Linköping University | Department of Computer and Information Science
Master's thesis, 30 ECTS | Statistics and Machine Learning
2023 | LIU-IDA/STAT-A–23/020–SE

Point Clouds in the application of Bin Picking

Abhijeet Anand

Supervisor : Amirhossein Ahmadian
Examiner : Jose M Peña

External supervisor : Christoffer Malmgren

Linköpings universitet
SE–581 83 Linköping
+46 13 28 10 00 , www.liu.se

Abstract

Automatic bin picking is a well-known problem in industrial automation and computer vision, where a robot picks an object from a bin and places it somewhere else. There is continuous ongoing research for many years to improve the contemporary solution. With camera technology advancing rapidly and available fast computation resources, solving this problem with deep learning has become a current interest for several researchers. This thesis intends to leverage the current state-of-the-art deep learning based methods of 3D instance segmentation and point cloud registration and combine them to improve the bin-picking solution by improving the performance and make them robust.

The problem of bin picking becomes complex when the bin contains identical objects with heavy occlusion. To solve this problem, a 3D instance segmentation is performed with Fast Point Cloud Clustering (FPCC) method to detect and locate the objects in the bin. Further, an extraction strategy is proposed to choose one predicted instance at a time. In the next step, a point cloud registration technique is implemented based on PointNetLK method to estimate the pose of the selected object from the bin.

The above implementation is trained, tested, and evaluated on synthetically generated datasets. The synthetic dataset also contains several noisy point clouds to imitate a real situation. The real data captured at the company 'SICK IVP' is also tested with the implemented model.

It is observed that the 3D instance segmentation can detect and locate the objects available in the bin. In a noisy environment, the performance degrades as the noise level increase. However, the decrease in the performance is found to be not so significant. Point cloud registration is observed to register best with the full point cloud of the object, when compared to point cloud with missing points.

Acknowledgments

First, I would like to thank Linköping University to provide me with this great opportunity to pursue a master's in Statistics and Machine Learning.

I would like to thank my university supervisor Amirhossein Ahmadian for the continuous supervision, thorough guidance, and illuminating regular discussions. I highly appreciate all the feedbacks and the ideas to improve at every stage during this research.

I would like to convey my sincere gratitude to SICK IVP for giving me this opportunity to work on this thesis. I am extremely grateful to my external supervisor Christoffer Malmgren for his immense support, mentorship, and invaluable insights throughout my research journey. His expertise, crucial feedback, and brainstorming discussions have been truly instrumental in shaping my work.

Further, I would like to thank Professor Jose M Peña for examining my thesis. I also thank my opponent Olof Josefsson for his valuable feedback. Their crucial comments have improved the overall quality of the research and the report. I would also like to thank Professor Oleg Sysoev for being the thesis course coordinator.

A special mention to Chayan, Elisio, Akash, Carol, Dinuke, Theo, Patrick, Hussnain, Nikhil, and Jaskirat for being a great support system here in Sweden and motivating me to accomplish more.

Lastly, I would like to thank my family for always inspiring me and being my strongest support, without which nothing of this could be possible.

Contents

List of Figures

List of Tables

1 Introduction

Robotic bin picking is a complex process in which a robot detects the object(s) from a container, picks it up, and places it somewhere else. Vision bin-picking robots are equipped with a camera(s) to detect the object and a gripper, e.g. a suction or parallel gripper to hold the object. With recent advancements in the industries, robotic bin picking is highly demanded as it becomes extremely crucial in the factory automation process. The application of robotic bin picking includes sorting items, picking and placing, packaging, or assembling. Automating bin picking is a long-known and core problem in industrial automation [37, 50, 24].

Robotic bin picking can improve accuracy and reduce labor costs. This can also be efficient in reducing the risk of injuries to workers and damage to products while picking. Fig. 1.1 shows a sample robot bin-picking setup.

Figure 1.1: A sample robot for bin picking applications

It appears to be a very simple task for humans to pick objects from an extremely cluttered environment. However, it is a highly challenging and complex exercise for robots. This composite task can be divided into the detection of individual instances, selecting one instance at a time, and estimating the pose of the selected object for the robot to pick from the best angle. Solving this problem with deep learning is an ongoing interest with several continuous enhancements [11, 18, 27, 31]. Current camera technologies are so accurate that they can capture the height of the object with millimeter precision. This growth in camera technology can generate 2.5D images from a scene. 2.5D image can be defined as the generation of an

actual 3D environment from a 2D retinal perception. In general, it is a viewer-centric 3D view, where the portion of the scene is missing which can't be seen with the placed camera. With the generated 2.5D image through the camera, a point cloud can be constructed which allows a deeper understanding of an image.

A point cloud is an accumulation of many points in 3D space. This cluster of points may represent a 3D shape or object. Each point in a point cloud has the Cartesian coordinates (X, Y, Z). Features like color or intensity might also be part of the cloud. Collecting point clouds from a 3D object has become very convenient due to rapid advancements in 3D measurement mechanisms. Processing on 3D Point Clouds has been a common start for many applications recently. Today, 3D point clouds are widely used in projects such as autonomous driving [26], automatic bin picking [37, 24, 50], pose estimations [25, 10, 21], and others. When several objects of the same class are present in the bin it becomes difficult for robots to distinguish between the objects. This problem can be tackled by training a neural network to perform 3D instance segmentation of point clouds [9, 15].

To find an unknown transformation between point clouds, point cloud registration is performed. In computer vision and robotics, point cloud registration also known as scan matching is a method to determine a spatial transformation (mainly, rotation, translation, and scaling) between two point clouds (source and target) which finally aligns these two point clouds. It is often researched with motion estimation or Simultaneous Localization and Mapping (SLAM) applications. However, it can also be useful for object detection and pose estimation. Industrial applications for point cloud registration can be random bin picking, i.e., picking known randomly oriented objects from a container.

1.1 Motivation

SICK IVP Linköping[1] is a subsidiary of SICK AG that specializes in developing industrial image processing systems. This division of SICK provides solutions for various industries, which include automotive, electronics, and packaging. Some of the products offered by SICK IVP Linköping are 2D/3D sensors, vision cameras, and smart cameras. The vision solutions provided by SICK IVP Linköping have many applications such as quality control, defect detection, robot guidance, and others. These solutions help companies improve their productivity, reduce waste, and enhance their overall competitiveness. Fig. 1.2 is the Ranger3, a 3D machine vision camera developed by SICK, which is capable of extracting the true 3D shape of an object with high precision. SICK IVP Linköping is also involved in the development of robotic bin-picking solutions. The company has a wide range of 3D vision sensors which can be used for bin-picking applications. These sensors leverage laser and camera technology to detect the position, orientation, and shape of objects in a bin, which makes it easier for a robot to pick them up.

Figure 1.2: Ranger3 - 3D machine vision camera

Solving the problem of robotic bin picking with the current advancement in deep-learning is the driving motivation behind this project. While leveraging contemporary solutions, the

[1] https://www.sick.com/se/en

solution for bin picking can be optimized. The complications associated with this task are mentioned below:

- Distinguishing individual instances of an object from a bin containing randomly piled identical objects is a highly challenging problem.
- 2.5D point clouds have missing points from the actual scene. Point cloud registration in this case of partial point cloud is a difficult task.
- During the real data collection, several noises like offset due to other light sources or reflection of objects can also get captured in the point cloud.
- Due to similar features in the symmetric objects, the complexity in point cloud registration increases.

1.2 Problem Statement

One of the main challenges in robotic bin picking is the problem of dealing with identical objects present in a cluttered environment. Within the scope of this thesis, a bin contains around 20 identical instances of an object, and a 2.5D point cloud of this image is available. The object is known and a computer-aided design (CAD) model of the object is present. The problem addressed in this thesis work is to detect individual instances of the object from the cluttered bin. From the predicted instances in the bin, chose one instance and estimate the pose of the chosen instance with respect to the CAD model of the object. The pose of the object involves rotation and translation in 3D space.

1.3 Research Questions

The main objective of this thesis is to research the existing enhancements in object detection, and pose estimation and implement these concepts in the field of bin picking. This thesis aims to answer the below research questions:

1. How to choose one instance from the predicted instances after 3D instance segmentation?
2. What is the impact on the performance of 3D instance segmentation with the added noise on synthetic data?
3. What is the change in performance when a partial point cloud and a full point cloud are registered with the CAD model of the objects?

1.4 Related Work

In earlier decades, robotics was introduced at large to production facilities. Due to the new advancements, robots are much more affordable now than in previous years and they share a common workplace as humans. With the human labor cost increasing in production, the inclination towards enhancing robotics is very demanding.

Intense research is ongoing by several companies including Amazon to enhance the current system. One of the most popular ones is Amazon Picking Challenge[8]. The challenge is about creating a robot that can pick a certain number of different objects from a shelf and put it in a container in a specified amount of time. The challenge also restricts the robot to being completely autonomous with zero human interaction during the process.

With the recent rise in deep learning-based applications, pose estimation of objects for bin picking is heavily researched [11, 18, 27, 31].

1.4.1 Robotic Bin Picking

Robotic bin picking is a very known, and challenging problem in industrial automation. With the constant improvement in industrial automation technologies, bin-picking systems have also been improvised over time resulting in better productivity and less human intervention [37, 50, 24]. With the increase in high computational power and 3D sensor technologies, deep learning-based solutions are the current interest of many researchers [20].

One of the many approaches is to train a neural network for planar objects [20]. This approach has shown great accuracy and flexibility for planar objects. This method can handle large volumes of objects in the bin and it is also efficient in terms of computation time. However, the disadvantage of this system is the limitation of the objects to be planar. Another approach is to train a support vector machine (SVM) to classify and locate the connectors in the bin [42]. This approach has shown a high accuracy and low hardware dependency. However, this network is limited to specific or similar objects.

One significant approach is to divide the task into two steps: Localize and registration [14]. This method uses advanced Density-based spatial clustering of applications with noise(DBSCAN) to cluster and localize based on geometric features of the input point cloud. Further, a point cloud registration is performed with the help of principal component analysis(PCA) and iterative closest point(ICP) for coarse and fine registration respectively. However, the registration technology used in this system is primitive.

1.4.2 Feature Extraction from Point Clouds

Point clouds are the simplest representation of a 2D or 3D object. Acquiring the point clouds has become one of the first steps in many applications. These acquired point clouds are often processed by bypassing expensive reconstruction techniques or de-noising filters. Applications like self-driving cars, robotics, and shape synthesis and modeling have taken advantage of point cloud processing. [23, 36, 13]. These modern applications need a prominent level of point cloud processing. During the feature extraction process, a feature vector is generated.

A feature vector can be explained as a numerical representation that contains main characteristics of the points. The representation of point clouds as feature vectors makes it possible use machine learning algorithms such as classification, segmentation or registration. Feature vector can encode details like geometric properties (corners, edges), semantic hues, or surface normal. These features can be local or global features. Local features contain features for every point of the point cloud while global features contain features for the entire point cloud. The choice of these features is task dependent. Local shape detail might be more interesting for task like object detection, while pairwise geometric information between points is more attractive for point cloud registration.

The advancement in the deep-learning for image processing has escalated the data-driven approach for feature extraction of point clouds. The point cloud processing through deep neural networks is outperforming the conventional approaches. [3]. Introducing deep learning to point clouds was not elementary; as standard deep neural network frameworks demand regular structure in the input data. Point clouds instead are considered to be unstructured due to their continuous distribution of positions in the space with any permutation. To handle this unevenness, state-of-the-art(SOTA) deep networks are designed can manipulate the raw point clouds. PointNet is a deep neural network architecture, which is capable of processing the unordered point clouds without requiring any pre-segmentation or voxelization. This SOTA architecture can extract both global features for the entire point cloud and local features for individual points too [32]. PointNet was extended to PointNet++ which exploits a hierarchical feature extraction process to store multi-scale information of the point cloud. The working principle behind this architecture involved dividing the point cloud into a hierarchy of nested partitions, and each partition individual with the help of PointNet architecture [33]. Another SOTA is PointCNN, a neural network architecture that leverages the

concept of convolutional neural networks and applies it to the point clouds. A direct convolution of kernels against the features of the points will have some consequences including desertion of shape information and variance to point ordering. This problem was addressed by learning χ-transformation on the point clouds, which helps in weighting the input features of the points and permutating the points to a latent and canonical order [22].

KPConv is a kernel-based point convolutional neural network architecture that is efficient to extract point features from point clouds in both local and multi-scale manner. Unlike other point convolution neural network architectures which utilize the fixed kernels, to adapt to the geometric characteristics of the point cloud, a learned kernel function is used. The usage of KPConv can be extended to deformable convolutions where it can learn to consider kernel points to local geometry [41].

1.4.3 3D Instance Segmentation

Instance segmentation can be defined as a process of identifying, segmenting, and classifying every individual object in the image. 3D segmentation is a computer vision technique, where a 3D point cloud is divided into meaningful parts or regions. 3D Instance segmentation is an ongoing interest of many researchers [44, 7].

The 3D U-net is a neural network architecture, which is specifically designed for dense volume segmentations concerning medical images [7]. This network is an extension of 2D U-net, which is very commonly used in biomedical imaging. The 3D U-net architecture has encoder-decoder blocks, where the encoder block extracts the features from the given input point cloud and the decoder block creates the segmentation map. A novel loss function called soft dice loss is also introduced to measure the similarity between predicted segmentation and the ground truth. Overall, the 3D U-net presents a great approach to address the challenge of dense volumetric segmentation. Along with being computationally expensive, 3D u-net also rely on manual annotations of the input data, which in turn is very time-consuming and expensive.

Another approach to perform 3D Instance segmentation is to add an associate embedding module, which combines information from both semantic and instance segmentation to perform better [44]. The method also has a multi-scale fusion network that uses contextual knowledge to increase the accuracy of the network. It has shown some benchmark results on known datasets. Overall, it is a robust technique that can be implemented in real-time use cases. It does have some limitations too. Due to the presence of several networks and modules which demands a good amount of computational resources. Another limitation is the need for annotated data for both instance and semantic segmentation.

1.4.4 Point Cloud Registration

Point cloud registration is the process of aligning two or more point clouds to create a single, unified representation of a scene or object. Researchers are constantly developing new techniques and algorithms to improve the accuracy and efficiency of point cloud registration [6, 29, 46, 45, 48]. Deep Global Registration, is a deep learning-based approach that uses deep neural networks to forecast the correspondences between the two point clouds [6]. This method is used to perform the global point registration. A large amount of training data is needed in this method to attain good performance.

Iterative closest point (ICP), which is a transformation optimization algorithm has been a point of interest to many researchers to fine-tune the existing registration methods. One of the research is a probabilistic point cloud registration (PPCR) for 3D point clouds[1]. PPCR is an enhanced ICP, which involves a probabilistic model to ameliorate the robustness of the model to perform better against noise and outliers. This is an iterative algorithm that continuously tries to reduce the error and improve the result until a defined convergence criterion is reached. This method may suffer the local minima and require multiple initializations.

3DRegNet is a deep neural network architecture to perform 3D point cloud registration[29]. The multi-scale feature extraction to record both local and global features of the point clouds makes it a state-of-the-art method. It also uses a correlation layer that estimates the transformation among the two point clouds. 3DRegNet is a little sensitive to the quality and completeness of the source point cloud and doesn't perform very well in a noisy and cluttered case.

1.5 Report Structure

Chapter 2 of this report describes the theoretical background needed to implement the methods. The next chapter presents a detailed description of data, data generation, and collection techniques. Chapter 4 explains the methodologies used to perform the 3D instance segmentation and registration of the point clouds. Experiments and results are described in Chapter 5. This chapter also includes the evaluation of the results obtained. A discussion of the obtained results and limitations of the methods is performed in Chapter 6. The conclusion of this thesis is done under Chapter 7.

2 Theoretical Background

2.1 3D Instance Segmentation

3D Instance Segmentation can be defined as a task to determine a unique label for all the individual objects present in the input point cloud. This task includes identifying the individual object and locating it in the point cloud [4].

Let P be a point cloud with N number of points such that $p_1, p_2, ..., p_N$ represent individual points. Let A be a set of points which is a subset of P representing one instance of the object in the point cloud. The task of 3D instance segmentation is to determine a function $P \rightarrow A$, which is capable of mapping the point P_i to the corresponding instance point A_j.

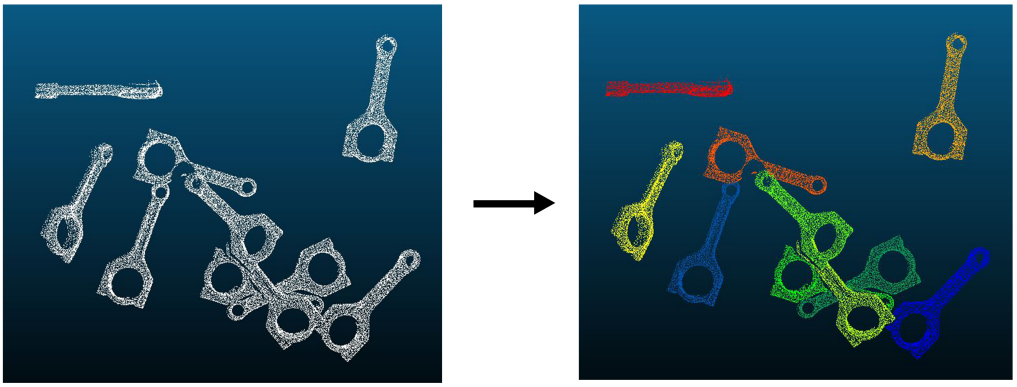


Figure 2.1: An ideal case of 3D Instance Segmentation

An ideal case of 3D Instance Segmentation of a conrod plastic point cloud is demonstrated in Fig. 2.1. Before the segmentation, all the points in the point cloud are labeled to one class which can be verified by one color (white). While we observe there are multiple colors in the point cloud after the 3D instance segmentation is performed. Here one instance of the object is identified by one color in the point cloud. Fig. 2.1 represents the ideal segmentation because every actual instance is identified by a unique color.

