is tight, there are still two issues that remain: 1) the point cloud in this 3D sphere contains both object points and non-object points, and 2) the precise distance between the object and the camera is still unknown. With these two issues, the network cannot estimate the pose accurately.

For the first issue, we use PointNets [Qi et al., 2017,] to segment the object point cloud from background point clouds. For the second issue, we train another tiny PointNet to output the residual vector between the mean value $\bar{\mathbf{T}}$ of the segmented points and object translation $\mathbf{T}$. This residual can be used to calculate the translation of the object. Then we employ the calculated translation vector to transfer the segmented point cloud to a canonical space. With this step, we can remove the shift property of the point cloud, which makes the rotation estimation task easier. An alternative way is to directly

output the translation value of the object, however, due to the large displacements among different poses, the network cannot converge easily in this fashion.

4.3.3 Rotation Estimation with Embedding Vector Features

We use PointNet as our main backbone. However, PointNet [Qi et al., 2017] directly operates on points coordinate, which means they are sensitive to points shift [Lin et al., 2020]. This makes directly estimating rotation from the point cloud difficult. To eliminate the impact of the points shift, we use the translation vector calculated from the previous step to transfer the point cloud of the object to local canonical space. Thereby, the viewpoint information is more evident and the PointNet can be free from the point cloud shift. In this local space, the network can estimate the rotation easily and accurately.

Embedding Vector Features

Theoretically, we need at least four different viewpoints to cover all parts of an object (see Figure 4.3) in 3D space [Draim, 1987]. For the pose estimation task, we usually have hundreds of different viewpoints for one object during training which should be enough to access the accurate pose of the object. However, the previous methods [Qi et al., 2018, Wang et al., 2019] still cannot get accurate pose results in real-time. One main issue is that they cannot extract the viewpoint information effectively.

To effectively extract the viewpoint information from limited training examples, inspired by the work described in Chapter 3, we propose the point-wise embedding vector features. Specifically, we design the rotation localization network architecture as shown in Figure 4.5 to predict point-wise unit vectors pointing to keypoints (illustrated in Figure 4.4). The keypoints are some pre-defined 3D points based on each 3D object model. Two aspects need to be decided for the keypoints: number and location. A sim-

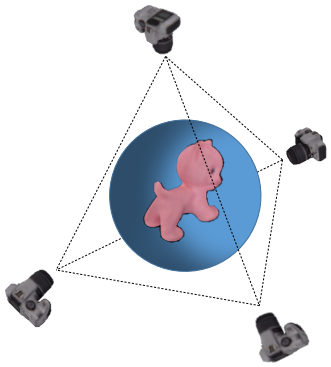


Figure 4.3: **Different viewpoints.** For a 3D object, we need at least four viewpoints to cover all the parts of the 3D object.

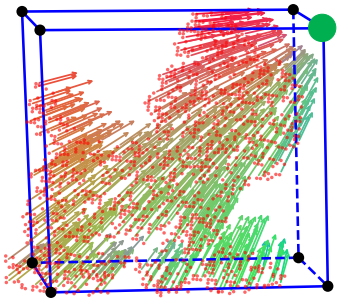


Figure 4.4: **Point-wise vectors.** Here we show point-wise vectors pointing to the green keypoint. We train our network to predict such directional vectors. The color is decided by the orientation of the vector. We map the angle of the vector relative to the three axes to RGB space.

ple way is to use the 8 corners of the 3D bounding box of the object model as keypoints, which is shown in Chapter 3, Figure 3.5 (a). This definition is widely used by many CNN-based methods in 2D cases [Oberweger et al., 2018, Rad and Lepetit, 2017, Rad et al., 2018, Tekin et al., 2018]. In our experiments, we also use this keypoints fashion, and in the experimental part, we show that this fashion can get the best performance.

The rotation localization network consists of three blocks: A, B, and C (as shown in Figure 4.5). We train block A with embedding vector feature extractor to predict the unit vectors pointing to the keypoints. By training this block, the network learns to extract point-wise embedding vector features from the input segmented point cloud. Then we use block B to integrate the point-wise embedding vector features with the

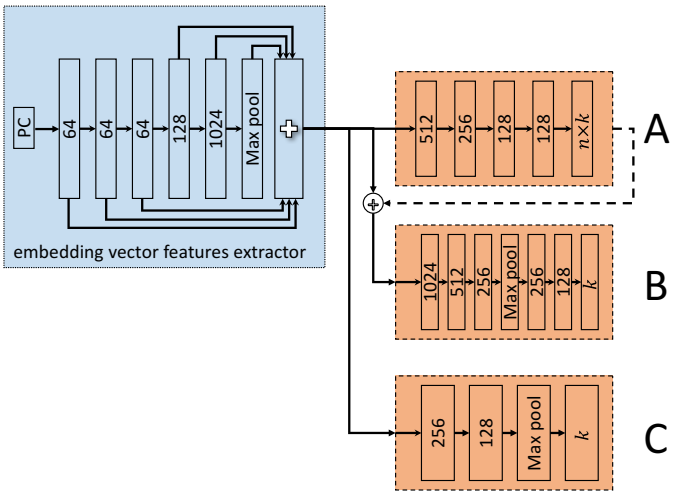


Figure 4.5: **Architecture of the rotation localization network.** In the training stage, there are three blocks in the rotation localization network. We use block A to predict the unit vectors pointing to the keypoints, the loss function of this block is shown in Equation 4.3. By training this block via the embedding vector feature extractor, the network can learn how to extract point-wise embedding vector features from the input point cloud. Then we use block B to integrate the point-wise embedding vector features with the output of block A to predict object rotation that is represented as the 3D coordinates of 8 keypoints in our pipeline. The loss function of this block is described in Equation 4.4. For rotation residual estimator block C, we use the Euclidean distance (see Equation 4.5) between the predicted 3D keypoints position (output of block B) and ground truth as ground truth. $k \in \mathbb{R}^{24}$ is the dimension of the output rotation vector. Hollow " $+$ " denotes feature concatenation.

output of block A to predict object rotation. The loss function of this block is the mean square error between the predicted rotation and ground truth. For rotation residual estimator block C, we use the Euclidean distance between the predicted 3D keypoints position (output of block B) and ground truth as ground truth (see Section 4.3.3 for details).

Compared to the global features used in [Qi et al., 2018], our new proposed EVF can learn more discriminative features for pose estimation. In Figure 4.6, we show the

Figure 4.6: **Different feature fashion with T-SNE for visualization.** The left image is the T-SNE distribution for EVF of 50 point cloud examples. The right image is the distribution for the global features of 50 point cloud examples.

T-SNE distribution of EVF and global features for randomly selected 50 point cloud examples from the training set. From Figure 4.6, we can see that the distribution of our EVF is well distributed, while some feature points are entangled with each other in global feature distribution.

Different from other state-of-the-art methods [Chen et al., 2020, Peng et al., 2018, Xiang et al., 2017], we adopt a multilayer perceptron (MLP) that takes point-wise embedding vector features as input and outputs the rotation of the object (shown in Figure 4.1). Please note, during inference, we use the rotation matrix to represent the rotation, which is computed from the keypoint positions using the Kabsch algorithm. Over the training process, as per the definition of point-wise vectors, we used the 3D keypoint positions to represent rotation. In experiments, we have found that our proposed method can make faster and more accurate predictions than other methods [Chen et al., 2020, Peng et al., 2018, Xiang et al., 2017].

Rotation Residual Estimator

To better utilize the viewpoint information from the point-wise embedding vector features, we add an extra network branch (block C in Figure 4.5) to estimate the residual between estimated rotation (block B in Figure 4.5) and ground truth. However, we do not have the ground truth for this residual estimator. To address this problem, we train this estimator in an online fashion. Assuming that the ground truth for block B of rotation localization network is GT_B and the output of block B is OT_B, then the target of our rotation residual estimator (block C) is the residual between GT_B and OT_B. As the rotation network converges, it becomes harder to learn the residual. If the rotation localization network fully exploits the embedding vector features, the rotation residual estimator can be ignored. However, when the rotation network cannot fully exploit the embedding vector features, the rotation residual estimator has a big impact on the final results, we show this property of the rotation residual estimator in Figure 4.8 (b). Please note, our proposed rotation residual estimator is different from the post-refinement module in the previous state-of-the-art methods [Li et al., 2018, Wang et al., 2019, Xiang et al., 2017]. Our proposed rotation residual estimator outputs rotation residual with estimated rotation synchronously, which saves the running time.

Our rotation estimator and rotation residual estimator together estimate the 6D pose of the object in a cascaded fashion. In [Dollár et al., 2010], the authors also estimated the pose in a cascaded fashion, however, in essence, their method is an iterative algorithm, our method output the rotation and corresponding residual at the same time.

4.4 Experiments

There are two parts in this experiments section. Firstly, we do ablation studies on keypoints selection schemes and empirically validate the three innovations introduced in

our new frame on the LINEMOD dataset: 3D sphere (‘SP”), point-wise embedding vector features (“EVF”) and rotation residual estimator (“RRE"). Then we test our proposed G2L-Net on several benchmark datasets, i.e. LINEMOD, YCB-Video, and TUD-L datasets. Our method achieves state-of-the-art performance on these datasets.

4.4.1 Implementation Details

We implement our framework using Pytorch. All the experiments are conducted on a PC with Intel i7-4930K 3.4GHz CPU and GTX 1080 Ti GPU.

Our pipeline contains two different deep network architectures: CNN and multi-layer perceptron (MLP). For the CNN part, we fine-tune the 2D detector, i.e. YOLOv3 [Redmon and Farhadi, 2018], which trained on the ImageNet [Deng et al., 2009] to locate the 2D region of interest and calculate the 3D sphere. For the MLP part, i.e. PointNets, we train the translation and rotation localization networks together. The translation localization network performs two related tasks: 3D segmentation and translation residual estimation. For 3D segmentation, we use cross-entropy loss $\mathcal{L}_{seg}$. For translation residual estimation, the loss function is defined as:

$$\mathcal{L}_{tran} = \left\| \Delta\mathbf{T} - \widetilde{\Delta\mathbf{T}} \right\|_2, \tag{4.2}$$

where $\Delta\mathbf{T} = \mathbf{T} - \bar{\mathbf{T}}$ is the residual between the ground truth translation $\mathbf{T}$ and the mean value $\bar{\mathbf{T}}$ of the segmented points, $\widetilde{\Delta\mathbf{T}}$ is the estimated translation residual.

The loss function of block A is defined as the mean square error between the predicted and ground truth directional vectors:

$$\mathcal{L}_A = \frac{1}{KN_O} \sum_{k=1}^{K} \sum_{\mathfrak{p} \in O} \left\| \widetilde{v}_k(\mathfrak{p}) - v_k(\mathfrak{p}) \right\|_2, \tag{4.3}$$

where K is the number of 3D keypoints. $\widetilde{v}_k(\mathfrak{p})$ and $v_k(\mathfrak{p})$ are the k_{th} predicted vector and the ground truth vector of the point $\mathfrak{p}$, respectively. N_O denotes the number of object points.

For block B, we use the modified corner loss proposed in [Qi et al., 2018]:

$$\mathcal{L}_B = \frac{1}{8}\sum_{k=1}^{8}\left\|\mathbf{p}_{k\mathbf{R}}^{key} - \widetilde{\mathbf{p}}_{kb}^{key}\right\|_2, \tag{4.4}$$

where $\mathbf{p}_{k\mathbf{R}}^{key}$ is k_{th} keypoint transferred by the ground truth rotation $\mathbf{R}$, $\widetilde{\mathbf{p}}_{kb}^{key}$ is the output of block B. For block C, the loss function is defined as:

$$\mathcal{L}_C = \frac{1}{8}\sum_{k=1}^{8}\left\|[\mathbf{p}_{k\mathbf{R}}^{key} - \widetilde{\mathbf{p}}_{kb}^{key}] - \widetilde{\mathbf{p}}_{kc}^{key}\right\|_2, \tag{4.5}$$

where $\widetilde{\mathbf{p}}_{kc}^{key}$ is the output of block C.

We combine all the losses together to simultaneously optimize all networks:

$$\mathcal{L} = \mathcal{L}_{tran} + \lambda_1\mathcal{L}_{seg} + \lambda_2\mathcal{L}_A + \lambda_3\mathcal{L}_B + \lambda_4\mathcal{L}_C, \tag{4.6}$$

where $\lambda_i = \{0.01, 1, 0.001, 0.001\}$.

Gap Between Different Architectures

There is one issue when we use different network architectures to process color images and depth images, these two architectures cannot seamlessly connect. For example, in the test phase, the state-of-the-art methods [Chen et al., 2020,, Wang et al., 2019,] first use the color image to generate a mask or bounding box to locate the corresponding position of the depth image and then convert the cropped depth image into point cloud for subsequent processing. However, in the training stage, the point cloud is generated by the ground truth mask or bounding box. When we use the points under these boxes (or masks), the generalization performance of the trained model will suffer. The reason is that there exist some differences between the ground truth and prediction, however, the previous method did not consider how to mitigate the differences, which may lead to the overfitting issue of the trained model.

Although, there are some data augmentation techniques proposed and employed in 3D tasks [Chen et al., 2020,, Choi et al., 2020, Qi et al., 2018, Shi et al., 2020], or 2D

computer vision tasks [Redmon and Farhadi, 2018], they either focused on 3D tasks or 2D tasks, how to bridge the gap between 2D augmentation and 3D augmentation is still an open problem. One possible solution is generating the data with various masks or bounding boxes offline, while this strategy will significantly increase the storage burden. Another way is to use the predicted results from the CNN for training, however, there are two issues here. First, if the detection is failed, we cannot train the subsequent networks. Second, the prediction of the CNN is fixed for a certain object, which cannot cover the various scenarios in test scenes. Here we provide an efficient way to alleviate this gap.

