A Deep Learning Network for Validation of 3D Point Cloud Surface Registration

By Joy Mazumder

A thesis submitted to the Graduate Program in Department of Electrical and Computer Engineering in conformity with the requirements for the Degree of Master of Applied Science

Queen’s University
Kingston, Ontario, Canada
September, 2020

Copyright © Joy Mazumder, 2020
Abstract

This thesis proposes a novel deep learning architecture called ValidNet to automatically validate the correctness of a 3D surface registration outcome. The performance of many tasks such as object detection mainly depends on the applied registration algorithms, which themselves are susceptible to local minima. Revealing this tendency and verifying the success of registration algorithms is a difficult task. We treat this as a classification problem, and propose a two-class classifier to distinguish clearly between true positive and false positive registration outcomes. Our proposed ValidNet system deploys a shared multilayer perceptron architecture which works on the raw and unordered point cloud data of scene and model points. This network is able to perform the two fundamental tasks of feature extraction and similarity matching using the powerful capability of a deep neural network. Experiments on a large synthetic dataset show that the proposed method can effectively be used in automatic validation of 3D surface registration.
Acknowledgments

I wish to express my sincere gratitude towards my thesis advisor, Prof. Michael Greenspan, without whom this research would not be possible. His support, patience, encouragement, and expertise guided me throughout this journey. I would also like to thank my lab mates, who had been an incredible source of inspiration throughout my Master’s. Finally, I would like to express my love for my family who had been my backbone throughout this arduous journey. This dissertation is dedicated to my parents, without whom I would not be the person I am today.
Statement Of Originality

The following works is my own and I hereby certify the intellectual content of this thesis is the product of my own work. All references and contributions of other individuals has been cited and sourced appropriately, as defined by the IEEE Citation Reference manual. An earlier version of this work was published in 16th International Conference on Computer Vision Theory and Applications (VISAPP), 2020 in Valletta, Malta, in a paper entitled ”ValidNet: A Deep Learning Network for Validation of Surface Registration” [1].
Contents

Abstract i
Acknowledgments ii
Statement Of Originality iii
Table of Contents iv
List of Tables vii
List of Figures viii
Glossary of Terms xii
Glossary of Symbols xiii
Glossary of Abbreviations xiv

Chapter 1: Introduction 1
1.1 Introduction .................................................. 1
1.2 Contributions ............................................... 4
1.3 Outline ................................................... 5
Chapter 2: Background

2.1 Point Cloud Registration ........................................ 7
   2.1.1 Classical Registration ..................................... 7
   2.1.2 Deep Learning Based Registration ......................... 12

2.2 Validation ......................................................... 14

Chapter 3: Problem Statement

3.1 Point Cloud Definition ........................................... 19

3.2 Iterative Closest Point Algorithm (ICP) ......................... 20
   3.2.1 Local Minima Problem of ICP ............................ 24

3.3 Limitation of RMSD Error and Residual Statistics for Identifying Local
   Minima .................................................................... 26

3.4 Overview of Approach to Identify Local Minima ................. 31

Chapter 4: Methodology

4.1 Shared MLP ........................................................ 34

4.2 Symmetric Function ............................................... 37

4.3 PointNet for Point Cloud Classification and Segmentation ...... 37

4.4 ValidNet ................................................................ 44
   4.4.1 ValidNet Feature Extraction Network ..................... 45
   4.4.2 Similarity Measurement Network .......................... 47

Chapter 5: Dataset Generation

5.1 Synthetic Bin Generation ........................................ 53

5.2 Prepossessing ....................................................... 54

5.3 Segmentation ......................................................... 56
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>Different levels of matching precision</td>
<td>68</td>
</tr>
<tr>
<td>6.1</td>
<td>Performance comparison on our dataset. ACA represents average class accuracy and OA represents overall accuracy</td>
<td>72</td>
</tr>
<tr>
<td>6.2</td>
<td>Performance of various feature extractor on our proposed architecture</td>
<td>83</td>
</tr>
<tr>
<td>6.3</td>
<td>Effect of color information on our proposed architecture</td>
<td>85</td>
</tr>
</tbody>
</table>
# List of Figures

1.1 Overview of ValidNet. (We propose a deep neural network that takes two raw point clouds (model and segment) from a registration technique as input and provides validity of the surface registration) [1]  

1.2 Synthetic Point Cloud Dataset. ........................................ 3

3.1 ICP First Iteration: (a) Initial transformation estimate, (b) Correspondences, (c) Application of calculated transformation. ................. 22

3.2 Two ICP Iterations .......................................................... 23

3.3 ICP local and global minimum and their relationship with initial pose. 25

3.4 Global and local minimum convergence of ICP ........................ 26

3.5 Convergence of ICP when source and target surface are different from each other ........................................................ 27

3.6 RMSD Error for Global ((a),(c)) and Local ((b),(d)) Minima .... 28

3.7 RMSD Error for Global ((a),(c)) and Local ((b),(d)) Minima .... 29

4.1 Perceptron (Green Circle represent summation and Non-linear operation) 35

4.2 Shared perceptron. Same weights and bias is used over the all points in the point cloud ........................................................ 36

4.3 1st layer of shared perceptron, \( r \) number of perceptron with same weight and bias is operating on an input sample .................... 38
4.4 2nd layer of shared perceptron, s number of perceptron with same weight and bias is operating on the first layer output.

4.5 *PointNet* for 3D Object Classification and semantic segmentation [2]  
[© 2017 IEEE] .......................................................... 40

4.6 T-Net architecture [3] .................................................. 41

4.7 *ValidNet* for registration validation. The left part is the feature learning network that learns the features of the model and the segment and the right part is the similarity finding network that measures the similarities between these features and provides the validation probability of registration success [1] ........................................... 44

4.8 Shared MLP on *ValidNet* .................................................. 46

5.1 Models for synthetic data generation ................................. 52

5.2 Data generation pipeline .................................................. 53

5.3 Synthetic Point Cloud Bin ................................................. 54

5.4 Comparison between real and synthetic data [1] .................. 55

5.5 Region Growing Segmentation Results ............................... 58

5.6 Neighbourhood points of a query. (The connections with marked 2 indicates that it will contribute twice in final FPFH feature calculation) [4] [© 2009 IEEE] .................................................. 61

5.7 uvw representation (based on [5]) ................................. 62

5.7 Output of RANSAC-based registration ............................... 65

6.1 *ValidNet* training setup ................................. 70
6.2 PointNet for registration validation. It treats the model and the segment point clouds as a single point cloud, captures their global relationship and provides the validation probability of registration success 72

6.3 Some samples where ValidNet classified correctly, with segment points rendered black, and model points rendered green. 74

6.4 Some samples where ValidNet failed to classify correctly, with segment points rendered black, and model points rendered green. 75

6.5 Use of ValidNet for 3D object detection and pose estimation bin picking pipeline, with detected objects overlayed in green. ValidNet effectively detects and removes false positive (i.e. inaccurate) results. 76

6.6 Segments with noise. Same segment from left to right with Gaussian noise mean 0 and S.D. 0, 0.5, 1, 1.5, 2, 2.5, 3 respectively 77

6.7 Effect of noise when trained on noiseless data and tested on noisy data. (Here, the random guess of ACA is 50% as it’s a two class classification problem) 78

6.8 Effect of noise when noise is added on the training data as well 80

6.9 Segments with missing data. Same segment from left to right 0%, 25%, 50%, 75%, 87.5%, 93.75% points are missing respectively 81

6.10 Effect of missing data on ValidNet and PointNet 82

6.11 Relative comparison of Average Class Accuracy as a function of Noise S.D, for various symmetric functions. ValidNet_X_Y means symmetric function X is used in the feature extraction network, and symmetric function Y is used in the similarity measurement network. 84
6.12 Effect of number of model points and segment points on \textit{ValidNet}.

Here, we keep the number of model and segment points same . . . . . 86

6.13 Heat Map Visualization of \textit{ValidNet} Output . . . . . . . . . . . . . 87
Glossary of Terms

**convergence basin** The range of initial poses for which ICP will converge to a particular minima. 8, 24–27

**local descriptor** A local descriptor of a point contains meaningful information to represent the local structure near the point. 9, 10, 59
Glossary of Symbols

M Model Point Cloud. 31, 33, 44

m Total number of model points. 44, 68, 85, 86

n Total number of segment points. 44, 68, 85, 86

S Segment Point Cloud. 31

L Class Label (L = 1 means positive class and L = 0 means negative class). 67, 69

p_q query point. 10, 56, 57

Θ th Angle Threshold. 67, 83

D th Distance Threshold. 67, 83

T Transformation Matrix. 21, 31

TM Transformed Model. 31

T^{−1}S Inverse Transformed Segment. 31, 33, 44
Glossary of Abbreviations

**3DSC** 3D shape contexts. 10

**CT** Computed Tomography. 15

**FDN** Fixed distance neighbors. 57

**FPFH** Fast Point Feature Histogram. 10, 13, 59, 60, 62, 63

**Go-ICP** Globally Optimal Iterative Closest Point. 12, 14

**ICP** Iterative Closest Point. 1, 5, 7–9, 11, 14, 21, 24–27, 30, 31, 59, 63, 89

**KNN** K Nearest Neighbors. 57

**MLP** Multi Layer Perceptron. 41–45, 47, 69, 89

**MR** Magnetic Resonance. 15

**PFH** Point Feature Histogram. 10, 59, 60

**PPF** Point Pair Feature. 11

**RANSAC** Random Sample Consensus. 9, 56, 59, 63
**RMSD** Root Mean Square Distance. 5, 27, 30, 89

**RSD** Radius based Surface Descriptors. 10
Chapter 1

Introduction

1.1 Introduction

The rigid registration of 3D surfaces is of central importance to the processing of 3D data. Since the initial introduction of the Iterative Closest Point (ICP) algorithm in the late 1990’s [6], there have been many variations proposed to the basic approach to improve robustness, efficiency and generality [7,8]. Current research aims to address non-rigid registration [9,10] and investigate machine learning approaches [11,12].

The basic approach to rigid 3D surface registration applies local optimization, and therefore depends upon initial conditions to resolve to a correct solution. In the case of ICP and its variants, an initial estimate of the true rigid transformation needs to be provided which is “close enough” to the true solution, lest the result be driven to a false positive (i.e., local vs. global minimum). It is therefore expected that a 3D surface registration task will sometimes fail, by returning a false positive. This failure mode is often difficult to recognize and diagnose, as the distribution of the surface residuals (i.e., distances between corresponding surface points and/or surfaces) between false positives and true positives can be similar.
1.1. INTRODUCTION

Figure 1.1: Overview of ValidNet. (We propose a deep neural network that takes two raw point clouds (model and segment) from a registration technique as input and provides validity of the surface registration) [1]

There are some systems where a human is in the loop to inspect the results from a registration task, in which case the occasional failure may be a nuisance, but is tolerable. There are also other desired uses of 3D surface registration, where registration failure is less tolerable, and can lead to a catastrophic system failure. One such desired use is automated robot grasping and manipulation, which are common tasks in industrial automation [13]. In this critical application, false positives in registration can result in damage to equipment, as the robot attempts to grasp a part that was incorrectly identified and/or localized. Even small error rates of a few percentage points, which are commonly accepted in many general approaches, can be unacceptable in this automation process.
This thesis proposes a new approach for automatic validation of registration algorithms. In our method, we focus on the validation of 3D registration results for rigid parts. A 3D segmentation algorithm is initially used to extract individual object instances from a scene. A registration algorithm is then used to localize object hypotheses in the scene. Each hypothesis is thus a segmented object matched with
1.2. CONTRIBUTIONS

the object model. Our algorithm categorizes each hypothesis as a *true positive* or *false positive*. In particular, the raw points of each segment and the object model are considered as the inputs to the network. The proposed network, called *ValidNet*, applies the powerful capability of a Deep Neural Network (DNN). This network can effectively extract required features and match similarities between them. Thus, this architecture can learn to extract features and classify validation hypotheses from the raw numeric data. An overview of the proposed method is shown in Figure 1.1.