2.2 Dynamic Graph Convolutional Neural Network (DGCNN)

DGCNN is a deep neural network architecture that is designed to work with point cloud data[49]. Point cloud data in general is unstructured 3D data that contains several points in the 3D space. The recent accomplishments of convolutional neural networks (CNNs) for image data, indicated the importance of using this concept for point cloud data. DGCNN works on the principle of creating a dynamic graph representation of a point cloud. In this graph, each point is considered as a node and the distance between the points defines the edge. This helps in capturing the local geometric details of the point cloud.

DGCNN comprises several stages, including edge construction, feature extraction, edge convolution, and pooling. During the edge construction, the nearest neighbors are deduced for each point of the input point cloud. This defines the edges between the point and its neighbors. With the edge and point as nodes a dynamic graph is developed. The dynamic graph is represented with an adjacency matrix A, where $A_{ij} = 1$ if point j is one of the nearest neighbors of point i, otherwise 0.

The point cloud data, together with the developed dynamic graph is fed into a neural network. To extract local features from each point, DGCNN uses shared multi-layer perceptrons (MLPs). The extracted features are used in the edge convolution layer (EdgeConv), where the dynamic graph is then updated. The EdgeConv layer plays a key role in the DGCNN architecture. It is introduced to capture the local geometric details of the point cloud. EdgeConv creates edge features between points which explains the relationship between a point and its neighbors. This is achieved by generating a local neighborhood graph and implementing convolution on the edges with its neighboring points. This graph is not static but rather dynamically updated after processing through every edgeConv layer in the network. This signifies that the nearest neighbors are updated in every layer.

Given a 3D point cloud P with N points: $P = p_1, p_2, ..., p_N$, the nearest neighbor graph for each point is computed. This graph contains a set of edges $E(i, j)$, for the point i to all of its neighbors. During edge convolution edge features are computed. The edge features are defined by: $e_{ij} = h_\theta(p_i, p_j)$, where h_θ is a non-linear function such as rectified linear units(ReLU) and θ signifies the set of trainable parameters. During the edgeConv layer operation, first, a linear transformation is performed on the node features followed by aggregation of features for each point i.

To generate a fixed-length global feature vector, DGCNN applies max pooling on the node features. This global feature vector summarizes the whole point cloud and represents the overall structure of the input point cloud.

The developed feature vector can further be utilized in tasks such as classification or segmentation. DGCNN is a widely adapted architecture for various tasks including 3D reconstruction[39], and 3D Instance segmentation [47].

2.3 Minimum Bounding Box(MBB) Algorithm

A bounding box is a geometric structure that encloses an object or multiple objects in 2D or 3D space. The bounding box is defined by a cuboidal shape for 3D point clouds and is recognized by position and orientation. MBB is an algorithm that is used to find the smallest bounding box in the 3D space that can accommodate all the points of the point cloud [28]. This algorithm first calculates the covariance matrix of the point cloud. The covariance matrix is a 3x3 matrix, which explains the spread and orientation of the point cloud. The equation below calculates the covariance matrix.

$$C = \frac{1}{N} \sum_{i=1}^{N} (x_i - \mu)(x_i - \mu)^T$$

In the equation above, N is the total number of points in the point cloud, x_i is the i_{th} point of the point cloud, and μ is the centroid of the cloud defined by

$$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i$$

From the calculated covariance matrix, eigenvectors of C are computed using eigenvalue decomposition. $C * v_i = \lambda_i * v_i$, where v_i and λ_i are the i_{th} eigenvector and eigenvalue respectively. For the corresponding eigenvalues, eigenvectors are sorted in a descending manner. The axes of the bounding box are defined by these eigenvectors. In the next step, minimum and maximum values along each axes are calculated by projecting the points onto each axes. The equation below shows the minimum and maximum value calculation along the x-axis.

$$x_{min} = \min_{i=1}^{N} \mathbf{x}_{i,x}$$

$$x_{max} = \max_{i=1}^{N} \mathbf{x}_{i,x}$$

The MBB needs eight corner points to be defined. With the minimum and maximum values from each axes, the eight corners of the bounding box are generated in the following way:

$$P_1 = (x_{min}, y_{min}, z_{min})$$
$$P_2 = (x_{min}, y_{min}, z_{max})$$
$$P_3 = (x_{min}, y_{max}, z_{min})$$
$$...$$
$$P_8 = (x_{max}, y_{max}, z_{max})$$

Using this algorithm, the length along the axes, the center of the bounding box, and the rotation of the bounding box can be extracted. It can be used in several computer vision and robotics applications like detecting collisions or tracking objects in a scene. The advantage of using the minimum bounding box algorithm is it gives a tight fit around the points. However, the rotation generated here is highly sensitive to noise and outliers. Fig. 2.2 shows the bounding box and minimum bounding box around the conrod plastic object.

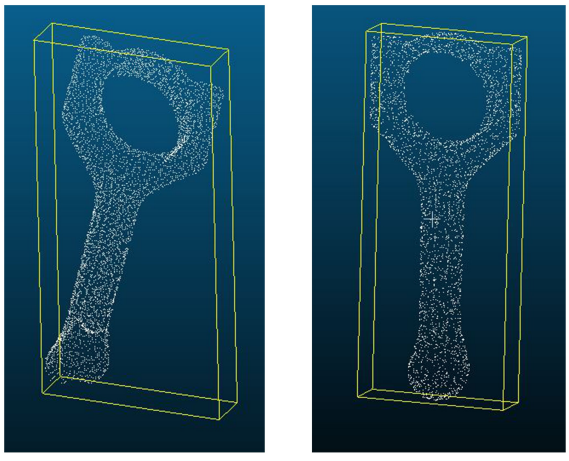


Figure 2.2: Bounding Box and minimum bounding box for Conrod Plastic

2.4 Non-Maximum Suppression (NMS)

In object detection tasks, there are several occasions when the detection of the same object overlaps with each other. To eliminate these redundant and overlapping detections, the Non-maximum Suppression technique is used. NMS is applied to generate a set of non-overlapping bounding boxes from the output of the 3D object detection [16]. The probability of the box containing the object of interest is defined by the confidence score and is assigned to all bounding box during the object detection task. In NMS algorithm, all the bounding boxes are sorted by the confidence score in descending order. The bounding box with the highest confidence score is selected and intersection over union (IoU) is calculated between the selected and all remaining bounding boxes. A threshold of IoU is set before. If any bounding box is found with IoU greater than the given IoU threshold, it signifies that both the bounding box indicates same object. Thus, it is removed. The selected bounding box is removed and saved in a separate list. This process is carried out in a loop until no bounding box remains. NMS is highly adapted in state-of-the-art frameworks like Fast R-CNN [12], YOLO [34], FPCC [47]. A simple demonstration of NMS is shown in Fig. 2.3.

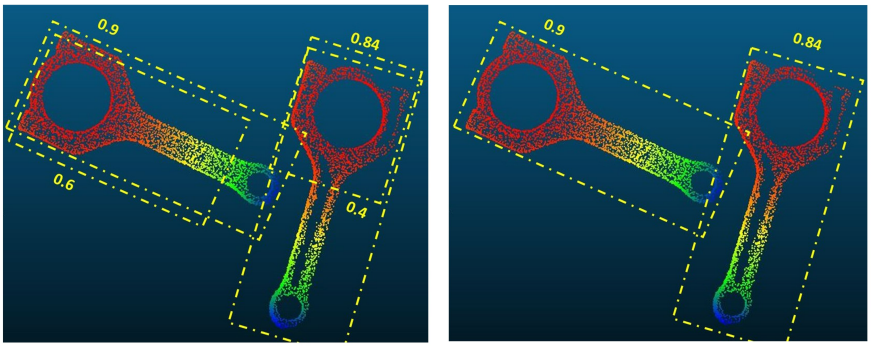


Figure 2.3: An example of Non-Maximum Suppression

In the Fig. 2.3, we observe that two conrod plastic objects. Each of the conrod plastic object have two predicted instances. After applying the NMS algorithm, only two instances remain, as other instances have lower confidence score and bigger overlap.

2.5 Smooth L1 Loss

Smooth L1 is a very known loss function in object detection, which is a variant of L1 loss. L1 loss is very commonly known as Mean Absolute Error (MAE) [12]. The key difference between the L1 loss and the smooth L1 loss is the introduction of a quadratic term in smooth L1 loss. This is applied when the absolute difference between the predicted value and the target value is smaller than 1. This helps in getting a smooth transition around zero, that reduces the impact of outliers. Thus, the smooth L1 loss has less sensitivity to the outliers when compared to the L1 loss. The smooth L1 can be defined as:

$$\text{Smooth}_{\text{L1}}(\text{x}) = \begin{cases} 0.5 * x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases}$$

where x is the difference between ground truth and predicted value.The smooth L1 loss is less sensitive to outliers because the squared term in the loss function gives a smooth transition between the L1 loss and the L2 loss, commonly known as mean squared error(MSE), which is more robust to outliers. A comparison between L1 loss, L2 loss and smooth L1 loss function is shown in Fig. 2.4.

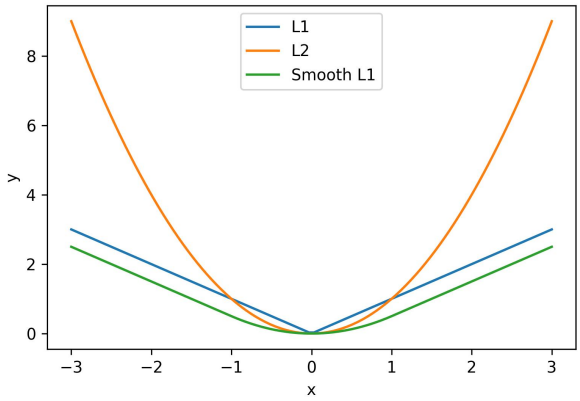


Figure 2.4: L1, L2 and Smooth L1 Loss curve

The smooth L1 loss function is used in the calculation of the loss in the FPCC-net [47].

2.6 Point Cloud Registration

Point cloud registration, also known as point-set registration is a well-known task in computer vision and robotics. It can be defined as a process of finding a spatial transformation (scaling, rotation, and translation) that aligns one point cloud to another [17]. This technique has numerous applications in fields such as robotics, augmented reality, and 3D scanning. The goal of registration is to find the optimal transformation that aligns the point clouds while minimizing the distance between corresponding points. The registration process can be divided into three main steps: feature extraction, correspondence estimation, and transformation optimization. Some common feature extraction methods include Scale-Invariant Feature Transform(SIFT), Speeded Up Robust Features(SURF), and others, while correspondence estimation methods include nearest neighbor search and random sampling. Transformation optimization methods typically involve iterative closest point (ICP) algorithms. For a rigid transformation, where scaling is not a point of concern, a transformation vector $T(a)$ is calculated as:

$$T(a) = Ra + t$$

Here, R is an orthogonal transformation, which represents a rotation matrix, and t is a vector signifying the translation of the source point cloud to the target point cloud. If A is a set of points $A = \{a_1, a_2, a_3, ...a_m\}$ and B is a set of points $B = \{b_1, b_2, b_3, ...b_n\}$, the optimal transformation T^* can be written as:

$$T^* = \underset{T}{\operatorname{armin}} \sum_{i=1}^{m} \sum_{j=1}^{n} w_{ij} ||T(a_i) - b_j||^2$$

In the equation above, $\mathbf{a}_i$ is the i_{th} point from point cloud A, $\mathbf{b}_j$ is the j_{th} point from point cloud B. T is the transformation matrix which includes rotation and translation information. w_{ij} is a non-negative weight linked to the corresponding points between point clouds A and B. To optimize both transformation T and weight w_{ij}, an iterative algorithm is used where the algorithm alternates between T and w until the convergence. In every iteration, T is updated which reduces the distance between corresponding points, and simultaneously, weight w is updated based on the alignment. A higher weight is assigned for more accurate correspondences while a lower weight is assigned in the other case.

Point cloud registration is a computationally intensive task and can be challenging in noisy or incomplete point clouds. In Fig. 2.5 an example of point cloud registration is demonstrated, where the blue bunny[1] is the target bunny and the red bunny is the source bunny. The source bunny is transformed in a way to match exactly with the target bunny.

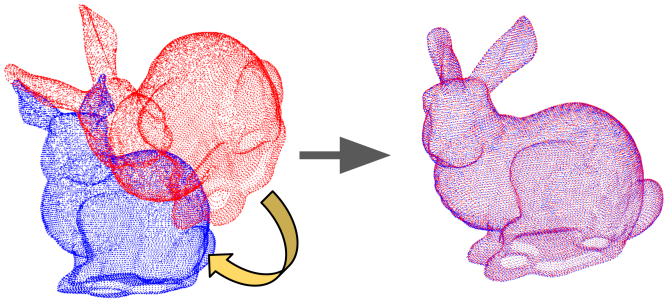


Figure 2.5: An example of Point Cloud Registration

2.7 Lucas-Kannade Algorithm

To measure the motion vectors of the pixels present in an image sequence (video), also known as optimal flow estimation, the Lucas-Kannade algorithm is commonly used [30]. There are a few assumptions in this algorithm like, motion between two adjacent frames is to be small which can be approximated by a linear transformation. The brightness of an individual pixel is also assumed to be constant over time, which develops a brightness constancy constraint given as:

$$I(x, y, t) \approx I(x + dx, y + dy, t + dt)$$

here I denote the intensity for the location (x, y) and time t in the image and the motion vector is represented by (dx, dy, dt).

Considering the above assumptions and constraints, the basic optical flow equation for all the pixels is solved by least square minimization as shown below.

$$\sum_{u,v}(I_x(u, v) * du + I_y(u, v) * dv + I_t(u, v))^2$$

here (u,v) is the location of a pixel, and I_x and I_y are the partial derivatives of the image intensity of x and y respectively. I_t can be defined by the temporal derivative for the image intensity. With this, the motion vector (du,dv) is calculated which minimizes this objective function with the help of a least-squares method. The equation is summed over all pixel locations(u,v), which measures the difference between the actual image intensity and estimated changes. The aim of this is to minimize this difference to achieve precise estimation in the optical flow.

In general, the LK algorithm is a well-known computer vision algorithm that is used in several computer vision-related tasks [40, 38, 43].

2.8 Evaluation Metrics

Evaluation metrics are needed to measure the performance of a statistical or machine learning model. Several metrics like accuracy, precision, F1 score, mean square error (MSE) can be

[1] http://graphics.stanford.edu/data/3Dscanrep/

used to evaluate the quality of the algorithm. The selection of these metrics are dependent on the data quality(balanced, unbalanced), type of algorithm (classification, segmentation, regression, ranking) and the task that the model is designed to perform. It is very important to carefully select the appropriate evaluation metrics for the task as it defines if the model is performing well and justifies the choice of parameters.

2.8.1 Intersection over Union (IoU)

IoU is a known metric in computer vision tasks like object detection and segmentation. It measures the amount of overlap between the true bounding box and predicted bounding box or the segmentation masks [35]. It is calculated as the ratio of intersection of predicted and true region over the union of predicted and true region. The graphical explanation is shown in Fig. 2.6.

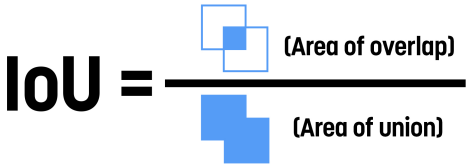


Figure 2.6: Graphical explanation of IoU

2.8.2 Mean Undersegmentation Error (MUCov)

Mean Undersegmentation Error(MUCov) is a quantitative measure in instance segmentation related tasks. MUCov is an estimation of degree of undersegementation, or missed detection in the prediction. Formally, undersegmentation is observed when the predicted mask receives less area than the ground truth of one instance of the object. The undersegmentation error can be defined as the ratio of number of points of undersegmentation to the number of points of ground truth mask of one instance and the mean of all the errors of all object instances is known as mean undersegmentation error.

Technically, it can be written as:

$$MUCov = \frac{1}{\mathrm{N}} \sum \frac{\text{Area of Undersegmentation}}{\text{Area of Ground Truth}}$$

Here, N is the total number of object instances in the prediction. MUCov ranges between 0 and 1, where 0 means perfect segmentation, whereas 1 defines total undersegmentation. Thus, lower the value of MUCov, better is the performance of model.

2.8.3 Mean Oversegmentation Error (MWCov)

Mean Oversegmentation Error(MWCov) is another quantitative measure in instance segmentation related tasks. MWCov is an estimation of degree of oversegmentation, or false positives in the prediction. Formally, oversegmentation is observed when the predicted mask receives more area than the ground truth of one instance of the object. The oversegmentation error can be defined as the ratio of number of points of oversegmentation to the number of points of ground truth mask of one instance and the mean of all the errors of all object instances is known as mean oversegmentation error.