Fast Point-Wise Relabelling

From Figure 4.7, we can see that there are three valid regions for the ground truth and augmented 2D boxes: intersection region (yellow), newly added region (blue), and region (purple) outside the augmented box. Our task is to access the point-wise label in the blue and yellow regions. In the yellow region, the points are labelled in the ground truth bounding box, and the points in the blue region are the background. The points in the purple region are abandoned after the augmented. It seems that we already know the label of all points in the augmented region, however, the fact is that the points in the augmented 2D box are transformed from the newly cropped depth region by the augmented 2D box, the order of these points are different from the pre-labelled points in the ground truth 2D box, which means we cannot find the corresponding label in the known label sequence. Our solution is to label the yellow region and blue region separately. First, we find the intersection region of the augmented and ground truth box (yellow region shown in Figure 4.7). Then for the yellow region, we use the pre-defined point label. The issue is how to efficiently detach the blue and yellow region in the augmented box. Our method is to set the depth value in the yellow region as zero, then for the transformed point cloud, the coordinate of z is zero. Via removing the points with z zero, the rest points are located in the blue region. Finally, we label these points

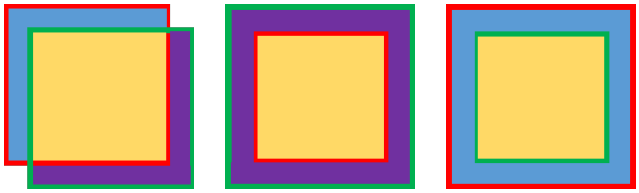


Figure 4.7: **Relations between bounding boxes**. Green 2D bounding box denotes ground truth. The red bounding box represents the augmented box. The yellow region is the intersection region of the two boxes; the purple region is the region in the ground truth box but outside the intersection region; the blue region is the region that belongs to the augmented box apart from the intersection region.

in the blue region as background points. Please refer Algorithm 1 for more details.

Algorithm 1: Online 2D Augmentation

Input: $bbox_{gt}$, $bbox_{aug}$, camera matrix $\mathcal{K}$, depth image

Output: points and point-wise label in $bbox_{aug}$

1. Find the intersection region of $bbox_{gt}$ and $bbox_{aug}$;
2. Find the point and label in the intersection region: $point_{inter}$ and $label_{inter}$;
3. Set the depth value in the intersection region as zero;
4. Transfer the depth in the $bbox_{aug}$ to 3D point cloud: $point_{3D}$;
5. Remove the zero value in the z coordinate of $point_{3D}$ to get $point_{rest}$;
6. Label the rest points of $point_{3D}$ as background points and store the label in $label_{rest}$;
7. Concatenate $point_{rest}$ and $label_{rest}$ with $point_{inter}$ and $label_{inter}$ to get new points and label: $point_{aug}$ and $label_{aug}$

4.4.2 Datasets

We employ the LINEMOD, YCB-Video, and TUD-Light datasets to test the proposed method. The details of these datasets are described in Chapter 2. However, all these

Table 4.1: **Ablation studies of different novelties on the LINEMOD dataset with ADD(-S) metric**. “SP" means 3D sphere, “EVF" means embedding vector features, and “RRE" denotes rotation residual estimator.

Method	SP	EVF	RRE	Acc	Speed(fps)
EXP1	×	×	×	93.4%	25
EXP2	✓	×	×	95.8%	25
EXP3	✓	✓	×	98.4%	23
EXP4	✓	✓	✓	98.7%	23

datasets do not contain the point-wise label. To train G2L-Net in a supervised fashion, we adopt an automatic way to label each point of the point cloud. As described in Section 2.4, we label each point in two steps. First, for the 3D model of an object, we transform it into the camera coordinate using the corresponding ground truth. We adopt the implementation provided by [Hodaň et al., 2016] for this process. Second, for each point on the point cloud in the target region, we compute its nearest distance to the transformed object model. If the distance is less than a value $\epsilon = 8mm$, we label the point as 1 (belonging to the object), otherwise 0.

4.4.3 Ablation Studies

Compared to the baseline method [Qi et al., 2018], our proposed method has three novelties. First, we fast locate the object point clouds with a 3D sphere that is different from the frustum method in [Qi et al., 2018]. Second, we use the proposed point-wise embedding vector features to estimate the rotation of the point cloud, which can better utilize the viewpoint information. Third, we propose a rotation residual estimator to estimate the rotation residual between ground truth and predicted rotation. From Table 4.1, we can see that the proposed three improvements can boost performance.

Table 4.2: **Ablation studies of different keypoints parameters on the LINEMOD dataset with ADD(-S) metric**. **BBX-8** means using the 8 corners of the 3D bounding box as keypoints. **FPS-K** denotes K keypoints generated by the FPS algorithm.

Method	BBX-8	FPS-4	FPS-8	FPS-12
Acc	98.7%	98.5%	98.4%	98.6%
Speed (fps)	23	23	23	23

We also compare the different keypoints selection schemes in Table 4.2. However, it shows that different keypoints selection schemes make little difference in the final results. For simplicity, we use the 8 corners of the 3D bounding box as keypoints in our experiments.

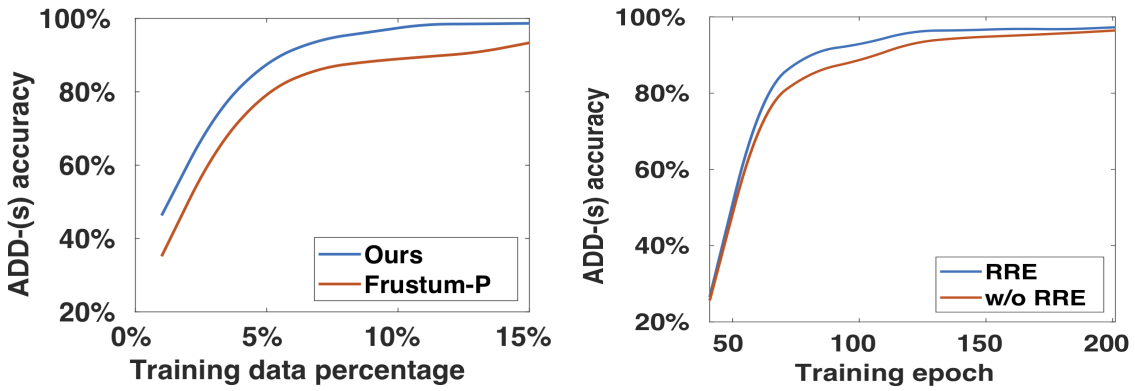
4.4.4 Generalization Performance

In this section, we evaluate the generalization performance of the proposed method G2L-Net. We gradually reduce the size of training data to see how the performance of the algorithm is affected by the LINEMOD dataset. From Figure 4.8 (a), we can see that even with only 5% of the training data, which is 1/3 of the normal setting, the performance (88.5%) is still comparable.

4.4.5 Comparison with State-of-the-Arts

In this section, we compare our method with the state-of-the-art pose estimation methods on two popular datasets: LINEMOD [Hinterstoisser et al., 2012] and YCB-Video [Xiang et al., 2017] datasets. Both visual and quantitative results are provided.

6D Object Pose Estimation on LINEMOD: Same as other state-of-the-art methods, we use 15% of each object sequence to train and the rest of the sequence to test on the



(a)Training size and performance (b) Training epoch and performance

Figure 4.8: **Performance on LINEMOD dataset.** (a) Influence of training data size using the ADD(-S) metric. When using the same training size, compared to Frustum-P [Qi et al., 2018], our method improves the performance significantly. For simplicity, here we provide ground truth 2D bounding box and randomly choose an object point as 3D sphere center for evaluation. (b) As the rotation localization network converges, the impact of the rotation residual estimator (RRE) decreases.

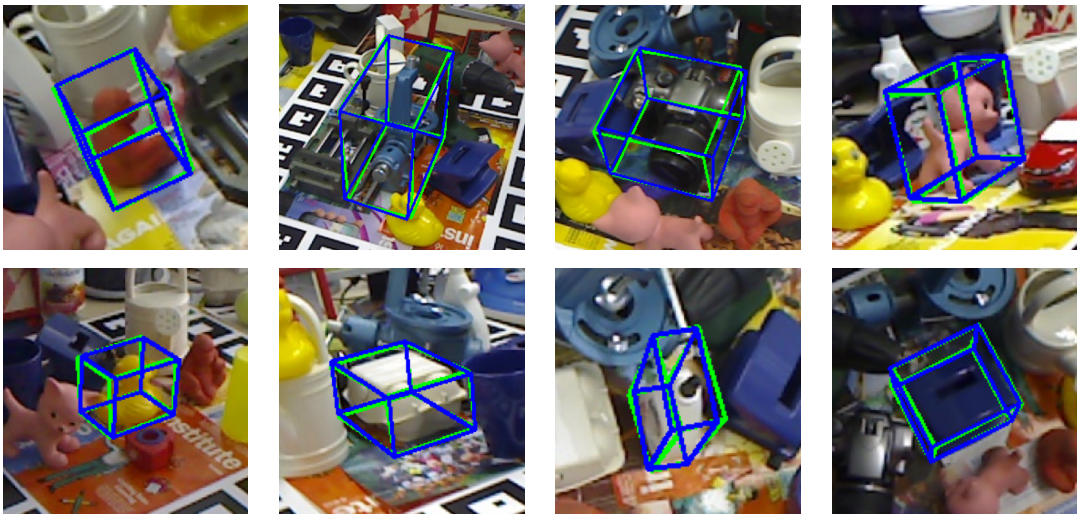


Figure 4.9: **Qualitative pose estimation results on the LINEMOD dataset**. Green 3D bounding boxes denote ground truth. Blue 3D bounding boxes represent our results. Our results match ground truth well.

LINEMOD dataset. In Table 4.3, we compare our method with state-of-the-art RGB and RGB-D methods. The numbers in brackets are the results without post-refinement.

Table 4.3: **6D pose estimation accuracy on the LINEMOD dataset.** We use ADD metric to evaluate the methods. For symmetric objects 'Egg Box' and 'Glue', we use the ADD-S metric. Note that, we summarize the pose estimation results reported in the original papers on the LINEMOD dataset. For refinement, 'ICP' means ICP refinement, and 'HV' means hypothesis generation/verification refinement.

Method	PVNet [Peng et al., 2018]	DeepIM [Li et al., 2018]	DPOD [Zakharov et al., 2019]	Frustum-P [Qi et al., 2018]	Hinterstoisser [Hinterstoisser et al., 2016]	DenseFusion [Wang et al., 2019]	PointPoseNet [Chen et al., 2020]	Ours
Input	RGB	RGB	RGB	RGB-D	Depth	RGB-D	RGB-D	RGB-D
Refinement	×	✓ (ICP)	✓$(ICP)(\times)$	×	✓(ICP)	✓$(ICP)(\times)$	✓(HV)	×
Ape	43.6%	77.0%	87.7% (53.3%)	85.5%	**98.5%**	92.3% (79.5%)	97.9%	96.8%
Bench Vise	**99.9%**	97.5%	98.5% (95.3%)	93.2%	99.0 %	93.2%(84.2%)	99.6%	96.1%
Camera	86.9%	93.5	96.0% (90.4%)	90.0%	**99.3%**	94.4%(76.5%)	98.5%	98.2%
Can	95.5%	96.5%	99.7% (94.1%)	91.4%	98.7%	93.1%(86.6%)	**99.4%**	98.0%
Cat	79.3%	82.1%	94.7% (60.4%)	96.5%	**99.9%**	96.5%(88.8%)	99.3%	99.2%
Driller	96.4%	95.0%	98.8% (97.7%)	96.8%	93.4%	87.0%(77.7%)	97.5%	**99.8%**
Duck	52.6%	77.7%	86.3% (66.0%)	82.9%	**98.2%**	92.3%(76.3%)	96.1%	97.7%
Egg Box	99.2%	97.1%	99.9% (99.7%)	99.9%	98.8%	99.8%(99.9%)	97.9%	**100%**
Glue	95.7%	99.4%	96.8% (93.8%)	99.2%	75.4%	**100%** (99.4%)	**100%**	**100%**
Hole Puncher	81.9%	52.8%	86.9% (65.8%)	92.2%	98.1%	92.1%(79.0%)	97.9%	**99.0%**
Iron	98.9%	98.3%	**100%** (99.8%)	93.7%	98.3%	97.0% (92.1%)	99.4%	99.3%
Lamp	99.3%	97.5%	96.8% (88.1%)	98.2%	96.0%	95.3%(92.3%)	99.1%	**99.5%**
Phone	92.4%	87.7%	94.7% (74.2%)	94.2%	98.6%	92.8%(88.0%)	**98.9%**	**98.9%**
Speed(FPS)	25	5	33(40)	12	8	16(20)	4	23
Average	86.3%	88.6%	95.2% (83.0%)	93.4%	96.3 %	94.3 %(86.2%)	98.4%	**98.7 %**

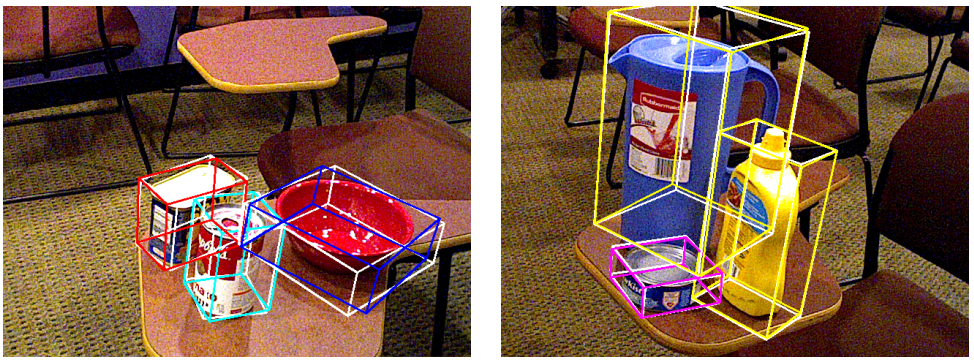


Figure 4.10: **Visualizing pose estimation results on YCB-Video**. White 3D bounding boxes are ground truth. Colorful 3D bounding boxes represent our results. For different objects, our prediction matches ground truth well.

We use Frustum-P [Qi et al., 2018] as our baseline. We re-implement Frustum-P to regress the 3D bounding box corners of the objects. From Table 4.3, we can see that our method outperforms the baseline by 5.4% in ADD accuracy and runs 2 times faster than

Table 4.4: **6D Pose estimation accuracy on the YCB-V dataset.** We use the ADD-S AUC metric to evaluate the methods. For refinement, 'ICP' means ICP refinement, and 'HV' means hypothesis generation/verification refinement.

