Xi'an Jiaotong-Liverpool University

西交利物浦大學

SCHOOL OF ADVANCED TECHNOLOGY
SAT301 FINAL YEAR PROJECT

Pose Estimation Based on Point Cloud for Robot Grasping

FINAL THESIS

In Partial Fulfillment

of the Requirements for the Degree of

Bachelor of Engineering

Name:	Yuliang Xiao
Student ID:	1929288
Supervisor:	Quan Zhang
Assessor:	Sanghuyk lee

Abstract

Compared with two-dimensional images, three-dimensional images could provide more details, such as depth information or 3D point cloud, to help the machine to understand the surrounding environment. Therefore, 3D images have become the key point for robotic grasping in 3D space. This final year project realized object classification and object pose estimation based on the point cloud. In the point cloud collection phase, we evaluated two types of depth cameras: dual-lens cameras and structure light cameras. Dual-lens camera exhibits a large work range and high frame rate, but their accuracy is so low that the point cloud collected by them will lose most of the features, so it is not suitable for our project. Additionally, the most target we design is smaller than $10cm$, which means accuracy is the most important parameter for the depth camera. Thus, the structure light camera can be better chosen for its high accuracy. Before inputting the point cloud into the neural network, we need to process the point cloud to extract the target from millions of points and remove the useless points that will affect predicting results. In the pre-procession phase, we applied the RANSAC algorithm to fitting and remove the plane or workbench in the point cloud, the radius-based filtering to filter outliers, and the DBSCAN algorithm to cluster targets. In the end, we applied four neural networks to predict the category, position, and orientation based on deep learning. We have chosen PointNet to basic our work because it can end-to-end learning for irregular point data and finish objection classification and other works efficiently and accurately with a simple modification. Based on the PointNet network, we designed three PointNet-like networks to realize pose estimation for the target. Furthermore, in the PointNet-like network used for translation prediction, we added a residual structure which leads to high performance. In the experiment, the accuracy of the PointNet network for the object classification task in our dataset is more than 99%. In addition, the error in translation and rotation prediction is $1.31cm$ and $4.75°$, respectively. The code is made available at https://github.com/Regen2001/PointNet-Like-Pose-Estimation.

Keywords: Point Cloud, Deep Learning, Object Classification, Object Pose Estimation

Acknowledge

Thanks to the supervisor Quan Zhang for his assistance in this project. Thanks to the PhD candidate Junwei Wu for his assistance in this project.

Contents

List of Tables

List of Figures

List of Acronyms

DoF	Degree of freedom
FPS	Farthest point sampling
RANSAC	The Random Sample Consensus algorithm
DBSCAN	The Density-Based Spatial Clustering of Applications with Noise algorithm
CNN	Convolutional neural network
MLP	Multi-layer perceptron
NLL	Negative Log-Likelihood

Chapter 1
Introduction

1.1 Background

Robot grasping, or the ability of robots to manipulate novel objects reliably with their manipulators or grippers has a wide range of applications in various industries [2,3]. In manufacturing, robot grasping enables manipulators to perform tasks such as assembly line work and material handling, which leads to increased efficiency and safety for human workers; In healthcare, robots with grasping functions have the ability to assist in rehabilitation, therapy, and other tasks, enabling patients to recover from injuries, and helping the elder and the disabled to capture and move objects [3,4]. Nowadays, since the technologies of robot control, human-robot collaboration, and robot motion planning are developing swiftly, robot capturing becomes a reality [4–6].

However, since the limitations in machine perception, robot grasping is difficult to realize. In practice, the noise that comes from sensors leads to the error of the position for each object and exact orientation; and some parameters such as the center of mass, are unable to be measured directly [2]. Therefore, how to classify object variety, and measure the localization and pose for target in three-dimensional space is an important and valuable problem to increasing the accuracy of robot grasping at present [1]. Whereas, different with 2D image only stores color information, 3D image data could provide more details to help the computer understand surrounding environments better as it could provide depth information [7]. Nowadays, 3D image is applied to several of fields, such as self-driving, robotic control, remote sensing, and medical treatment [8]. Owing to point cloud data stores the original 3D geometric information without discrete processing, it is the most common format of 3D data [9]. Thus, in order to improve the ability of environmental perception, and the accuracy of grasping, more 3D vision sensors, which obtain depth information or point cloud data sets, be applied to robotic capturing.

1.2 Aims and objectives

This project hopes to use a SIASUN 6 degree of freedom (6DoF) machine arm with a parallel gripper as its end effector to capture an object in the workbench, which means the object frame related to the manipulator needs to be observed. Due to the technologies of linear control of manipulators and trajectory generation being well established, and the SIASUN machine arm has loaded a supervisory computer, the key point for this project is the part of machine vision. Furthermore, the task of machine vision can be divided into three parts roughly: data pre-procession, object classification, and object pose estimation [1,3,9].

Thus, we try to realize point cloud data generation via dual-lens camera and structured light camera. However, owing to the original point cloud data including the number of useless points, it will be processed before being sent into the neural network. In the stage of pre-procession, the most important two operations are removing plane and clustering objects. Then, the PointNet network will classify these processed points and return a vector that includes the result, while these results are considered as a new feature and spliced with the original point cloud data. In the end, three PointNet-like networks will predict the pose of these targets. The final result of the object pose will be expressed as an Euler angle vector $[\phi,\ \theta,\ \psi]$ and a translation vector $[p_x,\ p_y,\ p_z]$.

After predicting the object pose and location, these results need to be transformed into the manipulator coordinate system from the camera coordinate system. Different from the general robotic grasp system, we will load the camera on the end effector of the robot, since the position and orientation of the camera can be changed via changing the location and posture of the end effector, which leads to an increased detection and working zone of the robot without add more camera. Furthermore, robotics can choose the best position and orientation to observe the objection. Figure 1.1 is the sketch for the location of the camera on the manipulator.

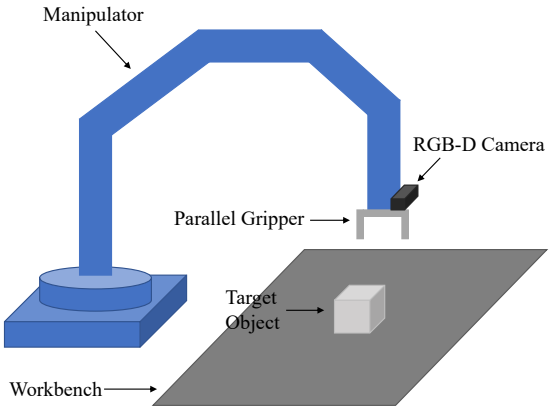

Figure 1.1: The sketch for the manipulator, where the camera is on the end effector

1.3 Related work

The traditional methods used to estimate six-dimensional pose can be divided into two main ideas roughly: feature-based and template-based [10]. In the first method, features are obtained in 2D images from points of interest or pixels, and then, these features will be matched to the feature on 3D models [11]. However, for feature-based method, the accuracy and efficiency of the result of pose estimation will be influenced by the extraction and description of features [1]. For instance, Wojciechowski and Suszynski [12] applied the RANSAC algorithm to realize rough recognition and got the pose result of an object via applying the iterative closest point algorithm to finish exact matching, while constructing the robot assembly process through the data collected from optical 3D scanners. Furthermore, the template-based method means a rigid template is constructed which is used to scan different positions of the input images [13]. In the second method, a similarity score will be calculated at each location and then, the score is compared with these similarity scores of each location, the best match one is the final pose information [14, 15]. For example, Zhu et al. [16] tried to track objects at times based on the CAD model by a monocular camera, which predicts the rotation matrix thought matching a global model library that estimated offline with images.

Nowadays, as the development of deep learning in these years, more deep-learning-based methods have been proposed to deal with the problem of 6D pose estimation [17]. For instance, PoseCNN, a convolutional neural network, is able to estimate target pose

while targets are occluded and symmetric in cluttered scenes [10]. Then, SSD-6D network to design to track 3D objects and estimate their 6D pose from RGB images, but this method is affected by the difference of color between synthetic model and scene appearance, also including local lighting variation [18]. Li et al. [19] showed a unified framework that can obtain the pose of the target from single and multiple views. Hu et al. [20] trained a segmentation-driven 6D pose estimation framework to solve the problem that several objects with poorly textured are occluded by each other. What is more, some methods are finished objection pose estimation from RGB-D images. For example, the DenseFusion network, which adds dense feature representations, and incorporates color and depth information, could give the 6D pose of the target from RGB-D images [21]. However, same with other methods based on RGB images, the accuracy of DenseFusion will be influenced by some factors which lead to chromatic aberrations such as lighting [20, 21].

In order to avoid the effect of lighting, some methods adopt point clouds to complete the object capturing through sampling and evaluating the position and orientation of the manipulator. For example, Mahler et al. [7] proposed a grasp quality convolutional neural network model (GQ-CNN) via Dex-Net 2.0 dataset to predict swiftly the probability of success of capture from depth maps and realize a high success rate on eight known objects. PointNetGPD, which is based on PointNet network, achieved end-to-end grasp evaluation, which directly processes the point cloud and grasps a complex geometric object even if the density of the input point cloud is low [22]. Such methods can realize a high accuracy and success rate in object capturing, but since the problem of object pose estimation cannot be solved reliably, it is impractical that apply them to industry safely [1].

Chapter 2

Problem Formulation

Generally, several homogeneous transformation matrices need to be collected in a vision system, while Fig 2.1 illustrates each frame of the system and the homogeneous transformation matrix used to transform positions and orientations, and for example, {Sen} is the coordinate system for the sensor and ${}^{Sen}_{Obj}T$ is the homogeneous transformation matrix of {Obj} reference to {Sen}. In this project, we set the center of mass for each object as the origin of {Obj} where is the grasping position, and since the density of the object is constant, the center of mass also is the geometric center.