Technically, it can be written as:

$$MWCov = \frac{1}{\mathrm{N}} \sum \frac{\text{Area of Oversegmentation}}{\text{Area of Ground Truth}}$$

Here, N is the total number of object instances in the prediction. MWCov ranges between 0 and 1, where 0 means perfect segmentation, whereas 1 defines total oversegmentation. Thus, lower the value of MWCov, better is the performance of model.

2.8.4 Precision and Recall

Precision calculation is a well known metric to calculate the performance of classification or detection related models. Precision is the measure of amount of correct predictions that are positive. It can also be defined as the number of positive class predictions which actually belongs to the positive class.

Mathematically, it can be represented as:

$$Precision = \frac{\text{TP}}{\text{TP+FP}}$$

Here, TP stands for True Positive and FP stands for False Positive

Recall is another important metric in machine learning tasks. It can be defined as the measure of correct predictions that are detected by the model.

Mathematically, it can be represented as:

$$Recall = \frac{\text{TP}}{\text{TP+FN}}$$

Here, TP stands for True Positive and FN stands for False Positive

Fig. 2.7 shows a graphical representation of precision and recall.

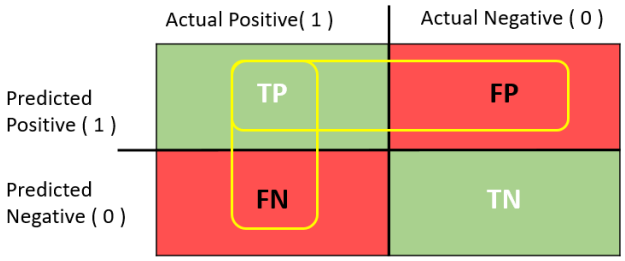


Figure 2.7: Visual Explanation of Precision and Recall

3 Data

This chapter provides a detailed description of the data used in this research. This chapter also brings out the outlines about the type of data, data collection methods, and, data pre-processing techniques performed in this study.

3.1 Data Type

The data used in this project are point cloud data. Point clouds are gaining popularity at a very fast pace in computer vision, industrial applications, and robotics. Point cloud data can be useful in projects that need 3D information such as robotics, autonomous driving, and others. Point clouds in real-life are generated using photogrammetry, or light scanning.

In this project, two types of data are used: Synthetic Point Clouds and Real point clouds. There are 4 different objects with synthetic data, namely IPA Gear Shaft, IPA Ring, Conrod Metal, and, Conrod Plastic. The objects present with real data are Conrod Metal and Conrod Plastic. These objects are shown in Fig. 3.1.

Figure 3.1: Synthetic and Real Data

3.2 Data Description

The synthetically generated data consists of X, Y, and Z values along with the ground truth of instance class, center location, and transformation with respect to the CAD models. One point cloud with 30 objects consists of approximately 70000 points. These numbers of points may vary due to the occlusion complexity or the object's position in the bin. It can also vary based on the shape of the object. The synthetically generated point clouds don't contain any color information.

The point cloud of real data consists of approximately 400,000 points with X, Y, and Z values and color(RGB) as well. This point cloud also contains a lot of noise (surroundings, reflection). A sample point cloud with 10 synthetic conrod plastic objects and 9 real conrod plastic objects is shown in Fig. 3.2.

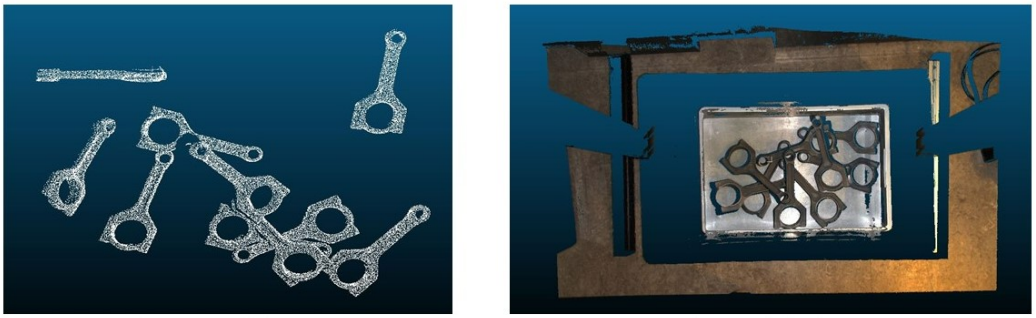


Figure 3.2: Synthetic and real image of conrod plastic

In the above image, we observe that the synthetic data is clean and completely noise free. While in the real image, we find some surrounding surfaces, and light reflections along with the instances of the conrod plastic object. In our data all the instances are from the same object, so one point cloud contains only one type of object.

3.3 Data Aquisition

3.3.1 Synthetic Data generation

In a point cloud dataset, there are several point clouds with varying numbers of instances of the object. As one instance of an object may have a large number of points, the overall size of the dataset and the number of points are huge. In such cases, labeling individual points is not feasible. Thus, synthetic data is generated as close as possible to the real data in various ways.

In this project, two objects from "Fraunhofer IPA Bin-Picking dataset" [19]: IPA Gear Shaft and IPA Ring, and two objects from SICK IVP: conrod metal and conrod plastic are taken to generate this synthetic dataset. In the process of generation, to create a scene alike view, the CAD model of each object is imported. This imported model is dropped randomly into the bin with a specified bin size one at a time [5]. In the first iteration, an empty bin is taken, one instance of the object is dropped, the scene is recorded and the bin is cleared. This process is iterated with the number of instances increasing in every iteration and is continued until the defined total number of instances in a bin is reached. This concludes one cycle. This data generation is done over multiple cycles. The synthetically generated point clouds hold the cartesian coordinates (X, Y, and Z), the centre of each object, rotation, and also the segmentation class for each point. Fig. 3.3 shows a sample synthetic data generation of IPA Gear Shafts.

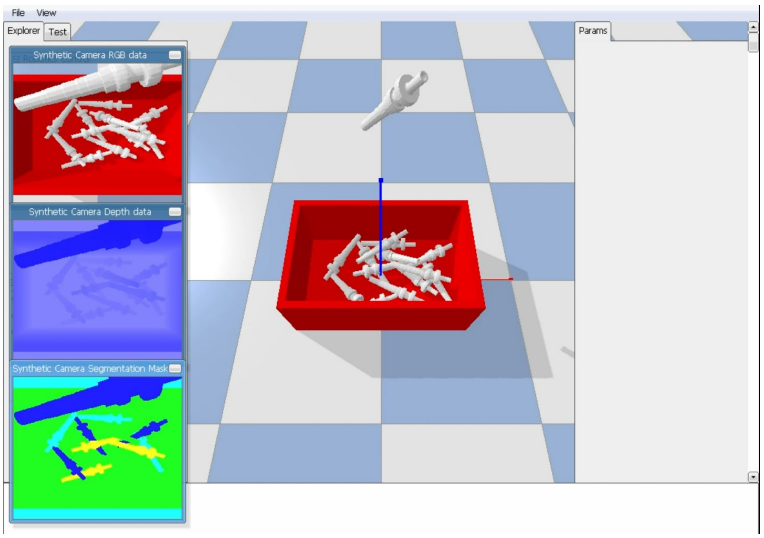


Figure 3.3: Synthetic Data Generation

The number of instances is set to 30. Thus one cycle will have 30 point clouds, one for each number of instances. The data is generated over 60 cycles. Thus, in total 1800 point clouds are generated for each object.

For training the 3D Instance segmentation network, the data from the first 40 cycles were used, ie, 1200 point clouds. The rest 20 cycle data were split between validation and training equally to have 300 point clouds for each set.

For point cloud registration, point clouds with only a single instance were needed. Thus, 200 individual instances were extracted from the previously generated point clouds. This includes both partial (camera view 2.5D point clouds) and full (complete instance) point clouds. To add to the training, the 50 best segments from the output of 3D instance segmentation were also used. For validation, 100 point clouds including partial and full point clouds and 50 best segments were included. To test this model, 80 point clouds including partial and full point clouds were taken along with 40 best segments from the previous output of 3D instance segmentation.

3.3.2 Real Data Collection

4 datasets with 50 point clouds each are collected for real data. Conrod plastic and conrod metal are the 2 objects for real data. To collect the real conrod plastic data, 9 and 18 objects of conrod plastics were used to create a dataset of 50 point clouds each. However, 10 and 23 objects were taken to collect the real conrod metal data with 50-point clouds. These point clouds don't contain any ground truth and are collected to perform qualitative analysis. All these images were collected using Zivid Two M70 camera, which uses structured light stereo as its 3D technology. The point cloud generated after capturing the image contains 3D(XYZ), Color(RGB), and Signal-to-noise ratio (SNR) values for each point. The focal distance of this camera is 700 mm with a spatial resolution of 0.39 mm.

3.4 Data Pre-processing

3.4.1 Noise removal from real data

The noise present in the real data is first reduced by cropping along the X, Y, and Z axes. Secondly, the data collected by the real camera is 1000 times more scaled than the synthetic data. It is due to the resolution of the camera with which the image is taken. Thus, the data

collected is scaled down by 1000 to bring the synthetic and real data at the same scale. It was also observed that the number of points in the real data was far more than in the synthetic data. Hence, the real data is down-sampled using random sampling with the sampling ratio of 0.22 to have a similar number of points with respect to the synthetic data. Fig. 3.4 shows the data pre-processing pipeline for the real data..

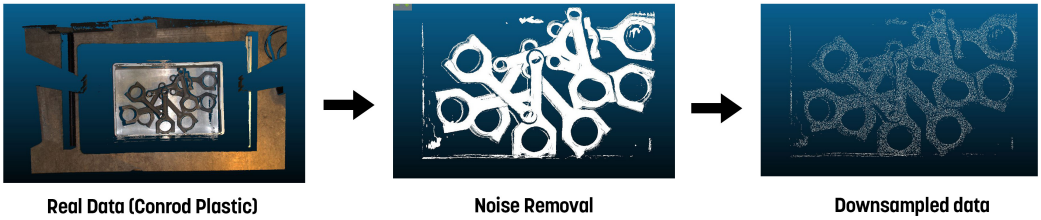


Figure 3.4: Noise removal pipeline for real data

3.4.2 Adding noise to synthetic data

Synthetic data is usually clean and noise-free. However, the real data can have several different types of noises like offset due to other light sources or reflection of objects can also get captured in the point cloud. This noise may affect the result of the real data. To observe, the impact of noises on the model, 36 test sets are generated with different percentages of noise. Out of the 36 test sets, 9 test sets for each object are generated synthetically, with different noise percentages. These noises are generated by offsetting a randomly chosen point in a random direction in 3D space. The distance of offset is drawn from a Gaussian distribution with zero mean and varying standard deviation. The percentage of noise is decided by the number of offset points. There are 3 noise percentages namely 10%, 30%, and 100%. For example, 10% noise signifies that randomly chosen 10% points of the given point cloud to get offset in a random direction by a fixed distance, ie. standard deviation. Fig. 3.5, represents the same point cloud with different noises.

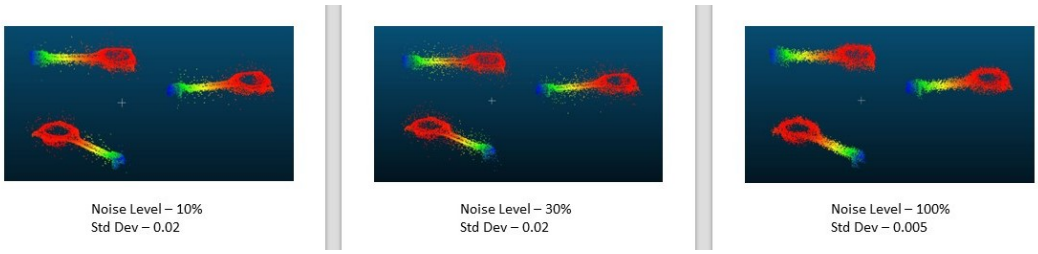


Figure 3.5: Adding several noise to real data

In the image above, it can be observed that the point cloud with 30% noise and a standard deviation of 0.02 looks noisier than the point cloud with 10% noise and the same standard deviation. Also, in the last image of Fig. 3.5, where 100% of the points are shifted with a small standard deviation, it can be observed it doesn't look so noisy, but that the shape of the instance is lost.

4 Methodology

This chapter presents a detailed discussion of the chosen methods mentioned in the theory chapter. The necessary steps involved in implementing the theoretical concepts to the company-specific data are also discussed. This thesis work is broadly divided into two main parts, namely 3D Instance Segmentation and Point Cloud Registration. An overview of the thesis work is shown in Fig. 4.1.

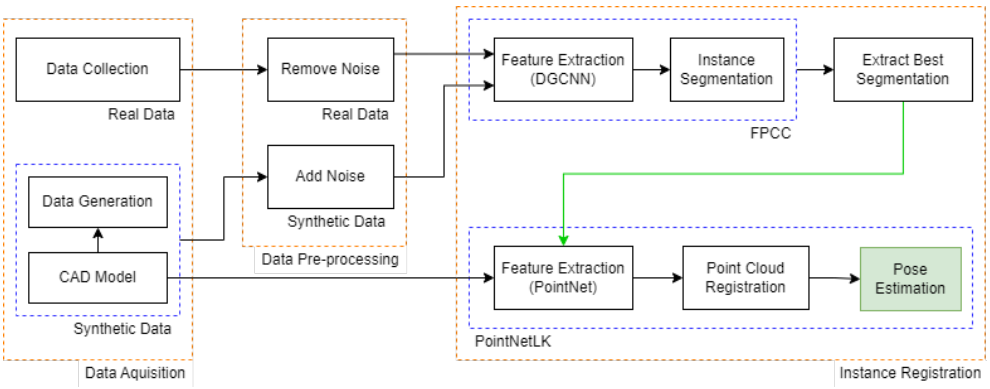


Figure 4.1: Overview of System Design

The overview begins with data acquisition where real data is collected with a stereo camera and synthetic data is generated using the available CAD models of the object. In the data pre-processing stage, the noise is removed from the real data and some offset noise is added to the synthetic data to imitate the real data. The third stage is instance registration, where the pre-processed input point cloud is first segmented to identify the instances of the object present in the bin, followed by choosing one predicted instance to perform point cloud registration. The pose estimation is done with the point cloud registration for the chosen point cloud with respect to the CAD model of the respective object.

4.1 3D Instance Segmentation

3D Instance Segmentation is a computer vision task, where individual instances of an object from a point cloud are detected by labeling every point in the cloud. 3D Instance Segmentation may become a complicated exercise because of the complex geometrical shapes of objects, size, orientation, and noise percentages. In this thesis, to perform 3D Instance Segmentation, we leverage a state-of-the-art approach Fast Point Cloud Clustering (FPCC) [47]. FPCC introduces deep learning-based, 3D instance segmentation which is capable of distinguishing individual objects from a large number of occluded objects efficiently. The key notion of this approach is to locate the geometric center of each object. These identified centers are further used for creating the clusters.

The network architecture of FPCC-net consists of Feature Extraction, Embedded Feature branch, Center Score branch, Inference phase, and Training phase. The FPCC-net architecture is shown in Fig. 4.2. A detailed discussion of each stage is done in later sections.

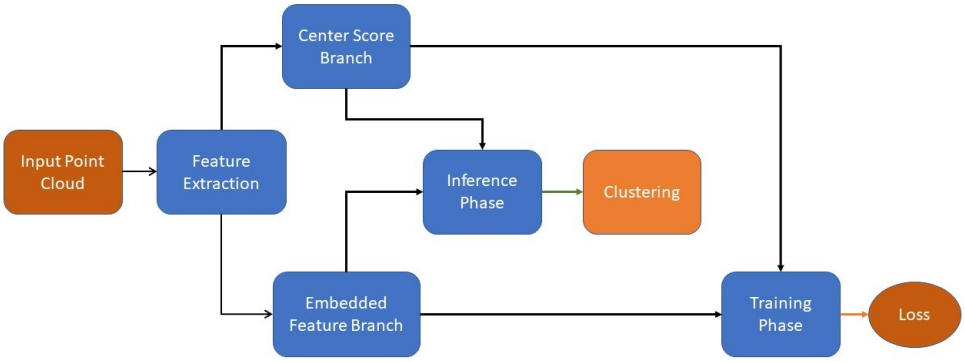


Figure 4.2: Architecture of FPCC Network

4.1.1 Input point cloud pre-processing

Point clouds with irregular and varying densities can lead to poor clustering. Thus, a pre-processing step is needed to address this issue. In the pre-processing step, all points of the point clouds are changed to a new coordinate system. Considering a point cloud with N points, represented as $p_i = (x_i, y_i, z_i)$, where $i = 1, 2, ..., N$ and (x_i, y_i, z_i) are the locations of i_{th} point in the X-axis, Y-axis, and Z-axis respectively. The location of points in the new coordinate system is calculated as:

$$\bar{x_i} = x_i - min\{x_1, x_2, ..., x_N\}$$
$$\bar{y_i} = y_i - min\{y_1, y_2, ..., y_N\}$$
$$\bar{z_i} = z_i - min\{z_1, z_2, ..., z_N\}$$

The new coordinate system is better aligned with the geometric property of the point cloud and is also less harmed by the varying density of the point clouds. Particularly, the new coordinate system has a uniform density along each axis and aligns better with principal axes.