Method	PoseCNN [Xiang et al., 2017] + ICP	MCN [Li et al., 2018]	DenseFusion [Wang et al., 2019] (no refinement)	Heatmap [Oberweger et al., 2018]	PVNet [Peng et al., 2018]	PointPoseNet [Chen et al., 2020]	Ours
Input	RGB+Depth	RGB+Depth	RGB+Depth	RGB	RGB	RGB+Depth	RGB+Depth
Refinement	✓(ICP)	✓(HV)	✓(ICP)	×	×	✓(HV)	×
002_master_chef_can	95.8%	**96.2**%	95.2%	69.0%	-	95.2%	94.0%
003_cracker_box	91.8%	90.9 %	**92.5**%	80.2%	-	89.1%	88.7%
004_sugar_box	**98.2**%	95.3%	95.1%	76.2%	-	96.0%	96.0%
005_tomato_soup_can	94.5%	**97.5**%	93.7%	70.0%	-	91.9%	86.4%
006_mustard_bottle	**98.4**%	97.0%	95.9%	84.8%	-	96.3%	95.9%
007_tuna_fish_can	**97.1**%	95.1%	94.9%	49.4%	-	97.0%	96.0%
008_pudding_box	**97.9**%	94.5%	94.7%	82.2%	-	96.5%	93.5%
009_gelatin_box	**98.8**%	96.0%	95.8%	81.8%	-	97.8%	96.8%
010_potted_meat_can	92.8%	**96.7**%	90.1%	66.2%	-	86.2%	86.2%
011_banana	**96.9**%	94.4%	91.5%	52.9%	-	94.3%	96.3%
019_pitcher_base	**97.8**%	96.2%	94.6%	69.9%	-	93.0%	91.8%
021_bleach_cleanser	**96.8**%	95.4%	94.3%	73.3%	-	93.5%	92.0%
024_bowl	78.3%	82.0%	86.6%	80.3%	-	82.8%	**86.7**%
025_mug	95.1%	96.8%	95.5%	50.5%	-	**97.2**%	95.4%
035_power_drill	**98.0**%	93.1%	92.4%	78.3%	-	91.1%	95.2%
036_wood_block	90.5%	**93.6**%	85.5%	65.2%	-	86.1%	86.2%
037_scissors	92.2%	94.2%	**96.4**%	28.2%	-	93.0%	83.8%
040_large_marker	**97.2**%	95.4%	94.7%	48.2%	-	97.0%	96.8%
051_large_clamp	75.4%	93.3%	71.6%	47.2%	-	96.1%	**94.4**%
052_extra_large_clamp	65.3%	90.9%	69.0%	47.5%	-	**93.6**%	92.3%
061_foam_brick	**97.1**%	95.9%	92.4%	85.6%	-	92.9%	94.7%
Average	93.0%	**94.3**%	91.2%	66.1%	73.4%	93.2%	92.4%
Speed (fps)	< 0.1	-	20	-	25	<2	21

Table 4.5: **6D Pose estimation accuracy on the TUD-Light dataset.** We use the ADD metric to evaluate the methods. "ITER_I" means iteration I times.

Method	G2L	G2L+ITER_1	G2L+ITER_2	G2L+ITER_4
Obj_001	91.5%	94%	94%	94.0%
Obj_002	82.5%	86.5%	88.1%	88.7%
Obj_003	86.5%	90%	92.5%	96.0%
Average	86.8%	90.2%	91.5%	92.4%
Speed (fps)	23	10	5	2

the baseline method. Compared to the method [Hinterstoisser et al., 2016] that used depth information, our method outperforms it by 2.4% in ADD accuracy and runs about

Figure 4.11: **2D detection results and iterative refinement**. Green 2D bounding boxes denote ground truth. Light green bounding boxes represent original prediction results, and yellow bounding boxes are refinement results.

3 times faster than it. When compared with our previous method [Chen et al., 2020], our method achieves better performance and faster speed. Although DPOD and PVNet are faster than our method, they only took RGB images as input. When using depth information, our method achieves the fastest inference speed. In Figure 4.9, we provide a visual comparison of the predicted pose versus the ground truth pose.

6D Object Pose Estimation on YCB-Video: Different from the LIMEMOD dataset, in the YCB-Video dataset, each frame may contain multiple target objects. Our method can also estimate the 6D pose for multiple objects at a fast speed. Table 4.4 compares our method with other state-of-the-art methods [Chen et al., 2020, Li et al., 2018, Wang et al., 2019, Xiang et al., 2017] on YCB-Video dataset under ADD-S AUC metric. From Table 4.4, we can see that our method achieves a comparable accuracy (92.4%) and is the fastest one (21fps) among all comparisons. In Figure 4.10, we also provide visualization results on this dataset.

6D Object Pose Estimation TUD-Light: During training, we use the data augmentation technique to do 2D box augmentation to cover the possible distribution of the output distribution. This works well in datasets such as LINEMOD and YCB-Video that have no significant lighting condition changes. However, when encounters significant lighting condition changes, the performance of the 2D detector will drop, which decreases the accuracy of the following step. As aforementioned, TUD-Light contains different

light conditions, which pose a huge challenge to our 2D detector part. To solve this, we adapt our pipeline in an iterative way, which can significantly improve the results (see Figure 4.11). The iterative version of our method is shown in Figure 4.12. From Table 4.5, we can see that with the proposed refinement procedure, G2L achieves better results.

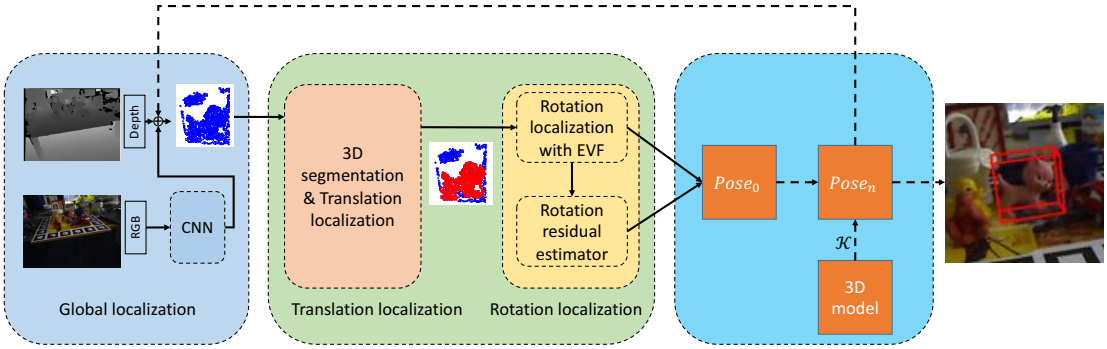


Figure 4.12: **Iterative refinement.** With known camera parameters $\mathcal{K}$ and the refined pose, we can render the object 3D model to 2D space to access the 2D bounding box. Then we feed this new 2D bounding box to the G2L part to calculate the new pose. We can iterate this procedure until it converges.

4.4.6 Running Time

For a single object, given a 480×640 RGB-D image, our method runs at 23fps on a PC environment (an Intel i7-4930K 3.4GHz CPU and one GTX 1080 Ti GPU). Specifically, the 2D detector takes $11ms$ for object location, and the pose estimation part that includes translation localization and rotation localization takes $32ms$. The rotation residual estimator takes less than $1ms$.

4.5 Conclusion

In this chapter, we propose a novel real-time 6D object pose estimation framework. Our G2L-Net decouples the object pose estimation into three sub-tasks: global localization, translation localization, and rotation localization with embedding vector features. In global localization, we use a 3D sphere to constrain the 3D search space into a more compact space than the 3D frustum. Then we perform 3D segmentation and object translation estimation. We use the 3D segmentation mask and the estimated object translation to transfer the object points into local coordinate space. Since viewpoint information is more evident in this canonical space, our network can better capture the viewpoint information with our proposed point-wise embedding vector features. In addition, to fully utilize the viewpoint information, we add the rotation residual estimator, which learns the residual between the estimated rotation and ground truth. In experiments, we demonstrate that our method achieves state-of-the-art performance in real-time.

Although our G2L-Net achieves state-of-the-art performance, there are some limitations. First, the extraction of the embedding vector features is based on the corners of the bounding box of the 3D object model, when the 3D object model of the object is unavailable, the performance will suffer. Second, while our G2L-Net is robust to the labelled data decrease, it cannot handle the case when the labelled data is unavailable of the object. In the next chapter, we will describe how we address the aforementioned limitations by extending our G2L-Net to category-level pose estimation tasks.

Chapter Five

Category-Level 6D Object Pose Estimation with Vector-Based Rotation Representation

5.1 Overview

In Chapter 3 and Chapter 4, we described our pose estimation methods at the instance level, although the proposed methods achieved good performance, they still need a large number of labelled data to train and cannot be easily extended to category-level pose estimation tasks. To alleviate this issue, in this chapter, we report how we estimate the object pose in category level with the proposed method.

In category-level pose estimation tasks, one major challenge is the intra-class variation that includes object shape and color variation. Existing deep learning based methods addressed this problem by mapping the different objects from the same category into a uniform model/map via RGB features or RGB-D fusion features. For example, Wang *et al.* [Wang et al., 2019] trained a modified Mask R-CNN [He et al., 2017] to predict the normalized object coordinate space (NOCS) map of different objects based

on RGB features and then computed the pose with observed depth and NOCS map by Umeyama algorithm [Umeyama, 1991]. Chen *et al.* [Chen et al., 2020] proposed to learn a canonical shape space (CASS) to tackle intra-class shape variations with RGB-D fusion features [Wang et al., 2019]. Tian *et al.* [Tian et al., 2020] trained a network to predict the NOCS map of different objects, with the uniform shape prior learned from a shape collection in NOCS-Synthetic dataset [Wang et al., 2019] and RGB-D fusion features [Wang et al., 2019].

Although these methods achieved state-of-the-art performance, there are still two issues. Firstly, the benefits of using the RGB features or RGB-D fusion features for category-level pose estimation are questionable. In [Vlach, 2016], Vlach et al. showed that people focus more on shape than color when categorizing objects, as different objects in the same category have very different colors but stable shapes (shown in Figure 5.1). Thereby, the adoption of the RGB features for category-level pose estimation can lead to low performance due to huge color variation in the test scene. To address this issue, the previous methods employed a large synthetic dataset [Wang et al., 2019] to increase the generalization performance of the methods. In contrast, to alleviate the color variation, we merely use the RGB features for 2D detection, while using the shape features learned from point cloud cropped in depth image for category-level pose estimation.

Secondly, learning a representative uniform shape requires a large amount of training data; therefore, the performance of these methods is not guaranteed with limited training examples. To overcome this issue, we propose to use 3D graph convolution (3DGC) based autoencoder [Lin et al., 2020] to effectively learn the category-level pose features via observed points reconstruction of different objects instead of uniform shape mapping, and we propose a new vector-based rotation (VDR) representation that uses two decoders to fully decode the learned category-level pose features. We further propose an online 3D data augmentation mechanism for data augmentation to reduce the dependencies of labeled data.

Figure 5.1: **Stable shape and various colors.** Top row: three mug instances randomly chosen from the NOCS-REAL dataset. Bottom row: three mug instances randomly cropped from the internet image search results (using the keyword 'mug'). The color is varied, while the shape is relatively stable.

To summarize, the main contributions of this paper are as follows:

- We propose a 3DGC-based autoencoder to reconstruct the observed points for latent orientation feature learning and estimate category-level 6D object size and pose. Due to the efficient category-level pose feature extraction, the framework runs at 20 FPS on a GTX 1080Ti GPU.
- To fully decode the rotation information from the learned latent features, we design a new VDR representation. This new rotation representation also allows us to naturally handle the circle symmetry object (in Section 5.3.3).
- Based on the shape similarity of intra-class objects, we propose a novel 3D deformation mechanism to augment the training data.

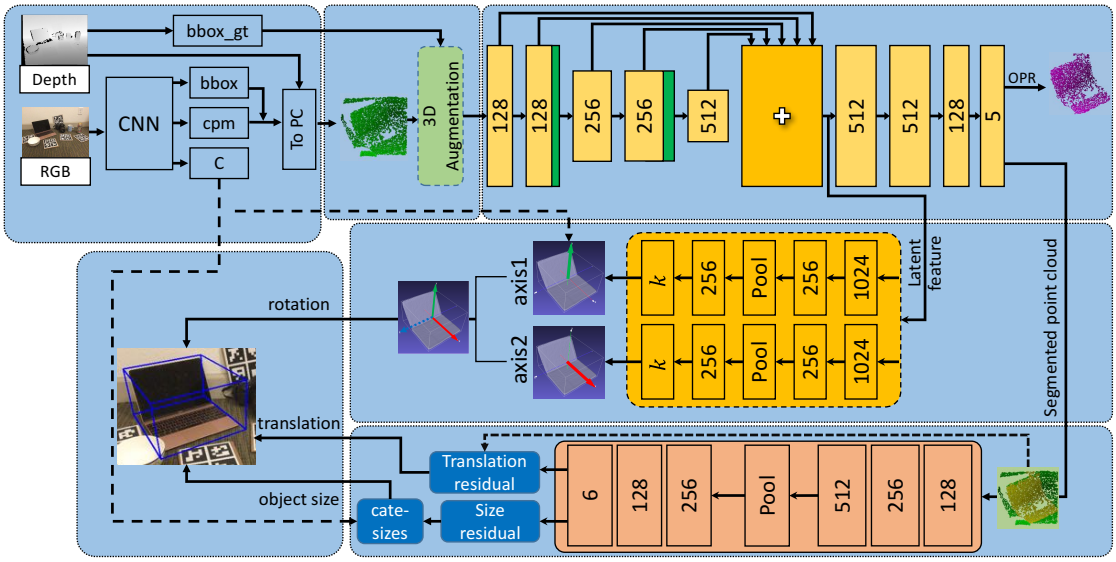


Figure 5.2: **Architecture of FS-Net.** The input of FS-Net is an RGB-D image. For RGB channels, we use a 2D detector to detect the 2D location of the object, category label 'C' (used for next tasks), and class probability map (cpm) (generate the 3D sphere center via maximum probability location and camera parameters). With the detected information and depth image, the points in a compact 3D sphere are generated. Given the points in the 3D sphere, we first use the proposed 3D augmentation mechanism for data augmentation. After that, we use a shape-based 3DGC autoencoder to perform observed points reconstruction (OPR), as well as point cloud segmentation for orientation latent feature learning. Then we decode the rotation information into two perpendicular vectors from the latent features. Finally, we use a residual estimation network to predict the translation and size residuals. 'cate-sizes' denotes the pre-calculated average sizes of different categories, 'k' is the rotation vector dimension, and the hollow '+' means feature concatenation. Please note, the 3D augmentation mechanism and ground truth bounding box (bbox_gt) are only deployed during training.

5.2 Related Works

Related works about instance- and category-level 6D object pose estimation have been reported in Chapter 2, in the following, we add the content close to the work in this chapter.

5.2.1 Data Augmentation

In 3D object detection tasks [Chen et al., 2020,, Qi et al., 2018, Shi et al., 2020], online data augmentation techniques such as translation, random flipping, shifting, scaling, and rotation are applied to original point clouds for training data augmentation. However, these operations cannot change the shape property of the object. Simply adopting these operations on point clouds is unable to handle the shape variation problem in the 3D task. To address this, [Choi et al., 2020] proposed part-aware augmentation that operates on the semantic parts of the 3D object with five manipulations: dropout, swap, mix, sparing, and noise injection. However, how to decide the semantic parts is ambiguous. In contrast, we propose a box-cage-based 3D data augmentation mechanism that generates the various shape variants (shown in Figure 5.6) and avoids semantic parts decision procedure.