Intuitively, a reliable validation for surface registration can be useful for general pose determination. However, the proposed method is evaluated for the time-sensitive application of robotic bin picking to emphasize its efficiency in real-time scenarios. It is well known that a large number (typically thousands to millions) of image data are required to train DNNs, and such a volume of ground truth datasets of real images are not available for this task, and would be expensive to generate. We therefore use a simple method to generate synthetic datasets for the bin picking problem. Some samples of the synthetic 3D images of bin picking scenes that are representative as input to the system are illustrated in Figure 1.2. Similar to the real-world data, the generated synthetic datasets contain occlusion, clutter, noise, and missing information (dropouts), and ground truth samples can be quickly and easily generated in high volume for training purposes.

1.2 Thesis Contributions

The main contributions of this thesis are as follows:

- We propose a novel and first deep neural architecture which can distinguish between correct and incorrect registration results of 3D objects, for robotic bin picking and other applications of 3D surface registration;
1.3. OUTLINE

- We propose a novel similarity measurement network which can measure similarities of the registered model and segment in feature space;

- We have developed and conducted multiple experiments to characterize our proposed method;

- We have designed a large 3D object detection and pose estimation dataset for robotic bin picking. The dataset is available at RCV lab, Queen’s University website.

- We have experimented with three different matching precision level on this dataset and were able to achieve more than 89.70% for all the precision level. We have also showed the robust behaviour of our method when there will be sensor noise and missing data.

1.3 Thesis Outline

The remainder of this thesis is organized as follows:

**Chapter 2, Background:** This chapter provides explanations of previous works of 3D surface registration techniques (both classical and deep learning) and the past efforts to identify the false convergence.

**Chapter 3, Problem Statement:** In this chapter, the motivation behind our work is provided. We have started with a discussion of ICP and its local minima problem, followed by an explanation of the drawbacks of Root Mean Square Distance (RMSD) error and residual distribution, with examples to filter out the false positives.

**Chapter 4, Methods:** In this chapter, we have provided a detailed explanation of our approach for the identification of false positives. In the beginning of the chapter,
we explained two of the most used terms, namely shared MLP and symmetric functions. We then discuss the PointNet architecture, since part of our proposed method is motivated by that system. Finally, we discuss our proposed ValidNet architecture and provide the explanation of its various components.

Chapter 5, Dataset Generation: This chapter discusses the approach of our dataset preparation. We start the discussion by selecting the model and then the generation of synthetic bins simulating the real-world scenario. Then we have discussed the segmentation and registration techniques for generating pose hypotheses. Finally, we have provided our approach of defining true and false positives from pose hypothesis and ground truth poses.

Chapter 6, Experiments and Results: This chapter provides the details of our experiments and the results of our method with sample output. We have also provided a robustness experiment and an ablation study of our ValidNet architecture in this chapter. We then explain the effect of the number of model and segment points in our proposed method. We conclude the chapter by providing a visualization of our network.

Chapter 7, Conclusions: This chapter concludes the thesis, with a brief review of our method and experiments. Some approaches which can be taken to improve this work in the future are also discussed.
Chapter 2

Background

In the last few years, there have been several research efforts and different approaches developed to find the rigid transformation between two surfaces. In the next two sections we are going to describe the past contributions to both the 3D surface registration, and the registration verification problems.

2.1 Point Cloud Registration

2.1.1 Classical Registration

The rigid registration of 3D surfaces is of central importance to the processing of 3D data. For the alignment of two point clouds, Besl and Mckay proposed the Iterative Closest Point (ICP) algorithm [6] which is one of the earliest algorithms and the most widely used method for point cloud registration.

This algorithm first assumes an initial transformation between the source and the target point cloud and the source point cloud is transformed based on that. After the transformation, for each source point the corresponding target point is calculated using the nearest neighbour algorithm. The transformation matrix is then updated.
based on the newly computed correspondences. This process continues iteratively until a certain number of maximum iterations, or a minimum error threshold has been reached.

A challenge is that the success of the ICP algorithm fully depends on the value of that initial transformation estimate. If the initial estimate is close enough to the global optimal solution, then ICP will converge to a good result, with the transformation reached at the final iteration falling very close to the true, global optimal solution. Otherwise, if the initial estimate is not close enough to the optimal solution, then ICP will converge to a local minimum which is called a false positive. (The ICP algorithm and it’s local minimum problem are discussed in detail in Chapter 3.2).

To deal with this local minimum problem and also increase the time efficiency of ICP (which can be expensive), a large number of variations of ICP have been proposed by various researchers. Fitzgibbon [14] proposed the use of non-linear optimization to reduce registration error. Using a non-linear optimizer increases the convergence basin radius of global minimum compared with regular ICP, and can therefore handle larger initial misalignments. The convergence basin is the range of initial poses for which ICP will converge to a particular minima (details in Chapter 3). Increasing the radius of the convergence basin of the global minimum makes the ICP algorithm more robust to the value of the initial pose.

Tsin and Kanade [15] extended 2D image correlation for alignment into 3D point cloud data. They defined the registration problem as finding a maximum kernal correlation configuration between two point clouds. Bouaziz et al. [16] proposed the use of sparsity inducing normal information to achieve registration optimization in order to reduce the effect of outliers and missing data. Apart from that, Masuda and
Yokoya [17] introduced MICP, Trucco et al. [18] proposed robust ICP (RICP), and Zinßer et al. [19] offered Picky ICP to make the ICP algorithm robust to the initial pose. In all of these variations, the dependency on the initial pose still exists, and they are not able to solve the local minimum problem exactly.

In the last few years many researchers have proposed various strategies to find a good initial transformation between two point clouds. A common strategy to find a good initial pose is to extract local descriptors from the source and target point clouds, and then use a Random Sample Consensus (RANSAC)-based [20] iterative algorithm to align those point clouds [21–23]. These algorithms start by calculating local descriptors from both clouds, which are mainly calculated from the point neighborhood relationships. After calculating the descriptors, 3 points or 4 points are selected randomly from one point cloud (often called the source) and their correspondences are computed based on those matching features in the other point cloud (the target). The transformation matrix between the two clouds is computed based on these correspondences. After the computation of the transformation matrix, a fitness score is calculated based on the number of inliers (the target points which are within a certain distance of the transformed source points) and outliers (the target points which are not within a certain distance of the transformed source cloud). These processes (selecting 3 or 4 points, calculation of transformation matrix and fitness score) run iteratively up to a certain number of iterations, which is based on the geometry and the number of points of the source and target point cloud, and the pose which has the best fitness score is ultimately selected as the best solution. The success of these types of approaches mainly depends on finding good correspondences which mainly depends on local descriptors.
In order to find good correspondences, various local descriptors have been proposed for 3D point cloud data. Spin Images is one of the original local descriptors proposed by Johnson et al. [24]. This feature is robust to clutter and occlusion, but its performance degrades with the increase of the noise [25]. Frome et al. [26] proposed a descriptor named 3D shape contexts (3DSC) which is a directly modified version of the 2D shape context descriptor to 3D point cloud images. This descriptor captures the local information of a point cloud point using a spherical support region near it. In this descriptor, there is no specific definition provided for the reference frame of a point, which is why multiple descriptor computations are required for a single feature point [27].

Rusu et al. [28] proposed the Point Feature Histogram (PFH) based on the point pair relations of a neighbourhood. In this feature, for every point pair within a certain radius of point \( p \), a number of geometric properties are calculated and then binned into a histogram, which is used as a descriptor of \( p \). Though PFH shows reasonably good performance for scene registration, it is not suitable for many real-time applications, because of its high computational expense. In order to simplify the representation and reduce computational expense, Rusu et al. [4] proposed a simplified version of PFH called Fast Point Feature Histogram (FPFH). This feature is able to provide much faster results than PFH without degrading the results significantly.

Radius based Surface Descriptors (RSD) was proposed by Marton et al. [29], in which the radius relationship was estimated from the distance and normal angles between two points. For a given point descriptor, this radius feature is calculated for all the neighbourhood points, and the maximum and minimum radius values are subsequently used as the feature. It is a simple and descriptive algorithm [30], but
2.1. POINT CLOUD REGISTRATION

in real world scenes where noise, occlusion and clutter are obvious, its performance degrades significantly [31].

Apart from that, there are a number of other features to note, for example Tombari et al. proposed unique signatures of histograms for surface and texture description (SHOT) [32] and unique shape context (USC) [33], Bo et al. proposed Depth Kernel descriptor [34], and Spectral Histogram was proposed by Behley et al. [35]. All of these approaches were able to find a good initial pose for ICP in some cases, but were sensitive to outliers, missing points and occlusion. In highly cluttered and occluded scenes, when there exists substantial noise and missing data, these approaches were typically unable to find good initial poses, and thus ICP converges to a local minimum.

Drost et al. [36] proposed oriented Point Pair Feature (PPF) extraction and a voting technique for point cloud registration. The PPF is based on the relative position and orientation of two points within a single point cloud. This algorithm starts by calculating the features of the model and scene points, and then after calculating point pair features, a few points of the scene are taken as reference points, and the normals of each source point are aligned with these target points. After alignment, for various rotation angles, the number of features matched are measured and counts are accumulated as votes. The pose which gets the highest vote is ultimately selected. This approach works well in many cases, but it’s performance degrades with the increase of the sensor noise and background clutter [37]. A few variations of this algorithm have been proposed to make the algorithm robust [37, 38], but still can’t guarantee a globally optimal solution.

Another type of approach to global registration that has been proposed is to
search the pose space to find the optimal solution. These types of techniques are based on performing branch-and-bound search over the pose space to find the global solution [39, 40]. Li and Hartley [39] proposed searching over the rotational space to find the global optimal solution. This algorithm doesn’t require any correspondences to find the optimal solution but their method has several limitations. Their method assumes that both point clouds have the same number of points, that there are no outliers or missing points between these clouds, that there exists 100% overlap between the clouds [41], and that there is only a rotation difference between these point clouds [42]. Due to these limitations, this algorithm is not suitable for many real applications. Olsson et al. [43] proposed using branch-and-bound search in Euclidean registration. Yang et al. [40, 42] introduced an algorithm called *Globally Optimal Iterative Closest Point (Go-ICP)* where they have solved the Euclidean registration problem for 3D point clouds and guarantee finding the global optimal solution. This algorithm is based on branch-and-bound search over the entire 3D pose space and finds the optimal translation and rotation between two point clouds. Searching the entire pose space in this way is inherently computationally expensive, which limits the application of the algorithm to offline calculations.

### 2.1.2 Deep Learning Based Registration

Since point cloud data are unordered sets, applying deep learning techniques for the processing of these types of data is quite challenging. In recent years, a few researchers have proposed various strategies that can work directly on raw unordered point cloud data such as *PointNet* [2], *PointNet++* [44], Pointwise CNN [45], DGCNN [46], A-CNN [47], and RSCNN [48]. These architectures are mainly designed
2.1. POINT CLOUD REGISTRATION

for 3D object recognition and semantic segmentation.

Inspired by this development, a few researchers have also proposed variations to do deep learning based point cloud registration. Pais et al. proposed 3DRegNet which is one of the earliest works that uses deep learning for 3D image registration [49]. For a given set of correspondences, the 3DRegNet architecture is able to classify the correct or incorrect correspondence pairs, and is able to align those point clouds based on correct correspondences. For correspondence selection, they have used the FPFH feature proposed by Rusu et al. [4].

Aoki et al. used PointNet for image registration in their PointNetLK system [50], where they have tried to solve the registration problem without any precomputed correspondences. In this paper, they have used the PointNet classification architecture for extracting the global features of the source and target cloud. These global features contain the global characteristics and pose information of those point clouds. They then used the classical Lucas & Kanade (LK) [51] algorithm to align those point clouds based on the features extracted by PointNet.