In the next step of pre-processing, each point from the new coordinate system is normalized to generate a normalized location, which is a vector of the form (n_x, n_y, n_z). To calculate the normalized location, the average of all points is computed to find the centroid of the point cloud. Further, the point cloud is scaled to the longest dimension to fit in a unit cube. To this end, all the coordinates of each point are divided by the length of the longest dimension to find the normalized location, which ensures that the values range between 0 and 1.

In the last stage, the new coordinates $(\bar{x}_i, \bar{y}_i, \bar{z}_i)$ is combined with the normalized location (n_x, n_y, n_z) to obtain a 6-dimensional vector for each point.

4.1.2 Feature Extraction

In the FPCC network architecture, the pre-processed input cloud of shape (N x 6) is fed into the feature extraction block. The feature extraction in this network is done using DGCNN model (2.2). During the implementation of DGCNN model, the last layers of classification and segmentation is ignored, in order to generate only the feature vectors of N input points. DGCNN generates point-wise features of N points from the input point cloud with the shape (N x 256). To further refine and enhance the input point cloud features FPCC-net uses the Embedded Feature branch.

4.1.3 Embedded Feature Branch

The feature extraction branch results in high-dimensional feature vectors, which can be computationally expensive and may also add noise or redundancy. FPCC-net introduces Embedded Feature Branch, which aims at refining the feature representations by reducing dimension and normalizing features. The feature vectors are normalized to ensure a consistent range and scale across all features. Embedded Feature Branch applies a shared Multi-Layer perceptron(MLP) and produces a global feature vector for all points by pooling through all points.

For a single point, it can be written as:

$$f_i = MLP(h_i)$$

where, h_i is the output of Feature Extraction layer of the i_{th} point, MLP here is a shared MLP and f_i is the output of this layer for i_{th} point. Pooling is done to collect the features of all the points from the point cloud and generate a global feature vector. This layer calculates the maximum value of each feature for all points in the cloud and can be written as:

$$v = max_{pool}(f_1, f_2, f_3, ..., f_N)$$

where, $f_1, f_2, f_3, ..., f_N$ are the output feature vectors from the previous feature embedded layer for each point in the point cloud and v is the output of this layer, which signifies a global feature vector for all the points.

4.1.4 Center Score Branch

A center score can be defined as the significance of a point within a point cloud to be the center of an instance. The Center Score branch estimates the center score for each point in the point cloud and generates a vector of shape (N x 1). This branch is in parallel to the Feature Embedded branch. In this branch, the features extracted are passed through two MLPs and are finally activated by a sigmoid function. Based the on features extracted, estimated centers are stored.

The center score for each point is calculated based on the distance from the estimated center. The Euclidean distance for each point is calculated with the nearest center. Considering a point cloud P with N points, the distance is calculated as follows:

$$d = \sqrt{\sum_{i=1}^{N} (p_i - center_i)^2}$$

here d is the calculated distance, p_i represents point i in the point cloud and $center_i$ is the nearest center for this point.

This distance is normalized by dividing it by d_{max}, where d_{max} is a hyper-parameter and is defined as the maximum distance from the cluster center of the object.

$$d_{rate} = \frac{d}{d_{max}}$$

here, d_{rate} is the normalized distance. Further, this normalized distance is clipped between 0 and 1, which signifies that any d_{rate} below 0 is given 0, and above 1 is given 1. Finally, the score calculation for each point is done by subtracting the squared distance rate by 1. Mathematically, it can be shown as:

$$score = 1 - d_{rate}^2$$

A high score indicates a more significant and tightly-cluster center.

4.1.5 Inference Phase

In the Inference Phase of FPCC-net, the outputs of the feature-embedded branch and center score branch are taken as input. The NMS algorithm (2.4) is adapted here to find the non-overlapping center points of the instances from the input point cloud.

Algorithm 1 Non-Maximum Suppression Algorithm

Input: Center score threshold (θ_{th}), Points $\mathbb{P}$, Screening radius (γd_{max}), Predicted center scores $\mathbb{S}$
Output: Non-overlapping center points $\mathbb{C}(c_1, c_2, ..., c_k)$

```
for i=1 to N do
    if s_i ⩽ θ_th then
        P ← P\{p_i}
        S ← S\{s_i}
    end if
end for
C ← {}
while P ≠ φ do
    m* ← arg max_m{s_m|s_m ∈ S}
    C ← p_m*
    P ← P\{p_m*}
    S ← S\{s_m*}
    for p_i in P do
        if d(p_m*, p_i) ⩽ γd_max then
            P ← P\{p_i}
            S ← S\{s_i}
        end if
    end for
end while
return C
```

A center score threshold (θ_{th}) is set for the points to qualify as a center of the instance. The NMS algorithm takes a center score threshold (θ_{th}), a set of points ($\mathbb{P}$), a hyper-parameter d_{max} which signifies the maximum distance of any point from the geometric center, and the predicted center scores ($\mathbb{S}$) for every point as an input. The output of the algorithm is a set of non-overlapping centers $\mathbb{C}$.

The algorithm is iterated for the total number of points in the point cloud, and if the predicted score (s_i) is less than the threshold for the center score (θ_{th}), then p_i is removed from the set of possible instance center points, and s_i is removed from the predicted center score list. It is done so that p_i does not suppress other nearby points which may be an instance center during other iterations of the algorithm.

The algorithm continues by initializing the center list $\mathbb{C}$. The algorithm is iterated then until the updated set of points is null. During each iteration, the highest center score m^* is taken from $\mathbb{S}$, and the corresponding point p_{m*} gets added to the center point list $\mathbb{C}$. Once it is added, the point p_{m*} is then removed from $\mathbb{P}$, and its corresponding score s_{m*} is eliminated from the scores list $\mathbb{S}$.

Further, the algorithm is iterated with remaining points in $\mathbb{P}$. All the points which fall within the screening radius γd_{max} are removed from the set of points $\mathbb{P}$ and their corresponding score s_i is removed from the center score list $\mathbb{S}$. This process is continued until every point is processed and finally, a list of center points $\mathbb{C}$ of length k is generated which doesn't contain any overlapping instances.

In the next part of the inference phase, the feature distances between computed center points and all the other points are calculated using:

$$d(e_F^{(i)}, e_F^{(k)}) = ||e_F^{(i)} - e_F^{(k)}||_2$$

The formula above represents the Euclidean distance between the feature vectors of i_{th} point from the remaining points and k_{th} center point determined by the Algorithm 1, where $e_F^{(i)}$ is the feature vector of i_{th} point and $e_F^{(k)}$ is feature vector for k_{th} center point. Based on these feature distances, all the points excluding center points $\mathbb{C}$ are clustered with the nearest center points. It is important to notice that the nearest center point for every point is calculated in the feature embedding space, while the distance between the point p_i and center point c_k is calculated with Euclidean distance in 3D space. At last, based on the nearest center point in 3D space instance label is assigned to every point, which distinguishes one instance cluster from another.

4.1.6 Training Phase

In the training phase of the network, the model tries to learn precisely to predict the center score and key geometric features of the points in the point cloud. Three matrices, feature distance matrix(FDM), valid distance matrix(VDM), and attention score matrix (ASM) are used to identify and prioritize the salient geometric features of the point cloud.

Feature Distance Matrix (FDM)

With the given input point cloud with N points, the FDM forms a square matrix with the size (N x N). This matrix is the pairwise distance between the feature vectors of each point in the point cloud. The purpose of this matrix is to collect local geometric features. The feature distance calculation is done using the formula below:

$$d_{F(i,j)} = ||e_F^{(i)} - e_F^{(j)}||_2$$

where, $d_{F(i,j)}$ is the distance between the feature vectors of i_{th} point and j_{th} point of the point cloud. The $d_{F(i,j)}$ should be small if i, j are in the same point cloud. The feature distance matrix is represented by $D_F \in \mathbb{R}^{NxN}$.

Valid Distance Matrix (VDM)

A binary matrix of elements either 0 or 1 of size (N x N) is introduced. This matrix aims at deciding if the two points belong to the same instance or not. If the distance between two points exceeds twice the maximum distance (d_{max}), then two points don't belong to the same instance. Mathematically, it can be written as:

$$\mathrm{d_V}(i,j) = \begin{cases} 1 & \text{if } \|p_i - p_j\|_2 < 2d_{\max} \\ 0 & \text{otherwise} \end{cases}$$

where, $d_V(i,j)$ is a binary value that signifies if i_{th} point and j_{th} point of the point cloud belongs to the same instance or not. The valid distance matrix is represented by $D_V \in \mathbb{R}^{NxN}$.

Attention Score Matrix (ASM)

The point pair close to the center should have a bigger weight than other points. To give importance to such point pairs attention score matrix is introduced in the FPCC-net. The ASM matrix is a square matrix of size (NxN), represented by $S_A \in \mathbb{R}^{NxN}$ and is calculated as:

$$S_{A(i,j)} = min(1, s_{center(i)} + s_{center(j)})$$

where, S_A signifies the point-pair weight between points i and j of the point cloud. $s_{center(i)}$ and $s_{center(j)}$ are the actual centre scores of the associated centre of the i_{th} and j_{th} point respectively.

4.1.7 Network Loss

The loss of the FPCC-net is a sum of two losses namely feature embedded loss (L_{EF}) and center score loss (L_{CS}). The total loss is calculated as below:

$$L = L_{EF} + \alpha L_{CS}$$

here, L is the total loss, L_{EF} is the loss of embedded feature branch and L_{CS} is the loss of center score branch. α is a constant term to equally weigh both losses.

Embedded Feature Loss

Two points in the point cloud can either belong to the same cluster or another. The Embedded Feature Loss is used to optimize the clustering with the given ground truth. The L_{EF} can be defined as:

$$L_{EF} = \sum_{i=1}^{N} \sum_{j=1}^{N} w_{(i,j)} k_{(i,j)}$$

where $w_{(i,j)}$ is the weight for i_{th} point and j_{th} point of the point cloud. $w_{(i,j)}$ is the product of element-wise multiplication of D_V and S_A.

$$w_{(i,j)} = d_{V(i,j)} s_{A(i,j)}$$

where, $d_{V(i,j)}$ is an element of D_V and $s_{A(i,j)}$ is an element of S_A. The feature distance between the two points of different instances should be greater than the points belonging to the same cluster.

The $k_{(i,j)}$ is a loss based on point pair features and is defined by:

$$\mathrm{k_{(i,j)}} = \begin{cases} max(0, d_{F(i,j)} - \epsilon_1) & \text{if } p_i \text{ and } p_j \text{ are in same instance} \\ max(0, \epsilon_2 - d_{F(i,j)}) & \text{otherwise} \end{cases}$$

here, ϵ_1 and ϵ_2 are constants. If the distance between the feature representations $d_{F(i,j)}$ is greater than ϵ_1, the value $d_{F(i,j)} - \epsilon_1$ is used as the value of $k_{(i,j)}$. If the distance is less than or equal to ϵ_1, the value of $k_{(i,j)}$ is set to 0. This is done when both points p_i, p_j are in the same instance. In other case, value $\epsilon_2 - d_{F(i,j)}$ is used for $k_{(i,j)}$ when ϵ_2 is greater than $d_{F(i,j)}$. If the distance is greater than or equal to ϵ_2, $k_{(i,j)}$ is set to 0. This determines the relationship between points of the same instances while considering both similarities and dissimilarities between them.

Center Score Loss

Smooth L1 loss function (2.5) is used in calculating the center score loss due to its robustness. The L_{CS} can be defined as:

$$L_{CS} = \frac{1}{N}\sum_{i=1}^{N} smooth_{L1}(s_{center(i)} - \hat{s}_{center(i)})$$

where, $s_{center(i)}$ is the actual center score and $\hat{s}_{center(i)}$ is the predicted center score. The actual center score (s_{center}) is calculated during the synthetic data generation process, while the predicted center score ($\hat{s}_{center}$) is computed in the center score branch.

4.2 Extracting Best Segmentation

The 3D Instance segmentation labels every point of the point cloud with the instance class. Usually, the number of predicted instances is more than the actual number of objects in the point cloud. Since the prediction is not 100% precise, some of the points of the actual instance get classified as different instances. Thus, the predicted instance may have some extra points from the actual instance and/or fewer points from the actual instance.

As the registration will be done with one of the predicted instances, it is important to decide which of the predicted instance will be the most suitable to do the registration. Two things can be considered while choosing one instance: the number of points in the predicted instance and the shape of the predicted object.

In the synthetic data with only one object, there are approximately 3000 points, while in the point cloud with 30 objects, there are approximately 70000 points. The average number of points per object is calculated to set a baseline as the expected number of points in one predicted instance. Assuming a scenario where the predicted instance may have the expected number of points or more, but many of the points are from different instances and are widely spread. This indicates that the shape of the object is lost in this point cloud. Thus, the minimum bounding box algorithm (2.3) is taken to check, if the predicted instance has a good shape.

4.2.1 Method 1: Fixed threshold

In this method, a threshold is set for the number of expected points and the volume of the predicted instance. The number of expected points is set to be the average number of points for the individual object. The volume of the predicted instance is set to be the volume of the CAD model of the respective object. According to this method, only those predicted instances are considered which have at least an average number of points and at max CAD model volume. Once the qualified instances are obtained, it is sorted based on the number of points. So, the qualified predicted instance with the maximum number of points is chosen as the best instance for registration.

4.2.2 Method 2: Score all the predicted instances

Considering the two parameters: number of points and volume of the predicted instance, a score calculation formula is proposed below.

$$iScore_k = \frac{1}{\mu Points_j} * nPoints_k - \frac{1}{aVolume_j} * dVolume_k$$

where j defines the object type (conrod metal, conrod plastic, gear shaft, or ring) and k is the predicted instance. $iScore_k$ is the calculated instance score for the k_{th} predicted instance. $nPoints_k$ is the number of points in the k_{th} instance and $\mu Points_j$ is the expected number of points for j_{th} object, which is the average number of points for the j_{th} object. The actual

volume of the minimum bounding box of the CAD model for j_{th} object is the $aVolume_j$ and $dVolume_k$ is the absolute difference in the minimum bounding box volume of the k_{th} predicted instance and CAD model.

The extracted best segmentation is taken as a source for point cloud registration with the CAD model to determine the 6D pose of the object.

4.3 Point Cloud Registration

Point Cloud registration is a technique to find the transformation of a source point cloud with respect to the template point cloud (CAD model). The task of registration becomes complex if the point cloud is incomplete, noisy or the object is symmetrical.

PointNetLK has combined deep learning with the iterative improvement from Lucas Kanade algorithm (2.7) to obtain benchmark results. With the given source and template point cloud, features for individual points are extracted using the deep learning architecture of PointNet [32]. These extracted feature vectors help in predicting the correspondence between the source and template point cloud by using the modified LK algorithm. Least Square Optimization is considered to compute the optimal rigid transformation (a transformation where the size or shape of the geometric figure doesn't change). This entire process is repeated until the convergence is achieved.

In this thesis, to perform the point cloud registration we leverage the method proposed in PointNetLK: Robust and Efficient Point Cloud Registration [2].

4.3.1 PointNetLK architecture

PointNetLK is a deep learning-based approach for estimating the rigid transformation between two point clouds. The mechanism of the PointNetLK can be divided into two main groups, namely feature extraction and iterative LK algorithm. The architecture of the PointNetLK is shown in Fig. 4.3. The red arrows in the image indicate that it is part of the iteration to find the optimal transformation while the black arrows suggest that the task is performed only once.

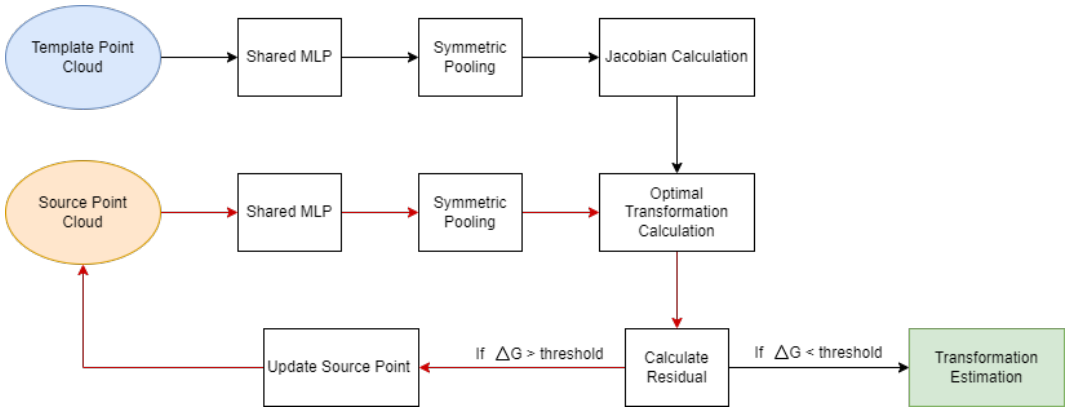


Figure 4.3: PointNetLK architecture

The feature extraction is based on PointNet architecture, where a point cloud $P = p_1, p_2, ..., p_N$ is passed as an input, and a feature vector is produced as an output. Let P_T be the target point cloud and P_S be the source point cloud. This registration task aims to find the transformation matrix G such that source point cloud P_S aligns best with the target point cloud P_T. The transformation G can be expressed as:

$$G = exp(\sum_i \xi_i T_i)$$

where ξ is a 6-dimensional transformation parameter containing both rotational and translational components. The first 3 components represent the rotation (angular velocities) and the next 3 represent the translation (linear velocities). T_i is the 4x4 twist matrix associated with the transformation. It also contains both rotational and translational information. A 3D point registration can be expressed as; $P_T = G.P_S$, which can also be written as $P_S = G^{-1}.P_T$. To generate a K-dimensional vector descriptor, a transformation function ϕ is applied.

$$\phi(P_S) = \phi(G^{-1}.P_T)$$

For a point cloud with N points, ϕ can defined as; $\phi : \mathbb{R}^{3XN} \rightarrow \mathbb{R}^K$.