5.2.2 Rotation Representation

Widely used rotation representations, such as Euler angle representation, Quaternion representation, and Axis-angle representation, suffer from discontinuous issue [Zhou et al., 2019], which is not suitable for network learning. Although the R_{6D} representation proposed in [Zhou et al., 2019] is free from the discontinuous issue, they took two columns/row from the 3×3 matrix as their new representation, the geometry meaning of this rotation representation is unclear. In contrast, our new VDR representation is

defined based on two vectors, and each vector has a clear geometry meaning: up vector and front vector.

5.3 Proposed Method

In this section, we describe the detailed architecture of FS-Net (shown in Figure 5.2). Firstly, we use YOLOv3 [Redmon and Farhadi, 2018] to detect the object location with RGB input. Secondly, we use 3DGC [Lin et al., 2020] autoencoder to perform 3D segmentation and observed points reconstruction for the latent feature learning. Then we propose the novel VDR representation to decode orientation information from the learned latent features. Thirdly, we use PointNet [Qi et al., 2017] to estimate the translation and object size. Finally, to increase the generalization ability of FS-Net and save storage space, we propose the box-cage-based 3D deformation for data augmentation.

5.3.1 Object Detection

Following our previous work[Chen et al., 2020], we train a YOLOv3 [Redmon and Farhadi, 2018] to fast detect the object bounding box in RGB images, and output class (category) labels. Then we adopt the 3D sphere to locate the point cloud of the target object quickly. The 2D detection part provides a compact 3D learning space for the following tasks. Different from other category-level 6D object pose estimation methods [Chen et al., 2020, Tian et al., 2020, Wang et al., 2019] that needed semantic segmentation masks, we only need object bounding boxes. Since object detection is faster and easier than semantic segmentation [He et al., 2017, Redmon and Farhadi, 2018], the detection speed of our method is faster than previous category-level pose estimation methods.

5.3.2 Shape-Based Network

The output points from object detection contain both object and background points. To access the points that belong to the target object and calculate the rotation of the object, we need a network that performs two tasks: 3D segmentation and rotation estimation.

Although many network architectures can directly process point cloud [Qi et al., 2017,, Zhou and Tuzel, 2018], most of the architectures calculate on point coordinates, which means their networks are sensitive to point clouds shift and size variation [Lin et al., 2020]. This may decrease the pose estimation accuracy.

To tackle the point clouds shift issue, Frustum-PointNet [Qi et al., 2018] and G2L-Net [Chen et al., 2020] employed the estimated translation to align the segmented point clouds to local coordinate space. However, their methods cannot handle the intra-class size variation.

To solve the point clouds shift and size variation problem, in this paper, we propose a 3DGC autoencoder to extract the point cloud shape features for segmentation and rotation estimation. 3DGC is designed for point cloud classification and part segmentation; our work shows that 3DGC can also be applied for category-level 6D pose estimation tasks.

3D Graph Convolution

3DGC kernel consists of m unit vectors. The m kernel vectors are applied to the n vectors generated by the center point with its n-nearest neighbors. Then, the convolution value is the sum of the cosine similarity between m kernel vectors and the n-nearest vectors. In a 2D convolution network, the trained network learned a weighted kernel that has a high response with a matched RGB value, while the 3DGC network learned the orientations of the m vectors in the kernel. The weighted 3DGC kernel has a high

response with a matched 3D pattern, which is defined by the center point with its n-nearest neighbors. For more details, please refer to [Lin et al., 2020].

Rotation-Aware Autoencoder

Based on the 3DGC, we design a 3DGC autoencoder for the estimation of category-level object rotation. To extract the latent rotation features, we train the autoencoder to reconstruct the observed points transformed from the observed depth region of the target object. There are several advantages to this strategy: 1) the reconstruction of observed points is view-based and symmetry invariant, 2) the reconstruction of observed points is easier than that of a complete object model (shown in Table 5.3), and 3) more representative latent features can be learned (shown in Table 5.2).

In [Sundermeyer et al., 2018, 2020], the authors reconstructed the input images to observed views as well. However, the input and output of their models are 2D images that are different from our 3D point cloud input and output. Furthermore, our network architecture is also different from theirs. Our work shows that this strategy can also be applied to point cloud reconstruction with a new architecture.

We utilize Chamfer Distance to train the autoencoder, the reconstruction loss function $\mathcal{L}_{rec}$ is defined as:

$$\mathcal{L}_{rec} = \sum_{\mathbf{p}_i \in \mathcal{P}} \min_{\hat{\mathbf{p}}_i \in \hat{\mathcal{P}}} \|\mathbf{p}_i - \hat{\mathbf{p}}_i\|_2 + \sum_{\hat{\mathbf{p}}_i \in \hat{\mathcal{P}}} \min_{\mathbf{p}_i \in \mathcal{P}} \|\mathbf{p}_i - \hat{\mathbf{p}}_i\|_2, \tag{5.1}$$

where $\mathcal{P}$ and $\hat{\mathcal{P}}$ denote the ground truth point set and reconstructed point set, respectively. $\mathbf{p}_i$ and $\hat{\mathbf{p}}_i$ are the points in $\mathcal{P}$ and $\hat{\mathcal{P}}$. With the help of 3D segmentation mask, we only use the features extracted from the observed object points for reconstruction.

After the network convergence, the encoder learned the rotation-aware latent features. Since the 3DGC is scale and shift-invariant, the observed points reconstruction enforces the autoencoder to learn the scale and shift-invariant orientation features

under corresponding rotation. In the next subsection, we describe how we effectively decode rotation information from the learned latent features with the proposed VDR representation.

5.3.3 Vector-Based Decoupled Rotation Representation

In this section, we first propose a new rotation representation, and then we analyse the properties of the proposed rotation representation.

As shown in Figure 5.3, we represent the rotation as two orthogonal vectors, called Vector-based Decoupled Rotation (VDR). The two vectors are parallel with two axes (up axis and front axis) of the object coordinate system. Then according to the definition of rotation, it is easy to show that these two vectors can completely represent the rotation information.

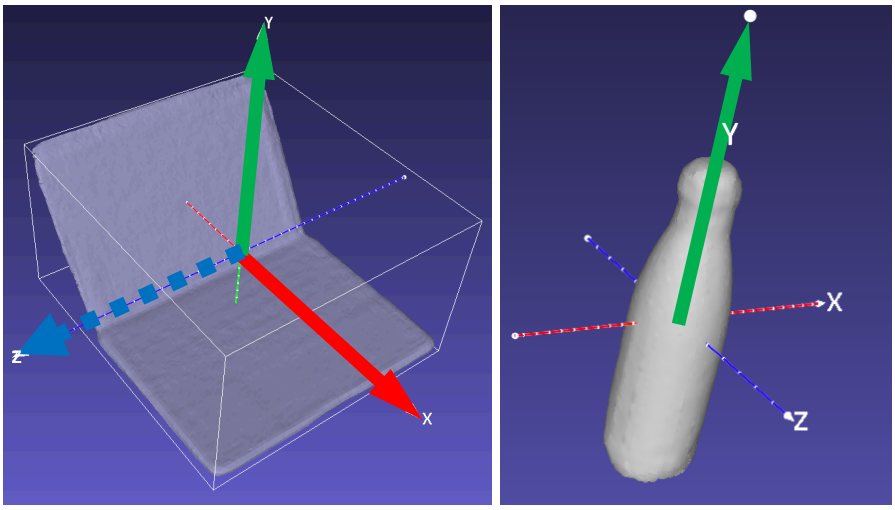


Figure 5.3: **Rotation represented by vectors.** Left: The object rotation can be represented by two perpendicular vectors (green vector and red vector). Right: For circular symmetry objects like the bottle, only the green(up) vector matters.

In the following, we provide the detailed analysis of the properties of the proposed VDR representation.

Decoupled Characteristic

Since the two vectors are orthogonal, the rotation information related to one of them is independent to the other, which means that we can use one of them to recover partial rotation information of the object. For example, in Figure 5.9, we use the green vector to recover the rotation. We can see that the green boxes and blue boxes are aligned well in the recovered vector orientation.

Given the latent features that contain rotation information, our task is to decode the category-level rotation features. To achieve this, we utilize two decoders to extract the rotation information in a decoupled fashion. The two decoders decode the rotation information into the corresponding greed vector and red vector.

Each decoder only needs to extract the orientation information along the corresponding vector, which is easier than the estimation of the complete rotation. Specifically, for each decoder, the output should be a vector with six variables: $[v_x^s, v_y^s, v_z^s, v_x^e, v_y^e, v_z^e]$, where $[v_x^s, v_y^s, v_z^s]$ is the start point of the predicted vector and $[v_x^e, v_y^e, v_z^e]$ is the endpoint of the predicted vector. Since the start point is always the origin, we only need to estimate the endpoint of the vector. Then in the inference stage, we combine the output from the two decoders with the origin to get a 3×3 matrix $\widetilde{\mathcal{V}}$. With matrix in the canonical frame and the estimated matrix $\widetilde{\mathcal{V}}$, we can calculate the final pose via the Kabsch algorithm. Since the Kabsch algorithm is based on singular value decomposition, we do not require the two predicted vectors are completely orthogonal.

Although other rotation representations, such as Euler angle, Quaternion, and Axis-angle, consist of several parts and each part has its particular meaning, the estimation of different parts separately cannot achieve similar performance as the estimation of the whole representation (shown in Table 5.7). One main reason is that the rotation information contained in the sub-part of these representations is entangled with other sub-parts.

One evident advantage of this characteristic is that we can easily handle the symmetry objects like the bottle, can, and bowl. Without loss of generality, we assume that the green vector is along the symmetry axis; then we can simply abandon the other vector while estimating the rotation to avoid rotation ambiguity.

Continuous Representation with Geometry Meaning

In [Zhou et al., 2019], Zhou et al. pointed out that due to the periodic property of rotation angle, the conventional rotation representations, such as Euler angle representation, Quaternion representation, and Axis-angle representation, have problems with continuity. To address this issue, Zhou et al. employed the first two columns as the new R_{6D} representation for rotation. Although their new R_{6D} representation can avoid the discontinuous problem, the geometry meaning of using such a representation is unclear.

In contrast, we cast the rotation information into 3D space with the help of two vectors. Then we can access a new continuous vector-based representation for the rotation that has a clear geometry meaning: one up vector and one front vector.

Point-Matching Variant

Point-Matching loss is a widely used loss for pose estimation that calculates the loss based on ADD-(S) metric [Hinterstoisser et al., 2012]. In [Wang et al., 2021], Wang et al. employed a new variant Point-Matching loss that accesses the loss only via rotation:

$$\mathcal{L}_{PM} = \frac{1}{|\mathcal{M}|} \sum_{\mathbf{p} \in \mathcal{M}} \|\mathbf{R}\mathbf{p} - \widetilde{\mathbf{R}}\mathbf{p}\|_2, \tag{5.2}$$

where $\widetilde{\mathbf{R}}$ and $\mathbf{R}$ are the prediction and ground truth. $|\mathcal{M}|$ is the number of points. $\mathbf{p} \in \mathcal{M}$ means the points belonging to the object 3D model $\mathcal{M}$.

However, there exists one issue here. As shown in Figure 5.4(a), when the rota-

tion changes θ, the distance d can be formalized as:

$$d = 2r \sin(\frac{\theta}{2}) \tag{5.3}$$

From the Eq. 5.3, we can see that the distance is determined by rotation angle θ and the radius r of the peripheral circle. When the point is closer to the center of rotation, the distance is shorter, which means the points away from the center are more important to the loss value with given θ. However, the conventional Point-Matching loss does not consider this imbalance problem.

The Point-Matching loss transformers the object 3D model with different rotation matrices and calculates the distance between the two transformed models. It transfers the rotation distance to the object point distance. Following this idea, we propose to use our VDR representation for Point-Matching loss.

Since our new representation is vector-based, we cannot directly calculate the loss by multiplying the VDR with the 3D model points of the object. Therefore, we use the add operation to replace the multiplication in Eq. 5.3. Then we can also transfer the vector distance to the object point distance.

As shown in Figure 5.4(b), we move the 3D object points to the directions pointed by the rotation vectors. Then the new Point-Matching loss can be defined as:

$$\begin{aligned} \mathcal{L}_R = &\operatorname{avg}_{\mathbf{p}\in\mathcal{M}} \|\mathbf{p} + \mathbf{v}_g - (\mathbf{p} + \hat{\mathbf{v}}_g)\|_2 \\ &+ \operatorname{avg}_{\mathbf{p}\in\mathcal{M}} \|\mathbf{p} + \mathbf{v}_r - (\mathbf{p} + \hat{\mathbf{v}}_r)\|_2 \end{aligned} \tag{5.4}$$

where $\hat{\mathbf{v}}_g$ and $\hat{\mathbf{v}}_r$ are the predicted vectors for the green and red vectors shown in Figure 5.3. $\mathbf{v}_g$ and $\mathbf{v}_r$ are the ground truth.

It is obvious that the new Point-Matching loss $\mathcal{L}_\mathbf{R}$ is equal to mean square error

(MSE) loss between the prediction and ground truth VDR:

$$\begin{aligned}
\mathcal{L}_R &= \text{avg}_{\mathbf{p}\in\mathcal{M}} \|\mathbf{p} + \mathbf{v}_g - (\mathbf{p} + \hat{\mathbf{v}}_g)\|_2 \\
&+ \text{avg}_{\mathbf{p}\in\mathcal{M}} \|\mathbf{p} + \mathbf{v}_r - (\mathbf{p} + \hat{\mathbf{v}}_r)\|_2 \\
&= \text{avg}_{\mathbf{p}\in\mathcal{M}} \|\mathbf{p} + \mathbf{v}_g - \mathbf{p} - \hat{\mathbf{v}}_g\|_2 \\
&+ \text{avg}_{\mathbf{p}\in\mathcal{M}} \|\mathbf{p} + \mathbf{v}_r - \mathbf{p} - \hat{\mathbf{v}}_r\|_2 \\
&= \text{avg}_{\mathbf{p}\in\mathcal{M}} \|\mathbf{v}_g - \hat{\mathbf{v}}_g\|_2 \\
&+ \text{avg}_{\mathbf{p}\in\mathcal{M}} \|\mathbf{v}_r - \hat{\mathbf{v}}_r\|_2 \\
&= \|\mathbf{v}_g - \hat{\mathbf{v}}_g\|_2 + \|\mathbf{v}_r - \hat{\mathbf{v}}_r\|_2
\end{aligned} \tag{5.5}$$

That means with MSE loss, the proposed VDR representation can naturally avoid the imbalance issue existing in Eq. 5.3.