Sarode et al. [52] proposed using a feed forward network in place of the Lucas & Kanade algorithm for the alignment of 3D point clouds. In this work, they have used the PointNet architecture for extracting global features and pose information of both source and target point clouds. After that, fully connected layers were used in order to identify the underlying transformation. They showed two versions of their architecture, namely simple PCRNet and iterative PCRNet. In simple PCRNet, source and target point clouds are used as input. In iterative PCRNet, source and target point clouds are used in the first iteration. For each subsequent iteration, the source point cloud is transformed by the transformation matrix found in the previous
2.2. VALIDATION

iteration, and used as input. Iterative PCRNet has been shown to perform better than PointNetLK in noisy data.

Wang et al. proposed the use of an attention model after learning the features in their Deep Closest Point paper [53]. In this work, they used the Edgeconv filter proposed by Wang et al. [46] to extract the features. After feature extraction, they used a Transformer network [54] as an attention model to generate a new set of features that contain the information of both source and target point clouds. From these new sets of features, a transformation matrix is computed using SVD.

2.2 Validation

While all of the approaches discussed in the previous section have attempted to make ICP robust to the local minimum problem, finding a global minimum solution remains challenging. Go-ICP is able to find the global minima between two surfaces, but is not suitable for many applications because of its inherent computational inefficiency. Another problem of Go-ICP is that it can’t differentiate between two different surfaces. If two surfaces which are not the same (e.g. which originated from two different objects) are provided as an input of Go-ICP, it will converge to the global minima and provides a result, rather than recognizing and rejecting the input.

For these reasons, it is essential to validate the accuracy of alignment between the source with the target point clouds. Each registration is finally concluded as successful if the source is accurately aligned with the target, and validation is required to reject false alignments. Although the local minimum problem of ICP has been known for years, this final hypothesis validation stage has not yet been sufficiently studied.
2.2. VALIDATION

Early methods for validation focused on visual assessment of the registration results. This was performed by viewing image pairs and comparing their contour overlays, alternate pixel displays, anatomical landmarks, or analytical models [55–58]. These methods were usually application-dependent. In medical imaging, for instance, Schnabel et al. [55] proposed to model tissue properties bio mechanically for Magnetic Resonance (MR) mammography images. They constructed finite element breast models and utilized a range of displacements. A validation method was also presented in [57] to assess the Computed Tomography (CT)–MR image registration accuracy locally, in all the areas of the volume, based on the correspondence analysis of cortical bone structures on the original images. In [58], an automatic validation was proposed for deep brain stimulation targeting for the treatment of movement disorders. Wang et al. [59] proposed a system to validate the segmentation outcome automatically of CT images and showed that using this techniques improves the subsequent registration step.

One interesting approach is to consider one hypothesis at a time and use thresholding for validation [60–62]. For example, Mian et al. [60] ranked all the hypotheses according to the quality of supporting correspondence sets. The validation of each hypothesis was achieved by evaluating the number of correspondences and a quality score of supporting correspondences. Bariya and Nishino [61] used a threshold set on the number of supporting correspondences and obtained a collection of hypotheses for each model. The final hypothesis selection was based on the calculation of overlap between the model and the scene.

Papazov and Burschka [62] introduced an acceptance function and a penalty function for pruning hypotheses and validating the model-scene correspondences.
acceptance function takes into account the number of target points that are within a
certain distance of the source point. The penalty term is calculated based on the num-ber of scene points occluded by the transformed model point. Finally the hypothesis
is selected by thresholding the acceptance and penalty functions. After thresholding,
they considered all of the hypotheses which fulfilled the predefined threshold. They
constructed a conflict graph and applied non-maximum suppression on the graph to
select the best hypothesis, if two hypotheses shared the same set of points.

Aldoma et al. [63] proposed an alternative approach for solving the hypothesis
verification problem. Instead of verifying one hypothesis at a time, they suggested to
take into account all the possible hypotheses and select the subset that can best rep-
resent the scene. In order to select the most representative subset of the hypotheses,
they defined a cost function over all of the hypotheses, and took the scene geometry
into consideration. After that, they tried to minimize that cost function and select
those hypotheses for which the cost function is optimal.

Gupta et al. used deep learning-based techniques for measuring the confidence
score of object detection and reduce the false positive from 2D RGB images in their
CullNet paper [64]. In that paper, they measured the confidence score of the 6DOF
estimated pose using cropped RGB image and 2D projection of 3D mesh. In that
work, they first used YOLOv3 for predicting the key points of the images. After
the prediction, they selected a few sets of key points based on the confidence scores
return by YOLOv3 and transformed the 3D mesh base on those key points. After
the transformation, they produced a 2D binary mask based on the translated model
and cropped the scene based on the dimension of the mask. They rescaled the mask,
and the scene, concatenated them along the channel and that 4 channel input feeds
into ResNet50 [65] architecture with group norm for calculating pose ware confidence score.

CullNet is similar in its objective to the ValidNet approach proposed in this thesis, and the two methods were developed independently in parallel: CullNet was submitted in September 2019 and appeared at ICCV in October 2019, whereas ValidNet was submitted in October 2019, prior to the publication of CullNet. The two methods, while motivated similarly, have significant differences, primarily stemming from the fact that CullNet operates on 2D data, whereas ValidNet was developed for 3D data.
Chapter 3

Problem Statement

In this work, we propose an approach to solve the hypothesis validation problem for 3D object detection and pose estimation. We propose to use a deep neural network called ValidNet that takes the scene and the transformed model hypotheses as input, and returns a matching score of their registration. Here, the scene is taken as the target point cloud, and the model hypothesis is the transformed source point cloud. Instead of considering the whole scene, which can be cluttered and occluded, we propose to first apply 3D segmentation to obtain the object hypotheses. Segmentation preserves most of the object shape and reduces the search space for registration, effectively removing outliers prior to registration.

Segmented object hypotheses in the scene will be obtained by applying a registration algorithm using the object model. These are the inputs to ValidNet for validation. In particular, two main steps which are also common in other methods, i.e. feature extraction and similarity matching, are effectively modeled by a deep network similar to PointNet [2]. Deep learning strategies are able to induce spatial–contextual features from 3D geometric data such as point clouds. Thus, the important features can be captured using a set of learnable filters during training.
3.1. POINT CLOUD DEFINITION

*PointNet* [2] is a leading approach that works directly on the raw point clouds as inputs. Likewise, we employ an unordered list of points and use a deep network to extract the local and global features. In particular, we use both the classification and segmentation architectures of *PointNet* in order to effectively extract the local and global features, without the need for explicit feature engineering. Similarity matching, however, is another essential step in which we propose to use a similarity matrix design based on the extracted features, and use a symmetric function to obtain the similarity scores.

3.1 Point Cloud Definition

A point cloud is an unordered set of 3D points and can be represented by \( \{ P_i \}_{i=1}^n \) where \( n \) is the total number of points in the set, and each \( P_i \) is a vector that contains the 3D position information (i.e. the \( x, y, \) and \( z \) coordinate) of the \( i^{th} \) point. Each \( P_i \) may also include other useful information such as color, intensity, or normal information. Since many 3D scanners can capture color information, in this work we have used point clouds with color information.

The point cloud representation of 3D image data has three main properties [2], the first of which is that it’s *unordered*. There is no natural way to order the points in a point cloud, such as exists in a 2D image, or in a volumetric representation of 3D data. For \( n \) total points, there therefore exist \( n! \) different permutations of order possible for a single point cloud [2], which is a huge number of possibilities for even a modest value of \( n \).

The second property of a point cloud is *interaction with neighbouring points*. The points in a point cloud are not completely isolated in space, and each point along
with its neighbourhood represents the local structure of the object.

The final property is that a point cloud’s local structure such as orientation, positions of points relative to their neighborhood points, and certain global geometric characteristics such as curvature are invariant to rigid transformations [2].

While designing a system for point clouds, it’s important to consider these properties. In particular, any system should be invariant to permutations of the point order, and should be able to capture it’s local and global properties depending on the application.

### 3.2 Iterative Closest Point Algorithm (ICP)

Let two point clouds be denoted as the source \( \{ P_i \}_{i=1}^m \) and target \( \{ Q_i \}_{i=1}^n \), where \( m \) and \( n \) are the total number of points in the source and target point clouds respectively. The goal of registration is to find the rigid transformation between these two point clouds. The rigid transformation is represented by the rotation matrix \( R_{st} \) and the translation vector \( t_{st} \). The rotation matrix is a 3 × 3 matrix and the translation is a vector of length 3. These two transformations can be combined and written in a single homogeneous matrix as:

\[
T = \begin{bmatrix}
R_{st} & t_{st} \\
0 & 1
\end{bmatrix}
\]

(3.1)
3.2. ITERATIVE CLOSEST POINT ALGORITHM (ICP)

Here, $T$ is a $4 \times 4$ matrix, which expands into rotational components $r_{ij}$ and translational components $t_i$ as:

$$
T = \begin{bmatrix}
r_{11} & r_{12} & r_{13} & t_1 \\
r_{21} & r_{22} & r_{23} & t_2 \\
r_{31} & r_{32} & r_{33} & t_3 \\
0 & 0 & 0 & 1
\end{bmatrix}
$$

(3.2)

The Iterative Closest Point Algorithm (Iterative Closest Point (ICP)), which is the most common approach for 3D point cloud registration, tries to find the transformation in such a way that minimizes the distance difference between the source and the target point cloud. The basic approach of ICP is simple and straightforward: It starts with an initial guess of rotational matrix $R_{st}$ and translation vector $t_{st}$ (Figure 3.1(a)), and the source is transformed based on this initial transformation matrix. After the transformation, for each source point, a corresponding target point is calculated as the closest point, using an efficient nearest neighbor routine. For simplicity, assume that the target point cloud has been permuted so that for a given transformation, each index $i$ denotes the nearest corresponding target point $Q_i$ to each source point $P_i$. The quantity that ICP minimizes is then given as the following error function [66]:

$$
RMSD = \sqrt{\frac{1}{m} \sum_{i=1}^{m} \| R_{st}P_i + t_{st} - Q_i \|^2}
$$

(3.3)

where, $RMSD$ stands for root means square distance, the $\| \cdot \|$ operation is usually the L-2 Norm (i.e. Euclidean distance).

An example of the computed correspondences are shown in Figure-3.1(b). After
3.2. ITERATIVE CLOSEST POINT ALGORITHM (ICP)

Figure 3.1: ICP First Iteration: (a) Initial transformation estimate, (b) Correspondences, (c) Application of calculated transformation.

computing the correspondences, the transformation matrix is calculated that minimizes the average distance difference between source points and their corresponding points in the target point cloud. The application of this calculated transformation to the source point cloud is shown in Figure 3.1(c).

For an optimal value of the calculated transformation, i.e. that minimizes Equation 3.3, it’s necessary to find the best (i.e. optimal) correspondences between the two point clouds [53]. Since it’s often hard to identify a good initial transformation, or to get optimal correspondences for a given source and target point cloud, it’s rarely

Joy Mazumder - Department of Electrical and Computer Engineering
3.2. ITERATIVE CLOSEST POINT ALGORITHM (ICP)

Figure 3.2: Two ICP Iterations

(a) Initial estimate

(b) Iteration 1 correspondences

(c) Iteration 1 transformation applied

(d) Iteration 2 correspondences

(e) Iteration 2 transformation applied
3.2. ITERATIVE CLOSEST POINT ALGORITHM (ICP)

possible to determine the transformation in one shot. Instead, ICP finds the correspondences and updates the transformation matrix iteratively in order to converge to the optimal solution. Figure 3.2 demonstrates this scenario, illustrating the effects of two consecutive iterations. This iterative process continues until a stopping condition has been reached, which can be that the value of the error in Equation (3.3) falls below a certain threshold, or that the error is not significantly decreasing further, or until a certain maximum number of iterations has been exceeded.