For a small perturbation in the transformation parameter, we expand the transformation function ϕ around Identity transformation with first-order Taylor expansion.

$$\phi(G^{-1}P_T) \approx \phi(I + \bigtriangledown\phi(I) * \xi)$$

where, I is the Identity transformation, $\bigtriangledown\phi(I)$ is the Jacobian matrix evaluated at the Identity transformation and ξ is the twist vector denoting a small change in the transformation parameter. It can further be written as:

$$\phi(G^{-1}P_T) \approx \phi(I) + \bigtriangledown\phi(I) * \xi$$

As we apply ground truth transformation to P_T, we substitute the $\phi(P_T)$ for $\phi(I)$.

$$\phi(G^{-1}P_T) \approx \phi(P_T) + \bigtriangledown\phi(I) * \xi$$

Upon differentiating both side of the equation with respect to ξ, we get:

$$\frac{\delta}{\delta\xi}\phi(G^{-1}P_T) \approx \frac{\delta}{\delta\xi}\phi(P_T) + \bigtriangledown\phi(I)$$

$$\frac{\delta}{\delta\xi}\phi(G^{-1}P_T) \approx \bigtriangledown\phi(I)$$

here the term $\frac{\delta}{\delta\xi}\phi(P_T)$ was neglected, as it is an approximation equation and the term $\frac{\delta}{\delta\xi}\phi(P_T)$ is negligible as compared to $\bigtriangledown\phi(I)$. Finally, replacing this derivative value in our initial expansion, we get:

$$\phi(P_S) = \phi(P_T) + \frac{\partial}{\partial\xi}[\phi(G^{-1}.P_T)]\xi$$

The G^{-1} can be defined as

$$G^{-1} = exp(-\sum_i \xi_i T_i)$$

and the derivative of the transformation function ϕ is denoted by

$$J = \frac{\partial}{\partial\xi}[\phi(G^{-1}.P_T)]$$

which is a $Kx6$ matrix. This representation of J can be expressed as the gradient of the transformation function with respect to the transformation parameter ξ. This traditional approach of the LK algorithm to calculate the Jacobian becomes difficult with unstructured data like point cloud data. The calculation here is the differentiation of the geometric transformation function with respect to the transformation parameter. As geometric transformation has complex operations like point-wise transformation, the equation becomes non-differentiable. This limitation hinders the deep learning optimization methods. Thus, the approach to calculating the Jacobian needs to be modified.

4.3.2 Modified LK Algorithm

To address the above challenges, the Jacobian is calculated by stochastic gradient. The Jacobian for every column is approximated as below.

$$J_i = \frac{\phi(exp(-t_i T_i).P_T) - \phi(P_T)}{t_i}$$

where t_i represents the extremely small deviation of ξ, and i refers to the i_{th} column of the Jacobian. In the above Jacobian calculation, first, the inverse transformation is computed with respect to the template point cloud. This undoes the effect of transformation and brings back the source point cloud into the coordinate frame of the template point cloud. This operation is identified as $\phi(exp(-t_i T_i).P_T)$. Further, the difference between source and template point cloud is calculated which measures the error between the two point clouds after the transformation was applied. Finally, the calculated error is normalized by dividing it by t_i. It is important to note that t_i should be extremely small for the modified Jacobian calculation to equal to the traditional Jacobian derivative.

To calculate the optimal twist, it is iterated over a loop while updating the source point cloud as $P_S \leftarrow \triangle G.P_S$ where $\triangle G$ can be expressed as $\triangle G = exp(\sum_i \xi_i T_i)$.

A minimum threshold ($\triangle G_{th}$) is set to stop the loop and final transformation is calculated as the product of estimates from every iteration. Mathematically, it can be written as:

$$G_{est} = \triangle G_n .....\triangle G_1 .\triangle G_0$$

4.3.3 Training Phase

The transformation matrix G is a 4X4 homogeneous matrix, which consists of rotation and translation in 3D space. To minimize the difference between the predicted output and the ground truth, a loss function is needed during the training. Here the loss is minimized by the following equation:

$$||(G_{est})^{-1}.G_{gt} - I_4||_F$$

This way of minimizing is simpler and efficient as it avoids matrix logarithmic operations. The loss function here is measuring the distance between the ground truth transformation and the estimated transformation. It encourages the estimated transformation to merge with the ground truth transformation while training.

Since point clouds are unordered set of points, it is quite important to apply the pooling operation as it helps in generating a global feature vector from all the local feature vectors. So, a pooling operation either a maximum pool or average is applied.

5 Experiment and Results

This chapter presents the outcome of the conducted experiments and also focuses on analyzing the obtained results. It aims upon evaluating the performance of various datasets to show the usefulness of the developed approach. Along with the detailed description of the experimental setup, the evaluation of the chosen methods is also discussed here. This chapter also motivates the choice of hyper-parameters and presents produced results in-depth with several visualizations.

5.1 Result of 3D Instance Segmentation

5.1.1 Experimental Setup

The training was performed for 50 epochs for each of the 4 synthetically generated objects. For each object, there are 1200 point clouds with the number of objects ranging from 1-30 in a point cloud. The FPCC network is trained in Pytorch with a learning rate of 0.0001, batch size of 2, and Adam Optimizer to reduce the overall loss and improve the accuracy. Nvidia GTX 1080 GPU with 12GB RAM is used during the entire training and testing of the model. During the training phase, d_{max} is varied with 0.07, 0.1, 0.18, and 0.25 to find the optimal d_{max}, while other parameters like γ, α, and k were kept constant at 0.1, 3, and 20 respectively. Here, γ is the coefficient of the screening factor, α is the weight to balance both losses (L_{EF}, L_{CS}), and is the number of nearest-neighbors to consider while performing the majority voting in K-nearest neighbor algorithm. Based on the original results of the FPCC paper, these parameters were chosen. Every batch takes 4096 unique points. The training consumes around 12 hours for one object with the above-mentioned parameter values.

5.1.2 Training Loss

The combined loss of the network is calculated as $L = L_{EF} + \alpha L_{CS}$. During the training, the loss of the network seems to decrease normally and saturate after some epochs. The model is saved upon every epoch and the best model is chosen with the least loss. The line chart for the training loss of different objects is shown in Fig. 5.1. The loss shown in Fig. 5.1 is the total loss during training for all the objects.

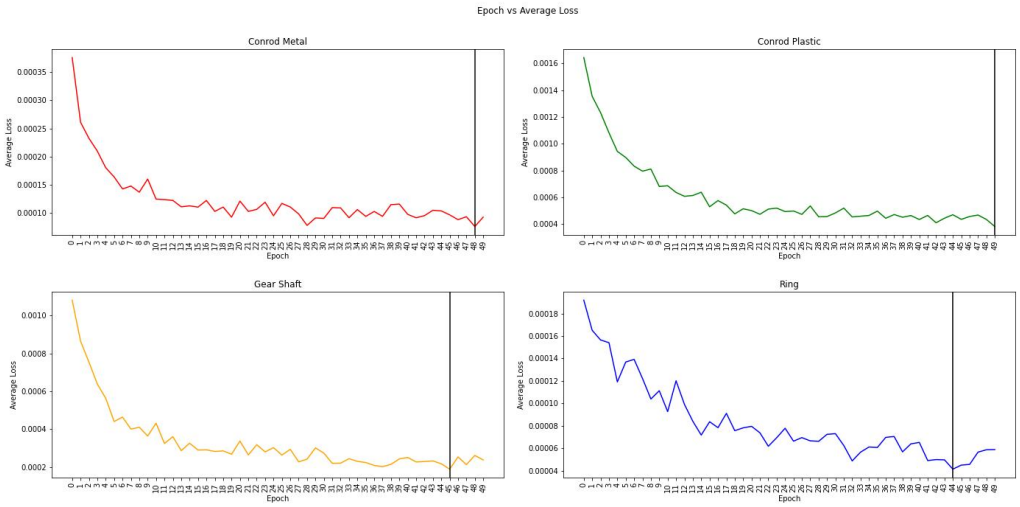


Figure 5.1: Training loss during 3D instance segmentation

We observe that the total training loss of the network saturates after some epoch for all the objects.

5.1.3 Tuning hyper-parameter dMax

d_{max} is a parameter that is dependent upon the shape and size of the object. We have tried different d_{max} values to check the performance of the model for different objects. The results from this hyper-tuning are mentioned in table 5.1 and table 5.2. All the evaluation of precision and recall is done with respect to IoU as 0.5. The value of IoU is chosen to be 0.5 as per the original FPCC implementation. This hyper-tuning is done on the validation data to choose the best d_{max} for every object.

Table 5.1: Performance of 3D instance segmentation with different values of d_{max} at IoU=0.5 on validation data (gear shaft and ring)

d_{max}	Gear Shaft		Ring	
	Precision	Recall	Precision	Recall
0.07	0.218 ± 0.021	0.588 ± 0.034	0.859 ± 0.030	0.970 ± 0.009
0.1	0.211 ± 0.019	0.580 ± 0.037	0.825 ± 0.026	0.965 ± 0.008
0.18	0.219 ± 0.020	0.606 ± 0.032	0.843 ± 0.013	0.967 ± 0.006
0.25	0.231 ± 0.020	0.609 ± 0.030	0.835 ± 0.014	0.968 ± 0.007

Table 5.2: Performance of 3D instance segmentation with different values of d_{max} at IoU=0.5 on validation data (conrod plastic and conrod metal)

d_{max}	Conrod Plastic		Conrod Metal	
	Precision	Recall	Precision	Recall
0.07	0.506 ± 0.026	0.856 ± 0.034	0.600 ± 0.019	0.915 ± 0.011
0.1	0.509 ± 0.010	0.867 ± 0.020	0.595 ± 0.018	0.902 ± 0.013
0.18	0.529 ± 0.019	0.900 ± 0.015	0.614 ± 0.022	0.917 ± 0.016
0.25	0.512 ± 0.026	0.874 ± 0.032	0.590 ± 0.010	0.895 ± 0.017

From the results of hyper-tuning d_{max}, we observe that gear shaft, ring, conrod plastic, and conrod metal objects have the best mean precision and mean recall with d_{max} values as 0.25, 0.07, 0.18, and 0.18 respectively on the validation data. The presented results contain

mean precision and mean recall with the respective standard deviation for every experiment. The deviation from the mean in the precision and recall for conrod metal is found to be less than 2%. In the case of conrod plastic, the deviation from the mean in the precision and recall is found to be close to 2.5% and 3% respectively. For the ring object, the deviation from the mean in the recall is observed to be very small that is 0.6%, while the deviation in the precision from the mean was less than 3%. Gear object was seen to have less than 2.5% and 4% deviation from the mean in precision and recall respectively.

To observe the performance of the trained model on test data, different d_{max} with IoU = 0.5 is also tested. Table 5.3 and 5.4, presents the result of this experiment.

Table 5.3: Performance of 3D instance segmentation with different values of d_{max} at IoU=0.5 on test data (gear shaft and ring)

d_{max}	Gear Shaft		Ring	
	Precision	Recall	Precision	Recall
0.07	0.248 ± 0.023	0.617 ± 0.035	0.843 ± 0.022	0.942 ± 0.005
0.1	0.256 ± 0.019	0.648 ± 0.039	0.836 ± 0.027	0.944 ± 0.006
0.18	0.260 ± 0.016	0.662 ± 0.028	0.826 ± 0.014	0.938 ± 0.007
0.25	0.262 ± 0.024	0.655 ± 0.030	0.805 ± 0.011	0.937 ± 0.005

Table 5.4: Performance of 3D instance segmentation with different values of d_{max} at IoU=0.5 on test data (conrod plastic and conrod metal)

d_{max}	Conrod Plastic		Conrod Metal	
	Precision	Recall	Precision	Recall
0.07	0.533 ± 0.022	0.804 ± 0.033	0.624 ± 0.018	0.853 ± 0.011
0.1	0.543 ± 0.020	0.822 ± 0.025	0.619 ± 0.014	0.838 ± 0.012
0.18	0.560 ± 0.027	0.850 ± 0.034	0.625 ± 0.012	0.844 ± 0.014
0.25	0.539 ± 0.023	0.823 ± 0.025	0.617 ± 0.010	0.841 ± 0.016

From Table 5.1, 5.2, 5.3, and 5.4 we observe a similar performance of the trained model on both validation and test data. The value of d_{max} for all objects is chosen based on the results of the validation set. The experiment of different d_{max} with test data is done to ensure the similar performance of the model on both datasets.

5.1.4 Results from Noisy Data

The chosen model is applied to the noisy test data created with several noise percentages. It is interesting to see the behavior of the evaluation metrics with the increasing noise percentage through line charts. Below are the line plots of individual evaluation metrics comparing all the objects with increasing noise percentages.

In all the figures 5.2, 5.3, 5.4, and 5.5, there are 3 line plots for noise percentages 10, 30, and 100 in the order left to right. Different colors in the line plots represent the different objects. All the plots are evaluated for 3 levels of increasing standard deviation. In Fig. 5.2, we observe that the mean precision is degrading with the increasing noise (standard deviation) for all three noise percentages. However, we also observe that the mean precision increases with the increasing noise in the case of the ring object. A similar trend was observed for the other evaluation metrics in Fig. 5.3, 5.4, and 5.5.

Table 5.5, presents the numerical representation for the results on the noisy data with conrod metal. The average precision clearly seems to drop with increasing standard deviation. However, the difference is not so significant.

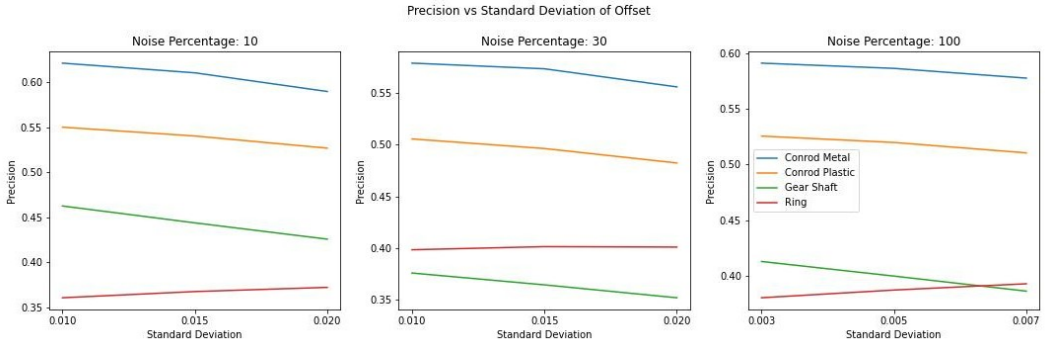


Figure 5.2: Precision vs Noise Percentages for all objects

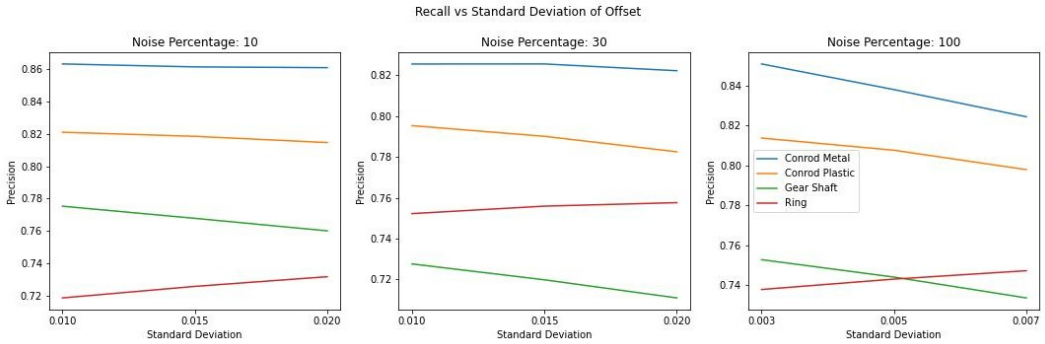


Figure 5.3: Recall vs Noise Percentages for all objects

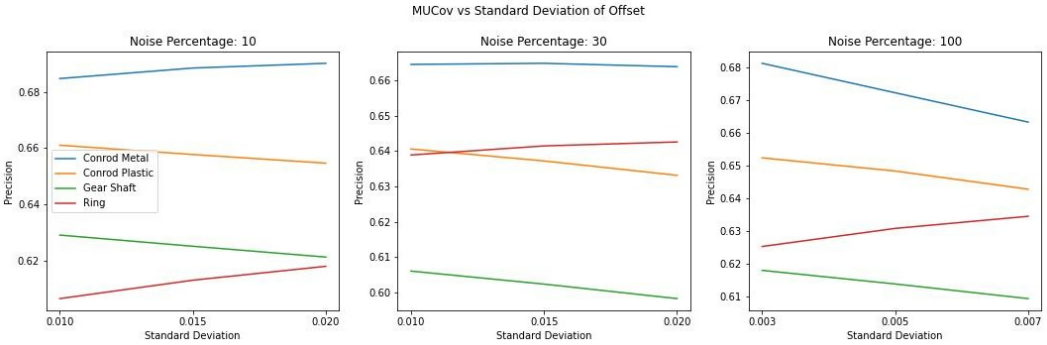


Figure 5.4: MUCov vs Noise Percentages for all objects

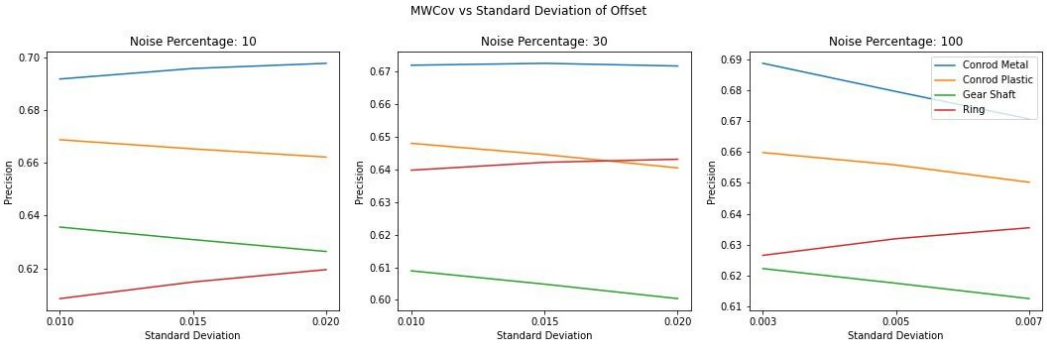


Figure 5.5: MWCov vs Noise Percentages for all objects

Table 5.5: Results for Conrod Metal on noisy test data with IoU = 0.5

Noise Percentage	Std. Dev.	Precision	Recall	MUCov	MWCov
10	0.01	0.620	0.863	0.684	0.691
	0.015	0.610	0.861	0.688	0.695
	0.02	0.589	0.860	0.690	0.697
30	0.01	0.578	0.825	0.664	0.671
	0.015	0.573	0.825	0.664	0.672
	0.02	0.555	0.822	0.663	0.671
100	0.003	0.591	0.850	0.681	0.688
	0.005	0.586	0.838	0.672	0.679
	0.007	0.577	0.824	0.663	0.670