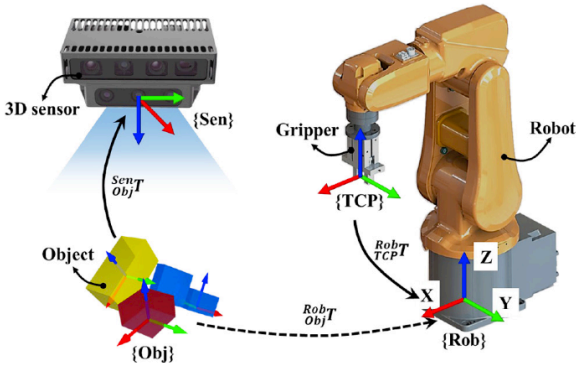

Figure 2.1: The illustration of the vision system of robot [1]

In grasping tasks, only when ${}^{Rob}_{TCP}T = {}^{Rob}_{Obj}T$, which means the locations and orientations of the end effector and objection related to the robot are equal, and the target can be captured successfully [23]. For instance, the homogeneous transformation matrix ${}^{Rob}_{Obj}T$, which means transform operation from frame {Obj} to frame {Rob}, is defined as

$$ {}^{Rob}_{Obj}T = \begin{bmatrix} {}^{Rob}_{Obj}R_{X'Y'Z'} & {}^{Rob}P_{ObjORG} \\ 0 & 1 \end{bmatrix} \in \mathbb{R}^{4\times4} \tag{2.1} $$

where ${}^{Rob}P_{ObjORG}$ is the position of the origin of {Obj} relative to {Rob} and expressed as

$$ {}^{Rob}P_{ObjORG} = \begin{bmatrix} p_x & p_y & p_z \end{bmatrix}^T \in \mathbb{R}^3 \tag{2.2} $$

where ${}^{Rob}P_{ObjORG}$ is the position of the origin of {Obj} relative to {Rob} and expressed as

For the Euler angle, there exist 12 possible sequences, and the X-Y-Z (intrinsic rotations) sequence and Z-Y-Z (extrinsic rotations) sequence are used commonly, while the first one is applied in this project since it describes the rotation in pose estimation better [24]. ${}^{Rob}_{Obj}R_{X'Y'Z'}$ means the rotation matrix for the X-Y-Z Euler angle, while it is written as

$$ {}^{Rob}_{Obj}R_{X'Y'Z'} = R_x(\phi)R_y(\theta)R_z(\psi) \in SO(3) \tag{2.3} $$

where $R_x(\phi)$, $R_y(\theta)$, and $R_z(\psi)$ are the rotation matrix around X-axis, Y-axis, and Z-axis, respectively, while if $\sin(\alpha)$ and $\cos(\alpha)$ are written as $s\alpha$ and $s\alpha$, these rotation matrices are defined as

$$ R_x(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\phi & -s\phi \\ 0 & s\phi & c\phi \end{bmatrix} \quad R_y(\theta) = \begin{bmatrix} c\theta & 0 & s\theta \\ 0 & 1 & 0 \\ -s\theta & 0 & c\theta \end{bmatrix} \quad R_z(\psi) = \begin{bmatrix} c\psi & -s\psi & 0 \\ s\psi & c\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.4} $$

Furthermore, the rotation matrix for the X-Y-Z Euler angle is calculated as

$$ {}^{Rob}_{Obj}R_{X'Y'Z'} = \begin{bmatrix} c\theta c\psi & -c\theta s\psi & s\theta \\ c\phi s\psi + c\psi s\phi s\theta & c\phi c\psi - s\phi s\theta s\psi & -c\theta s\phi \\ s\phi s\psi - c\phi c\psi s\theta & c\psi s\phi + c\phi s\theta s\psi & c\phi c\theta \end{bmatrix} \tag{2.5} $$

However, since the depth camera only could observe ${}^{Sen}_{Obj}T$, so ${}^{Rob}_{Obj}T$ should be calculated as

$$ {}^{Rob}_{Obj}T = {}^{Sen}_{Obj}T \, {}^{TCP}_{Sen}T \, {}^{Rob}_{TCP}T \tag{2.6} $$

Therefore, the most important part of a vision-based robot is object frame collection and Fig 2.2 shows the sketchy process that how to observe the object's position and pose relative to the robot.

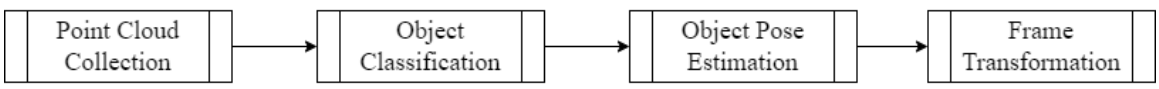

Figure 2.2: The sketchy flow chart of the process for object frame collection

Chapter 3
Methodology

In practical experiment, the point clouds always collected by 3D sensors such as dual-lens camera, laser radar, et cetera. In this project, dual-lens and structure light camera are applied because these two types camera could save the color information at the same time which is convenient for data visualization. After collecting the point cloud, as the original data includes more than 900,000 points, it cannot be used directly until finishing pre-procession. After extracting the objects, all of these points will be predicted their category and pose by PointNet network and PointNet-like network. Furthermore, as deep learning method was applied, building a dataset to train these networks is necessary. However, because of the limitation of experimental conditions, we built the simulated dataset via computer software and scripts, which is more convenient than actual shooting via depth camera.

3.1 Properties of Point Sets in $\mathbb{R}^3$

Point clouds also are described as a point set in three dimension $\{P_i | i = 1, \ldots, n\}$, while each point P_i is also written as a vector in Cartesian coordinate $(x,\ y,\ z)$. In different point cloud set, the different features will be added after the coordinate vector such as normal vector or color, but not common [25]. Therefore, a point cloud can be expressed as a $n \times 3$ matrix in Euclidean space. There are some main properties for point cloud set in $\mathbb{R}^3$:

- Unordered: Different with the pixels in an image have specific order, point cloud sets with same elements should be the same set, which means a network inputted a set with N points should be invariant for $N!$ permutation [25].
- Interaction among points: There exists interaction among points, which means all points are not isolated, and they are meaningful for neighboring points. Thus, the model need to have the ability that extract local structures from nearby points and their combinatorial interactions [26].

- Invariance under transformations: The point cloud set should be invariant under the operation of transformation, which means if rotate and translate all the points in one set and then, the new set is same with the original one [25].

Furthermore, for the point cloud collected by depth camera, most time they are non-uniform density in different areas [26]. For example, the area closed to the sensor has high-density points but the area away from the camera includes low-density points. Thus, in the down-sampling, farthest point sampling (FPS) algorithm is introduced to save as many features as possible in the original point set.

3.2 Point Cloud Collection

In this project since we need to collect point cloud data, we applied intel RealSense Depth D455, which is a dual-lens camera and shown in Fig 1(a), and Percipio FS20-E1, which is a structured light camera and shown in Fig 1(b).

(a) The photo of intel RealSense Depth D455 (b) The photo of Percipio FS20-E1

Figure 3.1: The photos for two types of depth camera

3.2.1 Dual-lens Camera

This camera, also known as a stereo camera, is a type of camera that uses two lenses to capture images, and so it captures two images of the same object from slightly different angles, which is similar to how human eyes view an object [27]. For dual-lens camera, the depth information is calculated by stereo triangulation, where the images captured by the two lenses are compared, and the differences in the images are used to calculate the depth

information, because objects that are closer to the camera will appear to move more between the two images than objects that are farther away. By analyzing the differences in the two images, the camera can calculate the distance to the object and create a 3D image [28].

For this type of camera, it can capture depth information in real-time with high resolution and accuracy, making it useful for applications that require fast processing. That is why it is widely used in robotics, autonomous vehicles, and object recognition and tracking [27].

However, since this type of camera is depended on external light source, the result of the depth information will be affected by lighting conditions. What is more, due to the theory to calculate depth information is analyzing the differences from two images, it requires two cameras to be aligned perfectly, which can be difficult to achieve. In the end, it may struggle with objects which has less texture, such as transparent or reflective object, as it may not be able to capture the necessary differences between the images [27].

For intel RealSense Depth D455, its ideal working range is $0.6m$ to $6m$ with 2% depth accuracy at $4m$, and its minimum depth distance at max resolution (1280×720) is around $0.52m$ [29]. Since the higher resolution could increase the depth accuracy, we set the color and depth resolution of D455 as 640×480 and 1280×720, while the RGB and depth frame rates were set as 30 fps and 5 fps, respectively.

3.2.2 Structure light Camera

The structure light camera is also a type of depth camera, but different with dual-lens camera depended on external light source, it has active light source, since it uses a projector and a camera to capture the information [30]. When it is working, this camera emits a pattern of light, such as a grid of horizontal and vertical lines, onto the object of interest. The light patterns are projected at different angles and positions, so that the object is illuminated from multiple directions. The camera captures the distorted light pattern and analyzes it to determine the shape and depth of the object. This is done by comparing the distorted pattern to the original pattern projected by the camera, and calculating the displacement of the pattern [31].

Structured light cameras have some advantages over other types of depth cameras. For

example, it provides highly accurate 3D measurements with sub-millimeter accuracy in some cases. They can capture 3D data quickly, often in real-time, making them suitable for applications that require rapid data acquisition [30].

However, structured light cameras have some limitations. They have a limited range of measurement, typically up to a few meters, which makes them unsuitable for applications that require long-range 3D measurement. Structured light cameras are sensitive to ambient light, which can interfere with the projected pattern and affect the accuracy of the 3D measurements. Additionally, structured light cameras may have limited resolution compared to other 3D imaging technologies, such as laser scanners or photogrammetry, which can limit their suitability for certain applications that require high levels of detail [30].

For Percipio FS20-E1, its ideal working range is $0.3m$ to $1.4m$ with 0.2% depth accuracy at $0.7m$. This camera has a high accuracy but low frame rate.

3.3 Point Cloud Pre-procession

After obtaining the depth image and color image though depth camera, we transformed the two images into a point cloud set via the internal parameters of the depth camera based on Open3d library in Python, which will be used in the following procession [32]. Then, we eliminated numerous points by distance since we determined that the points we needed were within a specific range. After removing the points belonging to the workbench in the point cloud set, the target points were collected. Finally, we sorted each point into different sets according to the object to which it belongs. Figure 3.2 is the flow chart for this part.

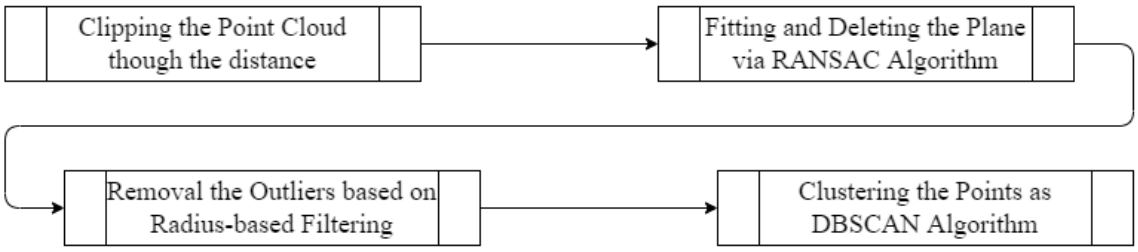

Figure 3.2: The flowchart of the pre-procession for point cloud collected by depth camera

3.3.1 Plane Fitting and Removal: RANSAC Algorithm

The Random Sample Consensus (RANSAC) algorithm is a widely-used robust estimation technique that is capable of identifying inliers in a dataset contaminated with outliers [33]. This algorithm works by randomly selecting a subset of points from the dataset and fitting a model to those points. Then, the remaining points are tested against the model, and those that fall within a certain tolerance are considered inliers. Finally, the process is repeated multiple times, and the model that has the most inliers is selected as the best estimate of the underlying structure of the data [33].