3.2.1 Local Minima Problem of ICP

The computation of correspondences in the first step of ICP depends on the initial estimate of the transformation. The subsequent steps rely on these computed correspondences. The initial estimate therefore plays an important role in determining the ICP outcome. From that initial estimate, ICP begins to converge, and successful convergence to the global minimum mainly depends on that initial estimate. If the initial estimate is near enough to the global minimum, then ICP converges to it and returns a true positive (i.e. a correct registration). Otherwise, if the initial estimate is too far from the global minimum, then ICP will still converge, but to a local (rather than global) minimum, which is called a false positive (i.e. an incorrect registration).

In the error curve of ICP, as discussed above there are multiple minima. Depending on the initial pose, ICP is able to reach any of the minima points. The range of poses for which ICP reaches a particular minimum point is called the convergence basin of that minimum point. When ICP starts execution from any initial pose within that convergence basin, it will converge to that minimum point.

Figure 3.3 shows the local and global minima and convergence basin of ICP and
3.2. ITERATIVE CLOSEST POINT ALGORITHM (ICP)

Figure 3.3: ICP local and global minimum and their relationship with initial pose.

Their relationship with the initial pose. As is illustrated, ICP will only result in a true positive if the initial pose is within the convergence basin of the global minimum; otherwise, a false positive will result. Figure 3.4 shows two examples of ICP convergences. One converges to the global minimum, and the other converges to a local minimum. Even if the source and target surfaces are different from each other,
3.3 LIMITATION OF RMSD ERROR AND RESIDUAL STATISTICS FOR IDENTIFYING LOCAL MINIMA

ICP will still converge to a minimum based on the initial pose and geometry of the surface. Figure 3.5 shows two examples where the source and target surfaces are different from each other, where ICP converges to a minimum.

3.3 Limitation of RMSD Error and Residual Statistics for Identifying Local Minima

Since convergence to a local minimum is a common occurrence in ICP, it is beneficial to identify and filter out those false positives. In reality, detection of ICP local minima convergence is a complicated task, as indicated by the convergence basin pattern shown in Figure 3.3. The shape of the convergence basin depends on the source and target point cloud and varies from each other. That is why the minima errors of

Joy Mazumder - Department of Electrical and Computer Engineering
3.3. LIMITATION OF RMSD ERROR AND RESIDUAL STATISTICS FOR IDENTIFYING LOCAL MINIMA

Figure 3.5: Convergence of ICP when source and target surface are different from each other

convergence basins are different for each object. The value of Root Mean Square Distance (RMSD) error (Equation 3.3) can be quite similar for local and global minima and it is not simple to find a suitable threshold or other statistical decision function to differentiate between them.

To illustrate this, Figure 3.6 shows four examples of ICP convergence where two instances converge to global minima, and two are trapped in local minima. Their corresponding RMSD values are also provided in the figure. In the first image (Figure 3.6(a)), an example is provided where ICP converges to the global minimum with an RMSD value of .707519, while in the second figure (Figure 3.6(b)), ICP converges to a local minimum with a higher RMSD value. This is an ideal outcome which could allow thresholding to differentiate between true positive and false positive convergences. Unfortunately, such a scenario is not always achieved. For example, in

Joy Mazumder - Department of Electrical and Computer Engineering
3.3. LIMITATION OF RMSD ERROR AND RESIDUAL STATISTICS FOR IDENTIFYING LOCAL MINIMA

(a) ICP global minimum RMSD=.707519
(b) ICP local minimum RMSD=.76411
(c) ICP global minimum RMSD=.756438
(d) ICP local minimum RMSD=.707751

Figure 3.6: RMSD Error for Global ((a),(c)) and Local ((b),(d)) Minima
3.3. LIMITATION OF RMSD ERROR AND RESIDUAL STATISTICS FOR
IDENTIFYING LOCAL MINIMA

Figure 3.7: RMSD Error for Global ((a),(c)) and Local ((b),(d)) Minima
the 3rd and 4th images (Figure 3.6(c)-(d)), the values of RMSD error for the global minimum is higher than that of the local minimum. It is therefore not obvious how to find a suitable simple threshold or decision function that can differentiate between all the 4 cases presented in the figure.

One possible approach to differentiate between true and false positive registration results would be to consider the histogram of residual values following convergence. Unfortunately, the distribution of residuals can be quite similar for positive and negative cases as well. To illustrate this, Figure 3.7 shows two ICP true convergence and two false convergence examples, and their corresponding residual histograms. In Figure 3.7(a) and 3.7(b), the residual distribution is quite different for positive and negative convergences. For the positive convergence in the Figure 3.7(a) histogram, there is a peak at a low value of residual distance, which means, for a large number of source points, the distance to their corresponding target points is small. That is what we expect for a true positive convergence case. On the other hand, for the negative case in Figure 3.7(b) the histogram is distributed over the region, which means that for many source points, residual minimization was not possible because of the wrong selection of corresponding target points. That is what we expect for the negative convergence cases.

If all true and false positive convergences followed similar distributions to those shown respectively in Figure 3.7(a) and (b), then it would be possible to design a decision rule to differentiate between these cases. Unfortunately, using this type of histogram distribution, it is not always possible to differentiate between true and false instances. As an example, Figure 3.7(c) and Figure 3.7(d) show another two distributions for positive and negative classes. In these cases, for the positive convergence,
there are some higher values at the low residual distance but it’s not as significant as occurred in Figure 3.7(a). Instead of one sharp peak at the low residual distance region, the histogram is gradual and decreasing with distance. For the negative case, somewhat evenly distributed values over the histogram range is expected. However in Figure 3.7(d), instead of the evenly distributed values, we see higher values at low residual distances which we might expect to find in global minima converge cases, and not in the local minima convergence case. Because of the similar residual distributions between positive and negative classes, it is difficult and not obviously possible to reliably differentiate between them based solely on residual histograms.

### 3.4 Overview of Approach to Identify Local Minima

In this work, we attempt to identify and filter out the local minima of ICP for 3D object detection. In our case, we have two point clouds, one a model $M$ which is our source point cloud, and another a scene segment $S$ which is our target point cloud. Notably, the scene segments are generated by applying a 3D segmentation algorithm such as region growing [67, 68], to a initial scene that includes a cluttered and occluded collection of objects. Segmentation can beneficially reduce the search space and make the registration process faster. The transformation between source and target point clouds found by the registration process is denoted by $T = [R_{st}, t_{st}]$.

The goal of validation is to quantify how well the transformed model $TM$ can represent the segment $S$, or equivalently how well the inverse transformed segment $T^{-1}S$ can represent the model $M$. The benefit of transforming the segment to the model through $T^{-1}S$, is that this maps the segment to a single canonical model pose, so that we do not have to consider different model poses, which reduces and simplifies
network training. In other words, the network does not need to learn the translation and rotational invariant properties since the segments and model are always mapped to the same position. Due to the 3D nature of the data, the particular model pose that is chosen as the canonical pose is arbitrary, and its selection will not impact the performance of the resulting system.
Chapter 4

Methodology

In this work, we propose a deep neural network called ValidNet which measures the probability of registration success based on similarities measures of model (M) points with respect to inversely transformed segment (T⁻¹S) points in feature space. Our key idea is, if registration is successful and the model and the segment are correctly aligned with each other, then the model points’ positional information and characteristics will be supported by the segment points’ characteristics, and we will be able to calculate the matching precision of the model and the segment from those characteristics.

Our proposed pipeline has two parts, one of which is a feature learning network, and the other a similarity measurement network. The feature learning network converts the model and segment points into a high dimensional feature space, and the similarity measurement network measures the similarities between the aligned model and segment in the feature space, and provides a probability of success of the registration.

In order to describe our ValidNet architecture, we first introduce and explain the two terms shared MLP and symmetric function, which are the basic building
blocks of our network. After that, we briefly discuss the PointNet [2] architecture since our ValidNet feature extraction network is motivated by and makes use of that architecture. We will then discuss how we have changed the PointNet architecture for extracting the features of the model and the segment points, and finally how we measured the similarities between the model and the segment.

4.1 Shared MLP

The perceptron is a basic building block of a deep neural network. A perceptron takes a vector as input and its output is just the weighted multiplications of the elements of the input vector with the addition of a bias term. A non-linear function is applied to the perceptron output to make it capable of dealing with non-linear separable cases. In this work, we have used Rectified linear unit (ReLU) [69] as a non-linear function. Figure 4.1 shows the basic operation of a perceptron:

- $v$ is the input vector with elements $v_i$;
- $w_i$ is a weight which is multiplied with the input element $v_i$;
- $b$ is the bias term;
- $f$ is a non-linear function, and;
- $o$ is the output of perceptron.

Typically, all of $v_i, w_i, b$ and $o$ are scalar values. Equation 4.1 shows the basic operation of the above illustrated perceptron:

$$o = f \left( \sum_{i=1}^{n} (w_i \times v_i) + b \right)$$ (4.1)
These weights and biases are the trainable parameters and are trained using a gradient decent algorithm.

The concept of a shared perceptron is that of using a single perceptron, having the same weights and biases, over multiple input vectors of a single data sample. It operates on each vector independently. The same weight values are multiplied with the input vectors, and the same bias values are added to produce the outputs. Figure 4.2 shows how a shared perceptron works on an input data sample having $n$ number of $(3 \times 1)$ vectors. Figure 4.2(a) shows the perceptron operating on the first vector $v_1 = (x_1, y_1, z_1)$, and producing output $o_1$. In Figure 4.2(b), the perceptron having the same weights and bias is operating on the second vector $v_2$ of the sample. The same strategy will apply for all the input vectors, up to $v_n$ (Figure 4.2(c)).

If there is at least one hidden layer in a shared perceptron network (i.e. an intermediate layer, which is not the input or output layer), then it’s called a shared multi-layer perceptron or shared MLP.  Figure 4.3 shows the first layer of a shared perceptron operating on the vectors of an input sample, and figure 4.4 shows the
4.1. SHARED MLP

Figure 4.2: Shared perceptron. Same weights and bias is used over the all points in the point cloud

Joy Mazumder - Department of Electrical and Computer Engineering
second layers of a shared perceptron working on the output of the first layer.

4.2 Symmetric Function

A point cloud is an unordered set of 3D data. In order to extract its characteristics invariant to the order of the points, a function is required for which the output will be the same whatever the order of the input is. A symmetric function serves this purpose. A symmetric function is a mathematical function of \( n \) variables which is unchanged under the permutation of its input arguments [70]. In this way, if \( f() \) is a two variable symmetric function, then \( f(x_1, x_2) = f(x_2, x_1) \). Sum, average and max are examples of common symmetric functions.

For example, suppose we have two input sets containing the same values but in a different order, e.g. \( A = \{1, 2, 6, 7\} \) and \( B = \{2, 7, 6, 1\} \). For these two sets the values of these symmetric functions are identical [3]:

\[
\begin{align*}
\text{sum}(1, 2, 6, 7) &= \text{sum}(2, 7, 6, 1) = 16 \\
\text{max}(1, 2, 6, 7) &= \text{max}(2, 7, 6, 1) = 7 \\
\text{average}(1, 2, 6, 7) &= \text{average}(2, 7, 6, 1) = 4
\end{align*}
\] (4.2)

4.3 PointNet for Point Cloud Classification and Segmentation

PointNet [2] is a revolutionary architecture proposed by Charles R. Qi et al., which is the first deep neural network-based approach that can operate directly on raw, unordered point clouds. Since a point cloud is a set of unordered data, without a natural ordering of the data like 2D images or voxel grids, previous deep learning techniques on this type of data were mainly based on converting the point cloud into some regular
4.3. POINTNET FOR POINT CLOUD CLASSIFICATION AND SEGMENTATION

Figure 4.3: 1st layer of shared perceptron, $r$ number of perceptron with same weight and bias is operating on an input sample
4.3. **POINTNET FOR POINT CLOUD CLASSIFICATION AND SEGMENTATION**

Figure 4.4: 2nd layer of shared perceptron, \( s \) number of perceptron with same weight and bias is operating on an the first layer output

Joy Mazumder - Department of Electrical and Computer Engineering
4.3. **POINTNET FOR POINT CLOUD CLASSIFICATION AND SEGMENTATION**

grid-like structures such as a voxel grid [71,72], 2.5D depth images [73] or multi-view 2D images [74]. After converting them into a regular structure, researchers applied regular 3D or 2D convolutional filters on them to extract meaningful information. The volumetric representation requires a lot of unnecessary computation, and can be space expensive, which often leads to a coarse resolution. Alternately, 2.5D depth images can’t capture the full properties of a 3D point cloud. Multi-view images can provide very good results on 3D object classification, but it requires 3D rendering and the processing of a large number of images which is computationally inefficient.

