

OBJECT LOCALIZATION IN 3D POINT CLOUD

CHUNG HUI SZE


**A project report submitted in partial fulfilment of the
requirements for the award of Bachelor of Engineering
(Honours) Biomedical Engineering**

**Lee Kong Chian Faculty of Engineering and Science
Universiti Tunku Abdul Rahman**

May 2020

DECLARATION

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at UTAR or other institutions.

Signature :  _____

Name : Chung Hui Sze _____


ID No. : 15UEB03715 _____

Date : 16 May 2020 _____

APPROVAL FOR SUBMISSION

I certify that this project report entitled “**OBJECT LOCALIZATION IN 3D POINT CLOUD**” was prepared by **CHUNG HUI SZE** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Engineering (Honours) Biomedical Engineering at Universiti Tunku Abdul Rahman.

Approved by,

Signature : 

Supervisor : Dr. Ng Oon-Ee

Date : 16 May 2020

The copyright of this report belongs to the author under the terms of the copyright Act 1987 as qualified by Intellectual Property Policy of Universiti Tunku Abdul Rahman. Due acknowledgement shall always be made of the use of any material contained in, or derived from, this report.

© 2020, Chung Hui Sze. All right reserved.

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to everyone who had supported me throughout this project. First, I would like to thank my FYP supervisor, Dr. Ng Oon-Ee for his invaluable suggestions, advices, practical guidance and patience which had successfully helped me in completing my final year project. In addition, I would wish to express my sincere thanks to all my friends and family for giving me endless support and encouragement throughout this period.

ABSTRACT

Object localization in 3D point cloud is one of the most complex yet interesting applications in computer vision, robotics and autonomous agents. The results of object localization are often affected by many factors such as the quality of the point clouds and the sensitivity of the algorithms to the occlusion in the point clouds.

This project provides an efficient algorithm that is able to recognize and localize more than one object from the scene at the same time and is also able to perform localization of an object which undergoes a transformation. There are a total of four major steps to perform the object localization in the 3D point cloud - Scale Invariant Feature Transform (SIFT) keypoint detection to mark the descriptive points in the cloud, Signature of Histograms of Orientations (SHOT) descriptor construction to store the geometrical properties of the keypoints, feature matching to collect point-to-point correspondences between the scene and the model and Hough Voting hypotheses generation to construct a model instance and localize it from the scene. In this project, adjustment of the parameters in each step was carried out to analyse their effects on the final localization result. The results obtained from each step based on the parameter adjustment were analysed and discussed.

Highly descriptive keypoints were detected by using SIFT detector as the keypoints were mostly located at the outlines of the point clouds. In the descriptor construction step particularly, two methods, Point Feature Histogram (PFH) and Signature of Histograms of Orientations (SHOT) were compared. SHOT's performance was better than PFH as it had a higher efficiency in computing the descriptors. The high accuracy rate of the feature matching process indicated that the process was able to generate correct correspondences between the scene and the model. In the final localization step, with the adjustment of the parameters, the result shows that this algorithm was able to correctly localize all input models from the scene point cloud, achieving a 100% localization accuracy.

TABLE OF CONTENTS

DECLARATION		i
APPROVAL FOR SUBMISSION		ii
ACKNOWLEDGEMENTS		iv
ABSTRACT		v
TABLE OF CONTENTS		vi
LIST OF TABLES		ix
LIST OF FIGURES		xi
LIST OF SYMBOLS / ABBREVIATIONS		xv
CHAPTER		
1	INTRODUCTION	
1.1	General Introduction	1
1.2	Importance of the Study	3
1.3	Problem Statement	4
1.4	Aim and Objectives	4
1.5	Scope and Limitation of the Study	5
1.6	Contribution of the Study	5
1.7	Outline of the Report	5
2	LITERATURE REVIEW	
2.1	Introduction	6
2.2	3D Keypoint Detection (Feature Extraction)	6
2.2.1	Fixed-Scale Keypoint Detection	7
2.2.2	Adaptive-Scale Keypoint Detection	8
2.2.3	Summary and Comparison between Fixed-Scale and Adaptive-Scale Keypoint Detection Methods	10
2.3	Local Surface Feature Description	11
2.3.1	Histogram-Based Methods	12
2.3.2	Signature-Based Methods	19

2.3.3	Summary and Comparison between Histogram-Based and Signature-Based Local Surface Feature Description Methods	20
2.4	Surface Matching (Coarse Recognition and Localization)	23
2.4.1	Feature Matching	24
2.4.2	Summary of Feature Matching	26
2.4.3	Hypothesis Generation	28
2.4.4	Summary of Hypothesis Generation	33
2.5	Fine Localization (Verification)	36
2.5.1	Individual Verification Methods	36
2.5.2	Global Verification Methods	36
2.5.3	Summary and Comparison between Local and Global Verification Methods	37
2.6	Summary	39
3	METHODOLOGY AND WORK PLAN	
3.1	Introduction	40
3.2	Point Cloud Library (PCL)	40
3.3	Project System Overview	40
3.3.1	Surface Normal Estimation	42
3.3.2	Keypoint Detection	43
3.3.3	Descriptor Construction	44
3.3.4	Feature Matching	46
3.3.5	Hypotheses Generation	47
3.4	Project Timeline	49
3.5	Summary	50
4	RESULTS AND DISCUSSION	
4.1	Introduction	52
4.2	Dataset	52
4.3	Normal Estimation	52
4.4	Keypoint Detection	56
4.4.1	Keypoint Quantity and Quality	57
4.4.2	Keypoint Repeatability	59

4.4.3	Keypoint Detector's Efficiency / Time Performance	71
4.5	Descriptor Construction	72
4.6	Feature Matching	77
4.7	Hypotheses Generation	78
4.8	Overall Parameter Set and Final Results	80
4.9	Summary	82
5	CONCLUSIONS AND RECOMMENDATIONS	
5.1	Conclusions	83
5.2	Recommendations for Future Work	83
	REFERENCES	85