Let $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ be a set of N data points, where each data point $\mathbf{x}_i$ has d dimensions, and $\mathbf{H}$ be the set of all possible hypotheses, where each hypothesis corresponds to a model with k parameters. In each iteration, RANSAC randomly selects a minimal subset of data points $\mathbf{X}_s \subseteq \mathbf{X}$, where $|\mathbf{X}_s| = k$. The model parameters $\boldsymbol{\theta}$ are then estimated by solving the equation $\mathbf{X}_s\boldsymbol{\theta} = \mathbf{y}$, where $\mathbf{y}$ is a vector of measurements corresponding to the selected data points. The quality of the fit is evaluated by counting the number of inliers n and outliers m using a distance threshold t, such that a data point $\mathbf{x}_i$ is an inlier if $\|\mathbf{x}_i\boldsymbol{\theta} - \mathbf{y}_i\| \leq t$, and an outlier otherwise [33]. Therefore, the algorithm repeats the process for a fixed number of iterations M, and selects the hypothesis with the largest number of inliers as the final model [33]. Algorithm 1 in Appendix B.1 is the pseudocode of RANSAC algorithm.

3.3.2 Outliers Removal: Radius-based Filtering

Radius-based filtering is a fundamental operation in computer graphics and geometric modeling, which involves selecting a subset of points or elements from a larger set based on their proximity to a given point or set of points [34]. The basic idea behind radius-based filtering is to consider a fixed radius around each point in a given set, and select all points or elements that fall within this radius. Formally, given a set of n points $X = \{x_1, x_2, ..., x_n\}$ in a d-dimensional space, and a query point q, the radius-based filter operation selects all points x_i in X that satisfy

$$|x_i - q|| \leq r \tag{3.1}$$

where r is the radius of the filter. The resulting set of points is often referred to as the neighborhood of the query point. Furthermore, this method can be extended to more complex structure while we applied this method to filter the outlier in point cloud set [34]. Algorithm 2 in Appendix B.2 is the pseudocode for radius-based Filtering.

Radius-based filtering is a simple and powerful operation that can be used in a variety of applications. However, selecting an appropriate radius parameter is crucial, as it can significantly affect the resulting set of points or elements, so finding a suitable radius parameter is an important in this algorithm.

3.3.3 Objects Clustering and Regression: DBSCAN Algorithm

The Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm is a popular unsupervised clustering method [35]. Unlike traditional clustering algorithms, such as k-means, which require the specification of the number of clusters in advance, DBSCAN is capable of discovering the number of clusters automatically, making it particularly useful in scenarios where the underlying structure of the data is unknown.

DBSCAN algorithm is to group together points that are densely packed in a high-dimensional space, while in particular, the algorithm defines two key parameters which are the radius of a neighborhood around each point ϵ, and the minimum number of points required to form a dense region $MinPts$, respectively. Points that are within ϵ distance of each other are considered neighbors, and a point is considered a core point if it has at least $MinPts$ neighbors, ant then a cluster is formed by recursively adding points that are within ϵ distance of a core point, but points that do not belong to any cluster are considered noise [35]. Algorithm 3 in Appendix B.3 is the pseudocode for DBSCAN algorithm.

Compared with traditional clustering algorithms, DBSCAN is robust to noise and outliers, as points that are not part of any cluster are classified as noise and what is more important, it is capable of discovering clusters of arbitrary shape without specify the number of clusters in advance.

3.4 Object Classification

Most of the methods for object classification always learn the embedding of each point first and then, gather the global features from the total data set. The result is given though calculate these global features into several one-dimensional convolution and fully connected layers [9, 25]. The existing methods of objection classification can be divided into multiview-based, volumetric-based, and point-based methods. In our project, we chose point-based method and applied PointNet network.

Multi-view based methods rely on rendering multiple views of the 3D object from different viewpoints and then processing each view as a 2D image with convolutional neural networks (CNN) [36]. Volumetric-based methods, on the other hand, represent the 3D object as a 3D grid of voxels, which can be processed with 3D CNNs [37]. These methods may be more effective in capturing local geometric features or handling occlusion, but they also require additional pre-processing and can be more memory-intensive [9].

In contrast, one of the key features of point-based method is its ability to directly process raw point clouds without the need for pre-processing steps, which can be more memory-efficient and computationally faster than voxel-based or multi-view based methods [9]. PointNet network is one of application of point-based method. To deal with the unordered point set, PointNet uses a hierarchical neural network architecture that applies shared multi-layer perceptrons (MLPs) to each point independently to learn point-wise features, and then aggregates these features to obtain a global feature representation of the entire point cloud.

In this project, we refereed [38] to realize PointNet network though PyTorch library in Python [39], while the network used to realize pose estimation also refer to this work.

3.4.1 PointNet Network Architecture

The structure of PointNet is shown in Figure 3.3, where is visualized the classification network the segmentation. In this network, the point cloud will be transformed into the best frame by a 3×3 transformation firstly, and then, all points will be up to 64-dimension from 3-dimension though 1-dimension (1D) convolution. After that, the same operation

be done that this $n \times 64$ set be transformation though a 64×64 transformation, and up to 1028-dimension by 1D convolution. The reason that the point cloud set is upped to 1028-dimension is that we hope as much as feature can be keep from the original point cloud after max pooling, and then, a 1028-dimensional vector of global feature is obtained. The transform matrix in this process are got via a small PointNet network. Additionally, this part also is being called as PointNet encoder. For object classification work, the global feature calculated though a multi-layer perceptron (MLP).

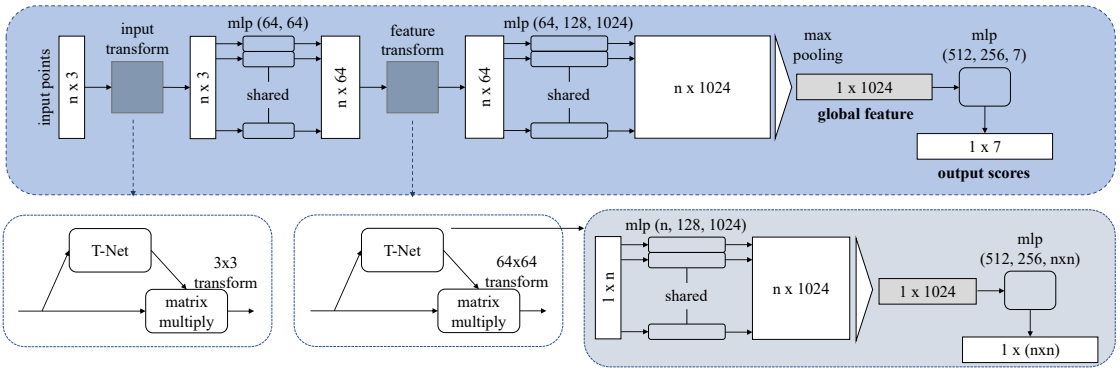

Figure 3.3: The architecture of PointNet used to finish object classification

Symmetry Function to Handle Unordered Point Set Since the unordered property of point cloud, we need a symmetry function to deal with these sets which could avoid the influence from the order change, and the mathematical expression is

$$f(\{x_1, x_2, \ldots, x_n\}) = f(\{x_{\pi(1)}, x_{\pi(2)}, ..., x_{\pi(n)}\}),\ x_i \in \mathbb{R}^3 \tag{3.2}$$

for any permutation π of the indices $\{1, 2, ..., n\}$.

Then, a function is defined on a point set which applied a symmetric function on its transformed factors, and it is shown as

$$f(\{x_1, x_2, \ldots, x_n\}) \approx g(\{h(x_1), h(x_2), \ldots, h(x_n)\}) \tag{3.3}$$

where $f : 2^{\mathbb{R}^N} \longrightarrow \mathbb{R}$, $h : \mathbb{R}^N \longrightarrow \mathbb{R}^K$, and $g : \underbrace{\mathbb{R}^K \times \cdots \times \mathbb{R}^K}_{n} \longrightarrow \mathbb{R}$ is a symmetric

function. For example, both accumulation and accumulative multiplication are symmetry function [25]. In PointNet, we used max pooling to finish this idea.

Joint Alignment Network Owing to the point cloud set are invariant under transformation, it is hoped that the network can be finished to this requirement. The solution is that align the input points before extracting features. In 2D images, though introducing spatial transformer, it can be aligned though sampling and interpolation [40]. Thus, this target can be easier because of form of point cloud, a mini-network which is T-Net shown in Figure 3.3 be used to predict an affine transformation matrix.

Then, the idea of mini-network can be expanded to feature space. However, since the dimension of feature space are higher than point space, a regularization term be added to the SoftMax training loss. Therefore, the transformation matrix could approach orthogonal matrix

$$L_{reg} = \left\| I - AA^T \right\|_F^2 \tag{3.4}$$

where A is the transformation matrix used to align feature, and it will nor loss information as an orthogonal matrix [25]. Furthermore, the addition of regularization term could increase the robustness and performance.

3.4.2 Loss Function for Back Propagation for PointNet

For this classification task, we applied the log-SoftMax function to the activate function in the last, since it voids computing the exponential function, which can easily overflow or underflow in numerical calculations, which means it is more numerically stable than Soft-Max function [41]. When given a vector of $\mathbf{x} = [x_1, x_2, ..., x_k]$, the Log-SoftMax function is defined as

$$\text{log\_softmax}(x_i) = \log(\frac{\exp(x_i)}{\sum_{j=1}^{k} \exp(x_j)}) \tag{3.5}$$

Therefore, the Negative Log-Likelihood (NLL) loss function, which widely used when the output of the model is a probability distribution over a set of k classes, is applied to calculate the loss of the model [42]. The NLL loss function measures the difference between the predicted probability distribution and the true probability distribution of the target variable,

by computing the negative log-likelihood of the true class label. If given a predicted probability distribution $\mathbf{p} = [p_1, p_2, ..., p_k]$ and a true class label y, NLL function is expressed as

$$\mathcal{L}(\mathbf{p}, y) = -\log(p_y) \tag{3.6}$$

The NLL loss function penalizes the model for making incorrect predictions by increasing the loss value when the predicted probability of the true class label is low [43].

3.5 Object Pose Estimation

Point-based methods are a popular approach for pose estimation using point clouds, while these methods use the geometric properties of the points to estimate the pose of the object. Point-based methods can be divided into two categories: feature-based and direct methods [3].

Feature-based methods extract distinctive features from the point cloud, such as corners or edges, and use them to estimate the object's pose. These methods are effective for objects with distinct and recognizable features, but may be less accurate for objects with complex shapes or surfaces [3].