*PointNet* was the first approach to solve the point cloud classification and segmentation problems from raw unordered input. This architecture takes raw positional information \((x, y, z)\) values of the point cloud and predicts its class label or point-wise semantic labels. It can also take color and normal information as input as well. For \(n\) points, the dimension of the input is \(n \times c\), where \(c\) is the number of input channels, which is: 3 for only positional information; 6 for position and color, and; 9 for the combination of position, color and normal information.

Joy Mazumder - Department of Electrical and Computer Engineering
4.3. **POINTNET FOR POINT CLOUD CLASSIFICATION AND SEGMENTATION**

For the classification of a point cloud, the most important property is to approximate the global features that are invariant under both the rigid transformation of the point cloud, and its point order. To find the global signature of the point cloud, their idea was to apply a symmetric function on the remapped elements of the point set:

\[
f(\{p_1, p_2, \ldots, p_n\}) \approx g(h(p_1), h(p_2), \ldots, h(p_n))
\]  \hspace{1cm} (4.3)

Here, \( h \) is a multi-layer perceptron that is shared between all the points of the point cloud and \( g \) is a symmetric function that is independent of the input point cloud’s point order. The shared Multi Layer Perceptron (MLP) layers convert the input point cloud into a high dimensional feature space, and the symmetric function accumulates the global information which is invariant to the point order.

*PointNet* initially started by approximating an input transform of the point cloud to achieve pose normalization [3]. They named this network *T-Net*. The purpose of this network was to make the point cloud invariant under rigid transformations. The network approximates a \( 3 \times 3 \) transformation matrix, which is used to transform all points in the point cloud to reduce the effect of rigid transformations. The T-Net network consists of 3 shared MLP layers which mapped the input points into a
1024 dimensional feature space. These shared MLP layers operated on each point independently and shared between all the points of the point cloud as discussed in Section 4.1. After transforming the point cloud into feature space, they used a symmetric function that captures the combined information of the whole point cloud. Using fully connected layers, they reduced the combined feature dimension from 1024 to 256. Finally using a trainable matrix (of dimension $256 \times 9$) and trainable bias (of dimension $9 \times 1$), they estimated the $(3 \times 3)$ transformation matrix as shown in Figure 4.6.

Once the transformation matrix has been predicted, it is then used to transform the input point cloud. Two shared MLP layers operate on these transformed point sets and converts the points into a 64-dimensional feature space as shown in Figure 4.5. They extended the concept of input transformation into feature space as well by using another T-net to predict the feature transform for feature alignment. This T-net predicts a $64 \times 64$ transformation matrix and the features learned by the first 2 MLP layer of the PointNet network is transformed according to that matrix.

The dimension of the transformation matrix in the feature transform network is large, which increases the model complexity so that it becomes hard to optimize the algorithm [3], which led to an over-fitting problem. To deal with that they added a regularization term with the final loss function of the network:

$$L_{reg} = \left\| I - AA^T \right\|$$

Here, $A$ is the transformation matrix predicted by the feature transformation network, and $I$ is a $64 \times 64$ identity matrix. This loss function was designed based on
the constraint that the transformation matrix $A$ is orthogonal, and orthogonal transformations preserve the information of the input. They showed that addition of these terms makes the network stable and able to achieve better performance.

The architecture of this feature transformation layer is quite similar to the input transformation layer except for the part of matrix multiplication and bias addition. Here, instead of $256 \times 9$ values, the multiplied trainable matrix size is $256 \times 4096$, following which 4096 bias parameters are added. After the feature transformation, 3 more layers of shared MLP were used to extract the point-wise local features of the point cloud and convert each point into 1024 dimensional feature space. These 1024 dimensional per point features contain the local information of the points. In order to accumulate the global information, a symmetric function is used on these features. The symmetric function captures this global feature vector of the point cloud which contains the transformation and order invariant characteristics of the whole point set. To predict the class label of the point cloud, a feed-forward network is used which takes the global vector as input and predicts its class probability.

The PointNet segmentation network predicts the class label of all points based on the global and local information of those points. To extract the features for each point having both local and global information, they concatenated the global feature vector of the point cloud with the output of the feature transform networks as shown in Figure 4.5. The output of the feature transform network provides the local information of each point while the global feature contains the global characteristics. After concatenating, they used another three shared MLP layers to extract a new set of per point features that contains both global and local information of the points. After the feature extraction has been completed, another shared MLP layer is used.
4.4. VALIDNET

to predict the segmentation label of the points. The bottom part of Figure 4.5 shows the PointNet segmentation network.

4.4 ValidNet

Figure 4.7: ValidNet for registration validation. The left part is the feature learning network that learns the features of the model and the segment and the right part is the similarity finding network that measures the similarities between these features and provides the validation probability of registration success[1]

Our proposed ValidNet architecture takes two point clouds as input, one the model (M) and the other the transformed scene segment (T⁻¹S). If the total number of model points is \(m\) and the total number of segment points is \(n\), then the input dimension of ValidNet is \((m \times c) + (n \times c)\), where \(c\) is the number of input channels. Since many 3D scanners capture co-registered color information of the point cloud as well, we used the colored 3D point cloud for this work. Each point is therefore represented by \(c = 6\) channel values, i.e. \((x, y, z, r, g, b)\). Our ValidNet can also take point clouds without color as well (i.e. \(c = 3\)) by only changing the shared MLP.

Joy Mazumder - Department of Electrical and Computer Engineering
input of the first layer. Our proposed architecture has two main parts, a feature learning part, and a similarity measurement part, which are described in the following subsections.

### 4.4.1 ValidNet Feature Extraction Network

Our network starts with a feature learning network which learns the positional information and characteristics of each model and segment point. In particular, the local and global information are accumulated in this part. Our feature learning network consists of shared MLP layers, skip connections, and symmetric functions. It is intuitively similar to the PointNet segmentation network. In the PointNet architecture, two sub-networks were used for input and feature transformations. In our work, since we have converted all the model and the segment point clouds into the model’s canonical position, all the segments and the model stay in the same position, which allowed us to ignore the input transform and feature transform network in the ValidNet architecture.

Our feature learning network starts with a five layer shared MLP. This shared MLP is operated on each point independently. In our problem we are dealing with two different point clouds (the model and the scene segment), and so we used the exact same filter to extract the model and segment features. That means the same learned weights and biases are used independently on all the model and segment points to extract the features. Figure 4.8 shows how our shared perceptron works to extract the model and the segment features. Using the same filter to extract the model and segment features will force the network to learn similar types of features on each channel which will be beneficial when subsequently measuring the similarities.
Figure 4.8: Shared MLP on ValidNet
These five shared layers extract the local information of the model and the segment points. After the extraction of the local features, two symmetric functions are then used on the remapped point-sets to leverage the global characteristics of the model and the segment, respectively. The captured global characteristics are invariant to the segment’s and the model’s point orders (as discussed in Section 4.2). Though PointNet proposes to use max-pool as a symmetric function, we have found that using average pooling provides the most robust behaviour in terms of noisy data. Since real world data will be noisy, we use use average pooling as a symmetric function here. Details of the experimental results that led to choosing average pooling as a symmetric function are described in Section 6.4.2.

After modeling the global characteristics, to make the per point feature combination of local and global information, the global vectors are concatenated to the output of the second MLP layer as shown in Figure 4.7. After the concatenation, another set of shared MLP layers are applied to the concatenated outputs of the model and the segment points’ features, to generate a new set of per point features that contains both local information of the point and the global information of that point cloud. These shared MLP layers are independently applied to the model’s and segment’s points’ features, and are shared between them.

4.4.2 Similarity Measurement Network

The next step after modeling the characteristics of the model and segment point sets with local and global information, is to measure their similarities in feature space. To measure the similarities, we have designed a similarity measurement network which can measure the matching score of registration in feature space.
Usually for measuring similarity, one-to-one correspondences are considered between the model and segment, which means each model point is checked as to whether it has a corresponding segment point or not. If the model point has a corresponding segment point, then the model point votes for the true positive; otherwise, it votes for the false positive. This strategy can be extended in the feature space as well by considering corresponding points’ features. In this way, if a model point has a corresponding segment point and their features are similar, then it votes for positive classes; otherwise, it votes for negative classes. Here, the vote is either 1 or 0, which represents whether a point is matched or not.

Sometimes, due to noise or occlusion, some point-to-point correspondences might be missing, and the matching strategy can fail. Even if for a model point, a corresponding segment point is found, these types of approaches don’t consider the matching precision between those points. In our work, instead of point-to-point matching we took a different approach. If the registration is successful and the model and the segments are correctly aligned with each other, then even without the existence of a corresponding segment point, a model point’s feature can be supported by other segment points in feature space. By this reasoning, instead of considering pure point-to-point feature matching, we considered all the segment points to determine whether a model point in feature space is relevant or not.

Our similarity measurement network consists of a similarity matrix, a symmetric function, and a feed forward network. The similarity matrix contains the similarity information between the model points and the inversely transformed segment points. The symmetric function captures the relevance information for a model point, and makes the network invariant to the segment’s point order. Finally, the fully connected
layer extracts the matching relevance.

Our similarity matrix $X_{m \times n}$ is calculated from the model and segment features which is learned from the feature learning part of our proposed method. Each element $x_{ij}$ indicates the similarity between the $i^{th}$ model point and the $j^{th}$ segment point. If $p_i$ and $q_j$ represent feature vectors for the $i^{th}$ model point and $j^{th}$ segment point, respectively, then:

$$x_{ij} = p_i^T q_j$$

(4.5)

The key idea of computing the similarity matrix is that, if the $i^{th}$ model point feature is supported by the $j^{th}$ segment point then the dot product value will be high; otherwise it will be low. In this way, the $i^{th}$ row of the similarity matrix represents the degree to which the $i^{th}$ point is accurately modeled by all segment points. If the $i^{th}$ model point is accurately represented by any inversely transformed scene point, the values along the $i^{th}$ row will be high. If it’s not supported by any of the segment points, then all the values along the row will be low. Hence, performing a symmetric function along the row preserves this matching information and makes the network invariant to scene input perturbation.

$$y_i = \text{symmetric function}(x_{i1}, x_{i2}, ..., x_{in}) \quad \forall i \in [1, m]$$

(4.6)

Equation 4.6 shows the symmetric function operation along the $i^{th}$ row of the similarity matrix. The $y_i$ values contain the matching information of the $i^{th}$ model point and the output vector $y$ contains the information of which model points are supported by the scene and their matching precision.

One characteristic of the problem that is worthwhile highlighting is that, since we
are motivated by the bin picking problem where the model identity is known and its transformation is fixed into a canonical pose, its point order is therefore fixed as well. For a single model, this output vector order will be fixed and we therefore don’t need to make our network invariant to that order.

After the computation of this similarity vector, a feed forward network is used to classify it. As a result, the output layer can provide the validation probability of the registration algorithm. More specifically, the outputs of the networks are two scores for two classes, i.e. the positive (correct registration) and negative (incorrect registration) class. These scores represent how accurately the registration method is able to find the transformation matrix.
Chapter 5

Dataset Generation

For training and testing of *ValidNet*, we need a large dataset with ground truth. As scanning a large number of scenes using a real scanner and labeling each object would be very manually intensive and time consuming, it is not feasible to do so. Instead, we generated a large synthetic dataset containing 104,904 training and 11,706 test samples. For generating our dataset, we used four different types of models: Horse, Sailor, T-rex, and Rhino. All of these models are taken from the Queen’s range image and 3D model database [75]. Figure 5.1 shows the four major views of our models which were used for the generation of the dataset.