LIST OF TABLES

Table 2.1: Summary of Methods of 3D Keypoint Detection.	10
Table 2.2: Summary of Methods for Local Surface Feature Description.	21
Table 2.3: Summary of Methods of Feature Matching.	27
Table 2.4: Summary of Methods of Hypotheses Generation.	33
Table 2.5: Summary of Methods of Verification.	38
Table 4.1: Time Taken for Surface Normal Estimation for Each Point Cloud at Different k.	55
Table 4.2: Number of Detected Keypoints for Crocodile, Seal, Basin Models and Scene Point Cloud at Different Minimum Scale.	58
Table 4.3: Table of Number of Original and Repeated Keypoints for Input Crocodile, Seal, Basin Models and Scene Point Clouds at Different Minimum Scale of Gaussian Scale Space.	60
Table 4.4: Number of Repeated Keypoints between Input Models and Scene at Different Minimum Scale.	63
Table 4.5: Mean Distance and Standard Deviation Between Model and Scene Point Cloud After Matching at Different Minimum Scale.	64
Table 4.6: Percentage of Keypoints Within Threshold Between Model and Scene Point Cloud After Matching at Different Minimum Scale.	65
Table 4.7: Numbers of Repeated Keypoints between Own Models Point Cloud Under Different Minimum Scale.	68
Table 4.8: Percentage of Keypoints Within Threshold for Input Models at Different Minimum Scale.	68
Table 4.9: Numbers of Exact Repeated Keypoints and Percentage of Repeated Keypoints within Threshold between Original and Rotated Keypoints for Each Model.	70
Table 4.10: Model's Keypoint Computation Time at Different Minimum Scale.	71
Table 4.11: Comparison of Computational Time for PFH and SHOT Descriptors at Different Radius for Neighbour Searching.	76
Table 4.12: Quantity of Original and Correct Correspondences and Accuracy of Feature Matching from Different Thresholds.	78

Table 4.13: Number of Model Instances Generated from Different <i>cg_thresh</i> .	79
Table 4.14: Parameters Set for Each Step in Model Localization.	80
Table 4.15: Dimension of Models Localized from Scene	80
Table 4.16: Parameters Set for Rotated Crocodile Localization.	81

LIST OF FIGURES

Figure 1.1: Object Recognition and Localization using Convolutional Neural Networks (Nicholson, 2019).	2
Figure 2.1: Detected Keypoints (red dots) (Mian, Bennamoun and Owens, 2010).	10
Figure 2.2: Oriented Point Basis (Johnson and Hebert, 1999).	13
Figure 2.3: Histogram of 2D Distribution of Parameters C1 versus C2 (Bielicki and Sitnik, 2013).	14
Figure 2.4: Building of Reference Object Descriptor (Bielicki and Sitnik, 2013).	14
Figure 2.5: Histogram Bins that Form 3D Shape Context (Frome, et al., 2004).	15
Figure 2.6: Comparison of Recognition Rate between Different Descriptors in Noise Experiments (Frome, et al., 2004).	16
Figure 2.7: Comparison of Recognition Rate between Different Descriptors in Clutter Experiments (Frome, et al., 2004).	16
Figure 2.8: A Keypoint with Two Least Squares Planes and their Related Normals for One Support Point on a Local Surface s (Flint, Dick and Hengel, 2014).	17
Figure 2.9: Tensor Computation (Mian, Bennamoun and Owens, 2006).	18
Figure 2.10: (a) Local Coordinate System. (b) Local Fingerprint of the Same Point from Different Views (Sun and Abidi, 2001).	19
Figure 2.11: Point Signature (Chua and Jarvis, 1997).	20
Figure 2.12: Sequence of Matching Two Points using Intrinsic Shape Signature (Zhong, 2009).	26
Figure 2.13: Schematic of Scale-Hierarchical Interpretation Tree (Bariya and Nishino, 2010).	29
Figure 2.14: Algorithm of RANSAC to extract Shapes in the Point Cloud (Schnabel, Wahl and Klein, 2007).	32
Figure 3.1: Overview of Object Localization in 3D Point Cloud.	41
Figure 3.2: Flowchart of Object Localization in 3D Point Cloud.	41

Figure 3.3: Timeline of Second Part of Project.	50
Figure 4.1: Visualization of Surface Normals for All Point Clouds at $k = 2$.	54
Figure 4.2: Visualization of Surface Normals for Crocodile (Top: $k = 4$; Bottom: $k = 10$).	54
Figure 4.3: Visualization of Surface Normals for Seal (Left: $k = 4$; Right: $k = 10$).	54
Figure 4.4: Visualization of Surface Normals for Basin (Left: $k = 4$; Right: $k = 10$).	54
Figure 4.5: Visualization of Surface Normals for Scene (Top: $k = 4$; Bottom: $k = 10$).	55
Figure 4.6: Graph of Models' Surface Normal Computational Time against k Neighbourhood Size.	55
Figure 4.7: Visualization of Crocodile's Detected Keypoints (Left: <i>Min Scale</i> 82; Right: <i>Min Scale</i> 65).	58
Figure 4.8: Visualization of Seal's Detected Keypoints (Left: <i>Min Scale</i> 82; Right: <i>Min Scale</i> 65).	58
Figure 4.9: Visualization of Basin's Detected Keypoints (Left: <i>Min Scale</i> 82; Right: <i>Min Scale</i> 65).	58
Figure 4.10: Visualization of Scene's Detected Keypoints (Above: <i>Min Scale</i> 82; Below: <i>Min Scale</i> 65).	59
Figure 4.11: Graph of Number of Total and Repeated Scene's Keypoints against Minimum Scale of SIFT Keypoint Detector.	61
Figure 4.12: Graph of Number of Total and Repeated Crocodile's Keypoints against Minimum Scale of SIFT Keypoint Detector.	61
Figure 4.13: Graph of Number of Total and Repeated Seal's Keypoints against Minimum Scale of SIFT Keypoint Detector.	61
Figure 4.14: Graph of Number of Total and Repeated Basin's Keypoints against Minimum Scale of SIFT Keypoint Detector.	62
Figure 4.15: Visualization of Unique Crocodile's Keypoints (Left: <i>Min Scale</i> 82; Right: <i>Min Scale</i> 65).	62
Figure 4.16: Graph of Number of Repeated Keypoints Between Three Input Models and Scene at Different Min Scale.	63