However, direct methods can handle noisy and incomplete data and can be applied to a wide range of objects with varying shapes and surfaces [3]. Additionally, point-based methods can be combined with other techniques, such as deep learning, to improve their accuracy and robustness, so this project chose direct method.

3.5.1 Analysis for Pose Estimation

Same with PointNet, these networks for pose estimation is an end-to-end framework which the input is point cloud matrix and output is the pose matrix predicted by the network, and the total framework is shown in Figure 3.4. But different with PointNet network only the point set is input, the result of object classification is a new feature for pose estimation network. For example, there are three types of objections and their order is a, b, and c, then if the result of classification is b, which is the second one in the order, so the class vector

$[0, 1, 0]$ will be spliced with point set like the point-class fusion vector in Figure 3.4.

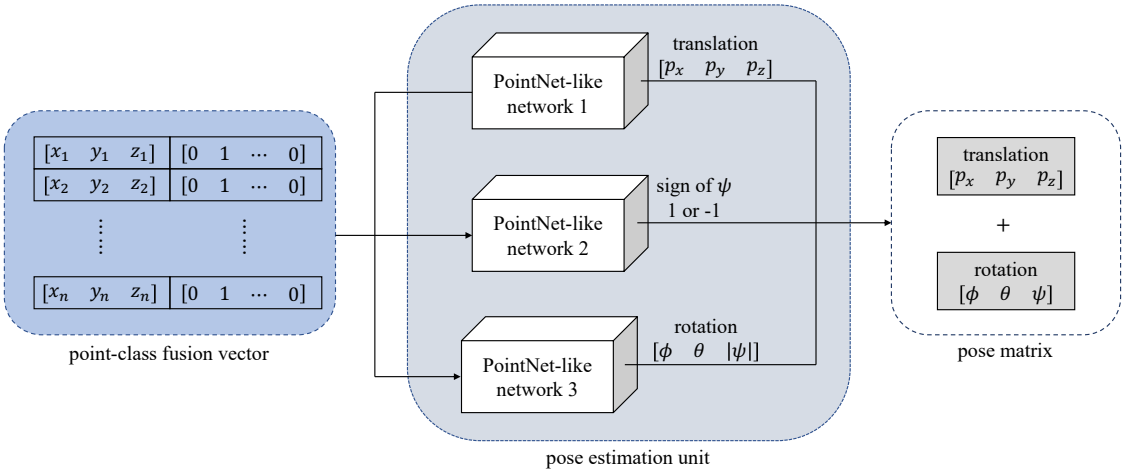

Figure 3.4: The overview for pose estimation unit

In this unit, there are three PointNet-like networks which used to output the pose matrix. Firstly, the point cloud data will be normalized into a unit sphere and then, input the three PointNet-like networks. PointNet-like network 1 will output the translation matrix $[p_x, p_y, p_z]$ while PointNet-like network 2 and 3 output the rotation matrix $[\phi, \theta, \psi]$ but PointNet-like network 2 outputs the sign of ψ and PointNet-like network 3 outputs $[\phi, \theta, |\psi|]$.

Compared with the translation estimation, the rotation estimation is more complex. If the targets are placed on a horizontal plane simply, the value of ϕ and θ are always zero so the end-effector is perpendicular to the plane and only the value of ψ need to be estimated [1].

However, it is more common that the objection is on an inclined plane with overleap and considering this condition, if ϕ and θ are the rotation values of the object on the x and y-axis, respectively, the angle of the inclined plane is decided as

$$\text{plane}_{\text{tilt}} = \max(|\phi|, \theta) \tag{3.7}$$

Owing to the capturing location of the target is always different when the plane angle is less than $45°$ and greater than $45°$, when the grasping position changes, it should be handled in different cases. If the inclination is greater than $45°$, the complementary angle is chosen.

In the end, the value of ψ need to be decided only [1].

3.5.2 PointNet-like Network Architecture

Figure 3.5 is the detailed structure for Pointnet-like network 1 for translation predicted. This is a simple network which only save the mlp structure in PointNet to increase the dimension of each point from 3-dimension to 1028-dimension. But different with Pointnet-like network 2, we added a residual structure to avoid the problem of gradient dissipation and degradation [44].

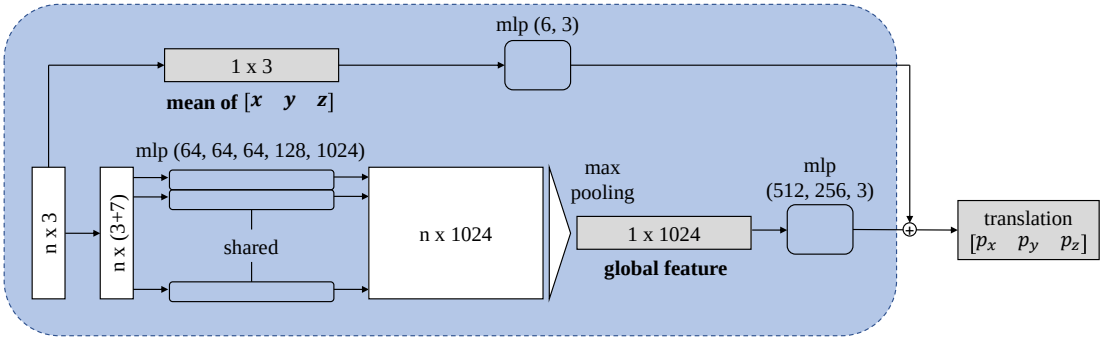

Figure 3.5: The detailed architecture for PointNet-like network 1

Furthermore, as the figure in Append A, the mean of point set should be calculated before the operation of normalization, while normalization will transform the points into an unit sphere. Given a set of n points $x = \{x_1, x_2, ..., x_n\}$ in a d-dimensional space, the mean of n points is expressed as $\bar{x} = \frac{1}{n}\sum x_i$, and the max radius of points set is got though $r = \max_{i=1}^{n} \|x_i - \bar{x}\|$. Therefore, the normalized point set X is calculated as

$$X = \frac{x - \bar{x}}{r} \tag{3.8}$$

where it is clear that the mean of normalized points is zero.

Figure C.2 is the detailed structure for Pointnet-like network 2 for rotation predicted rotation, which is same with Pointnet-like network 1 but no residual structure. In contrast, this network is more complex than the above network while we added a feature transform. In addition, both Pointing-like network 1 and 2 can be considered as fitting network.

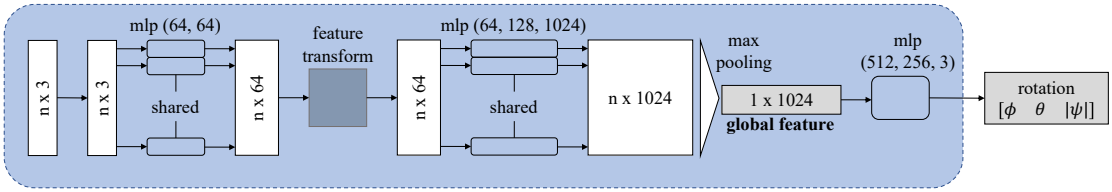

Figure 3.6: The detailed architecture for PointNet-like network 2

Figure 3.7 is the detailed structure for Pointnet-like network 3 for rotation predicted the sign of rotation. This is a simple network like Pointnet-like network 1, which have the same structure. However, PointNet-like 3 is a classification network.

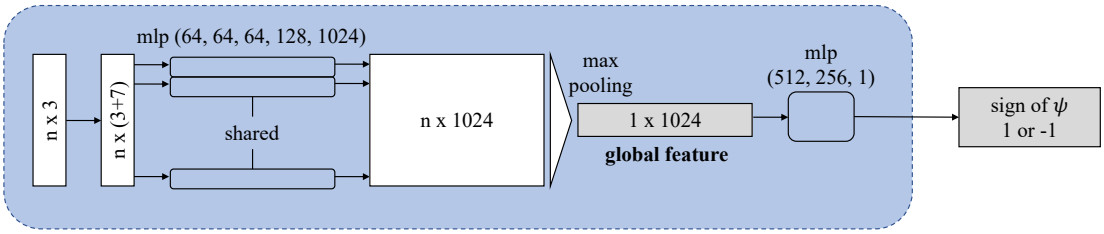

Figure 3.7: The detailed architecture for PointNet-like network 3

3.5.3 Loss Function for Back Propagation for PointNet-like network

For these two fitting network, the norm be used to measure the distance for target and predict result, while the Euclidean norm and the L1 norm are commonly used [41]. The Euclidean norm is a simple and intuitive geometric interpretation, which can be understood as the distance between two points, as written as

$$\|X\|_2 = \sqrt{\sum_i |x_i|} \tag{3.9}$$

However, the L1 norm, which expressed as

$$\|X\|_2 = \sum_i |x_i| \tag{3.10}$$

is more robust to outliers, gives more weight to large differences, and is generally faster to

compute [43]. In the next result, both two norm will be tested.

For PointNet-like network 3, although it is also a classifier like PointNet network, PointNet-like network 3 is work for a binary classification work. Therefore, we chose Sigmoid function to realize classification [45]:

$$\sigma(x) = \frac{1}{1 + e^x} \tag{3.11}$$

And the loss function for PointNet-like network 3 is Binary Cross function:

$$\mathcal{L})(x_n, y_n) = y_n \cdot \log(x_n) + (1 - y_n) \cdot \log(1 - x_n) \tag{3.12}$$

3.6 Dataset

A dataset is a fundamental component in the field of machine learning, which is a collection of data points that are used to train and test machine learning models. The goal of a dataset is to provide a representative sample of the problem domain that the machine learning model is attempting to solve. In order for a dataset to be useful, it must be large enough to provide the machine learning model with enough examples to learn from, and in our dataset, we have seven categories that each category includes 8,000 sample. However, it is also important that the dataset is not so large that it becomes difficult to manage and process. Additionally, the data must be diverse enough to represent the full range of possible inputs and outputs that the model may encounter in the real world.

In this project, we chose seven different but common object, which are cube, cuboid, cylinder, I beam, combination of two cuboids (C cuboid), combination of two cylinders (C cylinder), and combination of cuboid and cylinder (C cc), respectively. Since these workpieces are simple, we have written some Python scripts to build them, and for each type of workpiece, we set some variables to change the size and shape of the workpiece. For example, the length of the sides of the cube are randomly generated within 5 to $10cm$. The detail of size for other workpiece is shown in Table C.1 in Appendix C. Figure 3.8 shows the results of our script.