Recently Kleeberger *et al.* proposed a bin picking dataset [76] and the same approach as ours for generating synthetic bins. Since we had completed most of the experiments before their dataset was published, in this work, instead of their dataset we used our own dataset.

Figure 5.2 shows the basic flow diagram of our data generation pipeline. Details of each approach will be provided in the following sections.
Figure 5.1: Models for synthetic data generation
5.1 Synthetic Bin Generation

Our dataset generation procedure starts with modeling of synthetic bins, which are designed to be similar to the scenario in the industrial robotic bin picking task. In bin picking, the multiple objects in each bin are of a single class and randomly arranged in such a way that they exhibit clutter and occlusions. The challenge is to be able to determine the pose of each object to a high degree of confidence so that automated robotic grasping will succeed. For a realistic representation of the environment for objects falling in a bin, we used the Bullet Physics Library [77] which is a physics engine that simulates collisions, soft and rigid body dynamics. Using this library, we modeled various physical scenarios related to bin picking problems such as gravity, friction, and collisions with other objects. This results in reasonably realistic modeling of real-world scenarios of typical part bins, with clutter and occlusion in the scenes.

Using the Bullet Physics Library, we generated 10,000 synthetic scenes per model with varying numbers of objects (min=1, max=8). Therefore, in total, we have generated 40,000 synthetic scenes for four different models and divided this data into 36,000 training and 4,000 test scenes. Using the orientations returned by the simulator, we then converted these scenes into 2.5D point cloud data. Figure 5.3 shows four sample scenes from our dataset. Figure 5.4 shows some real data scanned by 3D scanner and some of our synthetic data for another model. In this work, we
haven’t use that model because of it’s missing color information.

5.2 Prepossessing

After generating the 2.5D point cloud dataset, we have down-sampled the point cloud to speed up the subsequent steps. We have applied uniform downsampling.
5.2. PREPOSSESSING

(a) Real data scanned by 3D scanner

(b) Synthetic dataset generated by us

Figure 5.4: Comparison between real and synthetic data  [1]

available in point cloud library [78]. In uniform downsampling method, a 3D voxel grid is created over the input point cloud and all the points in a voxel are represented by their centroid [79]. After downsampling, the background and the bin are subtracted using a point cloud scene without any objects. This helps to reduce the effects of outliers and to get better segments during segmentation step.
5.3 Segmentation

The next step in our data generation pipeline is object segmentation. The segmentation process divides the whole scene into multiple clusters. The purpose of the segmentation approach is to reduce the number of points and outliers of the target point cloud. As a result, good registration output can be achieved using a fewer number of Random Sample Consensus (RANSAC) iterations, and will make the registration process much faster.

We used the region growing segmentation [67] method to segment objects from the scene. This algorithm takes the whole raw unordered scene point cloud as input, and outputs clusters of points containing smooth surfaces. During the segmentation process, this algorithm only takes into account the surface normal. While calculating the normal, if the neighbourhood region is big enough, it can average out the effect of noise which can make the algorithm robust [67].

The region growing segmentation algorithm has two major steps, which are seed selection, and growing and merging regions. It starts from an initial selected point, and the region starts growing from it based on some predefined smoothness constraint. The initial point is known as a seed point. The selection of seed points plays an important role in the final output of this segmentation algorithm. The selection of wrong seeds can lead to imprecise results [80].

For the purpose of selecting the seed points, at first the curvature of all the points are calculated. The curvature of a point represents the surface variation around the point. It is a geometric property which is invariant under rigid transformations and is able to represent the shape of a surface [81,82]. The curvature value of a query point $p_q$ can be estimated from eigenvalues of the co-variance matrix of its neighbourhood.
points. The neighbourhood points are identified using either a K Nearest Neighbors (KNN) or Fixed distance neighbors (FDN) algorithm. The covariance matrix \( C \) of the neighbourhood region of the query point \( p_q \) can be expressed by Equation 5.1 [83,84]:

\[
C = \frac{1}{k} \sum_{i=1}^{k} \xi_i (p_i - \bar{p}) \cdot (p_i - \bar{p})^T
\]  

(5.1)

Here, \( k \) is the total number of neighbourhood points of \( p_q \), \( \bar{p} = \frac{1}{k} \sum_{i=1}^{k} p_i \) is the centroid of the neighbourhood region and \( \xi_i \) is the possible weight of \( p_i^{th} \) point. In our case, we have provided the same weights to all the points, so the value of \( \xi_i \) in our case is equal to 1. The covariance matrix \( C \) is a symmetric positive semi-definite matrix and its eigenvalues are \( \lambda_0, \lambda_1, \lambda_2 \) s.t. \((0 \leq \lambda_0 \leq \lambda_1 \leq \lambda_2)\). The variation of the surface (i.e. the curvature) at the query point \( p_q \) can then be calculated from these eigenvalues [83–85]:

\[
\sigma_p = \frac{\lambda_0}{\lambda_0 + \lambda_1 + \lambda_2}
\]  

(5.2)

After the calculation of the curvature of all the points, the first step would be sorting the points of the point cloud based on their curvature values, and the point which has the minimum curvature value is selected as a seed point. Points with low curvature values usually lie on a flat surface, and choosing such points helps to get fewer clusters with accurate and meaningful information.

After the selection of the seed point, its neighborhood points are identified using KNN or FDN. Following that, the smoothness values of all the neighborhood points are checked. The smoothness constraint is measured based on the angle between the seed point’s normal and its neighborhood point’s normal. If the angle is lower than
a certain predefined threshold, then the neighboring point is considered as a member of that cluster, and the region grows. After checking all the neighborhood points and their angle relationships, their curvature values are checked. If the curvature is less than a certain threshold, those points are added as a seed point, and the current seed point is removed from the list. These steps continue until the seed points set becomes empty, which means that a region has grown and clustered successfully. The process
then starts from the beginning for the other points of the point cloud, and terminates when all the points of the point cloud have been assigned to a cluster.

Four sample outputs (one for each model) of region growing segmentation (after prepossessing) are shown in Figure 5.5, with each segment rendered in a distinct color. Their corresponding scene is shown in Figure 5.3. The resulting segments were then filtered to exclude those containing too many or too few points, which would represent undersegmented and oversegmented regions, respectively.

5.4 Registration

To identify object poses from the segments returned by the segmentation algorithm, we perform registration between the model and the segment. In order to estimate the global initial alignment between the model and segment, we used a RANSAC-based registration algorithm and then further refined the pose using Iterative Closest Point (ICP). Another possibilities of generating object hypothesis for our dataset is using the ground truth pose. In that approach, at first the model has to transform based on the ground truth pose in the scene and then perturb the model a little bit in different direction to generate initial pose and apply ICP after that to check ICP can converge to global minima or not. We have used RANSAC-based method in this work because it can provide the object hypothesis similar to the hypothesis founds in real-world.

To find a good initial pose, we used a RANSAC-based registration algorithm using local descriptors. For our work, we used a local descriptor named Fast Point Feature Histogram (FPFH) proposed by Rusu et al. [4]. FPFH feature is the faster version of the Point Feature Histogram (PFH) [28]. FPFH feature reduces the complexity of
PFH and can provide output much faster, which makes it suitable for the bin-picking application. The computational complexity of FPFH is $O(nk)$ while the complexity of PFH is $O(nk^2)$, where $n$ is the number of points in the point cloud, and $k$ is the number of neighbors of each point.

This feature captures the local properties of a point using a multidimensional histogram which is invariant under the rigid transformation of the point cloud. These local properties are calculated based on the interaction of a point with its neighborhood points. For a query point $p_q$, the neighborhood points are shown in the Figure 5.6 [86], where the red point is the query point and the points within $r$ distance are considered as neighborhood points (all the points within the black dotted circle in the figure).

Let $p_s$ be a neighborhood point of the query point $p_q$, with respective normals $n_s$ and $n_q$. To calculate their angular relationship, they defined a fixed co-ordinate frame, named the $uvw$ frame. This frame is calculated based on their position and normal information as:

$$
\begin{align*}
  u &= n_q \\
  v &= u \times \frac{(p_s - p_q)}{\|p_s - p_q\|_2} \\
  w &= u \times v
\end{align*}
$$

The angular relationship features between the points $p_q$ and $p_s$ is calculated using the defined $uvw$ frame. The angular features is denoted by $\alpha, \phi$ and $\theta$. The definition
5.4. REGISTRATION

Figure 5.6: Neighbourhood points of a query. (The connections with marked 2 indicates that it will contribute twice in final FPFH feature calculation) [4] © 2009 IEEE

The calculation of the features is provided in Equation 5.4.

\[ \begin{align*}
\alpha &= v \cdot n_s \\
\phi &= u \cdot \frac{(p_s - p_q)}{\|p_s - p_q\|_2} \\
\theta &= \text{arctan}(w \cdot n_s, u \cdot n_s)
\end{align*} \]

(5.4)

For the query point \( p_q \), these features are calculated between \( p_q \) and all of its neighbourhood points. Since \( \alpha, \phi \) and \( \theta \) are measurements of angles, their values are
normalized to the same interval of the trigonometric circle \[87\]. After the normalization, these three features are binned into a histogram separately. For binning a feature, the entire range is divided into 11 subdivisions, and the number of occurrences in each division were counted. In total, 33 values were used to represent the local information of each point. These histograms are called Simplified Point Feature Histogram (SPFH).

After the calculation of SPFH feature for a query point \( p_q \), SPFH features were also calculated for all of its neighborhood points as well. Afterwards, the SPFH feature of \( p_q \) was re-weighted by the features of its neighborhood points, and the final weighted representation is the Fast Point Feature Histogram (FPFH) (Equation 5.5).

\[
FPFH(p_q) = SPFH(p_q) + \frac{1}{k} \sum_{i=1}^{k} \frac{1}{\omega_i} \cdot SPFH(p_i)
\]  

(5.5)

Here, \( \omega_i \) is the distance between a query point \( p_q \) and \( p_i \) in some given metric space \[4\]. FPFH feature can capture the weighted local information in a neighborhood up to
After the extraction of features from the model and the scene segments, these features were then used for performing RANSAC \cite{20} based registration \cite{88} between the model and the segment, resulting in an initial pose estimation for the segment. This RANSAC algorithm starts by selecting 3 random points from the source point cloud. These points are selected in such a way that their pairwise distances are higher than some predefined minimum distance threshold. After the selection of points, for each picked model point, all the points from the scene which have similar FPFH histograms are identified. From those similar points, a point is chosen randomly as a corresponding point. This processes are done for all three points and then the rigid transformation matrix is calculated from the three correspondences. An error metric is calculated for the measurement of the confidence of the transformation matrix. These processes continue for a certain maximum number of iterations, and then the pose which results in the minimum error metric is selected as an initial pose for ICP, which is applied to further refine the pose. The pseudocode of RANSAC registration is given in Algorithm 1.

A few examples of segments and the registration on those segments are shown in Figure 5.7. The model and the segment are shown by the red and white points, respectively.

After RANSAC-based registration, we built a dataset of 104,904 training and 11,706 test segments.
5.4. REGISTRATION

(a) Synthetic bin model: Horse

(b) Synthetic bin model: Sailor
Figure 5.7: Output of RANSAC-based registration

Joy Mazumder - Department of Electrical and Computer Engineering
Algorithm 1: Pseudocode of RANSAC-Based Registration

**Data:** Source Point Cloud $P$ and Target point Cloud $Q$

**Result:** Transformation matrix between the source and Target

$P_{f_{ec}} \leftarrow ExtractFeature(P)$

$Q_{f_{ec}} \leftarrow ExtractFeature(Q)$

$Best\_Score \leftarrow 0$

$Selected\_T \leftarrow I$

for Total number of RANSAC iteration do

$P_{key}, P_{keyf_{ec}} \leftarrow Select3Points(P, P_{f_{ec}})$

$Q_{key} \leftarrow CalculateCorrespondences(Q, Q_{f_{ec}}, P_{keyf_{ec}})$

$T \leftarrow CalculateTransformationMatrix(P_{key}, Q_{key})$

$TP \leftarrow TransformSource(P, T)$

$Matching\_Score = CalculateMachingScore(TP, Q)$

if $Matching\_Score > Best\_Score$ then

$Best\_Score \leftarrow Matching\_Score$

$Selected\_T \leftarrow T$

end

end

5.5 Ground Truth Label

The next step in the data generation pipeline is to assign ground-truth labels to the generated segments. Validation of registration results is a binary classification problem where the positive class is the one for which registration was successful and the negative class indicates unsuccessful registrations. To assign a class label to each segment, we used the registration output of the model and segment. After the registration, if the centroid and principal axes of the model and the segment were within some predefined threshold, we labeled it as a positive class segment; otherwise, it was labeled a negative class segment. Recently, Kleeberger et al. [76] suggested using the metric proposed by Bregier et al. [89] for defining the true and
false positive pose determination results for 3D bin picking, but in this work we apply our alternative form. The definition of true and false positives in our approach are as follows.