Figure 4.17: C2C Absolute Distance Display Range of Crocodile Model at Minimum Scale of 70.	65
Figure 4.18: Normal distribution of Crocodile Model at Minimum Scale of 70.	65
Figure 4.19: C2C Absolute Distance Display Range of Seal Model at Minimum Scale of 70.	66
Figure 4.20: Normal distribution of Seal Model at Minimum Scale of 70.	66
Figure 4.21: C2C Absolute Distance Display Range of Basin Model at Minimum Scale of 70.	66
Figure 4.22: Normal distribution of Basin Model at Minimum Scale of 70.	67
Figure 4.23: Histogram of C2C Absolute Distance of Crocodile Model at Minimum Scale of 70..	68
Figure 4.24: Histogram of C2C Absolute Distance of Seal Model at Minimum Scale of 70.	69
Figure 4.25: Histogram of C2C Absolute Distance of Basin Model at Minimum Scale of 70.	69
Figure 4.26: Crocodile's Keypoints Before and After 20° Rotation (Left: Rotated; Right: Original)..	69
Figure 4.27: Seal's Keypoints Before and After 20° Rotation (Left: Rotated; Right: Original).	70
Figure 4.28: Basin's Keypoints Before and After 20° Rotation (Left: Rotated; Right: Original).	70
Figure 4.29: Graph of Model's Keypoint Computation Time at Different Min Scale.	72
Figure 4.30: Visualization of PFH Output Histogram for Crocodile (Top: $r = 20$; Bottom: $r = 60$).	73
Figure 4.31: Visualization of PFH Output Histogram for Seal (Top: $r = 20$; Bottom: $r = 60$).	73
Figure 4.32: Visualization of PFH Output Histogram for Basin (Top: $r = 20$; Bottom: $r = 60$).	74
Figure 4.33: Visualization of PFH Output Histogram for Scene (Top: $r = 20$; Bottom: $r = 60$).	74

Figure 4.34: Visualization of SHOT Output Histogram for Crocodile (Top: $r = 20$; Bottom: $r = 60$).	74
Figure 4.35: Visualization of SHOT Output Histogram for Seal (Top: $r = 20$; Bottom: $r = 60$).	75
Figure 4.36: Visualization of SHOT Output Histogram for Basin (Top: $r = 20$; Bottom: $r = 60$).	75
Figure 4.37: Visualization of SHOT Output Histogram for Scene (Top: $r = 20$; Bottom: $r = 60$).	75
Figure 4.38: Graph of Comparison of Descriptor Computational Time between PFH and SHOT for Scene against Radius.	77
Figure 4.39: Localization of Three Models from Scene.	81
Figure 4.40: Localization of Rotated Crocodile Model from Scene.	82
Figure 4.41: Localization of Rotated Seal Model from Scene.	82

LIST OF SYMBOLS / ABBREVIATIONS

C	covariance matrix
C	central
$C1$	mean curvature
$C2$	Gaussian curvature
C^M	model's centroid
d	distance threshold
F	keypoint
I	point cloud
k	number of nearest neighbours
N	neighbours
n	surface normal
$O(k^2)$	complexity of PFH
\bar{p}	Centroid of neighbours in neighbourhood
p	targeted point
R	rotation matrix
R	radius of sphere
v	eigenvector
λ	eigenvalue
γ	constraints
α	radial coordinate
β	elevation coordinate
σ	surface variation
σ	standard deviation
θ	rotation angle
C2C	Cloud-to-Cloud
DoG	Difference-of-Gaussian
FS	feature space
FV	feature vector
GRF	global reference frame
ICP	iterative closest point

<i>kD</i>	k dimensional
LRF	local reference frame
LSH	locality sensitive hashing
LST	locality sensitive trees
LVD-LSDs	local variable dimension local shape descriptors
PCA	principal component analysis
PCD	point cloud data
PCL	point cloud library
PFH	point feature histogram
RANSAC	Random Sample Consensus
RoPS	rotational projection statistics
SHOT	Signature of Histograms of Orientations
SIFT	Scale Invariant Feature Transform
SURF	Speeded up Robust Feature

CHAPTER 1

INTRODUCTION

1.1 General Introduction

Today, computer vision is slowly encompassing both industrial and non-industrial applications such as civil engineering and entertainment. It combines both hardware and software technologies to contribute an operational guidance and instruction to systems and devices on image capturing and processing. According to Marr (2019), computer vision is one of the fields under artificial intelligence where the machine targets to replicate human perception to analyse visual information, understand the environment and situation, process them and thereafter work on an image. This technology is often applied to recognize and identify objects from images. In certain situations, it can be said to have outperformed human's recognition capabilities. This is because it can operate more consistently, measure orders of magnitude faster and more accurate and will not get tired as easy as human. There are some advanced technologies built from computer vision, such as autonomous vehicles, facial recognition, healthcare, agriculture, manufacturing and real-time sports tracking.

Computer vision is a fast-growing area of research. Tremendous amount of visual data is one of the reasons behind the fast growth of computer vision where they are used to train and test the machines. With these huge datasets, computer vision is able to do computations involving visual data such as image classification, image captioning, semantic segmentation, instance segmentation, object recognition (object detection) and object localization.

Object classification is an application of identifying and assigning a class label to an input image. Instance segmentation functions not just to find objects in an image, but also to create a mask for every object detected. Next, the function of object localization is to locate the presence of objects in an image and draw a bounding box around the position of objects. For object detection, it is more complex as it first needs to identify objects and then draw a bounding box around the object of interest in the image. In order to localize an object, object detection needs to be performed first. According to Brownlee (2019), when a user or practitioner refers to "object recognition", they often imply

“object detection”. To perform object localization, the algorithm is fed with an input image with one or more objects. After processing, it will return an object which is desired to be detected with an axis-aligned bounding box representing its orientation and scale.

Object recognition and localization in an image is one of the most complex applications in computer vision, robotics and autonomous agents. This task still remains challenging because of some constraints such as the number of objects need to be localized, complexity of the scene, background clutter and occlusions. According to Huang and You (2013), they mentioned that it is harder to detect and localize the target object in an unstructured, non-image-based data input, such as 3D point clouds than 2D images. This is because that the point cloud data contains lesser structural information and has a high complexity to extract information from 3D data. Within these few years, multiple algorithms that have been proposed to process 3D point cloud data to fix this problem. One of the most common methods to complete this task is to slide a window across the image and to detect whether the local window contains the target or background using Convolutional Neural Networks (CNN), as shown in Figure 1.1. However, inefficiency of algorithms to detect objects in a highly occluded scene still exists as there is lack of sufficient labelled data to train the classifier model. Therefore, the task is now focusing on constructing a good classifier to make the object localization easier.