In the scripts, for the convenience of follow-up work, we set the points used for grasp-

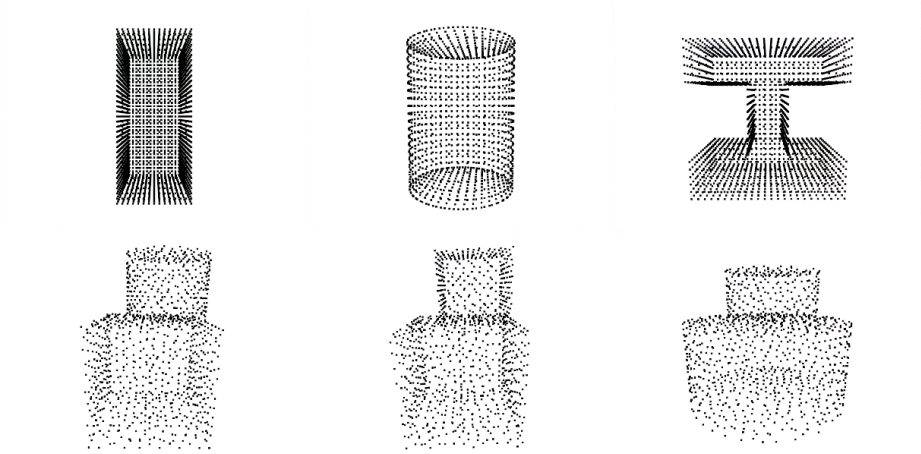

Figure 3.8: The detailed architecture for PointNet-like network 3

ing all are $\begin{bmatrix}0 & 0 & 0\end{bmatrix}$ and their Euler angle are $\begin{bmatrix}0 & 0 & 0\end{bmatrix}$. Then, though the different and random translation and rotation, numerous of sample can be built [46]. Furthermore, we have assumed our scope of work, the translation range for X, Y, and Z-axis are -0.5 to $0.5m$, -0.5 to $0.5m$, and 0.6 to $1m$, respectively. However, since the different rotation may lead to the same point cloud, the ranges of rotation are different for each workpiece. For example, every additional 90 degrees of rotation on the Z-axis produces the same point cloud, so the rotation range of Z-axis is $-45°$ to $45°$, while the rotation range for two others axis both are $-45°$ to $45°$. Then, the rotation range for other workpiece are shown in Table C.2 in Appendix C. Furthermore, in order to mimic the camera's ability to photograph only one side of an object, we cut half of our point cloud and shown in Figure 3.9. In the end, Figure A.2 in Appendix A shows the random transformation for models in dataset.

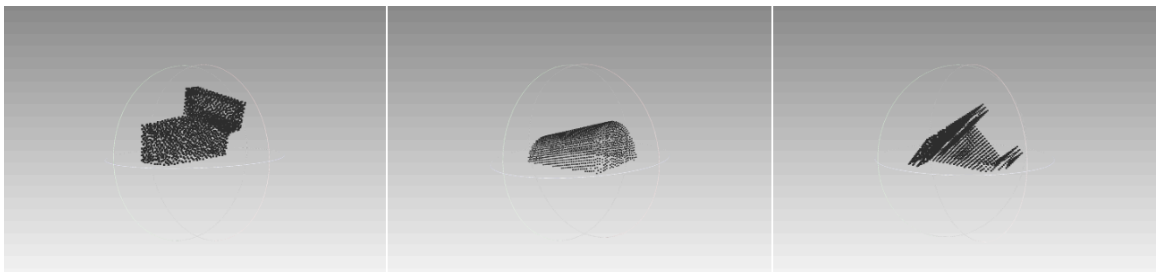

Figure 3.9: The point cloud model be cut half for dataset built via Python scripts

Chapter 4

Experimental Result

4.1 Result of Point Cloud Collected and Pre-procession

Before generating the point cloud set, only the RGB image and depth image can be captured directly, while the depth image is a two-dimensional representation of the distance information in a three-dimensional scene. Then, Figure 4.1 is the picture of RGB image on the left side and depth image on the right side, and we used the different color to stand for the different distance. What is more, the RGB image and depth image cannot be used directly until they are aligned.

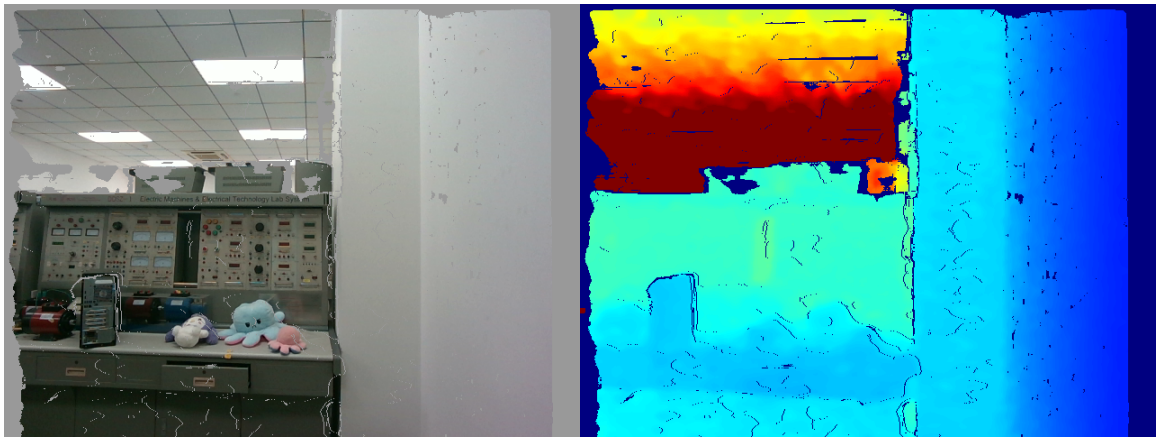

Figure 4.1: The picture captured from depth camera. The left image is RGB image and the right one is depth image

However, it is necessary to get the intrinsic parameters of the depth camera if we want to calculate the real position for each pixel in the depth image. Figure 4.2 is a model of a camera, where f is the focal length and the P is the real object coordinate $(X, Y, Z)^T$.

If the position P' on the camera is $(u, v, 1)^T$, and K is the matrix of camera intrinsics, the real position is calculated as

$$Z \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \mathbf{KP} \tag{4.1}$$

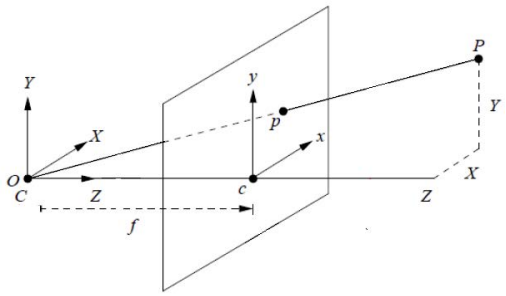

Figure 4.2: The model of camera

Then, Figure 4.3 is a point cloud, which is only a demo for the depth camera. And in order to distinguish between the two cameras, we grayed out the results of Realsense, and the colorful result belongs to percipio in the following section. Actually, the point cloud will not be processed the last three operations just what is shown in Figure 3.2. In the above section, the work range for our project about $1m$, so the point cloud can be clipped though the value of Z-axis. Then, the point clouds in Figure 4.4 are the clipping result, which will greatly reduce the amount of computation since these process removes numerous of useless points.

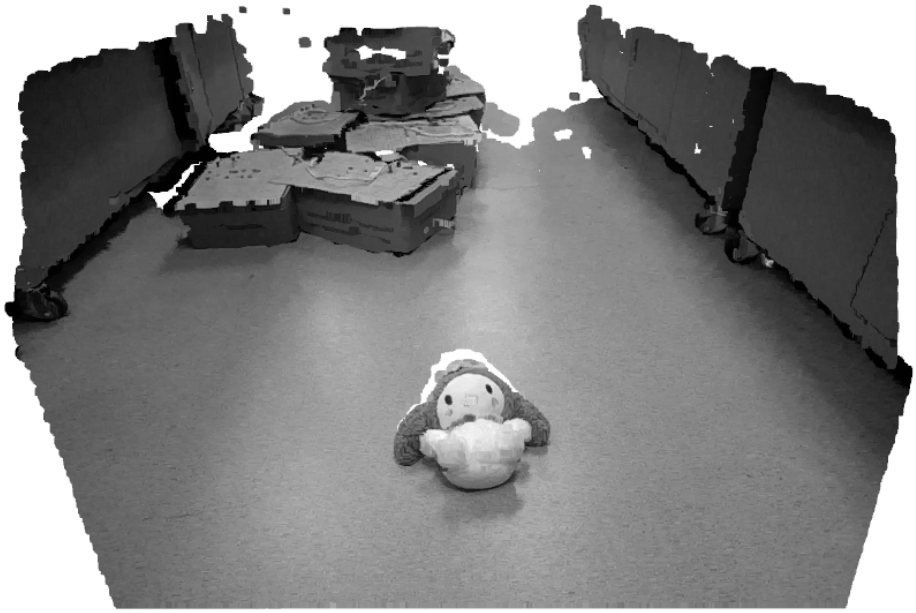

Figure 4.3: The demo of point cloud

4.1.1 The Result for Pre-procession

Figure A.3 in Appendix A is the overview for the total process of the pre-procession.

Plane Removal Figure 4.4 is the result of RANSAC algorithm and plane fitting. These images in up part is the original point cloud and in the down part, these blue points are plane which will be remove and the grey points should be saved.

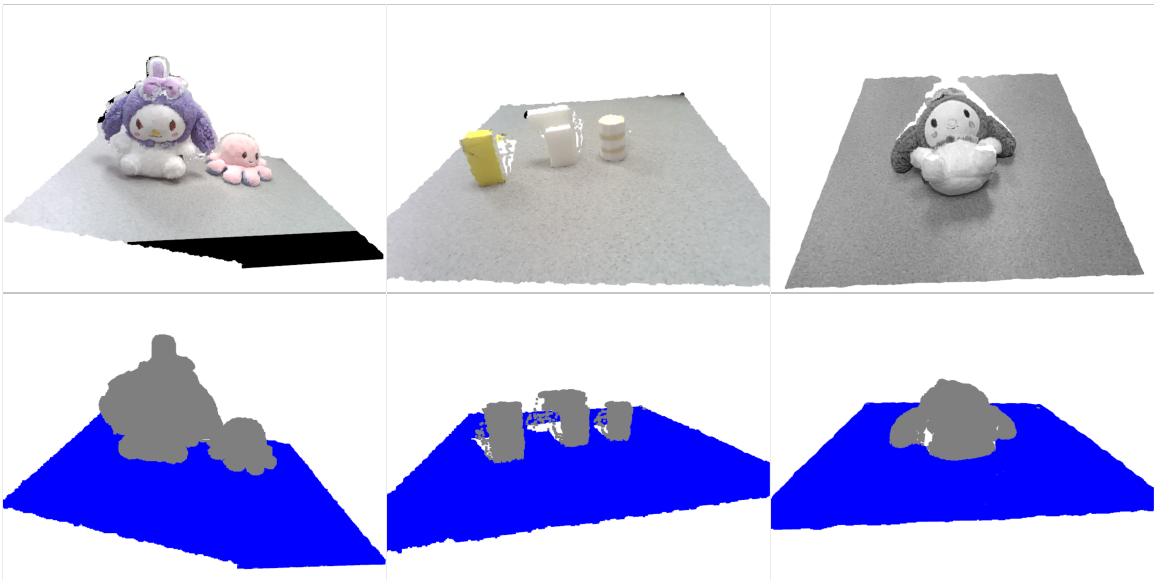

Figure 4.4: The demo for RANSAC algorithm. The up images are the original images; The down images are fitting result

Outliers Removal Figure 4.5 is the result of radius-based filtering while these images in the up part are the original point cloud and processed point cloud in the down part. It is clear that the processed results are cleaner than the original point clouds.