Let \((\bar{x}, \bar{y}, \bar{z}, \bar{\theta}, \bar{\phi}, \bar{\psi})\) be the 3 translational and 3 rotational components of a returned 6 DOF pose, which has been estimated through a registration process, and let \((\hat{x}, \hat{y}, \hat{z}, \hat{\theta}, \hat{\phi}, \hat{\psi})\) be the corresponding ground truth value. Further, let:

\[
\Delta D = \sqrt{(\bar{x} - \hat{x})^2 + (\bar{y} - \hat{y})^2 + (\bar{z} - \hat{z})^2}
\]

be the offset between the translational components of the returned and ground truth pose. Similarly, let:

\[
\begin{align*}
\Delta \vartheta &= |\bar{\vartheta} - \hat{\vartheta}| \\
\Delta \varphi &= |\bar{\varphi} - \hat{\varphi}| \\
\Delta \psi &= |\bar{\psi} - \hat{\psi}|
\end{align*}
\]

be the offsets between the rotational components of the returned pose and ground truth poses. The binary class label \(L\) of registration is then defined as:

\[
L = \begin{cases} 
1 & \text{if } \Delta D < D_{th} \text{ and } \Delta \vartheta < \Theta_{th} \text{ and } \Delta \varphi < \Theta_{th} \text{ and } \Delta \psi < \Theta_{th} \\
0 & \text{otherwise}
\end{cases}
\]

Here, \(L = 1\) means correct alignment (positive class) between the model and the segment and \(L = 0\) means incorrect alignment (negative class). \(D_{th}\) and \(\Theta_{th}\) are the translational and rotational thresholds respectively. To examine the effect of \(D_{th}\) and \(\Theta_{th}\), we used three different thresholds corresponding to three different precision levels as shown in Table 5.1.
Table 5.1: Different levels of matching precision

<table>
<thead>
<tr>
<th>Precision-level</th>
<th>Threshold</th>
<th>Class Distribution</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Train</td>
</tr>
<tr>
<td>Translation, $D_{th}$ (mm)</td>
<td>4</td>
<td>0.12</td>
</tr>
<tr>
<td>Rotation, $\Theta_{th}$ (radian)</td>
<td>4.5</td>
<td>0.14</td>
</tr>
<tr>
<td>Low</td>
<td>5</td>
<td>0.16</td>
</tr>
</tbody>
</table>

We next aligned the segment to the model’s initial pose using the transformation matrix returned by the registration step. Finally, before feeding into the network, we randomly sampled all the segments and models to a fixed number of points. In our experiments, we used $m=n=1,024$ points each for both the model and the segment.
Chapter 6

Experimental Results

6.1 Training and Experimental Setup

For training ValidNet, we feed the ground truth label of the validity of registration along with the model and segment point cloud in to the network. We have used Rectified linear unit (ReLU) [69] as an activation function on all the shared Multi Layer Perceptron (MLP) layers and the fully connected layers. ValidNet produces the output ($\nu$) based on the input sample. We have applied softmax function on the ValidNet output. The calculation of softmax is as follow-

$$\rho_i = \text{Softmax}(\nu_i) = \frac{\exp(\nu_i)}{\sum_j \exp(\nu_j)} \quad \text{for } i = 1, 2 \quad (6.1)$$

From the softmax output and ground truth label we have calculated the cross entropy loss. If the class label is denoted by $L$ (which is either 0 or 1), then the Loss function of our ValidNet network is defined by the equation- 6.2

$$Loss = -(L \log(\rho_1) + (1 - L) \log(\rho_2)) \quad (6.2)$$
6.2. RESULTS

This loss function is propagated backward and all the weights are updated based on the gradient of the loss. Adam optimizer [90] is used in order to update the model parameters.

The training setup is shown in Figure 6.1. In all of our experiments, we have used a batch size of 8 and an initial learning rate of 0.0001, which was reduced by a factor of 10 every 10 epochs. The network was trained for 60 epochs on a single NVIDIA Tesla P100 instance, which required \( \approx 13 \) hours of runtime. The points’ order of the segments were shuffled randomly everytime we feed it into the network. A dropout of 30% was used only on the fully connected layers. The code was implemented in Pytorch [91].

6.2 Results

To evaluate the effectiveness of our proposed method, we calculated overall accuracy and average class accuracy on all test samples in our dataset. The calculation
of overall accuracy and average class accuracy is provided in equation 6.3.

$$\text{Overall Acc.}(OA) = \frac{\text{Total Correct}}{\text{Total sample}} \times 100\%$$

$$\text{Avg. Class Acc.}(ACA) = \left( \frac{\text{Total correct positive}}{\text{Total positive sample}} + \frac{\text{Total correct negative}}{\text{Total negative sample}} \right) / 2 \times 100\% \quad (6.3)$$

We also compared ValidNet results with PointNet [2], and the method proposed by Papazov et al. [62]. PointNet is a well-known deep learning model for point cloud classification. In order to use PointNet for registration validation, we stacked two input point clouds (segment and model) before passing to the network, as shown in Figure 6.2. PointNet treats the combined point cloud as a single cloud and tried to classify it based on the global characteristics of the combined cloud. Since both point cloud (model and inverse transformed segment) are in the same canonical position, we haven’t used the input transform and feature transform network here as well. While using PointNet, we have tried both symmetric functions and found that max-pool is providing better performance than avg-pooling. We took PointNet with max-pool as our baseline.

While experimenting with the Papazov algorithm, we have transformed the model point cloud into scene position, as it requires the number of scene points under-neath it for measuring the penalty term. The different values of support and penalty thresholds were set for three precision levels for the Papazov algorithm. The results for ValidNet, PointNet with both symmetric functions, and the Papazov algorithm in terms of overall accuracy and average class accuracy for three different levels of matching precision are shown in Table 6.1.

From the table, it can be seen that different levels of matching precision do not have any impact on the relative performance of competing methods, or much impact.
6.2. RESULTS

Figure 6.2: PointNet for registration validation. It treats the model and the segment point clouds as a single point cloud, captures their global relationship and provides the validation probability of registration success.

<table>
<thead>
<tr>
<th></th>
<th>High Precision</th>
<th>Medium Precision</th>
<th>Low Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ACA</td>
<td>OA</td>
<td>ACA</td>
</tr>
<tr>
<td>PointNet (Avg pool)</td>
<td>81.65%</td>
<td>82.58%</td>
<td>82.23%</td>
</tr>
<tr>
<td>Papazov et al.</td>
<td>85.79%</td>
<td>85.58%</td>
<td>85.18%</td>
</tr>
<tr>
<td>PointNet (Max pool)</td>
<td>87.41%</td>
<td>87.71%</td>
<td>88.41%</td>
</tr>
<tr>
<td>ValidNet</td>
<td><strong>89.70%</strong></td>
<td><strong>89.82%</strong></td>
<td><strong>90.22%</strong></td>
</tr>
</tbody>
</table>

Table 6.1: Performance comparison on our dataset. ACA represents average class accuracy and OA represents overall accuracy.

on the accuracies attained. This indicates that the methods adapted well to different thresholds. Secondly, the results show that ValidNet outperforms Papazov et al. and PointNet for both evaluation criteria and all three precision levels. We believe that the major reason for ValidNet’s better performance over PointNet is that the characteristics learned by ValidNet’s feature extraction network are the combination of local and global characteristics of a point cloud, whereas PointNet captures only global characteristics. For a small misalignment between the model and the segment,
the overall global information does not change that much, which makes it difficult for \textit{PointNet} to classify those cases correctly. On the other hand, the local characteristics captured by \textit{ValidNet} change significantly resulting in better classification accuracy.

Figure 6.3 shows some sample inputs for which \textit{ValidNet} successfully validated the registration results, and Figure 6.4 shows some results where \textit{ValidNet} failed to validate correctly. By analyzing the failure cases for \textit{ValidNet}, we noticed that most of the misclassifications occur near the threshold level. In our problem, the difference between the positive and the negative classes are small as the data in our dataset is continuous and there is no sharp boundary between the two classes. We are selecting an arbitrary threshold (as discussed in section 5.5) to differentiate true and false detection which makes it difficult even for humans to differentiate between the two classes, and results in most misclassifications being near the threshold. In particular, the classification is determined by comparing the results against the ground truth pose, which is subtle and which may not be obvious solely by human visual comparison.

Figure 6.5 shows the effect of using \textit{ValidNet} on an object detection and pose estimation pipeline for industrial bin picking. Figure 6.5(a) shows the results of object detection and pose estimation before validation on few synthetic bins. In total there are 22 objects detected in the bin, with 13 results correct and 9 results incorrect. Figure 6.5(b) shows the results on the same bin after applying \textit{ValidNet}. The output demonstrates that \textit{ValidNet} can identify those false positives and remove these from the final detection result. As a result, in the final detection on those bins, there are 13 detections, all of which are correct results.
6.2. RESULTS

(a) True Positive Results: Positive classes detected as positive classes

(b) True Negative Results: Negative classes detected as negative classes

Figure 6.3: Some samples where ValidNet classified correctly, with segment points rendered black, and model points rendered green.
6.2. RESULTS

(a) False Negative Results: Positive classes detected as negative classes

(b) False Positive Results: Negative classes detected as positive classes

Figure 6.4: Some samples where ValidNet failed to classify correctly, with segment points rendered black, and model points rendered green.
6.2. RESULTS

(a) Results without ValidNet

(b) Results with ValidNet

Figure 6.5: Use of ValidNet for 3D object detection and pose estimation bin picking pipeline, with detected objects overlaid in green. ValidNet effectively detects and removes false positive (i.e. inaccurate) results.
6.3. ROBUSTNESS OF VALIDNET

6.3.1 Effect of Noise on ValidNet Architecture

Robustness against noise is an important property of any machine vision system to cater to any inherent unreliability of the sensor’s characteristics. We performed an experiment to test the robustness of our proposed method against noise. We have...
6.3. ROBUSTNESS OF VALIDNET

conducted the experiment on our low precision dataset. For each segment in the test data, we have added Gaussian noise on all the points independently, with mean 0 and specified standard deviations (S.D.), isotropic in all three dimensions. A segment of each model after the addition of noise is shown in Figure 6.6. We have tested our method, PointNet, and the Papazov algorithm with these data. While using PointNet we have used max-pool as the symmetric function, as it gave much better results then avg-pooling. While using the Papazov algorithm, as we used the complete scene for measuring the penalty term, we have added noise on all of the data in the bin scene, rather than just the isolated segment.

The comparative results of these algorithms in terms of average class accuracy on noisy data are shown in Figure 6.7. Though accuracy of all techniques suffer with increasing noise as expected, the performance of ValidNet degrades more gracefully

---

Joy Mazumder - Department of Electrical and Computer Engineering
As compared to that of PointNet and Papazov et al. As the S.D. of noise increases, the discriminative ability of both PointNet and Papazov et al. reduces steeply, whereas ValidNet continues to perform well as shown in Figure 6.7. The performance of ValidNet is almost unchanged up to noise S.D. = 1 (equal to approximately 0.70%-1.2% of the object radius, depending on the object) while PointNet performance starts degrading from noise S.D. = 0.5. The performance of the Papazov algorithm degrades nearly linearly with noise S.D. from the beginning.