5.3.4 Residual Prediction Network

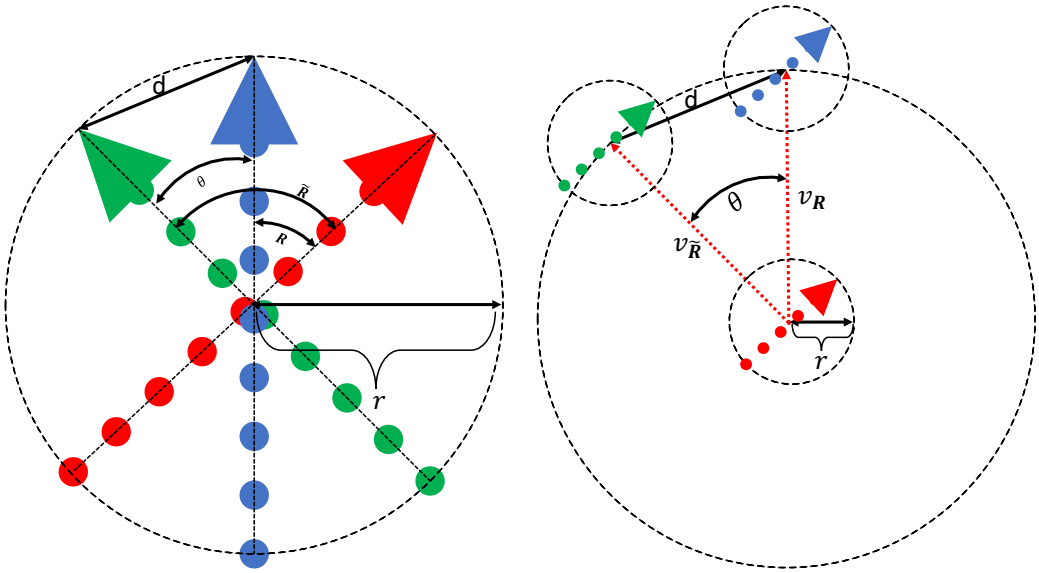
As both translation and object size are related to point coordinates, inspired by [Chen et al., 2020, Qi et al., 2018], we train a tiny PointNet [Qi et al., 2017] that takes segmented point cloud as input. More concretely, the PointNet performs two tasks: 1) estimating the residual between the translation ground truth and the mean value of the segmented point cloud; 2) estimating the residual between object size and the mean category size.

For size residual, we pre-calculate the mean size $[\overline{S}_x, \overline{S}_y, \overline{S}_z]^T$ of each category by

$$\begin{bmatrix} \overline{S}_x \\ \overline{S}_y \\ \overline{S}_z \end{bmatrix} = \frac{1}{N} \sum_{i=1}^{N} [S_x^i, S_y^i, S_z^i]^T, \tag{5.6}$$

where N is the amount of the object in that category. Then for object o in that category the ground truth $[\delta_x^o, \delta_y^o, \delta_z^o]^T$ of the size residual estimation is calculated as:

$$[\delta_x^o, \delta_y^o, \delta_z^o]^T = [S_x^o, S_y^o, S_z^o]^T - [\overline{S}_x, \overline{S}_y, \overline{S}_z]^T. \tag{5.7}$$



(a) Point Matching [Wang et al., 2021] (b) Point Matching with VDR

Figure 5.4: **Point-Matching variant in 2D case.** Red point cloud arrow is the original point cloud. Blue is the one transformed by the ground truth value. Green is transformed by the estimated value. The left image describes the Point-Matching variant loss used in [Wang et al., 2021] that only considers the rotation term. r is the radius of the peripheral circle. d is the distance between the point in the blue point cloud transformed by ground truth rotation $\mathbf{R}$ and the point in the green point cloud transformed by estimated rotation $\widetilde{\mathbf{R}}$. θ is the rotation residual between estimated rotation and the ground truth. The right image describes the Point-Matching variant based on our proposed VDR representation where the point clouds are transformed by the orientation of the vectors.

We use MSE loss to predict both the translation and size residual. The total loss function $\mathcal{L}_{res}$ is defined as: $\mathcal{L}_{res} = \mathcal{L}_{tra} + \mathcal{L}_{size}$, where $\mathcal{L}_{tra}$ and $\mathcal{L}_{size}$ are sub-loss for translation residual and size residual, respectively.

5.3.5 Data Augmentation: 3D Deformation Mechanism

One major issue in category-level 6D pose estimation is the intra-class shape variation. The existing methods employed two large synthetic datasets, i.e., CAMERA275(NOCS-

Synthetic) [Wang et al., 2019] and 3D model dataset [Chang et al., 2015] to learn this variation. However, this strategy not only needs extra hardware resources but also increases the (pre-)training time.

To alleviate the shape variation issue, based on the fact that the shapes of most objects in the same category are similar [Vlach, 2016] (shown in Figure 5.1), we propose an online 3D deformation mechanism for training data augmentation. We pre-define a box-cage for each rigid object (shown in Figure 5.6). Each point is assigned to its nearest surface of the cage; when we deform the surface, the corresponding points move as well.

Though box cage can be designed more refined, in experiments, we find that with a simple box cage, i.e. 3D bounding box of the object, the generalization ability of FS-Net is considerably improved (Table 5.2). Different to [Yifan et al., 2020], we do not need an extra training process to obtain the box-cage of the object, and we do not need target shape to learn the deformation operation either. Our mechanism can be applied during training on the fly, which can save training time and storage space.

To make the deformation operation easier, we first transfer the points to the canonical coordinate system and then perform 3D deformation. Finally, we transform them to the global scene:

$$\{\mathcal{P}_1, \mathcal{P}_2, \cdots, \mathcal{P}_n\} = \mathbf{R}(\mathbb{D}_{3D}(\mathbf{R}^{\mathbf{T}}(\mathcal{P} - \mathbf{T}))) + \mathbf{T}, \tag{5.8}$$

where $\mathcal{P}$ is the point set generated after the 2D detection step. $\mathbf{R}$, $\mathbf{T}$ are the pose ground truth. $\{\mathcal{P}_1, \mathcal{P}_2, \cdots, \mathcal{P}_n\}$ are the new generated training examples. $\mathbb{D}_{3D}$ is 3D deformation that includes cage enlarging, shrinking, changing the area of some surfaces.

In the canonical coordinate system, every box edge is parallel to an axis (shown in Figure 5.5). This property makes the 3D deformation operation easier. For example, when we need to elongate/shrink the mug along Y axis by n times. We enlarge the distance between surface $SF_{1,2,3,4}$ and surface $SF_{5,6,7,8}$ by n times. Since these two surfaces are parallel to the XZ-plane, the x and z coordinates are unchanged. Then

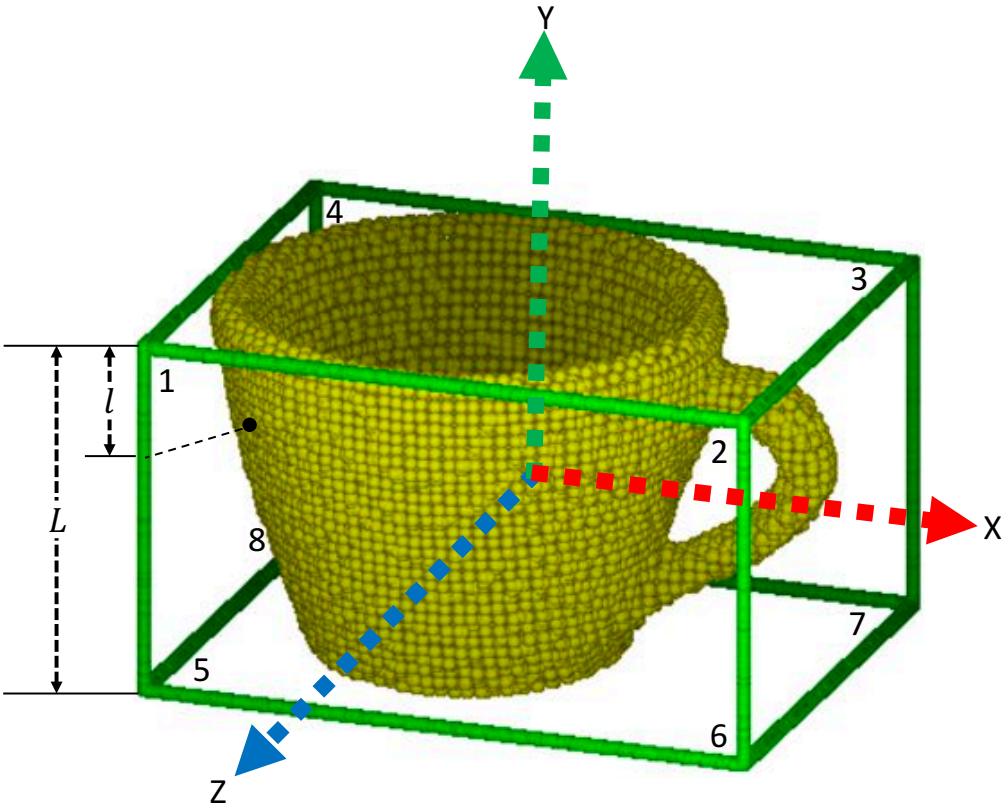


Figure 5.5: **3D object model**. We assume that the centre of the 3D bounding box is the origin point of the coordinate. The surface is represented as its four corners. For example, the top surface is represented as $SF_{1,2,3,4}$.

points coordinates are changed from $[\mathbf{x}, \mathbf{y}, \mathbf{z}]$ to $[\mathbf{x}, n\mathbf{y}, \mathbf{z}]$. Then for different axes we can use the following equations to access the points coordinates after operation:

$$[\mathbf{x}, n\mathbf{y}, \mathbf{z}] = \mathbb{D}_x([\mathbf{x}, \mathbf{y}, \mathbf{z}]), \tag{5.9}$$

$$[n\mathbf{x}, \mathbf{y}, \mathbf{z}] = \mathbb{D}_y([\mathbf{x}, \mathbf{y}, \mathbf{z}]), \tag{5.10}$$

$$[\mathbf{x}, \mathbf{y}, n\mathbf{z}] = \mathbb{D}_z([\mathbf{x}, \mathbf{y}, \mathbf{z}]), \tag{5.11}$$

where $\mathbb{D}_{x,y,z}$ is the elongate/shrink operation along corresponding axis.

Further, if the object is the mug or bowl, we may need to change the top or bottom size to generate new shapes (shown in Figure 5.6). In this case, assuming that we enlarge the bottom along X axis by n times, then from bottom to top, the coordinates are changed as:

$$\mathbf{x}_{new} = (1 + (n-1)\frac{l}{L})\mathbf{x}, \tag{5.12}$$

Table 5.1: **Shape similarity measurement**. We use Chamfer Distance($\times 10^{-4}$) to measure the shape similarity between the training set and testing set under circumstances w/o 3D deformation and with 3D deformation augmentation.

Category	w/o 3D deformation	3D deformation
Bottle	3.57	0.92
Bowl	0.78	0.64
Can	1.33	0.51
Camera	4.16	2.61
Laptop	2.67	1.95
Mug	0.67	0.53
Average	2.20	1.19

where l is the distance from a point to the top surface, i.e. $SF_{1,2,3,4}$ in Figure 5.5. L is the height of the object. Please note, all the edges are kept straight during deformation.

To show that the proposed 3D deformation mechanism can mitigate the shape variation issue in the category-level pose estimation task, in Table 5.1, we report the minimal Chamfer Distance (CD) error between objects model in the training set and testing set. In the instance-level 6D object pose estimation task, since the same object is shared in the training and testing set, the CD error is zero. However, in the category-level 6D object pose estimation task, due to the shape variation, there exists a gap between the object model in the training set and the object model in the testing set. We use CD error to measure this gap. When using the proposed 3D deformation data augmentation technique, the average CD error is reduced by 45.9%, from 2.20 ($\times 10^{-4}$) to 1.19 ($\times 10^{-4}$). This means with proposed data augmentation, the similarity between the training and testing set is significantly improved.

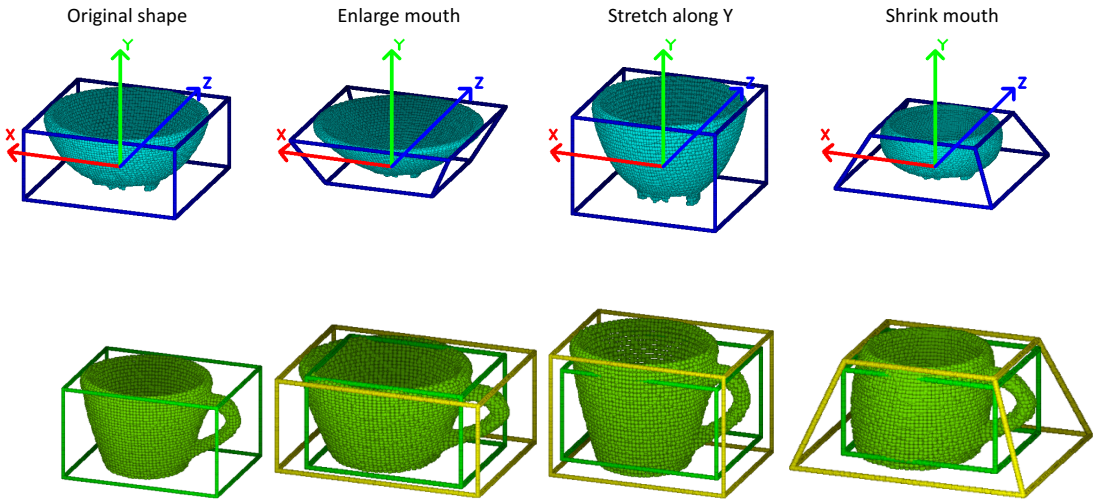


Figure 5.6: **3D deformed examples**. The new training examples can be generated by enlarging, shrinking, or changing the area of some surfaces of the box cages. The top row is the augmented examples of the 'bowl' with corresponding deformed box cages, the bottom is for 'cup'. In the bottom row, we also show the original 3D bounding boxes (green color) before deformation.

5.4 Experiments

5.4.1 Datasets

We use the NOCS-REAL and the LINEMOD datasets to the proposed method. Please refer to Chapter 2 for the detailed description of these datasets. However, the original 6D pose estimation datasets do not provide the point-wise label. To train the proposed network in a supervised way, as described in Chapter 2, Section 2.5.4, we use an automatic method to generate the label for each point of the point cloud.