Objects Clustering and Regression Figure 4.6 is the result of radius-based filtering while these images in the up part are the original point cloud and the down part shows the result of clustering and regression. In the result part, the different clusters are classified by different colors while if a point is labeled as black, it means this points does not belong to any cluster. That is because of this characteristic, DBSCAN algorithm also can be used

Figure 4.5: The demo for radius-based filtering. The up images are the original images; The down images are filtering result

to filtering outliers and most time, the outliers, which do not be removed by radius-based filtering algorithm, will be cleared at this process.

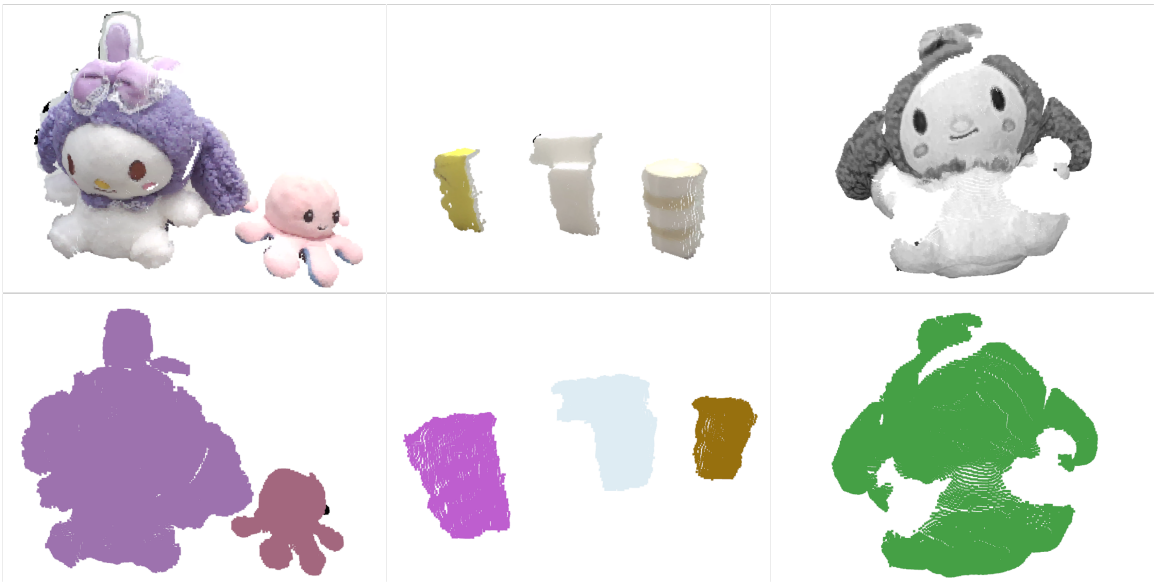

Figure 4.6: The demo for DBSCAN algorithm. The up images are the original images; The down images are clustering result

4.1.2 Comparison of Point Cloud

Although Realsense camera has a longer working range with higher frame rate, it will lose several features for the target. Figure 4.7 shown the result for two types of depth camera, and the left three images are collected by the structure light camera and the right one image is observed by the dual-lens camera. For the result collected by the dual-lens camera, it is seemed that we cannot identify the category of target, which means in our project, the structure light camera can be a better chosen although its close working distance and low frame rate. Furthermore, the noise also cannot be avoided as the result of left three images, it also lost some features while the second white object is a cylinder but it is seemed like a cuboid.

Figure 4.7: The point cloud collected by two types of depth camera

4.2 Result of Object Classification

Firstly, we evaluated the PointNet network in ModelNet40 dataset [47], which can be divided into 40 categories, and 9,843 samples used to train and 2,468 samples for testing. In the both training and test phase, we sampled uniformly 1,024 points form the face area of mesh and normalized them into a unit sphere. Furthermore, in the training phase, in order to increasing the robustness of the model, we rotated the point cloud randomly and shaken the location of each point by Gaussian noise with a mean of zero while a standard deviation of 0.02. In the end, the accuracy is 89.1%. Table 4.1 shows the accuracy for some different model used to realize 3D object classification. Although MVCNN have a higher accuracy, the number of parameters for MVCNN is 60.0 million and the number of floating-point operation is 62057 million while the number of parameters for PointNet is 3.5 million and

the number of floating-point operation is 440 million. It is seemed that if we hope to deploy a model in a machine arm, PointNet might be a better choose.

Table 4.1: The compared result for some different model used to realize 3D object classification

	input	view	accuracy avg. class
SPH [48]	mesh	-	68.2%
3DShapeNets [47]	volume	1	77.3%
VoxNet [37]	volume	12	83.0%
Subvolume [49]	volume	20	86.0%
LFD [47]	image	10	75.5%
MVCNN [36]	image	80	90.1%
PointNet	point	1	86.2%

In our dataset, there are only 7 categories, and 6,000 samples used to train and 2,000 samples used to test for each category. The accuracy is 99.9% while the accuracy for each category is shown in Table 4.2.

Table 4.2: The acuracy for each category of PointNet network. C cubodis means combination of two cuboids, c cylinder means combination of two cylinders, and C cc means combination of cuboid and cylinder

	Cube	Cuboid	Cylinder	I Beam	C cuboids	C cylinders	C cc
Accuracy	0.998	1	0.998	0.999	0.993	0.999	0.995

4.3 Result of Object Pose Estimation

Sign prediction network The accuracy for the sign of ψ prediction is 85.8% while the accuracy for each category is shown in Table 4.3. Furthermore, besides the result of cylinder and combination of two cylinders, the accuracy are more than 85% and most of them are higher than 90%. However, the accuracy for cylinder and combination of two cylinders are approaching than 0.5, since in the dataset, the rotation of Z-axis are always zero. Therefore, it is unnecessary for cylinder and combination of two cylinders to discuss the sign of ψ. Then, we also found a error in our dataset. When we labeled the rotation value for combination of two cylinders, the gripped points is set in the under cylinder but since the axises for two cylinders are not coincident, the rotation in Z-axis in dataset cannot be set

as a constant. Therefore, at this time, the labels of rotation for combination of two cylinders are wrong.

Table 4.3: The acuracy of the sign of ψ prediction for each category of PointNet network

	Cube	Cuboid	Cylinder	I Beam	C cuboids	C cylinders	C cc
Accuracy	0.934	0.868	0.513	0.862	0.947	0.659	0.960

Rotation prediction network In the experiment, we tested the result when the loss function is L1 norm and L2 norm respectively. The mean loss for every axis are $4.75°$ and $4.92°$, respectively, while the loss in each axis is shown in Table 4.4. It is clear that the L1 norm leads to a better performance, while the loss of each axis for each category is shown in Table 4.5, and these losses is calculated by the model trained by L1 norm. Furthermore, we found the loss in Z-axis for cylinder and combination of two cylinders are significantly greater than the loss for other category, which is seemed that these results can variety the conclusion in sign prediction. However, the rotation error for cube and cuboid in X and Y-axis are also significantly higher than other category. Therefore, we considered that the cube and cuboid cannot be divided into two categories while in geometry, cude and cuboid are classified as the same object in common.

Table 4.4: The mean loss of the rotation prediction for different loss function. The unit is degree

Loss function	Mean loss in every axis	Loss in X-axis	Y-axis	Z-axis
L1 norm	4.75	5.57	3.75	4.92
L2 norm	4.92	5.54	4.04	5.18

Table 4.5: The loss for each category of the rotation prediction when loss function is L1 norm. The unit is degree

	Cube	Cuboid	Cylinder	I Beam	C cuboids	C cylinders	C cc
Loss in X-axis	7.08	16.06	2.92	5.84	2.68	1.37	2.03
Loss in Y-axis	8.02	8.96	1.84	3.43	1.9	1.36	1.66
Loss in Z-axis	1.99	3.43	11.06	2.46	2.26	11.32	2.09

Translation prediction network Same with the experiment of rotation prediction, we evaluated the result when the loss function is L1 norm and L2 norm respectively, but we also tested the result whether we use the mlp to the mean of input points, while the results are shown in Table 4.6. The result shows that when the mlp to the mean of points is ignored, the model will have a better performance, while L1 norm also leads to a better result. Then, Table 4.7 expresses the loss of each axis for each category while the model is trained via L1 norm. In combination with the results of rotation prediction, we believer that in our dataset, L1 norm is more suitable, since compared with L2 norm, the L1 norm tends to produce sparse solutions, where many feature weights are exactly zero, which can simplify models and reduce overfitting, while the L1 norm is more robust to outliers and can handle situations where there are fewer samples than features [45].

Table 4.6: The loss for each category of the translation prediction for different loss function and whether use the mlp to the mean of input points. The unit is centimeter

mlp	Loss function	Mean loss	Loss in X-axis	Y-axis	Z-axis
Used	L1 norm	1.33	1.35	1.65	0.98
Used	L2 norm	1.41	1.29	1.45	1.5
Do not used	L1 norm	1.31	1.31	1.56	1.1
Do not used	L2 norm	1.31	1.37	1.43	0.99

Table 4.7: The loss for each category of the translation prediction when loss funciton is L1 norm and whether use the mlp to the mean of input points. The unit is centimeter

Use the mlp to the mean of input points

	Cube	Cuboid	Cylinder	I Beam	C cuboids	C cylinders	C cc
Loss in X-axis	1.39	1.08	1.02	1.58	1.51	1.43	1.46
Loss in Y-axis	1.61	1.42	1.07	1.69	1.86	2.13	1.72
Loss in Z-axis	0.87	0.92	0.8	0.84	1.15	1.2	1.05

Do not use the mlp to the mean of input points

	Cube	Cuboid	Cylinder	I Beam	C cuboids	C cylinders	C cc
Loss in X-axis	1.36	0.91	0.93	1.53	1.53	1.42	1.5
Loss in Y-axis	1.33	1.16	0.95	1.46	1.87	1.99	2.11
Loss in Z-axis	0.81	0.73	0.63	0.78	1.11	1.67	2.22

Chapter 5
Conclusion and Further work

In the previous work, we realized to collect point cloud though depth camera, and at the same time, we compared the advantages and disadvantages of two types of depth camera. After that, we have chosen the structure light camera to the sensor in our project since its high accuracy. Then, we designed some the pre-procession to extract the target from millions of points collected by the depth camera. In the end, after generating a dataset, we trained our network successfully. The accuracy of PointNet network for object classification is higher than 99%, the accuracy for sign of rotation in Z-axis prediction is 85.8%. Additionally, the error for translation and rotation prediction are $1.31cm$ and $4.75°$, while we verified the residual structure can improve the performance of model.