5.1.5 Number of predicted instances vs number of actual instances

It was observed that the number of predicted instances is usually higher than the actual number of instances in the bin. The test set contains 300 points clouds which were generated in 10 cycles. As each cycle contains 1-30 instances, therefore there are 10 point clouds for each number of instances (1-30) for every object. The average number of predicted instances for each number of actual instances is calculated and shown in Fig. 5.6.

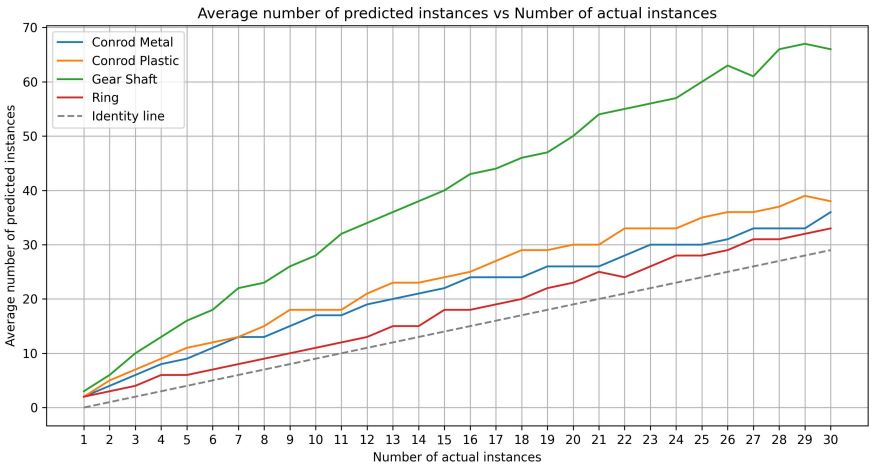


Figure 5.6: Number of predicted instances vs Actual number of objects

It can be seen that for conrod metal, conrod plastic, and ring objects, the number of predicted instances is slightly higher than the actual number of instances while it is approximately twice in the case of the gear shaft.

5.1.6 Qualitative Analysis

With the shown results above, it will also be interesting to observe the 3D instance segmentation performed on all the objects visually. 3D Instance segmentation performed on synthetic conrod plastic, gear shaft, and ring objects is shown in Fig. 5.7, Fig. 5.8, and 5.9 respectively.

The segmented images contain color gradients, where every color represents one predicted instance. We observe there are several instances that are well segmented, while there are some instances that have multiple predictions. We also observe that few predicted instances contain points from multiple actual instances.

The trained model is also tested on the real data. The point clouds of real conrod metal and real conrod plastic objects were collected at SICK to test the model. Conrod metal is a

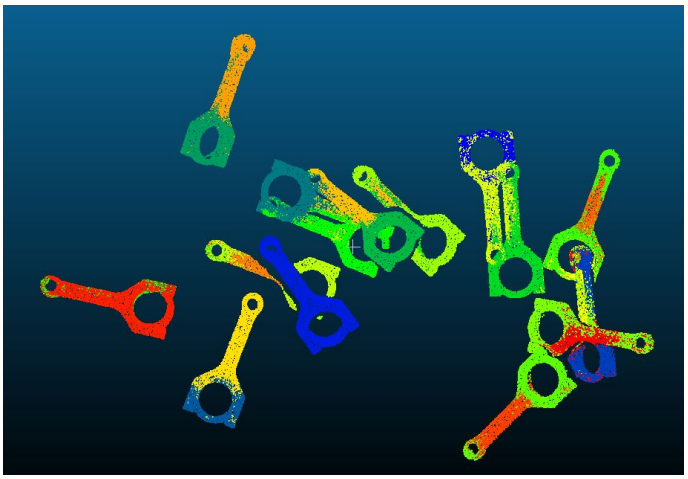


Figure 5.7: 3D Instance Segmentation of Conrod Plastic with 15 objects

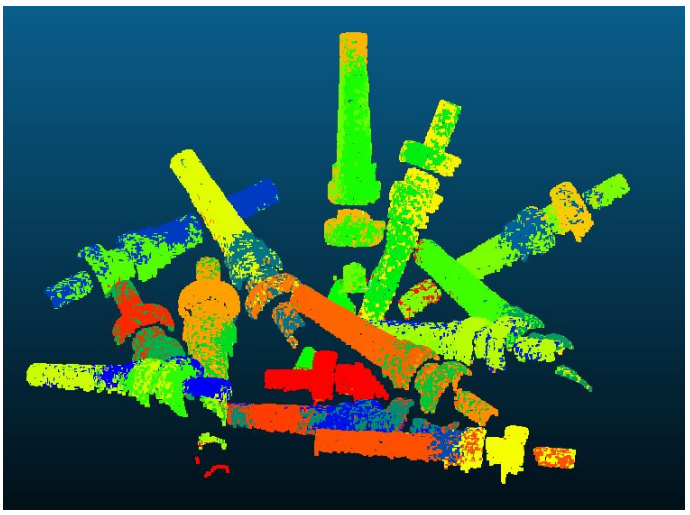


Figure 5.8: 3D Instance Segmentation of IPA Gear Shaft with 15 objects

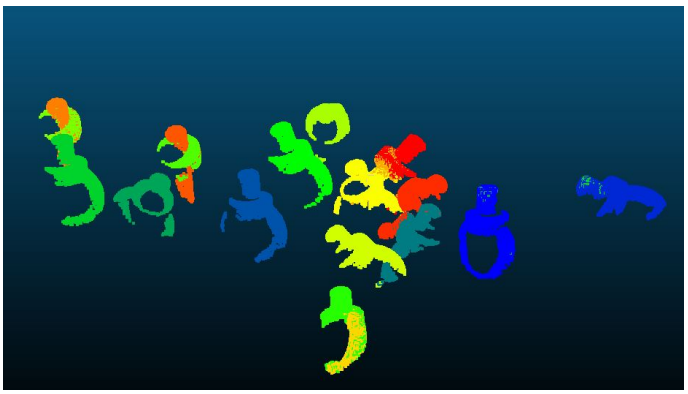


Figure 5.9: 3D Instance Segmentation of IPA Ring with 15 objects

customer data, which can't be demonstrated here. In the Fig. 5.10, the segmentation on real conrod plastic object with 9 instances is shown. The segmentation contains 15 instances.

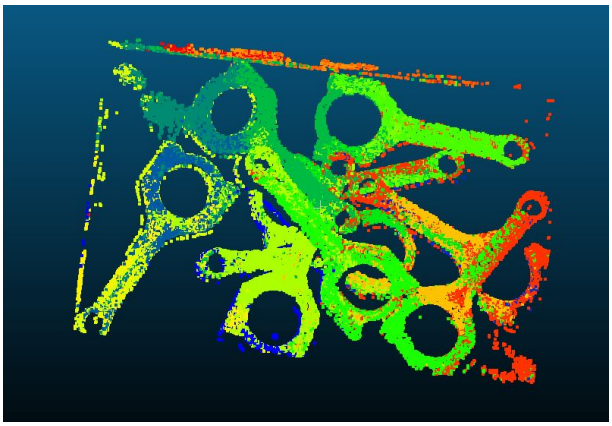


Figure 5.10: 3D Instance Segmentation of Real Plastic rod with 9 objects

5.2 Results for choosing Best Segmentation

As we observed in Fig. 5.6, the number of predicted instances is always higher than the actual number of instances. To chose the one instance from predicted instance we evaluate both the methods discussed previously on the validation data.

The average number of points and CAD model volume per object is calculated and displayed in the table 5.6.

Table 5.6: Average number of points per object

Object	avgPoints	CAD volume
Conrod Metal	2935	0.00034
Conrod Plastic	3061	0.00075
Gear Shaft	3221	0.00424
Ring	2106	0.00108

5.2.1 Results for method 1: Fixed thresholds

In this method, a threshold was set for both the number of points and the volume of the CAD model. The threshold for the number of points and CAD model volume per object can be referred at table 5.6. Considering an example of a conrod metal point cloud that contains 5 actual instances, the result of 3D instance segmentation is with segmentation id, number of points, and MBB volume of each predicted instance is shown in table 5.7.

Table 5.7: A sample 3D instance segmentation of conrod metal with 5 actual instances

Segmentation ID	nPoints	Instance volume
1	2849	0.00041
2	2702	0.00245
3	48	0.00039
4	1414	0.00070
5	3467	0.00105
6	561	0.00038
7	3086	0.00039
8	2361	0.00064

In the table 5.7 which demonstrates a sample segmentation of conrod metal, we observe that there are 8 predicted instances. Out of the 8 predicted instances, there are two segmentations (id: 5 and 7) which have more number of points than the average number of points but,

there are no segmentation which have equal or less volume than the CAD volume. Thus, we get no segmentation in this case to perform point cloud registration.

In general, out of 300 test point clouds for each object, 48, 56, 82, and 48 point clouds of conrod metal, conrod plastic, gear shaft, and ring respectively had the same issue. This was either due to none of the segmentation having more points than the average number of points or less than the volume of the CAD model.

5.2.2 Results for method 2: Scoring instances

The instance score calculation will help in choosing one segmentation to perform the point cloud registration with the CAD model as it will score every predicted instance. First, the score calculation was performed on the validation data. Based on the calculated scores for every point cloud of validation data, the minimum score and maximum score is taken to the normalize the score calculation.

The minimum and maximum instance score for every object is shown in table 5.8.

Table 5.8: Minimum and Maximum score from validation data

Object	min score	max score
Conrod Metal	-14.47	2.72
Conrod Plastic	-22.41	2.38
Gear Shaft	-15.25	2.63
Ring	-25.73	2.81

The sample score calculation for a point cloud with 16 objects is shown in table 5.9. The table is sorted on the score in descending order.

We observe that the number of predicted instances is 26 in this case and segmentation id 25 gets the highest score. This is because the number of points is higher than the average number of points and the difference between the volume of the predicted instance and the actual object is very low.

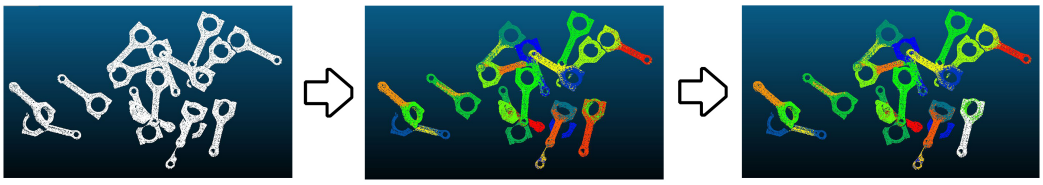


Figure 5.11: Choosing the best segmentation

The extraction of the best segmentation for the same point cloud with 16 actual instances is demonstrated in Fig. 5.11. The figure shows the 3D Instance segmentation of the input point cloud followed by selecting one predicted instance based on the score. The white-colored instance in the best segmentation part of Fig. 5.11 is the best segmentation with id 25.

Table 5.9: Normalized scores for all the predicted instances of a conrod plastic point cloud

Segmentation ID	nPoints	volume	dVolume	score
25	2614	0.000797	0.000038	0.940164
4	2163	0.000980	0.000221	0.908573
11	1829	0.000641	0.000118	0.903540
13	1884	0.000420	0.000339	0.888728
24	1908	0.001127	0.000369	0.887360
7	2242	0.001355	0.000597	0.882815
9	3238	0.002061	0.001302	0.867336
21	642	0.000719	0.000040	0.863430
17	723	0.000852	0.000093	0.862483
22	1566	0.001281	0.000522	0.862309
26	1471	0.001248	0.000490	0.861112
15	2476	0.001906	0.001148	0.849590
14	1615	0.001531	0.000772	0.845004
19	523	0.000502	0.000257	0.842159
20	618	0.000267	0.000492	0.827811
23	1080	0.001526	0.000767	0.824622
16	1523	0.001825	0.001067	0.818802
3	1591	0.002421	0.001663	0.775684
1	2163	0.002872	0.002113	0.763304
8	2769	0.003299	0.002540	0.754076
6	2299	0.003271	0.002512	0.737963
2	1199	0.002858	0.002100	0.726910
10	3805	0.004297	0.003539	0.717616
12	797	0.004511	0.003753	0.584379
18	1859	0.006362	0.005603	0.483551
27	1052	0.006937	0.006178	0.408052
5	1558	0.012328	0.011570	0.013742

5.2.3 Comparison of methods

The two implemented methods for choosing one segmentation is demonstrated qualitatively in the Fig. 5.12.

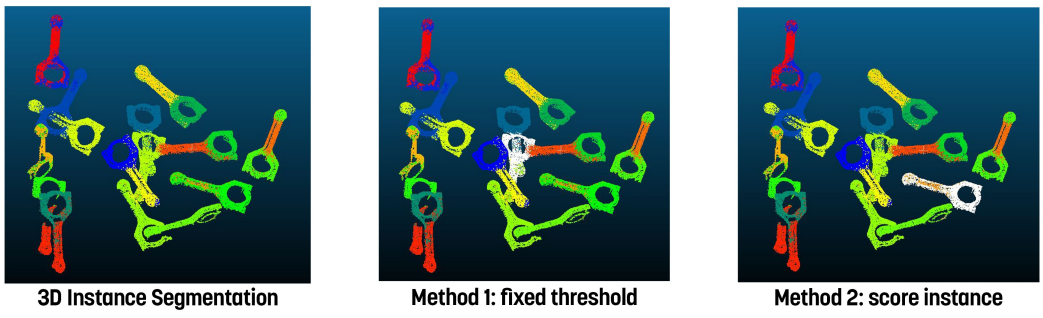


Figure 5.12: Comparison of methods for selecting one instance

In the Fig. 5.12, we observe that the segmentation chosen by both methods are different. The white points in the image signify the selected instance by the methods. Method 1 chooses an instance which has high number of points but, the instance is highly occluded and contains points from multiple actual instances. Method 2 selects an instance which is isolated and has good shape.

5.3 Result of Point Cloud Registration

5.3.1 Experimental Setup

PoinetNetLK trains all the objects at once. First, it trains for a classification network to distinguish between different objects, and then a registration network is trained to find the optimal transformation between the source and template point cloud. The classification network is trained for 70 epochs, while the registration network is trained for 50 epochs. In the training dataset, 2.5 D point cloud, 3D point cloud, and best segmentation from the 3D instance segmentation are taken for all 4 objects. Nvidia GTX 1080 GPU with 12GB RAM is used during the entire training and testing of the model. During the training phase of PoinetNetLK, the batch size is kept at 32 with Adam optimizer and a fixed delta (step size for Jacobian) of 0.001. Every batch takes 1024 unique points. Each training consumes around 2 hours with the above-mentioned parameter values.

5.3.2 Network Loss

The losses of the network seem to decrease normally and gets steady after some epochs. The best model is saved with the least loss. The line chart for the training and validation loss of different objects for both classification and registration is shown in the Fig. 5.13 and Fig. 5.14 respectively.