5.4.2 Training Details

We use Pytorch [Paszke et al., 2017] to implement our pipeline. All experiments are deployed on a PC with i7-4930K 3.4GHz CPU and GTX 1080Ti GPU.

First, to locate the object in RGB images, we fine-tune the YOLOv3 pre-trained on COCO dataset [Lin et al., 2014] with the training dataset. Then we jointly train the 3DGC autoencoder and residual estimation network. For 3D deformation augmentation, we randomly choose the n in the range $[0.8, 1.2]$ for Eq. 5.9-5.12 according to the uniform distribution. The total loss function is defined as:

$$\mathcal{L}_{Shape} = \lambda_{seg}\mathcal{L}_{seg} + \lambda_{rec}\mathcal{L}_{rec} + \lambda_R\mathcal{L}_R + \lambda_{res}\mathcal{L}_{res}, \tag{5.13}$$

where λs are the balance parameters that are set to keep different loss values at the same magnitude. We use cross entropy for 3D segmentation loss function $\mathcal{L}_{seg}$.

We adopt Adam [Kingma and Ba, 2014] to optimize the FS-Net. The initial learning rate is 0.001, and it is halved every 10 epochs. The maximum epoch is 50.

5.4.3 Evaluation Metrics

For the category-level pose estimation, we use the IoU_x and $n^\circ\ m$ **cm** (see Section 2.6 in Chapter 2 for details) to measure different methods.

For instance-level pose estimation, we compare the performance of FS-Net with other state-of-the-art instance-level methods using the ADD-(S) metric [Hinterstoisser et al., 2012].

5.4.4 Ablation Studies

We use the G2L-Net [Chen et al., 2020] as the baseline method which extracted the latent features for rotation estimation via point-wise orientated vector regression, and the ground truth of rotation is the eight corners of the 3D bounding box with the corresponding rotation. The loss function for rotation estimation is the MSE loss between predicted 3D coordinates and ground truth. Compared to baseline, our proposed work has three

Table 5.2: **Ablation studies on the NOCS-REAL dataset**. We use two different metrics to measure performance. '3DGC' means the 3D graph convolution. 'OPR' means observed points reconstruction. 'VDR' represents the decoupled rotation mechanism. 'DEF' denotes the online 3D deformation. In the last row, the values in the bracket are the performance for the reconstruction of the complete object model transformed by the corresponding rotation. Please note, for the sake of the ablation studies, we provide the ground truth 2D bounding box for different methods.

Method	3DGC	DEF	OPR	VDR	IoU_{50}	10°10 **cm**
G2L [Chen et al., 2020]	×	✓	×	×	94.65%	31.0%
G2L+VDR	×	✓	×	✓	96.21%	47.81%
Med1	✓	✓	×	×	97.98%	46.4%
Med2	✓	✓	✓	×	95.61%	46.8%
Med3	✓	✓	×	✓	97.34%	61.1%
Med4	✓	×	✓	✓	97.30%	58.2%
Med5	✓	✓	✓	✓	98.04% (94.44%)	65.9% (58.0%)

novelties: a) view-based 3DGC autoencoder for observed point cloud reconstruction; b) VDR representation; c) 3D augmentation mechanism.

In Table 5.2, we report the experimental results of three novelties on the NOCS-REAL dataset. Comparing Med3 and Med5, we find that reconstruction of the observed point cloud can learn better pose features. The performance of Med2 (Med1, G2L) and Med5 (Med3, G2L+VDR) shows that the proposed VDR representation can effectively extract the rotation information. The results of Med4 and Med5 demonstrate the effectiveness of the 3D deformation mechanism, which increases the pose accuracy by 7.7% in terms of 10°10 **cm** metric. We also compare the different reconstruction choices: the reconstruction of observed points and the complete object model with corresponding rotation. From the last row of Table 5.2, we can see that the observed points reconstruction can learn better rotation features. Overall, Table 5.2 shows that the proposed novelties can improve the accuracy significantly.

5.4.5 Generalization Performance

NOCS-REAL dataset provides 4.3k real images that cover various poses of different objects in different categories for training. That means the category-level pose information is rich in the training set. Thanks to the effective pose feature extraction, FS-Net achieves state-of-the-art performance even with part of the real-world training data. We randomly choose different percentages of the training set to train FS-Net and test it on the whole testing set. Figure 5.7 shows that: 1) FS-Net is robust to the size of the training dataset and has good category-level feature extraction ability. Even with 20% of the training dataset, the FS-Net can still achieve state-of-the-art performance; 2) the 3D deformation mechanism significantly improves the robustness and performance of FS-Net.

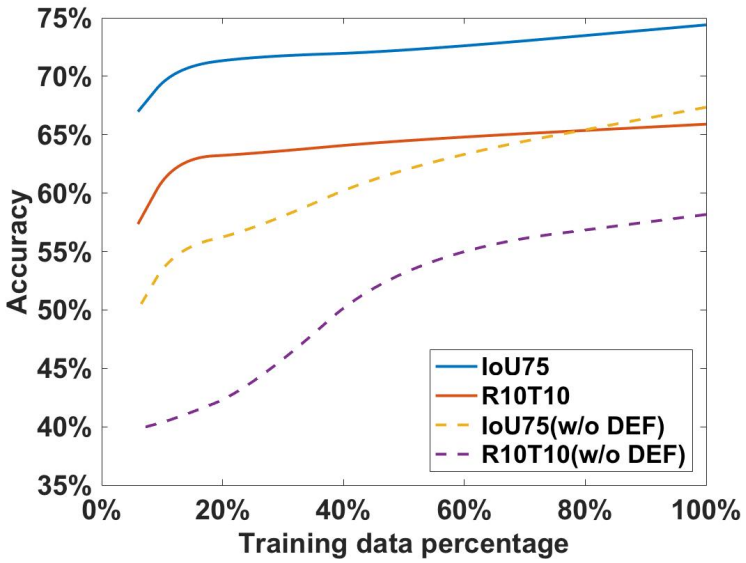


Figure 5.7: **Generalization performance.** With the given 2D bounding box and a randomly chosen 3D sphere center, we show how the training set size affects the pose estimation performance. 'w/o DEF' means no 3D deformation mechanism is adopted during training.

5.4.6 Evaluation of Reconstruction

Point cloud reconstruction has a close relationship with pose estimation performance. We computed the Chamfer Distance of the reconstructed point cloud with the ground truth point cloud and compared it with other reconstruction types used by other methods. From Table 5.3, we can see that the average reconstruction error of our method is 0.86, which is 72.9% and 18.9% lower than that of Shape-Prior [Tian et al., 2020] and CASS [Chen et al., 2020], respectively. It shows that our method achieves better pose estimation results via a simpler reconstruction task. i.e. observed points reconstruction rather than complete object model reconstruction.

5.4.7 Comparison with State-of-the-Arts

Category-Level Pose Estimation

We compare FS-Net with NOCS [Wang et al., 2019], CASS [Chen et al., 2020], Shape-Prior [Tian et al., 2020], and 6-PACK [Wang et al., 2020] on NOCS-REAL dataset in Table 5.4. We can see that our proposed method outperforms the other state-of-the-art methods on both accuracy and speed. Specifically, with 3D detection metric IoU_{50}, our FS-Net outperforms the previous best method, NOCS, by 11.7% and the running speed is 4 times faster. In terms of 6D pose metric 5°5 **cm** and 10°10 **cm**, FS-Net outperforms the CASS by the margins of 4.7% and 6.3%, respectively. FS-Net even outperforms 6-PACK under 3D detection metric IoU_{50}, which is a 6D tracker and needs an initial 6D pose and object size to start. See Figure 5.8 for more quantitative details.

The qualitative results are shown in Figure 5.9. Please note, we only use real-world data (NOCS-REAL) to train our pose estimation part. Other methods use both synthetic dataset (CAMERA) [Wang et al., 2019] and real-world data for training. The number of training examples in the CAMERA is 275K, which is more than 60 times than

Table 5.3: **Reconstruction type comparison.** The comparison is on the NOCS-REAL dataset with the Chamfer Distance metric ($\times 10^{-3}$). 'Complete' means the reconstruction of the complete 3D model. 'Observed' denotes the reconstruction of the observed points.

Methods	CASS [Chen et al., 2020]	Shape-Prior [Tian et al., 2020]	Ours
	Complete	Complete	Observed
Bottle	**0.75**	3.44	1.2
Bowl	**0.38**	1.21	0.39
Camera	0.77	8.89	**0.44**
Can	**0.42**	1.56	0.62
Laptop	3.73	2.91	**2.23**
Mug	0.32	1.02	**0.29**
Average	1.06	3.17	**0.86**

that of NOCS-REAL (4.3K). It shows that FS-Net can efficiently extract the category-level pose features with fewer data.

Table 5.4: **Category-level performance on the NOCS-REAL dataset with different metrics**. We summarize the pose estimation results reported in the origin papers on the NOCS-REAL dataset. '-' means no results are reported under this metric.

Method	IoU_{25}	IoU_{50}	IoU_{75}	5°5**cm**	10°5 **cm**	10°10 **cm**	Speed(FPS)
NOCS [Wang et al., 2019]	84.9%	80.5%	30.1%	9.5 %	26.7%	26.7%	5
CASS [Chen et al., 2020]	84.2%	77.7%	-	23.5 %	58.0%	58.3%	-
Shape-Prior [Tian et al., 2020]	83.4%	77.3%	53.2%	21.4%	54.1%	-	4
6-PACK [Wang et al., 2020]	94.2%	-	-	**33.3 %**	-	-	10
Ours(w/o 2D aug)	81.3%	77.6%	53.1%	26.5 %	47.4%	51.2%	**20**
Ours	**95.1%**	**92.2%**	**63.5%**	28.2 %	**60.8%**	**64.6%**	**20**

Table 5.5: **Instance-level comparison on the LINEMOD dataset**. Our method achieves a comparable performance with the state-of-the-arts in both speed and accuracy.

Method	Input	ADD-(S)	Speed(FPS)
PVNet [Peng et al., 2018]	RGB	86.3%	25
CDPN [Li et al., 2019]	RGB	89.9%	33
DPOD [Zakharov et al., 2019]	RGB	95.2%	33
G2L-Net [Chen et al., 2020]	RGBD	98.7%	23
Densefusion[Wang et al., 2019]	RGBD	94.3%	16
PVN3D [He et al., 2020]	RGBD	99.4%	5
Ours	RGBD	97.6%	20

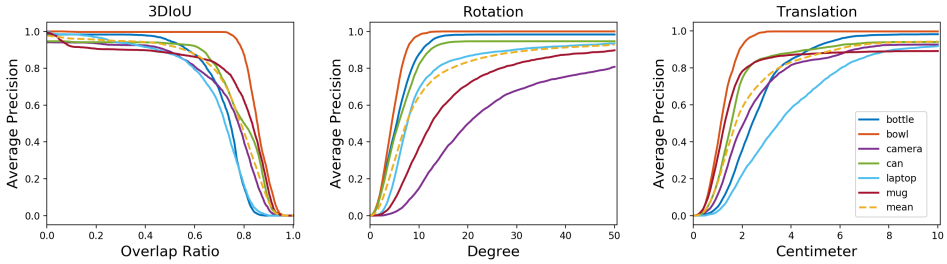


Figure 5.8: **Result on NOCS-REAL**. The average precision of different thresholds tested on NOCS-REAL dataset with 3D IoU, rotation, and translation error.

Instance-Level Pose Estimation

We compare the instance-level pose estimation results of FS-Net on the LINEMOD dataset with other state-of-the-arts instance-level methods. From Table 5.5, we can see that FS-Net achieves comparable results on both accuracy and speed. It shows that our method can effectively extract both category-level and instance-level pose features.

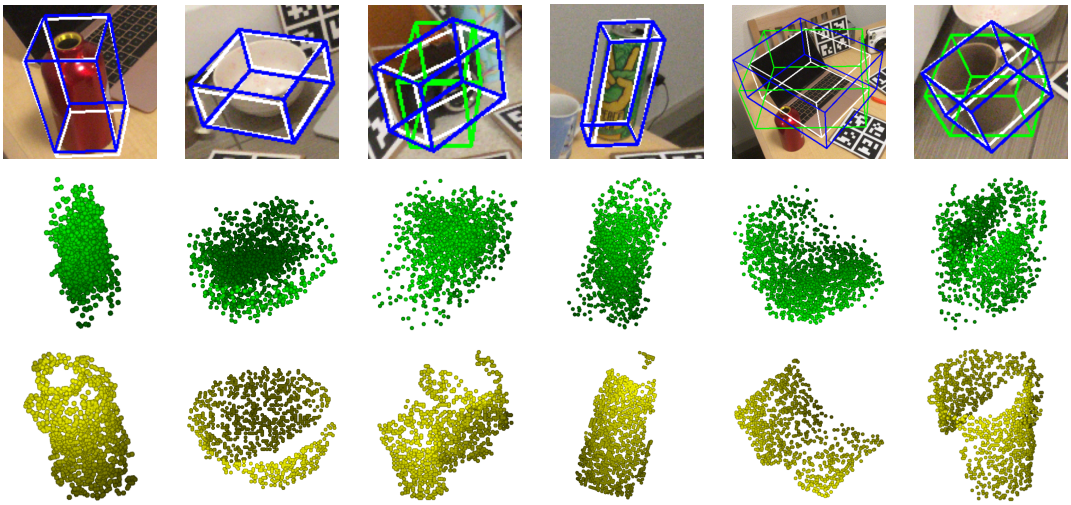
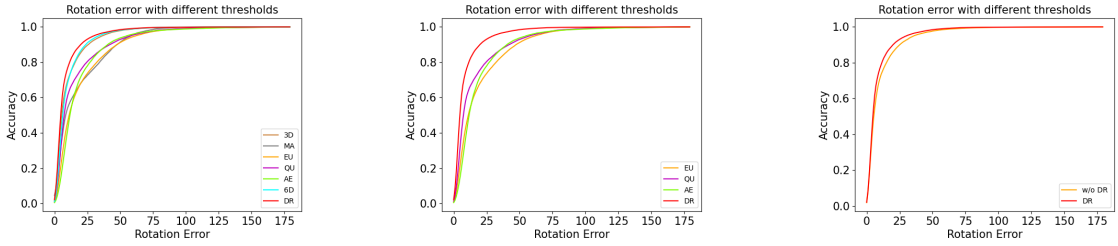


Figure 5.9: **Qualitative results on NOCS-REAL dataset**. The first row is the pose and size estimation results. White 3D bounding boxes denote ground truth. Blue boxes are the poses recovered by two estimated rotation vectors. The green boxes are the poses recovered from the estimated green vector. Our results match the ground truth well in both pose and size. The second row is the reconstructed observed points under corresponding rotation, although the reconstructed points are not perfectly in line with the target points, the basic orientation information is kept. The third row is the ground truth of the observed points extracted from the depth map.