Figure 1.1: Object Recognition and Localization using Convolutional Neural Networks (Nicholson, 2019).

1.2 Importance of the Study

In the past, most of the object recognition and localization strategies in conventional computer machine mainly focus on the processing of RGB images using image-based algorithms. However, in recent years, the acquisition and processing of 3D point clouds data of real world scenes in object recognition and localization is gaining more and more attention.

Object recognition and localization is very important in today's world as it can assist human in accelerating and improving performance. The applications of object recognition and localization can be found in many areas, including image retrieval, object tracking, surveillance and security, autonomous cars, robotics, indoor navigation and others. For example, self-driving cars require object recognition and localization to localize vehicles or road signs in the environment to analyse when to accelerate or apply brakes. Besides, robots utilized for domestic housekeeping or elderly care must have the ability to detect and localize certain objects efficiently while performing searching tasks. In medical image analysis, object recognition and localization can be used to identify the object (for examples: heart and tumour) correctly together with the location and scale detected.

Today, 3D scanners can scan objects in an image and convert the raw data into point cloud representation. A 3D point cloud of an image is basically presented in a form of a set of points in a 3D coordinate system with x, y, and z coordinates. In a 3D point cloud, it usually lies on the surface of the object without containing any information of the object surface, such as the colour, texture or features. The reason why recognition and localization of object in 3D point clouds is so important is that the point clouds generated by the 3D scanners and 3D imaging are easy for measurement. Besides, the cameras used for indoor mapping purposes which convert image into unstructured point clouds require less power and are significantly cheaper than laser scanners. Furthermore, acquisition of point clouds of real world scenes can improve the processing ability of the computers and contribute to advanced multiple-image reconstruction algorithms.

1.3 Problem Statement

As mentioned before, recognition and localization of objects in a 3D image are the primary research problem exist in computer vision. They are considered one of the most challenging as there is a lack of generic 3D data or labelled training data to build efficient classifiers. There are plenty of image-based algorithms available but insufficient point cloud processing algorithms to perform object recognition and localization. This is because that the properties of the point clouds constructed from real world scene produce more challenges than general algorithms.

First, point cloud data might consist of noise points which are the outliers that are not part of the scene. These noise points can disturb the detection of the keypoints which further reduce the overall accuracy. Also, some of the point cloud images available are low in resolution which are constructed from poor sensors that only perform sparse reconstructions. Furthermore, certain existing algorithms are sensitive to loss of information or occlusion as the algorithms are difficult to identify the actual shape of the underlying surface of an object. This problem can affect the performance of surface matching in object recognition step. Besides, there are not many methods available in open source that can be used to measure the scale and dimensions of the detected objects.

In short, a method that can perform robustly against all constraints such as noise, clutter and missing data needs to be designed which can recognize and localize objects on 3D point cloud images, rather than normal RGB images.

1.4 Aim and Objectives

The aim of performing object localization in 3D point clouds is to explore robust methods for locating and measuring objects within a 3D point cloud.

In order to localize an object, object searching and recognizing have to be performed first. There are mainly three objectives to perform this task. The first objective is to search and identify existing rich 3D point cloud datasets which consist of multiple objects inside to be used in this project. The second objective is to develop a method for matching an object (2D or 3D) to objects within the point cloud and the third objective is to measure the dimensions of the located object.

1.5 Scope and Limitation of the Study

The scope of this research and study is to develop an algorithm which is robust and able to recognize an object in the scene with multiple objects efficiently and calculate the dimensions of the located object. The study is mainly focusing on detecting and locating an object in 3D point clouds. The methodology proposed is divided into two main parts, where object recognition is performed first, followed by object localization.

There are a few limitations whilst performing the study of object localization in 3D point clouds. First, there is lack of available resources in terms of point cloud images. Most of the image resources used in this study are RGB images. Even with the point cloud images, it is difficult to find a point cloud image with multiple objects inside. Furthermore, most of the latest advanced algorithms used in this study involve deep learning. Deep learning in image processing is getting popular in the recent years, thus there is a lack of research studies on this method.

1.6 Contribution of the Study

This project provided an insightful summary of all the existing methods that had been implemented in the object localization in 3D point cloud. Besides, the comparison of performance between each technique that was presented in this study could provide a quick understanding of all the methods for those who wish to perform a similar project. This project provides an algorithm that could recognize and localize rotated objects and multiple objects from a scene point cloud at the same time. The study could also let others to understand how the parameters set in the project affect the results.

1.7 Outline of the Report

In this report, Chapter 2 outlines the comprehensive summary and analysis of all existing methods used in each step to perform object localization in 3D point cloud. Details of all the techniques implemented are presented in Chapter 3. Results are shown and the detailed analysis and discussion are provided in Chapter 4, followed by the conclusion and recommendations in Chapter 5.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

The task of object recognition and localization is getting more and more popular and most of the task is performing on digital images. Guo, et al. (2014) stated that object recognition and localization in the rich scenes can be separated into two types, the global and local feature-based methods. The difference between these two methods is that the global feature-based methods consist of a group of global features which can outline the whole desired object while the local feature-based methods only concern about the local surfaces that surround the particular interest points.

For global feature-based methods, there is a need to perform object segmentation from the scene. This method does not take the object's shape information into account. Thus, global feature-based methods are more often used in 3D shape retrieval and classification rather than recognition of objects that might be occluded from the cluttered scenes. Since local feature-based methods can generally deal with clutter and occlusion better, therefore they are often utilized to recognize and localize object from scenes. Local 3D object recognition and localization in cluttered scenes has a sequence of steps: 3D interest point detection (feature extraction), construction of descriptor for local surface feature, surface matching (coarse recognition and localization) and fine localization (verification). This chapter reviews current work related to each of these steps.