We believe that this greater robustness of ValidNet against noise is due to its use of local characteristics of points. For higher noise levels, points move farther away from their original positions resulting in a drastic change in the global shape of the combined object. The change in points' local characteristics remains relatively less significant. Since ValidNet is dealing with the local characteristics of the points, it is capable of dealing with data having more noise. Furthermore, we are considering all the points of the model and calculating their matching precision. If due to noise few point’s matching score failed to capture required information for measuring the alignment verification, the other points’ matching score are able to encapsulate that information, which is why ValidNet is able to perform better than PointNet in these conditions. As noise increases for the Papazov algorithm, the support and penalty terms suffer, as it’s a calculation based on the distance of the model and segment points, which degrades the discriminative ability of the Papazov algorithm more drastically.

We saw that ValidNet performed well when it was trained on noiseless data and tested on noisy data. We performed another robustness experiment to evaluate its performance when noise is added to training data as well. The results are shown in
6.3. ROBUSTNESS OF VALIDNET

Figure 6.8: Effect of noise when noise is added on the training data as well. For both ValidNet and PointNet, the reduction in accuracy with noise is now much lower, which is expected as the networks have already seen noisy data during training. ValidNet again shows more robustness by outperforming PointNet. When the noise S.D. is 3, the performance of PointNet drops by approximately 3.5% from its original performance on noiseless data, while the ValidNet performance drops only by approximately 1.5%, as shown in Figure 6.8.

6.3.2 Effect of Missing Data from Segments

In real world in-the-wild applications, sometimes due to the specularity of the object surface, some data-points can’t be captured by the sensor, which results in some missing points on the sensed object surface. In order to examine the effect of missing data on our proposed model, we have conducted an experiment where we have trained ValidNet on perfect synthetic data, and while testing we have randomly...
Figure 6.9: Segments with missing data. Same segment from left to right 0%, 25%, 50%, 75%, 87.5%, 93.75% points are missing respectively.
removed some data point from the segments. We have randomly removed 25%, 50%, 75%, 87% and 93.75% points from the segments, so that the resulting segments had 768, 512, 256, 128, and 64 points respectively.

Figure 6.9 shows a segment of each object after removing the percentages of points described above. As the number of points decreases, the classification task became more and more difficult. We have tested ValidNet on these challenging data. For comparison purposes, we have also tested PointNet on missing data. While using PointNet we have used max-pooling as a symmetric function as it was giving much better performance than average pooling. The results are plotted in Figure 6.10.

From the figure, it’s visible that ValidNet performs well in terms of missing data. The performance change is very little up to 75% of the missing data. The performance starts degrading after that as data become more and more sparse, as shown in Figure 6.9. The performance of PointNet starts degrading when the missing data is
6.4. ABLATION STUDY

In order to select suitable components for ValidNet, we have tried multiple feature extractor and symmetric functions, which we describe in detail in later sections.

6.4.1 Feature Extractor

We have tried multiple feature extraction networks to see their impact on the ValidNet architecture. In particular, we have tried the DGCNN local feature extractor network [46], the local feature learning network of the PointNet classification network, and the feature learning network of the PointNet segmentation network. We have trained these three networks on our dataset with a low precision level ($D_{th} = 5\, mm$ and $\Theta_{th} = 0.16\, rad$). The results of these feature extractor network trials are given in Table 6.2. This test shows that that PointNet segmentation network feature outperforms both DGCNN and PointNet classification network’s local feature.

<table>
<thead>
<tr>
<th>Feature Extractor</th>
<th>Average class accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>DGCNN [46]</td>
<td>79.40%</td>
</tr>
<tr>
<td>PointNet classification network feature [2]</td>
<td>83.27%</td>
</tr>
<tr>
<td>PointNet segmentation network feature [2]</td>
<td>89.75%</td>
</tr>
</tbody>
</table>

Table 6.2: Performance of various feature extractor on our proposed architecture

higher than 25%. When there are only 6.25% points on the segments it’s very challenging even for humans to identify the true and false positive cases, but still ValidNet performs higher than 82.5% on that challenging situation, whereas the performance of PointNet at that stage dropped to approximately 63%. The experiment supports the greater robustness of ValidNet compared with PointNet in terms of missing data.
6.4. ABLATION STUDY

Figure 6.11: Relative comparison of Average Class Accuracy as a function of Noise S.D., for various symmetric functions. ValidNet\textsubscript{X,Y} means symmetric function X is used in the feature extraction network, and symmetric function Y is used in the similarity measurement network.

6.4.2 Symmetric Function

In the proposed ValidNet architecture, we used symmetric functions in both feature extraction network and similarity measurement network. In the feature extraction network, the symmetric functions are used for accumulating the global information of the model and segment point cloud, and in the similarity measurement network, the symmetric function is used for capturing the matching score of each model point. In both parts of the network, we have tried the two most popular symmetric functions, which are average and max-pool. We have tried combinations of the average and max-pooling functions in the feature extraction and similarity measurement networks.
6.5 Effects of Color Information

To examine the effect of color information, we have conducted an experiment by removing the color component from the point cloud prior to training ValidNet. Without the color information, each point is represented only by the \((x, y, z)\) values, so that the input dimension of our network is \((m \times 3) + (n \times 3)\). To make ValidNet compatible with this representation, we have changed the number of inputs from 6 to 3 in our first shared perceptron layer, and kept the rest of the network unchanged.

The results of executing ValidNet with and without color information is provided in Table 6.3. This experiments shows that the addition of color information along with the points’ \((x, y, z)\) coordinates improves overall performance.

<table>
<thead>
<tr>
<th>ValidNet without color</th>
<th>89.57%</th>
</tr>
</thead>
<tbody>
<tr>
<td>ValidNet with color</td>
<td>89.75%</td>
</tr>
</tbody>
</table>

Table 6.3: Effect of color information on our proposed architecture

Figure 6.11 shows the relative performance of the symmetric functions with respect to noise. From this experiment we have found that without noise, max pool in the feature extraction network and average pool in the similarity measurement network provides the best performance, but that this performance degrades sharply with increasing the noise on the test data. With the increase of noise, using average pooling in both parts of the network provides the best performance, which is higher than others in noisy data, as shown in Figure 6.11. As real-world scanners capture noisy data, we have selected to use average pooling in both cases for our proposed network.

Joy Mazumder - Department of Electrical and Computer Engineering
6.6. EFFECTS OF NUMBER OF POINTS OF THE MODEL AND THE SEGMENT

In order to check the effect of the number of model and segment points (i.e. the values of \( m \) and \( n \), respectively), we have conducted another experiment where we have set the different values of \( m \) and \( n \) to 512, 256, 128 and 64, and trained ValidNet for the low precision dataset. The performance of ValidNet with respect to number of points is plotted in Figure 6.12. In this figure, as the number of points decreases, the performance of ValidNet decreases as expected, although the decrease is quite gradual. For example, when the number of model and scene points \( m = n = 128 \), which is an 800\% decrease from \( m = n = 1024 \), the performance only drops by 0.5\%. Similarly, when the number of points is \( m = n = 64 \), it still performs at a level of more than 87\%.

Figure 6.12: Effect of number of model points and segment points on ValidNet. Here, we keep the number of model and segment points same

6.6 Effects of Number of Points of the Model and the Segment

Joy Mazumder - Department of Electrical and Computer Engineering
6.7 Visualization of Results

To visualize how well ValidNet captures the required matching information, we display activation output of the avg-pooling layer (after similarity matrix) as a heat map in Figure 6.13. In this figure, registration outputs along with heat maps of four samples each for positive and negative classes are shown. The figure shows two alternate heat map representations for each segment. The first one, which we call vector representation, shows the activation output for 1024 points sequentially in a vector (some width is added for better visualization). The second one, which we call point cloud representation, shows the activation output for 1024 points mapped to a point cloud.
their locations in the input point cloud segment.

The heat maps in Figure 6.13 show a clear differentiation between the positive and the negative classes. For the positive classes, the heat map is more towards the yellow-red side (showing higher activation outputs) while for the negative classes, the heat map goes more towards the green-blue region (showing lower activation outputs). The final fully connected layers of our network perform the final classification based on these values.
Chapter 7

Conclusion

The existence of local minima problem in Iterative Closest Point (ICP) and the challenges of automatic validation of registration results have been open and well known problems for many years. As Root Mean Square Distance (RMSD) error and residual distribution is not sufficient to solve that problem properly, a few other approaches were proposed before by the research community to solve that problem. Those approaches are not able to identify local minima accurately all the time, however, and their performance highly depends on the level of noise and missing data in the scene.

In this work, we have presented a novel approach and according to our knowledge, the first deep learning based architecture to verify 3D surface registration results for rigid parts. Our proposed method is able to provide good performance in different scenarios, such as with or without noise, and missing data. This method treats the validation problem as two class classification problem. We have explored the potential of using deep learning for the validation task. Our proposed architecture can extract the meaningful information from the point cloud using a deep shared Multi Layer Perceptron (MLP) architecture. Then these features were used to calculate the
point-wise relationship between the model and segment. Afterwards the symmetric function can accumulate the necessary information for the measurement of the validity of registration and makes the model invariant under the perturbation of the segments point order. Finally, the fully connected layers are capable of classifying that information.

To evaluate the performance of our proposed method, we have designed a large synthetic dataset for this hypothesis verification problem. While generating the data, we have modeled various physical properties such as gravity, friction and collision so that our generated data will look like the real world data with clutter and occlusion. For generating object hypotheses from the bin we have used RG segmentation and RASNAC based registration. To define true and false hypotheses, we have selected three different thresholds and evaluate the performance of our proposed ValidNet architecture on all three. We have shown that our method outperforms PointNet and Papazov et al. on our synthetic dataset for all the different threshold levels.

In order to examine the behavior of our model in noisy data and in scenes with missing information, we have also conducted robustness experiments on our ValidNet architecture. We have added various level of noise and missing data on the test set and have shown that our model is capable of exhibiting robust behaviour in these types of situations, and is able to outperform PointNet and the Papazov et al. algorithm. This robustness experiment supports the potential of using ValidNet at the real world industrial level.

For a particular sensor, sometimes it is possible to find the characteristics of the sensor noise. In order to demonstrate how ValidNet performs if noise can be modeled more accurately, we have conducted another set of experiments where noise is added.
on training data as well. We found that adding noise on the training data can make the architecture perform better in noisy data. This experiment informs us that, if the characteristics of the sensor noise can be modeled accurately, then ValidNet can provide strong performance in real world scenes.

In this work, we have also created a full 3D object detection and pose estimation pipeline, and shown some sample examples of the effect of using ValidNet in that pipeline. In this pipeline we have worked on object segments instead of the complete scene which reduces the search space and can be beneficial for time critical (e.g. real-time) applications. Another interesting approach of our method is that the network does not need to learn the translation and rotational invariance properties, since the scene segments are mapped to the same canonical model pose, which is very beneficial in real world applications. In the real world scenario, the dimension of the bin, the height of the bin from the camera, and the pose of the camera might vary. Our method can be applied in all those cases without any modifications.

In future, further investigation of this work can be performed. Experiments can be performed by increasing ICP iterations where ValidNet predicts negative classes to see it’s impact on final object detection accuracy. Experiments on different objects with color variations (as in our dataset color variation is limited) can be done to see the effect of color texture of the proposed architecture. A point’s local feature information along with its position and color information can be added into the network to see the effect of that. Another potential set of approaches can be explored by converting the point cloud of the model and scene into depth images. The scene can be cropped based on the dimension of the transformed model and those depth images used as input to design a network that can deal with that representation.

Joy Mazumder - Department of Electrical and Computer Engineering
Bibliography


Joy Mazumder - Department of Electrical and Computer Engineering


Joy Mazumder - Department of Electrical and Computer Engineering