5.4.8 Rotation Representation Comparison

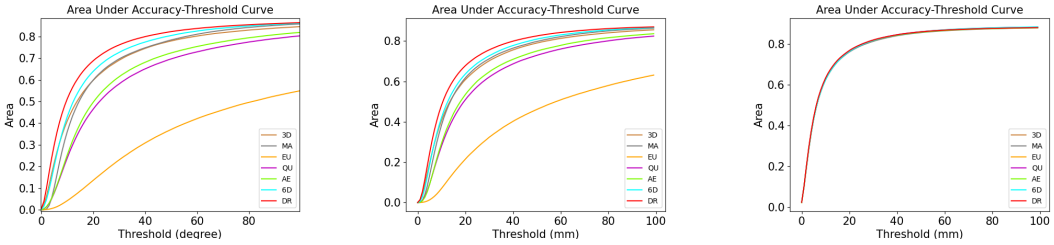
In this section, we compare the VDR representation with other rotation representations on 6D object pose estimation tasks. For the sake of fair comparison and following the experimental protocol in [Zhou et al., 2019], we adopt MSE loss for all rotation representations.

For category-level pose estimation, we use the proposed FS-Net pipeline as the baseline method. Same as [Zhou et al., 2019] we use the geodesic errors to measure different rotation representations. We report the results on Table 5.6 and Figure 5.10. For instance-level pose estimation, we use our previous work G2L-Net [Chen et al.,



(a)Without Decoupled Training (b)Decoupled Training (c)Decoupled and w/o Decoupled

Figure 5.10: **Comparison on category level**. The curve of accuracy under different rotation error thresholds with different training fashions on FS-Net. '3D' means using eight corners of the 3D bounding box as the rotation representation. 'MA' denotes using rotation matrix. 'EU' means Euler angle. 'QU' means Quaternion. 'AE' means Axis-angle. '6D' means R_{6D} proposed in [Zhou et al., 2019]. 'DR' means the proposed VDR representation.



(a) AUC of Rotation (b)AUC of ADD(S) (c) AUC of Translation Error

Figure 5.11: **Comparison on instance level.** We report the AUC of rotation error, ADD(S) error, and translation estimation error for different representations based on G2L-Net. '3D' means using eight corners of the 3D bounding box as the rotation representation. 'MA' denotes using rotation matrix. 'EU' means Euler angle. 'QU' means Quaternion. 'AE' means Axis-angle. '6D' means R_{6D} proposed in [Zhou et al., 2019]. 'DR' means the proposed VDR representation.

Table 5.6: **Rotation error comparison.** For different representations on category-level 6D object pose estimation, we provide the geodesic distance error of the rotation prediction on the test set. The 'RM' means rotation mapping that proposed in [Pitteri et al., 2019]. 'w/o decoupled' means training rotation as a whole term. 'decoupled' means training the rotation sub-terms with different network branches. '3D' means using the eight 3D corners to represent the rotation as in G2L[Chen et al., 2020]. 'R_{6D}' is the rotation representation proposed in [Zhou et al., 2019].

	Category	Bottle	Bowl	can	camera	laptop	mug	Average
w/o decoupled	FS-Net+3D+RM	6.14	5.08	**4.38**	25.55	**7.06**	19.42	11.27
	FS-Net+Matrix+RM	8.94	**4.84**	5.33	41.87	17.70	32.92	18.60
	FS-Net+Euler+RM	13.06	6.18	7.23	44.50	13.92	37.39	20.38
	FS-Net+Quaternion+RM	7.24	5.07	5.81	41.94	13.08	27.47	16.77
	FS-Net+Axis-angle+RM	10.30	6.96	10.33	41.00	16.07	31.41	19.35
	FS-Net+R_{6D}+RM	8.51	5.12	4.98	**23.35**	9.27	**17.13**	11.40
	FS-Net+VDR+RM	**5.88**	4.92	4.40	24.72	7.96	19.12	**11.17**
decoupled	FS-Net+Axis-angle+RM	17.27	12.92	16.18	39.14	18.08	25.54	21.52
	FS-Net+Euler+RM	17.46	16.91	19.05	52.58	40.30	55.81	33.68
	FS-Net+Quaternion+RM	15.25	12.78	15.96	41.52	36.65	41.56	27.29
	FS-Net+VDR [Chen et al., 2021]	**5.71**	**3.85**	**3.97**	**21.59**	**8.00**	**13.77**	**9.48**

2020] and the state-of-art method GDR-Net [1] [Wang et al., 2021] as baselines to test different rotation representations. We summarize the results in Table 5.7 and Figure 5.11.

In Table 5.6, in the 'decoupled' part, we decouple the Axis-angle representation as two sub-terms: vector part and the length of the vector; we decouple Euler representation as three angles, and we decouple Quaternion as four sub-terms which are four variables in the Quaternion representation.

From Table 5.6, we can see that our new proposed VDR can achieve the best performance for rotation estimation. Compared with the R_{6D} rotation representation

[1] We use the code provide by GDR-Net to test different rotation representations: https://github.com/THU-DA-6D-Pose-Group/GDR-Net

Table 5.7: **Rotation error for different representations on instance-level 6D pose estimation.** The best one is bolded. 'PM' means Point-Matching loss.

Method	Ape	Bench Vise	Camera	Can	Cat	Driller	Duck	Egg Box	Glue	Hole Puncher	Iron	Lamp	Phone	Average
G2L [Chen et al., 2020]	**5.07**	15.34	5.99	8.96	**6.03**	13.04	6.84	25.13	11.31	5.13	17.20	12.54	7.24	10.75
G2L+Matrix	7.86	11.73	8.28	7.76	9.14	9.83	8.58	**17.02**	13.36	6.69	8.26	13.49	9.87	10.14
G2L+Euler	42.79	54.67	42.33	53.59	43.07	46.06	39.41	52.32	49.91	51.66	45.03	53.87	48.20	47.92
G2L+Quaternion	13.29	17.36	12.35	18.70	12.42	14.54	15.28	22.52	17.32	13.52	16.91	20.70	16.26	16.24
G2L+Axis-angle	14.60	19.12	9.37	13.70	10.70	17.74	10.91	20.46	18.44	10.13	13.65	18.18	13.91	14.69
G2L+R_{6D}[Zhou et al., 2019]	7.01	12.28	8.16	8.97	6.69	**7.70**	8.41	18.77	**10.79**	6.73	8.42	11.74	6.59	9.40
G2L+VDR	5.75	**11.69**	**4.96**	**7.44**	7.33	10.85	**5.35**	18.85	11.17	**4.30**	**6.38**	**11.63**	**5.86**	**8.58**
GDR-Net[Wang et al., 2021](+R_{6D}(PM))	2.11	1.85	1.81	1.82	2.02	2.02	1.98	1.72	2.37	1.96	2.33	1.87	2.30	2.01
GDR-Net+R_{6D}	2.73	2.71	2.80	2.71	2.71	2.88	2.77	2.49	3.18	2.98	3.04	2.74	3.43	2.86
GDR-Net+VDR	**1.83**	**1.82**	**1.67**	**1.73**	**1.80**	**1.83**	**1.83**	**1.58**	**1.95**	**1.88**	**2.04**	**1.68**	**2.19**	**1.83**

proposed by Zhou et al. [Zhou et al., 2019], in instance-level pose estimation, our rotation prediction error is 16.84% smaller than that of R_{6D} representation. When comparing VDR representation in different training fashions, the decomposable property of VDR can decrease the rotation error by 15.13%, from 11.17 to 9.48.

From Table 5.7, we find that the proposed VDR representation achieves the best performance for both G2L-Net and GDR-Net. Compared with the R_{6D} representation, in G2L-Net, VDR decreases the rotation error from 9.4 to 8.58, a decrease of 8.72%. In GDR-Net, we retrain the network with R_{6D}(PM), R_{6D} and VDR representation. All other parameters are fixed except the rotation loss part. Compared with R_{6D}(PM) and R_{6D}, our proposed VDR decreases the rotation error by 8.9% and 36%.

To see how the rotation representation can affect the translation estimation, based on G2L-Net, we show the translation estimation error for different representations in Figure 5.10.c. From Figure 5.10.c, we can see that the area under threshold curves of translation for different rotation representations are almost overlapped with each other, which means the rotation representation can merely affect the accuracy of rotation estimation.

5.4.9 Running Time

Given a 640× 480 RGB-D image, our FS-Net runs at 20 FPS on Intel i7-4930K CPU and 1080Ti GPU. Specifically, the 2D detection takes about 10ms to proceed. The pose and size estimation takes about 40ms.

5.5 Conclusion

In this chapter, we propose a fast category-level pose estimation method that runs at 20 FPS, which has the potential for real-time applications. The proposed method first extracts the latent features by the observed points reconstruction with a shape-based 3DGC autoencoder. Then the category-level orientation features are decoded by the two decoders based on VDR representation. Finally, for translation and object size estimation, we use the residual network to estimate them based on residual estimation. In addition, to increase the generalization ability of FS-Net and save the hardware source, we design an online 3D deformation mechanism for training set augmentation. Extensive experimental results demonstrate that FS-Net is less data-dependent and can achieve state-of-the-art performance on category- and instance-level pose estimation in terms of both accuracy and speed. The experimental results also show that the proposed VDR representation is more suitable for the 6D object pose estimation tasks than other widely used rotation representations. Please note, our 3D augmentation mechanism and VDR representation are model-free, which can be easily plugged into other pose estimation methods to boost the performance.

Chapter Six

Conclusion and Future Work

In this thesis, we report our works about 6D object pose estimation at the instance and category level. In this chapter, we summarize the thesis and discuss the potential research direction for future work.

6.1 Thesis Conclusion

In this thesis, we mainly have three findings. First, apart from RGB, the point cloud can also be applied for 6DoF pose estimation in both instance level and category level.

In the first chapter, we give the overview of the whole thesis including thesis motivation, the formulation and challenges of the research problems, and the proposed methods with their contributions. To better understand the context of the proposed methods and the research problems, we summarize the related works, evaluation metrics, and relevant datasets in Chapter two. Via addressing the problems in 6D object pose estimation, this thesis produces three works that distribute in Chapters three, four, and five. These three works have shown that:1) point cloud representation can also be used for 6D pose estimation;2)feature design is essential for 6D pose estimation tasks;3)different rotation representations have a big influence on the 6D pose estimation results.

Specifically, these three works use RGB information for object bounding box detection and estimate 6D pose based on the point cloud representation. Then Chapter 3 describes how we extract point-wise features from the point cloud for the occlusion scenario. In this chapter, we demonstrated that the features extracted from 3D space is more suitable for 6D object pose estimation. However, the running speed of the proposed method is slow due to inefficient feature extraction. In Chapter 4, we introduced a novel embedding vector features for real-time instance-level 6D object pose estimation. When comparing the 3D rotation estimation and 3D translation estimation, we found that 3D rotation prediction is the hardest part of the 6D object pose estimation task which is regarded as the viewpoint variation challenge in this field. Therefore, the embedding vector features are proposed for better viewpoint information capture. With the novel features, given an RGBD image, the proposed pipeline estimates the 6D pose of the target object in real-time without post-refinement. While the method proposed in Chapter 4 is fast and accurate, it needs a large number of labelled data to train, when the annotation of the particular object is unavailable, the performance will drop. To address this, we extend the method to category-level pose estimation, then Chapter 5 is about how we address the intra-class variation challenges in category-level tasks with the proposed 3D deformation mechanism and vector-based decoupled rotation representation. In addition, we firstly use the 3D graph convolution kernel for category-level 6D object pose estimation. With these novelties, the proposed framework estimates the size and the 6D pose of the unseen object at a fast speed (20fps).

6.2 Future Work

Although the methods described in Chapters 3,4, and 5 can achieve state-of-the-art performance, they still have several limitations. First, the pose estimation parts of the methods are based on the 2D detection results from the 2D detector. Once the detection failed, the pose estimation part could not work anymore. Second, the methods need

a large number of labelled data to train. However, annotation 6D pose for the target objects is harder than other computer vision tasks such as object detection and segmentation. Third, we only test the proposed methods on benchmark datasets, how the proposed methods perform on real-world applications is still unknown.

To address these limitations, our future work can focus on three points. First, to make the method more robust, it should have the ability to directly detect the target object from the point cloud. For this purpose, we can borrow some ideas from 3D object detection methods [Misra et al., 2021, Shi et al., 2019] that can directly detect 3D objects from the point cloud. Second, to make the method less dependent on the labelled data, we can apply the unsupervised methods [Chen et al., 2021, Mariotti et al., 2021] or few-shot learning methodology [Wang et al., 2021,] to the 6D object pose estimation field. However, for unsupervised learning, how to design the architecture and the loss function of the network is an open question. For few-shot learning, the main challenge is effectively extracting information from limited labelling data. Third, to test the performance of the proposed methods, we can apply the method to some real-world applications, such as robot grab [Wang et al., 2020] and virtual reality [Han et al., 2020].

Bibliography

[1] Aitor Aldoma, Federico Tombari, Luigi Di Stefano, and Markus Vincze. A global hypotheses verification method for 3d object recognition. In *European conference on computer vision*, pages 511–524. Springer, 2012.

[2] 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.

[3] Eric Brachmann. Learning to predict dense correspondences for 6d pose estimation. 2018.

[4] Eric Brachmann, Alexander Krull, Frank Michel, Stefan Gumhold, Jamie Shotton, and Carsten Rother. Learning 6d object pose estimation using 3d object coordinates. In *The European Conference on Computer Vision (ECCV)*, pages 536–551. Springer, 2014.

[5] Eric Brachmann, Frank Michel, Alexander Krull, Michael Ying Yang, Stefan Gumhold, et al. Uncertainty-driven 6d pose estimation of objects and scenes from a single rgb image. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3364–3372, 2016.

[6] Grigore C Burdea and Philippe Coiffet. *Virtual reality technology*. John Wiley & Sons, 2003.

[7] 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.

[8] 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.

[9] Wei Chen, Jinming Duan, Hector Basevi, Hyung Jin Chang, and Aleš Leonardis. Ponitposenet: Point pose network for robust 6d object pose estimation. In *The IEEE Winter Conference on Applications of Computer Vision (WACV)*, March 2020.