2.2 3D Keypoint Detection (Feature Extraction)

In the first step of object recognition, keypoint detection or feature extraction needs to be performed to search for the keypoints which are 3D points with discriminative information content. This step is performed to detect the inherent scale of each keypoint. The location and scale of a keypoint that are obtained here will be used to determine a local surface. These will be further used to generate descriptors in next step. Therefore, detection of keypoint locations is very important as it strongly determines the success of local feature descriptors.

In 2D images, methods like Harris corner detector and SURF are often used to identify keypoints that have a high chance to be well-localized. Example of salient points is the corner points which have a high intensity gradient in all directions. For 3D images, the main idea is the same where keypoints that can be well localized need to be defined. The only difference is that it needs a keypoint which has a high surface spatial in all three directions to extract the unique local 3D coordinate basis. There are generally two methods to perform keypoint detection, fixed-scale and adaptive-scale methods.

2.2.1 Fixed-Scale Keypoint Detection

In this method, a point that is unique around its point neighbourhood is detected as keypoint by using either curvatures or surface variation measures.

A few authors have utilized surface variation measures such as using eigenvalues to extract keypoints. First, Matei, et al. (2006) figured out that before constructing descriptor, the scene features of keypoints that have rich 3D information need to be computed first. Selection of salient point is performed by computing the eigenvalues of the scatter matrix at all 3D points. Since the smallest eigenvalues λ_3 of the scatter of the neighbouring points contributes a good 3D saliency measure, they are used to compute the surface variation that surrounds a point p . Then, all point candidates are arranged based on their surface variations into a list and the keypoints will be chosen from the sorted list.

A similar idea was utilized by Zhong (2009) to detect keypoints before building a shape based descriptor to recognize 3D objects. The surface normal vector of the local surfaces is often utilized to construct descriptors. However, Zhong (2009) mentioned that by using only the normal vector of a surface, a 3D coordinate structure of a point still cannot be determined as there is not sufficient information. Therefore, they decided to build a local reference frame at each point. First, they calculated a scatter matrix for the point by utilizing all the neighbouring points. Then, they calculated three eigenvalues and the corresponding eigenvectors of the matrix. The salient points which are rich in 3D structure possess huge 3D point variations among the neighbouring points. To find the points, the smallest eigenvalue is used to determine these variations. When two eigenvalues are the same, they applied additional constraints on the

ratio of two successive eigenvalues only selected the point which satisfies $\lambda_2/\lambda_1 < \gamma_{21}$ and $\lambda_3/\lambda_2 < \gamma_{32}$. This Eigen analysis method which was implemented by Matei, et al. (2006) and Zhong (2009) are useful as they can be computed efficiently and achieve excellent outcomes in terms of repeatability.

Glomb (2009) mentioned the advantages of using Harris operator such as it is more robust to noise, has high repeatability and sufficient information content. They summarized four reasons and propositions to extend Harris operator from 2D images to 3D meshes. Examples of the propositions are using Gaussian function built from the point cloud, utilizing derivative of fitting quadratic surface and others. Details of the methods are discussed in the paper of Sipiran and Bustos (2011) as they referred to the work of Glomb (2009) as a basis to build an improved version of Harris operator in detecting keypoint on 3D meshes. First, centroid of point neighbourhood is calculated and a set of neighbouring points is translated to the centroid. The best-fitting plane is computed to the translated points and the points are rotated so that the normal of the fitting plane aligns with the z-axis. Next, the points are fitted to a quadratic surface to compute derivatives. A quadratic surface is chosen as it is simple enough to express a function of two variables using quadratic terms. Then, they used the derivatives of the function to formulate a matrix to eliminate noise. Now in the matrix, each vertex is associated with its Harris operator value. Finally, the vertex which fulfilled the stated condition is selected as keypoint.

2.2.2 Adaptive-Scale Keypoint Detection

In the paper “Local 3D Structure Recognition”, Flint, Dick and Hengel (2014) mentioned that it is important to detect the significant keypoint locations in order to further construct the 3D descriptors. They utilized the adaptive-scale keypoint detection to extract interest points. This method constructs a scale space for input images. Points that are unique and contain high distinctiveness measures in scale and spatial neighbourhoods will be selected as keypoints. A descriptor that contains strong keypoint information can recognize 3D models more efficiently in the range data of scene. In range data, the keypoints need to be well-localized in three dimensions. Since the range data is a set of points, they first constructed a 3D density map to get a density function by sampling the points across the data set. Next, they convolved the 3D density map with a

group of Gaussian kernels to form a density scale-space. To detect the keypoints located in the density scale space, they applied the determinant of Hessian matrix. In the matrix, the local maxima will become keypoints.

The reason for applying Hessian matrix to search for the keypoints is that it contributes to an accurate calculation and it is determined for arbitrary scale. Based on the experiment conducted, Hessian matrix method can detect the same keypoint under a range of transformations, achieving a high level of repeatability. However, it is quite time-consuming to perform the sampling step in the matrix scale-space over the data.

Mian, Bennamoun and Owens (2010) presented an algorithm used to detect keypoints which have a high repeatability between 3D models and its partial views. They also created a keypoint quality measurement technique to rank the keypoints in order to choose the best ones. There are three constraints proposed to define the keypoints. First, the keypoints must contain high repeatability. Second, the keypoints must have a 3D coordinate basis building from local surface in neighbourhood. Third, the surface of keypoints must have enough descriptive information.

First, they cropped out a local surface from the 3D model to obtain a local reference frame which is insensitive to noise. They then rotated the neighbouring points on the cropped surface in order to align the normal of the point with the positive z-axis. Then, Principal Component Analysis (PCA) is executed on the neighbourhood's covariance matrix to remove polygons that are occluded in the cropped surface. The first two principal axes ratio is used to measure the surface variations. A threshold is set for surface variation comparison to choose the keypoints. Moreover, an automatic scale selection is proposed to define the scale of a keypoint as the neighbourhood size when there is a local maximum occurs in the surface variations, as shown in Figure 2.1. The keypoints detected have high repeatability and are insensitive to noise. It still has a drawback which is the incapability to perform efficient computation.

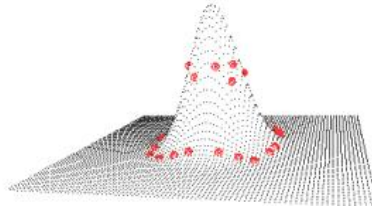


Figure 2.1: Detected Keypoints (Red Dots) (Mian, Bennamoun and Owens, 2010).