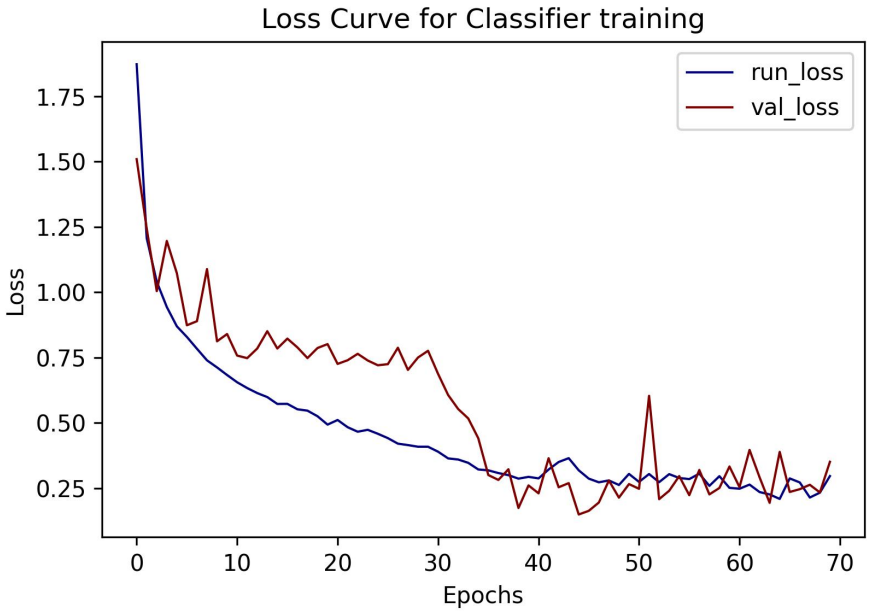


Figure 5.13: Training and validation loss during classification training

5.3.3 Difference in transformation

The logarithm of estimated transformation and ground truth is calculated to compare the difference between both matrices. This calculation results in a vector of length 6. For the estimated transformation the vector has $h_{w1}, h_{w2}, h_{w3}, h_{v1}, h_{v2}, h_{v3}$ and $g_{w1}, g_{w2}, g_{w3}, g_{v1}, g_{v2}, g_{v3}$ for ground truth. h_{w1}, h_{w2}, h_{w3} represents the rotation, h_{v1}, h_{v2}, h_{v3} represents the translation of estimated transformation. g_{w1}, g_{w2}, g_{w3} represents the rotation, g_{v1}, g_{v2}, g_{v3} represents the translation of ground truth transformation. The difference between the respective compo-

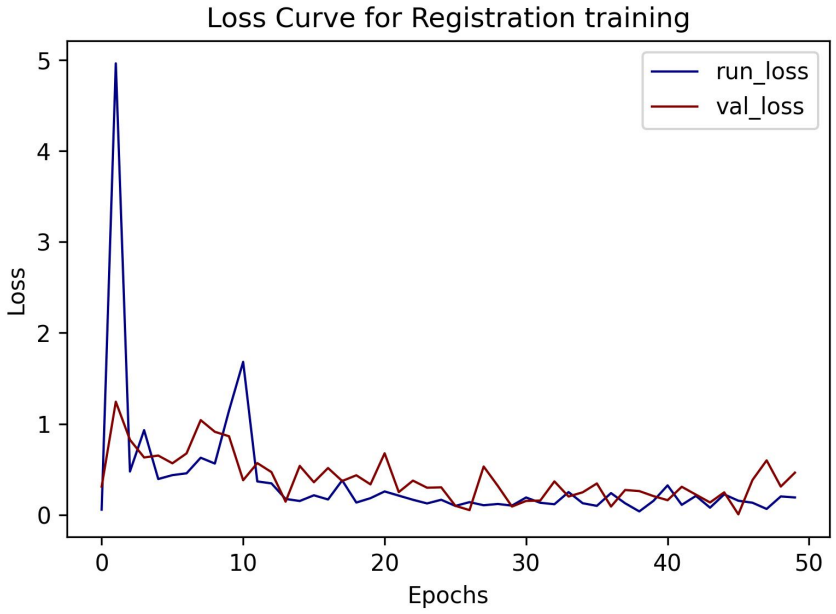


Figure 5.14: Training and validation loss during registration training

nents is taken to observe the deviation of our predicted transformation from the actual transformation.

The difference between the actual transformation and predicted transformation is calculated and presented in the Fig. 5.15.

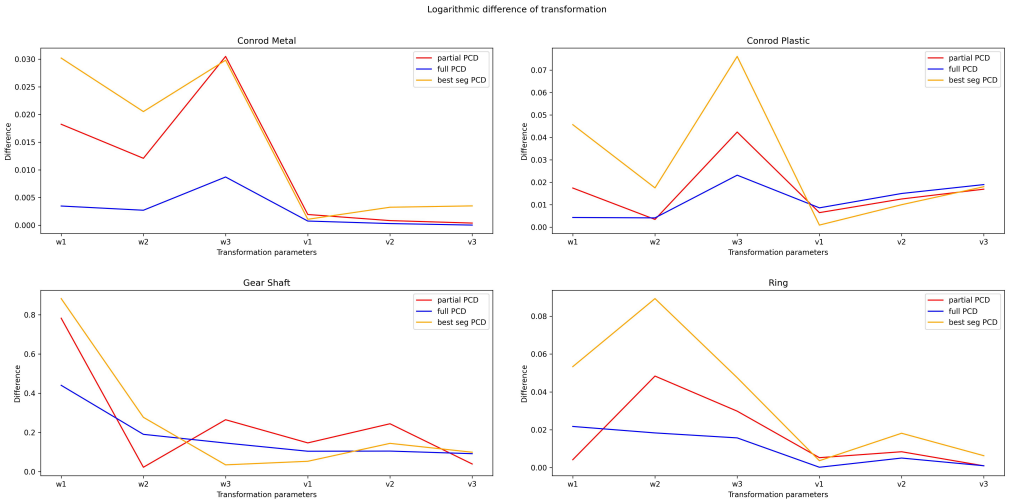


Figure 5.15: Difference in transformation

The x-axis in all the subplots is transformation parameters $(w1, w2, w3, v1, v2, v3)$. They represent the difference in the transformation for both rotation and translation. For all the objects, we observe that the difference in the positional parameters is very low, which signifies that the difference in translation between the source and template is very low. We also observe that the difference in the rotational part for all objects except the gear shaft is also very small. For all objects except the gear shaft, we see that the maximum difference is around 0.08.

5.3.4 Rotation-Translation Error Analysis

Several errors are computed to evaluate the trained model. *ep* represents the position error which is calculated as euclidean distance between the estimated position and ground truth position. It computes the accuracy of the estimated translation. *ew* represents the rotation error. It measures the accuracy of the estimated rotation. *ex* represents the twist error, which is a combination of both rotation and translation components. *ev* represents the translation error, which can be calculated as the euclidean norm of the translation from the twist error. It measures the accuracy of the estimated translation. For a perfect registration, all the above-mentioned errors should be zero.

The rotation-translation errors for all the objects are calculated and presented in the table below.

Table 5.10: Translation-Rotation Errors for Conrod Metal

Object type	ep	ex	ew	ev
Partial	0.38031	1.34632	1.37934	0.39500
Full	0.37912	1.30548	1.35808	0.37407
Best Seg	0.39700	1.36725	2.07961	0.39110

Table 5.11: Translation-Rotation Errors for Conrod Plastic

Object type	ep	ex	ew	ev
Partial	0.27434	1.797175	1.81321	0.29993
Full	0.27708	1.79090	1.79023	0.28256
Best Seg	0.28043	1.87993	1.85586	0.31800

Table 5.12: Translation-Rotation Errors for Gear Shaft

Object type	ep	ex	ew	ev
Partial	0.54907	3.01634	3.11720	0.786520
Full	0.52877	1.95358	2.04684	0.61082
Best Seg	0.54907	3.01635	3.11720	0.78652

Table 5.13: Translation-Rotation Errors for Ring

Object type	ep	ex	ew	ev
Partial	0.33294	2.87993	2.90534	0.38340
Full	0.33874	1.70664	1.74756	0.38849
Best Seg	0.33319	2.69094	2.71884	0.37597

In the table 5.10, 5.11, 5.12, and 5.13, all the errors related to translation and rotation for all objects are presented. All the objects with different source type (partial, full or best segmentation) are evaluated. We observe from all the tables that the error for position and translation is low, which signifies that source is well translated to template point cloud. We also observe that the errors related to rotation is slightly higher than the positional, this motivates that object is well-registered but not so accurate in every case.

5.3.5 Qualitative Analysis

After the deep evaluation on the difference in transformation and the error of every object type, it is interesting to observe the registration of these objects visually.

The source and template before point cloud registration is shown in Fig. 5.16.

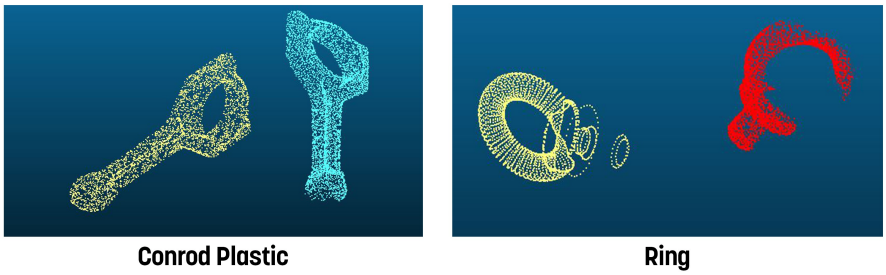


Figure 5.16: Point clouds before point cloud registration

In all the images, the yellow color point cloud is the CAD model (template) and the other color one is the source point cloud. Both the source and template point clouds before registration is seen to be clearly distant and with significant rotation difference.

Point clouds after performing the point cloud registration is demonstrated below.

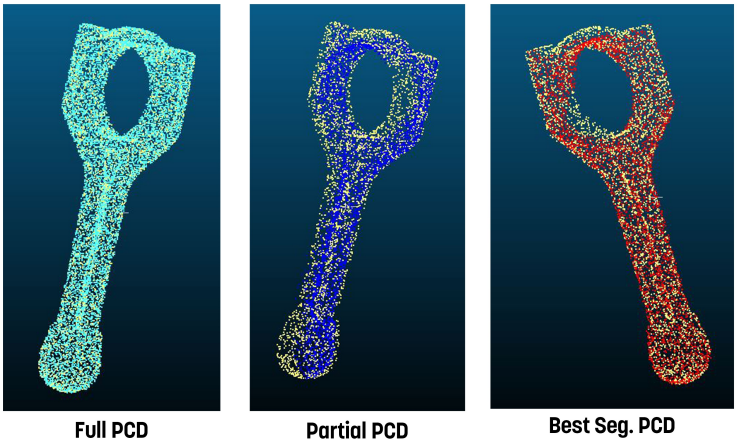


Figure 5.17: Point cloud registration on Conrod Plastic

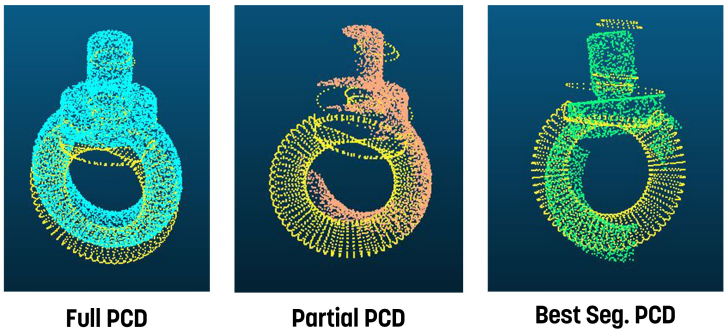


Figure 5.18: Point cloud registration on Ring

Fig. 5.17 and 5.18 shows the result of point cloud registration on conrod plastic and ring object respectively. Both images have 3 sub-images representing the 4 source type. The alignment of the two point clouds (source and template), motivates the quality of registration.

6 Discussion

This chapter focuses on discussing the results obtained, methods used, limitations, and ethical considerations. It also presents a comprehensive analysis of the results obtained from applying these methods, and their effectiveness in approaching our problem, i.e. robotic bin picking. After evaluating the results in the previous chapter, we are interested in gathering insights about the strength and weaknesses of the applied methods and discussing the scopes for further enhancements. The prime objective of this thesis was to research robust algorithms for 3D Instance segmentation and point cloud registration. Further, combine these two techniques by selecting one instance from the predicted instances efficiently to have a smooth point cloud registration.

6.1 Results

The result section presents the interpretation of the output generated by applying 3D instance segmentation using FPCC (Fast Point Cloud Clustering) and point cloud registration with PointNetLK on our data. The discussion on the results is done individually for 3D instance segmentation, selection of one instance, and point cloud registration.

6.1.1 3D Instance Segmentation

The results obtained from the implemented FPCC-net give some valuable insights into the performance and robustness of this method to tackle the challenging task of 3D Instance Segmentation. The implementation was evaluated using IoU, precision-recall, mean under-segmentation and mean over-segmentation. This was performed for 4 objects including the gear shaft, ring, conrod metal, and conrod plastic. The hyper-tuning of d_{max} on the validation data is well justified as the ring is the smallest object amongst all, while the gear shaft is the largest, thus a smaller value of $d_{max} = 0.07$ and a larger value of $d_{max} = 0.25$ for ring, gear shaft respectively. Based on the chosen parameters, the test dataset was evaluated 5.1, 5.2, 5.3, and 5.4. It was observed that the performance of the model doesn't vary a lot, which signified that the model is not over-fitting.

To further test the robustness and bring it as close as to real data, the trained model is also applied to the separate noisy datasets, which contain several levels of noise for all the objects. The performance on the noisy datasets is presented in the Fig. 5.2, 5.3, 5.4, and 5.5. It is

observed that the average precision of the model decreases in most cases with the increasing amount of noise, which is an expected behavior.

Table 5.5, presents the evaluation of conrod metal object with different noise percentages for 3 levels of standard deviation. It can be observed that the performance of the model degrades with the added noise, which is an expected behaviour. From table 5.3, 5.4 and 5.5, it is inferred that the performance on the noisy dataset is not significantly different from the non-noisy dataset. This motivates the trained model is performing well with noisy data.

The number of predictions to the number of actual instances analyses is also performed to test the trained mode. Since there were multiple point clouds with the same number of objects in the test set. Thus, the mean value of the number of predictions for each number of objects (ranging from 1 to 30) is calculated and presented in Fig. 5.6. It is observed that for all the objects except the gear shaft, the number of predictions is closer to the number of actual objects. It was also observed that the ratio of the number of actual objects to the ratio of the number of predictions decreased with the increasing number of objects.

In the qualitative analysis part, the instances can be clearly distinguished for all objects. The performance of the model on the real conrod plastic object looks similar, where the instances well-segmented. However, we also observe that a few instances are not very well segmented. This can impact the point cloud registration if chosen. Thus, a strategy is needed to choose the most suitable predicted instance.

6.1.2 Selecting one instance from prediction

Selecting one instance from the predicted instances is a crucial part of this research. A predicted instance with just a high number of points may not have a good shape like the actual object. It may include points from other actual instances and may affect our point cloud registration. Method 1 with fixed thresholds had several scenarios where no segmentation was selected because of hard thresholds.

Thus, scoring all the predicted instances based on the number of points and minimum bounding box volume is a better idea. A sample calculation of the score for choosing the one best instance out of all the predicted instances for the conrod plastic object is shown in 5.9. It was observed that the instances with a good number of points and low difference in the volume of minimum bounding boxes of predicted instance and CAD model gets the high score. It can also be observed that the predicted instance with id 21 has a very low difference in the volumes, but due to a very low number of points, it gets a low score. This motivates that the score calculation formula proposed in this research is a good approach to chose one segmentation. Fig. 5.11 demonstrates the selection of best segmentation for conrod plastic with 15 objects. We see that the white instance in the best segmentation image has the maximum score. This selection of instance is considered good as the implementation can detect the object with the least occlusion. This will not only provide ease at registration but also will be simple for the robot to pick.

6.1.3 Point Cloud Registration

For point cloud registration using PointNetLK, 3 different types of sources were taken to be registered with the CAD model. A partial point cloud is a 2.5D point cloud generated by the camera view during synthetic generation, a full point cloud is the complete point cloud of the same instance generated during the synthetic data generation and the best-segmented point cloud is the best segmentation chosen based on the score calculation after FPCC implementation. In Fig. 5.15, it can be observed that the translation of source point cloud is well estimated. In the table 5.10, 5.11, 5.12, and 5.13, where the rotation translation analysis is performed it is also interesting to observe that the performance is better with the full point cloud, which is expected as the full point cloud has better shape and more concrete features

to get aligned to the template point cloud. It is also important to notice that the performance of the model doesn't vary much between the different types of sources.

6.2 Limitations

While the results from FPCC and PointNetLK have shown some good results for 3D instance segmentation and point cloud registration, it is also important to describe the limitations of the methods. In this section, we describe key aspects where these methods may face difficulties.

During the clustering phase of 3D instance segmentation, the pairwise distance calculation is performed, which is very demanding. FPCC provides a very tailored solution based on objects, which means the same model will not perform well on different objects. Thus, separate training is needed for every object. This also signifies that the method is sensitive to parameter values. FPCC first finds the potential centers for the instances and then clusters. So, in cases when the geometrical centers of the object are missing it chooses a point on the object somewhere which is not a geometrical center as the center of the instance.

The selection of one instance from the predicted instances has shown some good results. However, this method is very sensitive to outliers. In case a few extra points outside the predicted instance, will result in a high volume of the minimum bounding box, which in turn will give a bad score. Thus, this solution is not robust to the outliers.