Although we added the pre-process phase in the project, the noises are inevitable in the point cloud even collecting though structure light camera, which can be an important work in the further. Another problem is the results of PointNet-like network which used to predict translation and rotation. Unexpectedly, to predict results of cube and cuboid are not good and higher than the mean error, while the predict result for these combinations are satisfying. However, the most important problem for our project is the dataset. In order to build the dataset conveniently and randomly, we used some Python scripts to generate these target and then, we just cut half of these object to simulate the characteristic that camera only can photograph only one view for an object. Our way to deal with this characteristic is too simple that it cannot simulate well.

Therefore, the priority work in the further is build a better dataset to train our network. Owing to simulate camera photograph result of target, we will use some simulated software BlseSor [50] to build our dataset. Then, we will add more method to process our point cloud, such as deep learning methods, the PointCleanNet [51]. In the end, since these architectures for PointNet-like networks are so simple, they cannot capture the local feature. Thus, we will try to design our network again based on PointNet++ network.

References

[1] Z. Wang, Y. Xu, G. Xu, J. Fu, J. Yu, and T. Gu, "Simulation and deep learning on point clouds for robot grasping," *Assembly Automation*, vol. 41, no. 2, pp. 237–250, 2021.

[2] J. Mahler, M. Matl, V. Satish, M. Danielczuk, B. DeRose, S. McKinley, and K. Goldberg, "Learning ambidextrous robot grasping policies," *Science Robotics*, vol. 4, no. 26, p. eaau4984, 2019.

[3] G. Du, K. Wang, S. Lian, and K. Zhao, "Vision-based robotic grasping from object localization, object pose estimation to grasp estimation for parallel grippers: a review," *Artificial Intelligence Review*, vol. 54, no. 3, pp. 1677–1734, 2021.

[4] W. He, C. Xue, X. Yu, Z. Li, and C. Yang, "Admittance-based controller design for physical human–robot interaction in the constrained task space," *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 4, pp. 1937–1949, 2020.

[5] X. Yu, W. He, Y. Li, C. Xue, J. Li, J. Zou, and C. Yang, "Bayesian estimation of human impedance and motion intention for human–robot collaboration," *IEEE transactions on cybernetics*, vol. 51, no. 4, pp. 1822–1834, 2019.

[6] X. Yu, W. He, H. Li, and J. Sun, "Adaptive fuzzy full-state and output-feedback control for uncertain robots with output constraint," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 11, pp. 6994–7007, 2020.

[7] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg, "Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics," *arXiv preprint arXiv:1703.09312*, 2017.

[8] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3d object detection network for autonomous driving," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 1907–1915, 2017.

[9] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun, "Deep learning for 3d point clouds: A survey," *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 12, pp. 4338–4364, 2020.

[10] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes," *arXiv preprint arXiv:1711.00199*, 2017.

[11] Y. Liu, B. Fan, S. Xiang, and C. Pan, "Relation-shape convolutional neural network for point cloud analysis," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 8895–8904, 2019.

[12] J. Wojciechowski and M. Suszynski, "Optical scanner assisted robotic assembly," *Assembly Automation*, vol. 37, no. 4, pp. 434–441, 2017.

[13] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab, "Model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes," in *Computer Vision–ACCV 2012: 11th Asian Conference on Computer Vision, Daejeon, Korea, November 5-9, 2012, Revised Selected Papers, Part I 11*, pp. 548–562, Springer, 2013.

[14] Z. Cao, Y. Sheikh, and N. K. Banerjee, "Real-time scalable 6dof pose estimation for textureless objects," in *2016 IEEE International conference on Robotics and Automation (ICRA)*, pp. 2441–2448, IEEE, 2016.

[15] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, K. Konolige, G. Bradski, and N. Navab, "Technical demonstration on model based training, detection and pose estimation of texture-less 3d objects in heavily cluttered scenes," in *Computer Vision–ECCV 2012. Workshops and Demonstrations: Florence, Italy, October 7-13, 2012, Proceedings, Part III 12*, pp. 593–596, Springer, 2012.

[16] W. Zhu, P. Wang, R. Li, and X. Nie, "Real-time 3d work-piece tracking with monocular camera based on static and dynamic model libraries," *Assembly Automation*, vol. 37, no. 2, pp. 219–229, 2017.

[17] S. Caldera, A. Rassau, and D. Chai, "Review of deep learning methods in robotic grasp detection," *Multimodal Technologies and Interaction*, vol. 2, no. 3, p. 57, 2018.

[18] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab, "Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again," in *Proceedings of the IEEE international conference on computer vision*, pp. 1521–1529, 2017.

[19] C. Li, J. Bai, and G. D. Hager, "A unified framework for multi-view multi-class object pose estimation," in *Proceedings of the european conference on computer vision (eccv)*, pp. 254–269, 2018.

[20] Y. Hu, J. Hugonot, P. Fua, and M. Salzmann, "Segmentation-driven 6d object pose estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3385–3394, 2019.

[21] C. Wang, D. Xu, Y. Zhu, R. Martín-Martín, C. Lu, L. Fei-Fei, and S. Savarese, "Densefusion: 6d object pose estimation by iterative dense fusion," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 3343–3352, 2019.

[22] H. Liang, X. Ma, S. Li, M. Görner, S. Tang, B. Fang, F. Sun, and J. Zhang, "Pointnetgpd: Detecting grasp configurations from point sets," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 3629–3635, IEEE, 2019.

[23] S. B. Niku, *Introduction to robotics: analysis, control, applications*. John Wiley & Sons, 2020.

[24] P. Allgeuer and S. Behnke, "Fused angles and the deficiencies of euler angles," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5109–5116, IEEE, 2018.

[25] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 652–660, 2017.

[26] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *Advances in neural information processing systems*, vol. 30, 2017.

[27] R. Blake and H. Wilson, "Binocular vision," *Vision research*, vol. 51, no. 7, pp. 754–770, 2011.

[28] A. K. Prasad, "Stereoscopic particle image velocimetry," *Experiments in fluids*, vol. 29, no. 2, pp. 103–116, 2000.

[29] U. Guide, "Intel® realsense tm product family d400 series calibration tools," *no. July*, 2020.

[30] J. Geng, "Structured-light 3d surface imaging: a tutorial," *Advances in Optics and Photonics*, vol. 3, no. 2, pp. 128–160, 2011.

[31] S. Zhang and P. S. Huang, "Novel method for structured light system calibration," *Optical Engineering*, vol. 45, no. 8, pp. 083601–083601, 2006.

[32] Q.-Y. Zhou, J. Park, and V. Koltun, "Open3D: A modern library for 3D data processing," *arXiv:1801.09847*, 2018.

[33] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[34] M. Kazhdan, M. Bolitho, and H. Hoppe, "Poisson surface reconstruction," in *Proceedings of the fourth Eurographics symposium on Geometry processing*, vol. 7, p. 0, 2006.

[35] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise.," in *kdd*, vol. 96, pp. 226–231, 1996.

[36] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-view convolutional neural networks for 3d shape recognition," in *Proceedings of the IEEE international conference on computer vision*, pp. 945–953, 2015.

[37] D. Maturana and S. Scherer, "Voxnet: A 3d convolutional neural network for real-time object recognition," in *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 922–928, IEEE, 2015.

[38] X. Yan, "Pointnet/pointnet++ pytorch," 2019.

[39] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," *Advances in neural information processing systems*, vol. 32, 2019.

[40] M. Jaderberg, K. Simonyan, A. Zisserman, *et al.*, "Spatial transformer networks," *Advances in neural information processing systems*, vol. 28, 2015.

[41] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[42] A. R. Webb, *Statistical pattern recognition*. John Wiley & Sons, 2003.

[43] S. Marsland, *Machine learning: an algorithmic perspective*. CRC press, 2015.

[44] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[45] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*, vol. 4. Springer, 2006.

[46] J. Mahler, F. T. Pokorny, B. Hou, M. Roderick, M. Laskey, M. Aubry, K. Kohlhoff, T. Kröger, J. Kuffner, and K. Goldberg, "Dex-net 1.0: A cloud-based network of 3d objects for robust grasp planning using a multi-armed bandit model with correlated rewards," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1957–1964, IEEE, 2016.

[47] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1912–1920, 2015.

[48] M. Kazhdan, T. Funkhouser, and S. Rusinkiewicz, "Rotation invariant spherical harmonic representation of 3 d shape descriptors," in *Symposium on geometry processing*, vol. 6, pp. 156–164, 2003.

[49] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. J. Guibas, "Volumetric and multi-view cnns for object classification on 3d data," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5648–5656, 2016.

[50] M. Gschwandtner, R. Kwitt, A. Uhl, and W. Pree, "Blensor: Blender sensor simulation toolbox," in *Advances in Visual Computing: 7th International Symposium, ISVC 2011, Las Vegas, NV, USA, September 26-28, 2011. Proceedings, Part II 7*, pp. 199–208, Springer, 2011.

[51] M.-J. Rakotosaona, V. La Barbera, P. Guerrero, N. J. Mitra, and M. Ovsjanikov, "Pointcleannet: Learning to denoise and remove outliers from dense point clouds," in *Computer graphics forum*, vol. 39, pp. 185–203, Wiley Online Library, 2020.