2.2.3 Summary and Comparison between Fixed-Scale and Adaptive-Scale Keypoint Detection Methods

Table 2.1 summarizes all methods used to extract keypoint or features. In fixed-scale based method, there is a possibility where only low number of keypoints detected as it is using a fixed scale, which subsequently will result in poor object recognition rate. Besides, this method defines the scale empirically, which means that it does not extract the scale information entirely in the local surfaces. For adaptive-scale based method, it might perform better than the fixed scale based method as it samples all candidate points in a 3D density map, ranks all the keypoint using quality measurement technique and finally chooses the qualified keypoints based on certain threshold.

Table 2.1: Summary of Methods of 3D Keypoint Detection.

No.	Method Category	Method Name	Data Type	Outcomes	Reference
1	Fixed Scale	Eigen Analysis	Point Cloud	This method has high repeatability and can be computed efficiently.	Matei, et al. (2006)
2	Fixed Scale	Eigen Analysis	Point Cloud	This method is computationally efficient and highly repeatable.	Zhong (2009)
3	Fixed Scale	Harris Operator	Mesh	Reasons and propositions of	Glomb (2009)

				extending Harris operator from 2D images to 3D meshes are concluded.	
4	Fixed Scale	Harris Operator	Mesh	It is robust to noise, local scaling, holes and has high repeatability and sufficient information content.	Sipiran and Bustos (2011)
5	Adaptive Scale	Gaussian kernels & Hessian Matrix	Point Cloud	The method is accurate and efficient but it is sensitive to point density variations and time-consuming.	Flint, Dick and Hengel (2014)
6	Adaptive Scale	Principal Component Analysis (PCA) & Automatic Keypoint Scale Selection	Point Cloud	This approach is insensitive to noise and has high repeatability, but it is computationally inefficient.	Mian, Bennamoun and Owens (2010)

2.3 Local Surface Feature Description

Once the keypoints are extracted, the information of each keypoints needs to be presented clearly. To complete this, the descriptive local surface information of each keypoint will be used to build a keypoint descriptor. According to Guo, et al. (2014), there are mainly two categories of descriptors for interest feature points, the histogram-based and the signature-based methods.

2.3.1 Histogram-Based Methods

Local feature histogram descriptors mainly determine the local neighbourhood of a feature by grouping geometric or topological measurements into histograms. In short, a simple descriptor can be created with the distribution of the pixel intensities represented by histograms.

In the paper “3D shape-based object recognition system in scenes containing clutter and occlusion”, Johnson and Hebert (1999) created spin image descriptors to efficiently recognize multiple objects in cluttered 3D scenes. The spin image is also known as a data level shape descriptor which is used to pair or match surfaces represented as surface meshes. In this method, oriented points that associated with a direction are utilized to construct spin image descriptors.

First, every oriented point lying on an object’s surface is further described with a surface position and surface normal. Now, the point contains an information of two dimensional local coordinate basis. The two coordinates basis are the radial coordinate α and elevation coordinate β . By using this oriented point basis, a spin map can be defined which projects three dimensional points to the two dimensional coordinates basis related to the oriented point. Now, each oriented point on the object surface has its own unique spin map coordinates (α, β) . Figure 2.2 shows the basis of an oriented point. A 2D accumulator indexed by both α and β is constructed. Next, the bin which is indexed by the coordinate in the accumulator is then increased to update the 2D accumulator. The 2D array accumulator is bilinear interpolated to smooth the contribution of the vertex, causing the accumulator to have a less sensitivity to the vertex’s position, to obtain ideal spin image descriptors. All these steps are repeated to process all the points located on the model’s surface. In the end, spin image descriptor with a two dimensional array representation is developed.

By using spin image descriptors in object recognition algorithm, the algorithm can handle clutter and occlusion well as proved in the experiment part of the paper of Johnson and Hebert (1999). However, spin image descriptor consists of some weakness where it can be affected easily by mesh resolutions and non-uniform sampling. In fact, it is very challenging to build spin images to recognize objects as there are few parameters need to be taken into consideration. The first spin image parameter is the bin size, where it regulates both storage

volume and averaging of the spin image descriptors. Users have to set the suitable bin size so that the object scale and resolution do not overly depend on the bin size setting. The closer the bin size to mesh resolution, the better the matching of spin images.

The next parameter is the spin image width which represents total rows and columns. The number of rows and columns must set to be equal to produce a square spin image. In other words, the image width controls the size of the square spin image. By properly setting this parameter, amount of a spin image's global information can be regulated. Lastly, the last parameter in generating a spin image is the support angle. It is the largest angle in between the surface normals and the direction of an oriented point basis. This parameter will affect the descriptiveness of spin image.

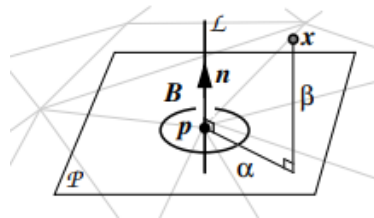


Figure 2.2: Oriented Point Basis (Johnson and Hebert, 1999).

Bielicki and Sitnik (2013) had proposed a method to recognize and localize 3D objects in a cloud of points by using locally calculated feature vector (FV) descriptors. In their training phase, the outcomes from pre-processing (PP) are used to compute local feature vectors (FVs). Then, all the local feature vectors are compiled into histograms to construct a reference object global descriptors. This local FV consists of two histograms: first is the 2-D distribution $C1$ versus $C2$ which is shown in Figure 2.3 and the second is a local surface type distribution. Both of these histograms are computed by the number of intervals. By using the number of intervals, the dimensionality of the feature space FS can be calculated. In order to ensure that only the most significant combination of features will be used in the recognition and localization phase, principal component analysis (PCA) is used to perform a reduction of feature space dimensionality. Figure 2.4 shows the brief steps of building a reference object descriptor.

By using the local FVs to build descriptors, the threshold algorithm can be used even with insufficient and noisy data. Besides, these descriptors allow detecting even highly occluded objects. FVs can also reduce the clutter effect on the recognition rate. However, the main weakness of this method is that the descriptiveness of the proposed descriptors might result in high false-positive ratio. Geometric representation of the reference object can be further improved to reduce this error.