For point cloud registration using PointNetLK, the architecture is computationally expensive. For larger point clouds, computational time and resource requirements increase a lot. The model is not very robust to the outliers as well. Also, every point is treated equally, thus information like local density is neglected.

Also, the real image provided by SICK had approximately 400,000 points in every point cloud and the data wasn't labeled. In general, labeling the point cloud data is very impractical as the point clouds have a huge number of points and several point clouds make a dataset.

6.3 Ethical Considerations

When working with point cloud segmentation and registration-related projects, there can be various ethical considerations that need to be taken care of to ensure that the project is developed responsibly and ethically. First, ethical considerations can be privacy. As these projects may capture images inside industries or factories with a camera, it is important to consider the privacy of the people involved. This leads to creating anonymous data, which may demand taking consent from every individual if captured or blurring faces.

The safety of the system is another important aspect here. Bin-picking applications may have a high risk of injuring the human workforce. Thus, the system should be designed with proper safety measures, including emergency stop buttons or safety barriers.

Ethical considerations such as transparency and accountability are also important. The algorithms, source of data, and decision-making process should be kept transparent. The developers should hold accountability for any unethical behavior in the workflow which is generated as an outcome of this research.

Overall, these ethical considerations promote fairness, safety, accountability, and privacy which will result in carrying out this project in an ethical environment.

7 Conclusion

This thesis focused on adapting state-of-art methods in 3D instance segmentation and point cloud registration and combining them to build an efficient technique for robotic bin picking. At first, to detect individual objects from the bin, FPCC (fast point cloud clustering) network is trained with the company's data. The model was trained to fit the data available. The trained model was robust in detecting the individual instances from the bin. The model was also applied to the real dataset provided by SICK. Based on the qualitative analysis, the model seemed to have a similar performance. Further, to select one instance from the predicted instances, a score calculation is proposed, where every predicted instance gets a score based on shape and density. With the selected instance, another state-of-the-art method, PointNetLK is adapted to perform the point cloud registration. This task also showed promising results with close to accurate registrations. This thesis also aims to answer below research questions:

- *How to choose one instance from the predicted instances after 3D instance segmentation?* We observed that the number of prediction of instances is approximately 1.5 times the actual number of instances. It is crucial to decide on one instance, as it will have an impact on the registration in the next step. A scoring mechanism is proposed in this thesis to decide upon the best segmentation. Two things were considered: the shape and density(number of points) of the instance. To validate the shape, the minimum bounding box of the instance with highest score is calculated. To validate the density, the number of points in the selected instance was considered. Unlike keeping a threshold for both density and volume, which removes many predicted instances, this method keeps all the predicted instances and scores them. Hard thresholds may result in no best segmentation, while this approach guarantees one. It was also observed that the high score is given to the instances which are out of high occlusion areas, which is good for the robotic bin picking.

- *What is the impact on the performance of 3D instance segmentation with the added noise on synthetic data?*

  Noisy datasets were synthetically generated, to test the robustness of the trained model with respect to the noisy data. It was observed that the performance degrade for all the objects as the noise level increased, except for the ring object. The FPCC-net performs best when the object is not hollow at the geometrical center. In the case of the ring object,

where there are no points at the geometrical center, with the added noise it observes some points close to the geometrical center. Thus, when the noise levels increase, the center prediction shifts towards the geometrical center and the performance for the ring object continuously increase with added noise. It is also important to note that there is not a significant decrease in the performance, which makes the model robust towards the noise and motivates for a good performance on the real dataset.

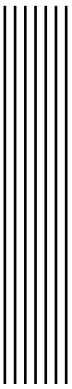
- *What is the change in performance when a partial point cloud and a full point cloud are registered with the CAD model of the objects?*

  Point cloud registration with PointNetLK has shown robust performance. It was observed that the alignment in the case of conrod metal and conrod plastic is very close to accurate. This registration task was performed for 3 different types of source point clouds (partial, full, and best segmentation). The model has shown the best performance with the full point cloud, while with the partial and best segmentation, it has shown similar performance. Overall, there is not a significant difference in the performance among the three. It was also observed that in some cases the translation of the source point cloud was very well estimated, while the rotation wasn't up to the mark. This mostly was observed when the number of points wasn't enough in the source point cloud or the CAD model had lacking points from the edges.

7.1 Future Work

This section proposes potential directions for further exploration and improvements. This is based on the results obtained and the limitations of the current work. In this section, we discuss the possible advancements that could enhance the performance of the current implementation.

1. Hyper-tune all the parameters for both methods (FPCC and PointNetLK) to improve the overall performance.
2. As the score is in the decreasing order for all the predicted instances, a threshold is needed for the instance score after which all the points of the selected instances will be removed and a re-segmentation will be performed.
3. Feature extraction is performed twice, one in each 3D instance segmentation and point cloud registration. A common feature extraction will reduce the computation time at large.



Bibliography

[1] Gabriel Agamennoni, Simone Fontana, Roland Siegwart, and Domenico Sorrenti. "Point Clouds Registration with Probabilistic Data Association". In: Oct. 2016, pp. 4092–4098. DOI: 10.1109/IROS.2016.7759602.

[2] Yasuhiro Aoki, Hunter Goforth, Rangaprasad Arun Srivatsan, and Simon Lucey. *PointNetLK: Robust Efficient Point Cloud Registration using PointNet*. 2019. arXiv: 1903.05711 [cs.CV].

[3] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. *ShapeNet: An Information-Rich 3D Model Repository*. 2015. arXiv: 1512.03012 [cs.GR].

[4] Liang-Chieh Chen, Alexander Hermans, George Papandreou, Florian Schroff, Peng Wang, and Hartwig Adam. *MaskLab: Instance Segmentation by Refining Object Detection with Semantic and Direction Features*. 2017. arXiv: 1712.04837 [cs.CV].

[5] Yiting Chen and Miao Li. *Data Generation for Learning to Grasp in a Bin-picking Scenario*. 2021. arXiv: 2108.06532 [cs.RO].

[6] Christopher Choy, Wei Dong, and Vladlen Koltun. "Deep Global Registration". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 2511–2520. DOI: 10.1109/CVPR42600.2020.00259.

[7] Özgün Çiçek, Ahmed Abdulkadir, Soeren S. Lienkamp, Thomas Brox, and Olaf Ronneberger. *3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation*. 2016. arXiv: 1606.06650 [cs.CV].

[8] Nikolaus Correll, Kostas E. Bekris, Dmitry Berenson, Oliver Brock, Albert Causo, Kris Hauser, Kei Okada, Alberto Rodriguez, Joseph M. Romano, and Peter R. Wurman. "Analysis and Observations From the First Amazon Picking Challenge". In: *IEEE Transactions on Automation Science and Engineering* 15.1 (2018), pp. 172–188. DOI: 10.1109/TASE.2016.2600527.

[9] Chunchao Dong, Naijian Chen, Yahui Li, and Jiawei Bao. "Point Cloud Segmentation Algorithm Based on Deep Learning and 3D Reconstruction". In: *2022 IEEE 5th Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*. Vol. 5. 2022, pp. 476–480. DOI: 10.1109/IMCEC55388.2022.10019815.

[10] Zhikai Dong, Sicheng Liu, Tao Zhou, Hui Cheng, Long Zeng, Xingyao Yu, and Houde Liu. "PPR-Net:Point-wise Pose Regression Network for Instance Segmentation and 6D Pose Estimation in Bin-picking Scenarios". In: *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2019, pp. 1773–1780. DOI: 10.1109/IROS40897.2019.8967895.

[11] Xinjian Fan, Xuelin Wang, and Yongfei Xiao. "A combined 2D-3D vision system for automatic robot picking". In: *Proceedings of the 2014 International Conference on Advanced Mechatronic Systems*. 2014, pp. 513–516. DOI: 10.1109/ICAMechS.2014.6911599.

[12] Ross Girshick. *Fast R-CNN*. 2015. arXiv: 1504.08083 [cs.CV].

[13] Aleksey Golovinskiy, Vladimir G. Kim, and Thomas Funkhouser. "Shape-based recognition of 3D point clouds in urban environments". In: *2009 IEEE 12th International Conference on Computer Vision*. 2009, pp. 2154–2161. DOI: 10.1109/ICCV.2009.5459471.

[14] Jiaxin Guo, Lian Fu, Mingkai Jia, Kaijun Wang, and Shan Liu. *Fast and Robust Bin-picking System for Densely Piled Industrial Objects*. 2020. arXiv: 2012.00316 [cs.RO].

[15] Naufal Muhammad Hirzi, Muhammad Anwar Ma'sum, Mahardhika Pratama, and Wisnu Jatmiko. "Large-scale 3D Point Cloud Semantic Segmentation with 3D U-Net ASPP Sparse CNN". In: *2022 7th International Workshop on Big Data and Information Security (IWBIS)*. 2022, pp. 59–64. DOI: 10.1109/IWBIS56557.2022.9924988.

[16] Jan Hosang, Rodrigo Benenson, and Bernt Schiele. *Learning non-maximum suppression*. 2017. arXiv: 1705.02950 [cs.CV].

[17] Xiaoshui Huang, Guofeng Mei, Jian Zhang, and Rana Abbas. *A comprehensive survey on point cloud registration*. 2021. arXiv: 2103.02690 [cs.CV].

[18] Kyekyung Kim, Joongbae Kim, Sangseung Kang, Jaehong Kim, and Jaeyeon Lee. "Vision-based bin picking system for industrial robotics applications". In: *2012 9th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*. 2012, pp. 515–516. DOI: 10.1109/URAI.2012.6463057.

[19] Kilian Kleeberger, Christian Landgraf, and Marco F. Huber. *Large-scale 6D Object Pose Estimation Dataset for Industrial Bin-Picking*. 2019. arXiv: 1912.12125 [cs.CV].

[20] Tuan-Tang Le and Chyi-Yeu Lin. "Bin-Picking for Planar Objects Based on a Deep Learning Network: A Case Study of USB Packs". In: *Sensors* 19.16 (2019). ISSN: 1424-8220. DOI: 10.3390/s19163602.

[21] Deping Li, Hanyun Wang, Ning Liu, Xiaoming Wang, and Jin Xu. "3D Object Recognition and Pose Estimation From Point Cloud Using Stably Observed Point Pair Feature". In: *IEEE Access* 8 (2020), pp. 44335–44345. DOI: 10.1109/ACCESS.2020.2978255.

[22] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. *PointCNN: Convolution On $\mathcal{X}$-Transformed Points*. 2018. arXiv: 1801.07791 [cs.CV].

[23] Ming Liang, Bin Yang, Shenlong Wang, and Raquel Urtasun. *Deep Continuous Fusion for Multi-Sensor 3D Object Detection*. 2020. arXiv: 2012.10992 [cs.CV].

[24] Diyi Liu, Shogo Arai, Zhuang Feng, Jiaqi Miao, Yajun Xu, Jun Kinugawa, and Kazuhiro Kosuge. "2D Object Localization Based Point Pair Feature for Pose Estimation". In: *2018 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. 2018, pp. 1119–1124. DOI: 10.1109/ROBIO.2018.8665097.

[25] Diyi Liu, Shogo Arai, Fuyuki Tokuda, Yajun Xu, Jun Kinugawa, and Kazuhiro Kosuge. "Deep-Learning based Robust Edge Detection for Point Pair Feature-based Pose Estimation with Multiple Edge Appearance Models". In: *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. 2019, pp. 2920–2925. DOI: 10.1109/ROBIO49542.2019.8961752.

[26] Weixin Lu, Yao Zhou, Guowei Wan, Shenhua Hou, and Shiyu Song. "L3-Net: Towards Learning Based LiDAR Localization for Autonomous Driving". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 6382–6391. DOI: 10.1109/CVPR.2019.00655.

[27] Carlos Martinez, Heping Chen, and Remus Boca. "Automated 3D vision guided bin picking process for randomly located industrial parts". In: *2015 IEEE International Conference on Industrial Technology (ICIT)*. 2015, pp. 3172–3177. DOI: 10.1109/ICIT.2015.7125566.

[28] Joseph O'Rourke. "Finding minimal enclosing boxes". In: *International Journal of Computer & Information Sciences* 14.3 (June 1985), pp. 183–199. ISSN: 1573-7640. DOI: 10.1007/BF00991005.

[29] G. Dias Pais, Srikumar Ramalingam, Venu Madhav Govindu, Jacinto C. Nascimento, Rama Chellappa, and Pedro Miraldo. *3DRegNet: A Deep Neural Network for 3D Point Registration*. 2020. arXiv: 1904.01701 [cs.CV].

[30] Dhara Patel and Saurabh Upadhyay. "Optical Flow Measurement using Lucas Kanade Method". In: *International Journal of Computer Applications* 61 (Jan. 2013), pp. 6–10. DOI: 10.5120/9962-4611.

[31] Ales Pochyly, Tomas Kubela, Vladislav Singule, and Petr Cihak. "3D vision systems for industrial bin-picking applications". In: *Proceedings of 15th International Conference MECHATRONIKA*. 2012, pp. 1–6.

[32] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. *PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation*. 2017. arXiv: 1612.00593 [cs.CV].

[33] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. *PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space*. 2017. arXiv: 1706.02413 [cs.CV].

[34] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. *You Only Look Once: Unified, Real-Time Object Detection*. 2016. arXiv: 1506.02640 [cs.CV].

[35] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. "Generalized Intersection Over Union: A Metric and a Loss for Bounding Box Regression". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 658–666. DOI: 10.1109/CVPR.2019.00075.

[36] Radu Bogdan Rusu, Zoltan Csaba Marton, Nico Blodow, Mihai Dolha, and Michael Beetz. "Towards 3D Point Cloud Based Object Maps for Household Environments". In: *Robotics and Autonomous Systems Journal (Special Issue on Semantic Knowledge in Robotics)* 56.11 (30 November 2008), pp. 927–941.

[37] Jane Shi and Gurdayal S. Koonjul. "Real-Time Grasping Planning for Robotic Bin-Picking and Kitting Applications". In: *IEEE Transactions on Automation Science and Engineering* 14.2 (2017), pp. 809–819. DOI: 10.1109/TASE.2017.2671434.

[38] Tao Song, Bing Chen, Fu M. Zhao, Zheng Huang, and Mu J. Huang. "Research on image feature matching algorithm based on feature optical flow and corner feature". In: *The Journal of Engineering* 2020.13 (2020), pp. 529–534. DOI: https://doi.org/10.1049/joe.2019.1156.

[39] Zejia Su, Haibin Huang, Chongyang Ma, Hui Huang, and Ruizhen Hu. *Point cloud completion via structured feature maps using a feedback network*. 2022. arXiv: 2202.08583 [cs.CV].

[40] Viacheslav Tarasenko and Dong-Won Park. "Detection and Tracking over Image Pyramids using Lucas and Kanade Algorithm". In: 2016.

[41] Hugues Thomas, Charles R. Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J. Guibas. *KPConv: Flexible and Deformable Convolution for Point Clouds*. 2019. arXiv: 1904.08889 [cs.CV].

[42] Pedro Torres, Janis Arents, Hugo Marques, and Paulo Marques. "Bin-Picking Solution for Randomly Placed Automotive Connectors Based on Machine Learning Techniques". In: *Electronics* 11.3 (2022). ISSN: 2079-9292. DOI: 10.3390/electronics11030476.

[43] Chaoyang Wang, Hamed Kiani Galoogahi, Chen-Hsuan Lin, and Simon Lucey. "Deep-LK for Efficient Adaptive Object Tracking". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. 2018, pp. 627–634. DOI: 10.1109/ICRA.2018.8460815.

[44] Xinlong Wang, Shu Liu, Xiaoyong Shen, Chunhua Shen, and Jiaya Jia. *Associatively Segmenting Instances and Semantics in Point Clouds*. 2019. arXiv: 1902.09852 [cs.CV].

[45] Yue Wang and Justin M. Solomon. *Deep Closest Point: Learning Representations for Point Cloud Registration*. 2019. arXiv: 1905.03304 [cs.CV].

[46] Yue Wang and Justin M. Solomon. *PRNet: Self-Supervised Learning for Partial-to-Partial Registration*. 2019. arXiv: 1910.12240 [cs.LG].

[47] Yajun Xu, Shogo Arai, Diyi Liu, Fangzhou Lin, and Kazuhiro Kosuge. "FPCC: Fast point cloud clustering-based instance segmentation for industrial bin-picking". In: *Neurocomputing* 494 (July 2022), pp. 255–268. DOI: 10.1016/j.neucom.2022.04.023.

[48] Jiaolong Yang, Hongdong Li, Dylan Campbell, and Yunde Jia. "Go-ICP: A Globally Optimal Solution to 3D ICP Point-Set Registration". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38 (2016), pp. 2241–2254.

[49] Chenke Yue, Yong Wang, Xintong Tang, and Qiuyi Chen. "DRGCNN: Dynamic region graph convolutional neural network for point clouds". In: *Expert Systems with Applications* 205 (2022), p. 117663. ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2022.117663.

[50] Chungang Zhuang, Zhe Wang, Heng Zhao, and Han Ding. "Semantic part segmentation method based 3D object pose estimation with RGB-D images for bin-picking". In: *Robotics and Computer-Integrated Manufacturing* 68 (2021), p. 102086. ISSN: 0736-5845. DOI: https://doi.org/10.1016/j.rcim.2020.102086.