Appendix A Some Figures

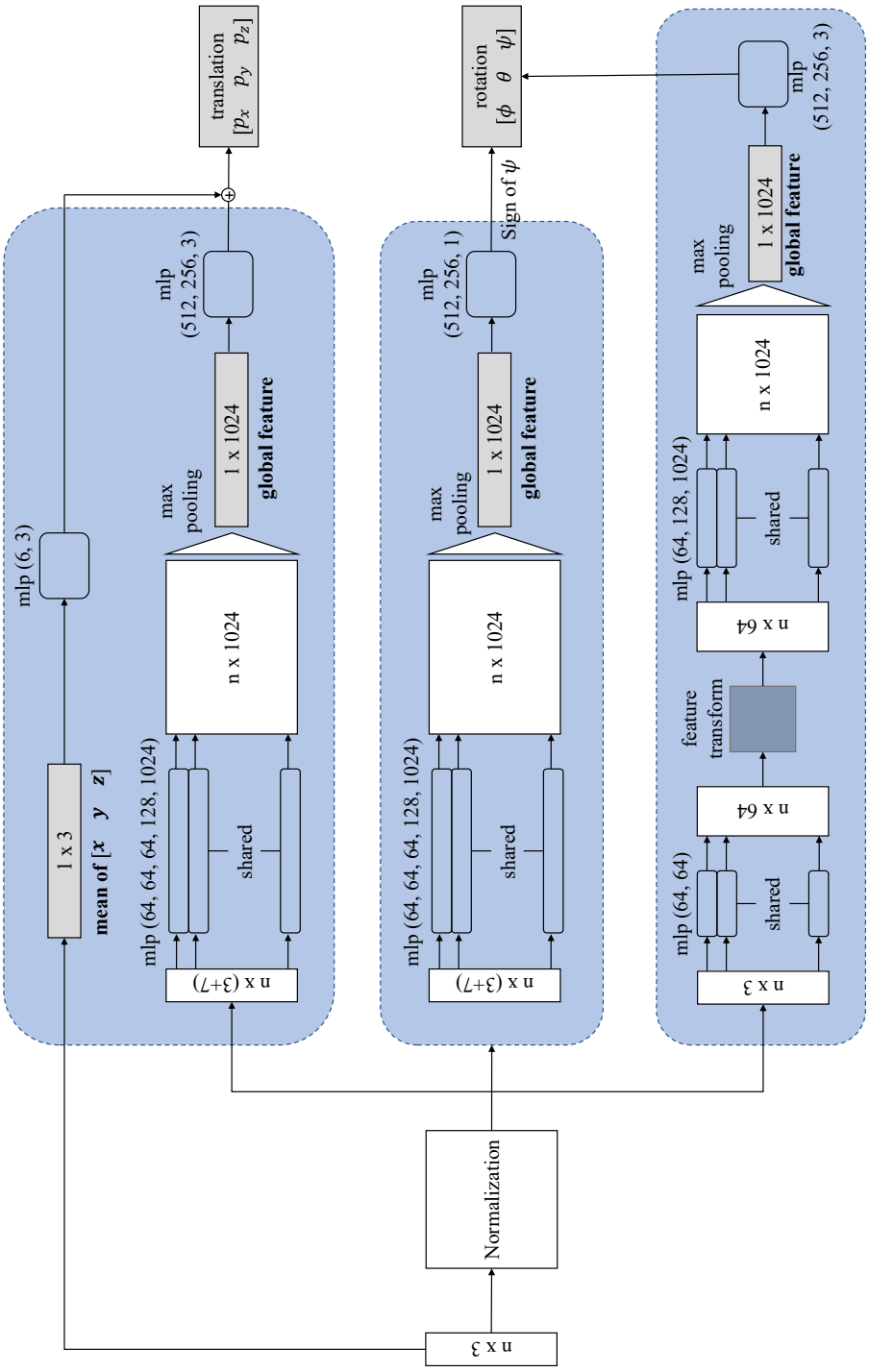

Figure A.1: The illustration of the pose estimation unit

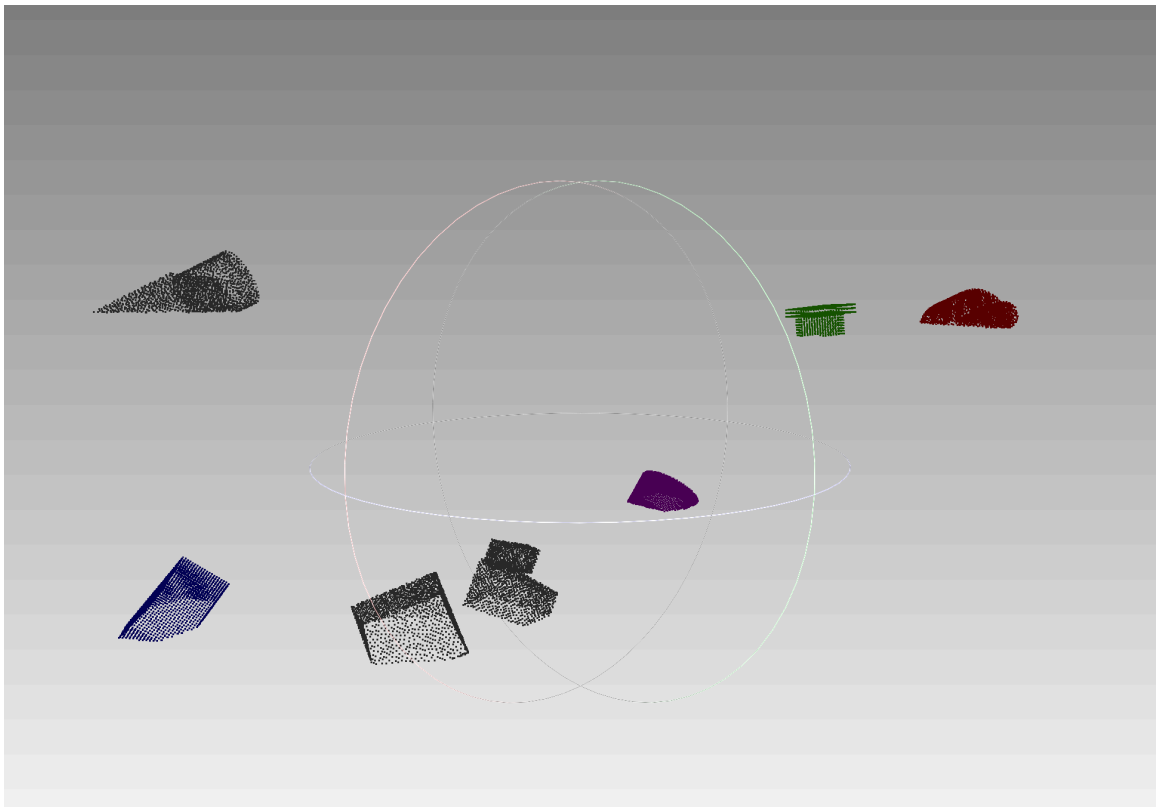

Figure A.2: The demo for the object in different position and pose

Figure A.3: The overview for the total process of the result of pre-procession

Appendix B The Pseudocode for Point Cloud Pre-procession

B.1 The Pseudocode of RANSAC Algorithm

Algorithm 1: RANSAC algorithm

```
Input: Data: a set of observed data points
Model: a model to be fitted to the data points
n: the minimum number of data points required to fit the model
k: the maximum number of iterations allowed in the algorithm
t: a threshold value for determining when a data point fits a model
d: the number of close data points required to assert that a model fits well to data
Output: BestModel: the model with the most inliers
Initialize BestModel to null;
for i = 1 to k do
    randomly select n data points from the data;
    Fit a model to the selected data points;
    for each data point not in the selected set do
        if the point fits the model with an error less than t then
            add the point to a set of inliers;
        end
    end
    if the number of inliers exceeds d then
        re-estimate the model using all inliers;
        if the new model fits better than the current BestModel then
            update BestModel with the new model;
        end
    end
end
return BestModel
```

B.2 The Pseudocode of Radius-Based Filtering

Algorithm 2: Radius-based filtering algorithm

Input: points: a set of observed data points
radius: the maximum distance between two points in order to be considered nearby
Output: filteredPoints: a set of filtered data points
filteredPoints = empty list of points;
for *each point* $p1$ *in points* **do**
    hasNearbyPoint = False;
    for *each point* $p2$ *in points* **do**
        if $p1 \neq p2$ *and* $distance(p1, p2) \leq radius$ **then**
            hasNearbyPoint = True;
            break;
        end
    end
    if *not hasNearbyPoint* **then**
        filteredPoints.add($p1$);
    end
end
return *filteredPoints*;

B.3 The Pseudocode of DBSCAN Filtering

Algorithm 3: DBSCAN algorithm

```
Input: Data: a set of observed data points
ϵ: the radius of the neighborhood around each data point
MinPts: the minimum number of data points required to form a dense region
Output: Clusters: a set of clusters
Initialize all points as unvisited;
Initialize an empty set of clusters;
for each unvisited point p do
    Mark p as visited;
    Find all points within the ϵ-neighborhood of p;
    if the number of points in the neighborhood is less than MinPts then
        Mark p as noise;
    end
    else
        Create a new cluster C, and add p to C;
        for each point q in the neighborhood do
            if q is unvisited then
                Mark q as visited;
                Find all points within the ϵ-neighborhood of q;
                if the number of points in the neighborhood is greater than or
                  equal to MinPts then
                    Add all points in the neighborhood to C;
                end
            end
            if q is not yet a member of any cluster then
                Add q to C;
            end
        end
        Add C to the set of clusters;
    end
end
return Clusters
```

Appendix C Detailed Data for The Dataset

The unit for Table C.1 is centimeter. The order of size of cube in combination is: length, width, and height. The order of size of cylinder in combination is: radius and height.

Table C.1: The detail size for each workpiece

Cube	Cuboid		
Size	Length	Width	Height
5-10	3-5	6-10	4-8

I Beam			
Width	Height	Web	Flange
8-10	8-10	1-3	1-3

Cylinder		Combination of two cuboids	
Radius	Height	Size of cuboid 1	Size of cuboid 2
		(3,8,5)	(8,10,8)
2-4	4-8	(4,6,8)	(7,9,9)
		(2,7,3)	(10,7,9)

Combination of two cylinders		Combination of cuboid and cylinder	
Size of cylinder 1	Size of cylinder 2	Size of cylinder	Size of cuboid
(2,7)	(4,8)	(2,7)	(8,10,8)
(3,6)	(5,10)	(3,6)	(7,9,9)

The unit in Table C.2 is degree. ϕ means the rotation range on the X-axis, θ means the rotation range on the Y-axis, and ψ means the rotation range on the Z-axis

Table C.2: The rotation range for each workpiece on each axis

	ϕ	θ	ψ
Cube	-45 - 45	-45 - 45	-45 - 45
Cuboid	-90 - 90	-45 - 45	-45 - 45
Cylinder	0	-45 - 45	-45 - 45
I Beam	-90 - 90	-45 - 45	-45 - 45
Combination of two cuboids	-90 - 90	-45 - 45	-45 - 45
Combination of two cylinders	-90 - 90	-45 - 45	-45 - 45
Combination of cuboid and cylinder	-90 - 90	-45 - 45	-45 - 45