[10] Wei Chen, Xi Jia, Hyung Jin Chang, Jinming Duan, and Aleš Leonardis. G2l-net: Global to local network for real-time 6d pose estimation with embedding vector features. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[11] Wei Chen, Xi Jia, Hyung Jin Chang, Jinming Duan, Shen Linlin, and Ales Leonardis. Fs-net: Fast shape-based network for category-level 6d object pose estimation with decoupled rotation mechanism. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021.

[12] Yang Chen and Gérard Medioni. Object modelling by registration of multiple range images. *Image and vision computing*, 10(3):145–155, 1992.

[13] Zhi Chen, Wei Yang, Zhenbo Xu, Zhenbo Shi, and Liusheng Huang. Vk-net: Category-level point cloud registration with unsupervised rotation invariant keypoints. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1900–1904. IEEE, 2021.

[14] Jaeseok Choi, Yeji Song, and Nojun Kwak. Part-aware data augmentation for 3d object detection in point cloud. *arXiv preprint arXiv:2007.13373*, 2020.

[15] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255. Ieee, 2009.

[16] Piotr Dollár, Peter Welinder, and Pietro Perona. Cascaded pose regression. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1078–1085. IEEE, 2010.

[17] Andreas Doumanoglou, Tae-Kyun Kim, Xiaowei Zhao, and Sotiris Malassiotis. Active random forests: An application to autonomous unfolding of clothes. In *European Conference on Computer Vision*, pages 644–658. Springer, 2014.

[18] Andreas Doumanoglou, Rigas Kouskouridas, Sotiris Malassiotis, and Tae-Kyun Kim. Recovering 6d object pose and predicting next-best-view in the crowd. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[19] John E Draim. A common-period four-satellite continuous global coverage constellation. *Journal of Guidance, Control, and Dynamics*, 10(5):492–499, 1987.

[20] PN Druzhkov and VD Kustikova. A survey of deep learning methods and software tools for image classification and object detection. *Pattern Recognition and Image Analysis*, 26(1):9–15, 2016.

[21] 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.

[22] Xiao-Shan Gao, Xiao-Rong Hou, Jianliang Tang, and Hang-Fei Cheng. Complete solution classification for the perspective-three-point problem. *IEEE transactions on pattern analysis and machine intelligence*, 25(8):930–943, 2003.

[23] Shangchen Han, Beibei Liu, Randi Cabezas, Christopher D Twigg, Peizhao Zhang, Jeff Petkau, Tsz-Ho Yu, Chun-Jung Tai, Muzaffer Akbay, Zheng Wang, et al. Megatrack: monochrome egocentric articulated hand-tracking for virtual reality. *ACM Transactions on Graphics (TOG)*, 39(4):87–1, 2020.

[24] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *The IEEE International Conference on Computer Vision (ICCV)*, pages 2961–2969, 2017.

[25] 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.

[26] Stefan Hinterstoisser, Cedric Cagniart, Slobodan Ilic, Peter Sturm, Nassir Navab, Pascal Fua, and Vincent Lepetit. Gradient response maps for real-time detection of textureless objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(5):876–888, 2012.

[27] 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 *Asian conference on computer vision*, pages 548–562. Springer, 2012.

[28] Stefan Hinterstoisser, Vincent Lepetit, Naresh Rajkumar, and Kurt Konolige. Going further with point pair features. In *The European Conference on Computer Vision (ECCV)*, pages 834–848. Springer, 2016.

[29] Tomáš Hodaň, Jiří Matas, and Štěpán Obdržálek. On evaluation of 6d object pose estimation. In *The European Conference on Computer Vision (ECCV)*, pages 606–619. Springer, 2016.

[30] Tomáš Hodan, Pavel Haluza, Štepán Obdržálek, Jiri Matas, Manolis Lourakis, and Xenophon Zabulis. T-less: An rgb-d dataset for 6d pose estimation of texture-less objects. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 880–888. IEEE, 2017.

[31] Tomas Hodan, Frank Michel, Eric Brachmann, Wadim Kehl, Anders GlentBuch, Dirk Kraft, Bertram Drost, Joel Vidal, Stephan Ihrke, Xenophon Zabulis, et al. Bop: Benchmark for 6d object pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 19–34, 2018.

[32] Yinlin Hu, Joachim Hugonot, Pascal Fua, and Mathieu Salzmann. Segmentation-driven 6d object pose estimation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[33] Omid Hosseini Jafari, Siva Karthik Mustikovela, Karl Pertsch, Eric Brachmann, and Carsten Rother. The best of bothworlds: Learning geometry-based 6d object pose estimation. *arXiv preprint arXiv:1712.01924*, 2017.

[34] 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.

[35] 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 *The European Conference on Computer Vision (ECCV)*, pages 205–220. Springer, 2016.

[36] 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 *The IEEE International Conference on Computer Vision (ICCV)*, pages 1521–1529, 2017.

[37] Leonid Keselman, John Iselin Woodfill, Anders Grunnet-Jepsen, and Achintya Bhowmik. Intel realsense stereoscopic depth cameras. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–10, 2017.

[38] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[39] Alexander Krull, Eric Brachmann, Frank Michel, Michael Ying Yang, Stefan Gumhold, and Carsten Rother. Learning analysis-by-synthesis for 6d pose estimation in rgb-d images. In *The IEEE International Conference on Computer Vision (ICCV)*, pages 954–962, 2015.

[40] Alexander Krull, Eric Brachmann, Sebastian Nowozin, Frank Michel, Jamie Shotton, and Carsten Rother. Poseagent: Budget-constrained 6d object pose estimation via reinforcement learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6702–6710, 2017.

[41] Vincent Lepetit, Pascal Fua, et al. Monocular model-based 3d tracking of rigid objects: A survey. *Foundations and Trends® in Computer Graphics and Vision*, 1(1):1–89, 2005.

[42] Chi Li, Jin Bai, and Gregory D. Hager. A unified framework for multi-view multi-class object pose estimation. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[43] Yi Li, Gu Wang, Xiangyang Ji, Yu Xiang, and Dieter Fox. Deepim: Deep iterative matching for 6d pose estimation. In *The European Conference on Computer Vision (ECCV)*, September 2018.

[44] Zhigang Li, Gu Wang, and Xiangyang Ji. Cdpn: Coordinates-based disentangled pose network for real-time rgb-based 6-dof object pose estimation. In *The IEEE International Conference on Computer Vision (ICCV)*, October 2019.

[45] Zhigang Li, Gu Wang, and Xiangyang Ji. Cdpn: Coordinates-based disentangled pose network for real-time rgb-based 6-dof object pose estimation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7678–7687, 2019.

[46] 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 *European conference on computer vision*, pages 740–755. Springer, 2014.

[47] Zhi-Hao Lin, Sheng-Yu Huang, and Yu-Chiang Frank Wang. Convolution in the cloud: Learning deformable kernels in 3d graph convolution networks for point cloud analysis. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[48] Zhi-Hao Lin, Sheng-Yu Huang, and Yu-Chiang Frank Wang. Convolution in the cloud: Learning deformable kernels in 3d graph convolution networks for point cloud analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1800–1809, 2020.

[49] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *The European Conference on Computer Vision (ECCV)*, pages 21–37. Springer, 2016.

[50] David G Lowe. Object recognition from local scale-invariant features. In *iccv*, page 1150. IEEE, 1999.

[51] 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, 2016.

[52] Eitan Marder-Eppstein. Project tango. In *ACM SIGGRAPH 2016 Real-Time Live!*, page 40. ACM, 2016.

[53] Octave Mariotti, Oisin Mac Aodha, and Hakan Bilen. Viewnet: Unsupervised viewpoint estimation from conditional generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10418–10428, 2021.

[54] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 922–928. IEEE, 2015.

[55] Frank Michel, Alexander Kirillov, Eric Brachmann, Alexander Krull, Stefan Gumhold, Bogdan Savchynskyy, and Carsten Rother. Global hypothesis generation for 6d object pose estimation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 462–471, 2017.

[56] Ishan Misra, Rohit Girdhar, and Armand Joulin. An end-to-end transformer model for 3d object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2906–2917, 2021.

[57] Douglas Morrison, Adam W Tow, M McTaggart, R Smith, N Kelly-Boxall, S Wade-McCue, J Erskine, R Grinover, A Gurman, T Hunn, et al. Cartman: The low-cost cartesian manipulator that won the amazon robotics challenge. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7757–7764. IEEE, 2018.

[58] Markus Oberweger, Mahdi Rad, and Vincent Lepetit. Making deep heatmaps robust to partial occlusions for 3d object pose estimation. In *The European Conference on Computer Vision (ECCV)*, pages 119–134, 2018.

[59] Adam Paszke, Sam Gross, Soumith Chintala, and Gregory Chanan. Pytorch, 2017.

[60] Sida Peng, Yuan Liu, Qixing Huang, Hujun Bao, and Xiaowei Zhou. Pvnet: Pixel-wise voting network for 6dof pose estimation. *arXiv preprint arXiv:1812.11788*, 2018.

[61] Giorgia Pitteri, Michaël Ramamonjisoa, Slobodan Ilic, and Vincent Lepetit. On object symmetries and 6d pose estimation from images. In *2019 International Conference on 3D Vision (3DV)*, pages 614–622. IEEE, 2019.

[62] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (CVPR)*, July 2017.

[63] Charles R. Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J. Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[64] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, pages 5099–5108, 2017.

[65] Mahdi Rad and Vincent Lepetit. Bb8: a scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3828–3836, 2017.

[66] Mahdi Rad, Markus Oberweger, and Vincent Lepetit. Feature mapping for learning fast and accurate 3d pose inference from synthetic images. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4663–4672, 2018.

[67] Waseem Rawat and Zenghui Wang. Deep convolutional neural networks for image classification: A comprehensive review. *Neural computation*, 29(9):2352–2449, 2017.

[68] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7263–7271, 2017.

[69] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[70] 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*, September 2018.

[71] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 770–779, 2019.

[72] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10529–10538, 2020.

[73] Younghak Shin and Ilangko Balasingham. Comparison of hand-craft feature based svm and cnn based deep learning framework for automatic polyp classification. In *2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 3277–3280. IEEE, 2017.

[74] Jamie Shotton, Ben Glocker, Christopher Zach, Shahram Izadi, Antonio Criminisi, and Andrew Fitzgibbon. Scene coordinate regression forests for camera relocalization in rgb-d images. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2930–2937, 2013.

[75] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[76] Shuran Song and Jianxiong Xiao. Deep sliding shapes for amodal 3d object detection in rgb-d images. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 808–816, 2016.

[77] Martin Sundermeyer, Zoltan-Csaba Marton, Maximilian Durner, Manuel Brucker, and Rudolph Triebel. Implicit 3d orientation learning for 6d object detection from rgb images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 699–715, 2018.

[78] Martin Sundermeyer, Maximilian Durner, En Yen Puang, Zoltan-Csaba Marton, Narunas Vaskevicius, Kai O Arras, and Rudolph Triebel. Multi-path learning for object pose estimation across domains. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13916–13925, 2020.

[79] Alykhan Tejani. Latent-class hough forests for 3d object detection and pose estimation of rigid objects. 2014.

[80] Bugra Tekin, Sudipta N. Sinha, and Pascal Fua. Real-time seamless single shot 6d object pose prediction. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[81] Meng Tian, Marcelo H Ang Jr, and Gim Hee Lee. Shape prior deformation for categorical 6d object pose and size estimation. *arXiv preprint arXiv:2007.08454*, 2020.

[82] Jonathan Tremblay, Thang To, Balakumar Sundaralingam, Yu Xiang, Dieter Fox, and Stan Birchfield. Deep object pose estimation for semantic robotic grasping of household objects. *arXiv preprint arXiv:1809.10790*, 2018.

[83] Shinji Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (4):376–380, 1991.

[84] Haley A Vlach. How we categorize objects is related to how we remember them: the shape bias as a memory bias. *Journal of experimental child psychology*, 152: 12–30, 2016.

[85] Angtian Wang, Shenxiao Mei, Alan L Yuille, and Adam Kortylewski. Neural view synthesis and matching for semi-supervised few-shot learning of 3d pose. *Advances in Neural Information Processing Systems*, 34, 2021.

[86] 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. 2019.

[87] 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.

[88] Gu Wang, Fabian Manhardt, Federico Tombari, and Xiangyang Ji. GDR-Net: Geometry-guided direct regression network for monocular 6d object pose estimation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16611–16621, June 2021.

[89] 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 Conference on Computer Vision and Pattern Recognition*, pages 2642–2651, 2019.

[90] Pengyuan Wang, Fabian Manhardt, Luca Minciullo, Lorenzo Garattoni, Sven Meier, Nassir Navab, and Benjamin Busam. Demograsp: Few-shot learning for robotic grasping with human demonstration. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5733–5740. IEEE, 2021.

[91] Xia Wang, Tong Zhao, and Ding Wang. Grab application of remote operation service robot. In *Journal of Physics: Conference Series*, volume 1633, page 012031. IOP Publishing, 2020.

[92] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *arXiv preprint arXiv:1801.07829*, 2018.

[93] Paul Wohlhart and Vincent Lepetit. Learning descriptors for object recognition and 3d pose estimation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3109–3118, 2015.

[94] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1912–1920, 2015.

[95] 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.

[96] Bin Yang, Wenjie Luo, and Raquel Urtasun. Pixor: Real-time 3d object detection from point clouds. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 7652–7660, 2018.

[97] Wang Yifan, Noam Aigerman, Vladimir G Kim, Siddhartha Chaudhuri, and Olga Sorkine-Hornung. Neural cages for detail-preserving 3d deformations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 75–83, 2020.

[98] Yuan Yuan, Jia Wan, and Qi Wang. Congested scene classification via efficient unsupervised feature learning and density estimation. *Pattern Recognition*, 56: 159–169, 2016.

[99] Nida M Zaitoun and Musbah J Aqel. Survey on image segmentation techniques. *Procedia Computer Science*, 65:797–806, 2015.

[100] Sergey Zakharov, Ivan Shugurov, and Slobodan Ilic. Dpod: 6d pose object detector and refiner. In *The IEEE International Conference on Computer Vision (ICCV)*, pages 1941–1950, 2019.

[101] Zhengyou Zhang. Microsoft kinect sensor and its effect. *IEEE multimedia*, 19 (2):4–10, 2012.

[102] 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.

[103] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4490–4499, 2018.

[104] Menglong Zhu, Konstantinos G Derpanis, Yinfei Yang, Samarth Brahmbhatt, Mabel Zhang, Cody Phillips, Matthieu Lecce, and Kostas Daniilidis. Single image 3d object detection and pose estimation for grasping. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 3936–3943. IEEE, 2014.