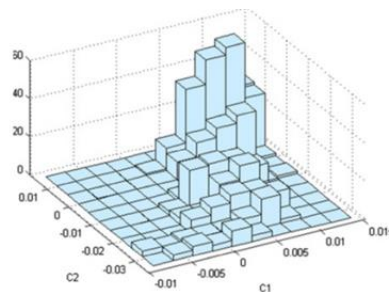


Figure 2.3: Histogram of 2D Distribution of Parameters C1 versus C2 (Bielicki and Sitnik, 2013).

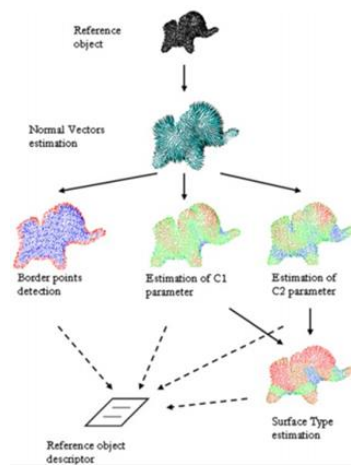


Figure 2.4: Building of Reference Object Descriptor (Bielicki and Sitnik, 2013).

In the paper of Frome, et al. (2004), they had proposed two new regional shape descriptors. The first is known as 3D contexts and the second is harmonic contexts to recognize three dimensional objects (in this case, vehicles) in noisy and cluttered point cloud scenes. For shape contexts, histograms directly function as the descriptors but for harmonic shape contexts, extra transformation needs to be performed. Before starting to build this descriptor, there are two

parameters need to be decided first which are the support region pattern and the methods of distributing all the histogram bins which located in three-dimensional space into vector.

For 3D shape context descriptors, they are basically the extension of 2D shape contexts to a 3D surface. The support region is a sphere centered on the basis keypoint and a surface normal estimated for the keypoint. Then, the support region which also known as the spherical neighbourhood is divided equally by using elevation and azimuth dimensions and logarithmically using radial dimension into histogram bins as shown in Figure 2.5. Step of performing logarithmical sampling is to improve the descriptor to become more robust to distortions in shape. Lastly, by accumulating the weighted count of the number of points inserting into each bin, a 3D shape context descriptor is constructed.

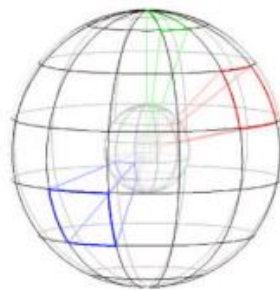


Figure 2.5: Histogram Bins that Form 3D Shape Context (Frome, et al., 2004).

Next, a harmonic shape context descriptor can be built by applying a spherical harmonic transform to the 3D shape context. First, it starts with the same histogram created for 3D shape context. Bin values are used to perform a spherical harmonic transformation for the shells to build a new histogram. In short, the harmonic shape context is basically a histogram vector that results from the amplitudes of the transformation.

Frome, et al. (2004) compared the recognition ability of both shape context descriptors and spin image descriptor which was used in 3D shape-based object recognition system by Johnson and Hebert (1999). Although the descriptors utilized by Johnson and Hebert (1999) is to define surface meshes, but its implementation to point clouds is quite fast and direct. The result in testing the descriptors in vehicle recognition showed that both shape context

methods obtained better recognition rates than spin image in noisy scenes. In cluttered scenes, 3D shape context descriptor achieved the best performance among other methods. These results are presented in Figure 2.6 and Figure 2.7.

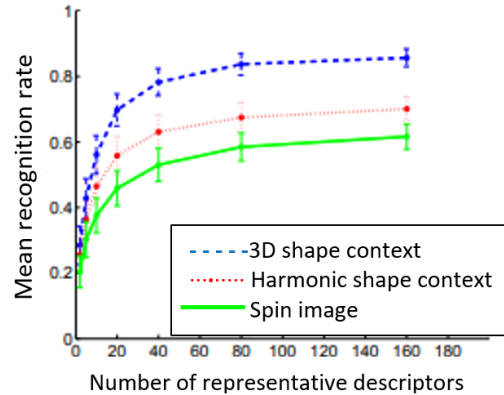


Figure 2.6: Comparison of Recognition Rate between Different Descriptors in Noise Experiments (Frome, et al., 2004).

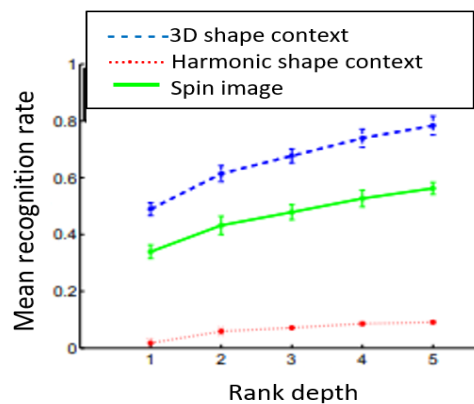


Figure 2.7: Comparison of Recognition Rate between Different Descriptors in Clutter Experiments (Frome, et al., 2004).

In addition, a new 3D feature descriptor known as THRIFT was presented by Flint, Dick and Hengel (2014) in their local 3D structure recognition method. The idea of THRIFT is the extension of successful Scale Invariant Feature Transform (SIFT) and also Speeded up Robust Feature (SURF) algorithms used in keypoint extraction, identification and matching in range data. By using the orientation information extracted in keypoint detection, THRIFT can create a descriptor by counting up the all points into a single dimensional histogram in accordance with their angles between two surface normals n_{small} and n_{big} . Lastly, all the surface normals' angles are fitting into

the bins of the histogram to form a descriptor. The reason of using surface normal information is that it can improve the descriptor by handling the changes in sampling density better than certain types of descriptors which only utilize the information of the location extracted from keypoint detection such as shape contexts and spin images. Figure 2.8 shows the graphical interpretation of the descriptor.

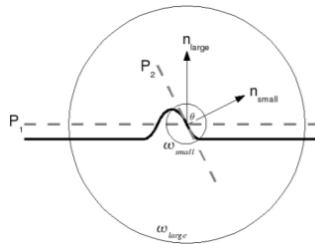


Figure 2.8: A Keypoint with Two Least Squares Planes and their Related Normals for One Support Point on a Local Surface. (Flint, Dick and Hengel, 2014).

Taati, et al. (2007) proposed a method for 3D object recognition and pose determination between a range data by using local shape descriptors with variable dimension (VD-LSDs). Similarly, in their paper, they presented three phases to perform object recognition: point matching, pose reconstruction and pose finalization. Building of local shape descriptors falls in the first phase, where they will be used later to determine the point correspondences between the input range data and full model. In this paper, model point cloud and scene point cloud are given. The main interest is to determine the rigid transformation that is able to align the instance found with the model in the scene by if that particular instance lies there.

First, on the covariance matrix of each keypoint in the local neighbourhood, they executed Principal Component Analysis (PCA) to generate a Local Reference Frame (LRF) and three eigenvalues scalars which represent vector lengths along each LRF to each point. Next, with all scalars and vectors, they generated several properties for each point: position properties, direction properties and dispersion properties. Selection of all these property sets can affect the effectiveness and robustness of point matching. Then, they chose a small part from these properties by implementing a feature selection method. In

point clouds, there is no need to construct LSDs for every point. Therefore, only salient points which have a special geometry are selected in order to build more descriptive LSDs. The last step is to create a scalar-quantized or vector-quantized histogram. After extracting and accumulating all the chosen properties into bins of a histogram, the descriptor is finally created. Based on the experiment conducted, variable dimensional local shape descriptor had a better recognition rate compared to the spin image on a few data sets.

According to Mian, Bennamoun and Owens (2006), they created a tensor descriptor using histogram-based method in their object recognition and segmentation in cluttered scenes paper. The method aims to successfully detect 3D objects and also predict their location and orientation in a complex scene. At first, an input point cloud scene is converted into decimated triangular meshes, as shown in Figure 2.9a and Figure 2.9b. Normals are computed for each vertex and triangular face. Next, two vertices (in pairs) which fulfil particular distance and angle conditions are chosen to define its 3D coordinate basis as shown in Figure 2.9c. The 3D coordinate basis is then used to construct a local 3D grid at the origin. The grid size indicates the amount of locality while the grid's bin size controls the granularity level. After determining the tensor grid, surface areas of the meshes which intersected each bin of the grid are calculated and summed to construct a 3D tensor descriptor.

Tensors can handle noise and also changing mesh resolutions very well. Based on the experiment performed in this paper, Mian, Bennamoun and Owens (2006) proved that tensor outperformed the spin image in recognizing objects in cluttered scenes. One of the potential drawbacks is its combinatorial explosion of vertex pairs.

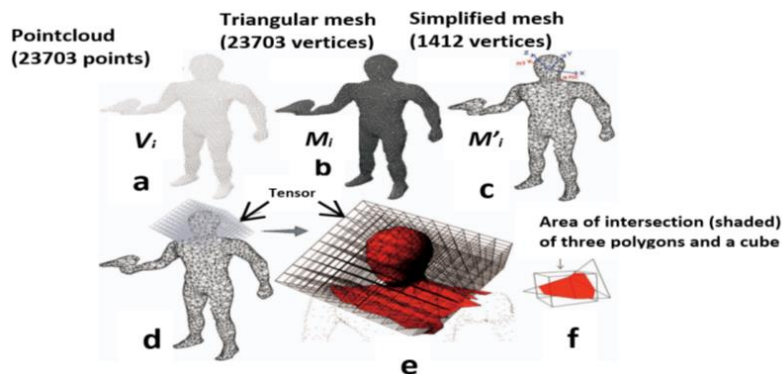


Figure 2.9: Tensor Computation (Mian, Bennamoun and Owens, 2006).

2.3.2 Signature Based Methods

The main idea of this method is that it basically encodes one or more geometric measures calculated individually at each neighbouring point to describe the local neighbourhood of a keypoint. Signature descriptors contribute more descriptive power.

Besides creating descriptors using histogram-based method, Mian, Bennamoun and Owens (2010) also built a feature descriptor by using the depth values of the local surface. Before building the descriptor, they presented a method to measure and rank the quality of the keypoints. After that, the most outstanding keypoints are chosen for detecting local features. Next, they derived a local 3D coordinate system for the local feature and fitted a lattice to all local surfaces.

Sun and Abidi (2001) had developed point's fingerprint descriptor to perform surface matching efficiently. To generate the point's fingerprint, geodesic circles was constructed for each interest point by utilizing the surrounding points which have the similar geodesic distance to the interest point. Then, a local coordinate system is constructed with the normal and tangent plane at the keypoint, as shown in Figure 2.10a. 2D contours which also known as point's fingerprint descriptor can be obtained by projecting all geodesic circles onto a tangential plane of the surface. Figure 2.10b shows the local fingerprint of the same point from different views. Point's fingerprint can perform better as it contains more descriptive information than methods that only utilize one contour or 2D histogram. Besides, the cost of computation is low compared to descriptors using 2D image representation.

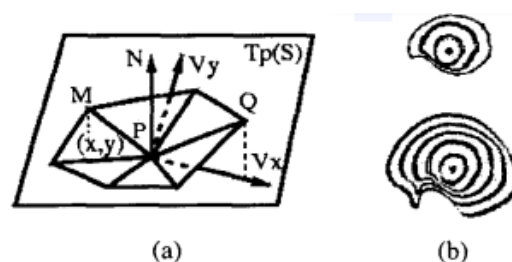


Figure 2.10: (a) Local Coordinate System. (b) Local Fingerprint of the Same Point from Different Views (Sun and Abidi, 2001).

Chua and Jarvis (1997) had created descriptors known as point signatures to perform the task of object recognition. In the paper, they mentioned that by using point signatures, recognition of an object in both single-object scene and complex scene containing few partially overlapping objects can be done. A point signature is known as a 1D signature that expresses the surface surrounding a keypoint. First, a plane is fit to all contour points and then translated to the keypoint. After the contour points are being projected onto the fitted plane to create a curve, each contour point is defined by two parameters which are the signed distance and the clockwise rotation angle θ . Figure 2.11 shows a point signature. The reference direction of the signature may not be unique as several signatures could be collected from the same point. This method consists of a drawback which is sensitive to mesh resolutions.

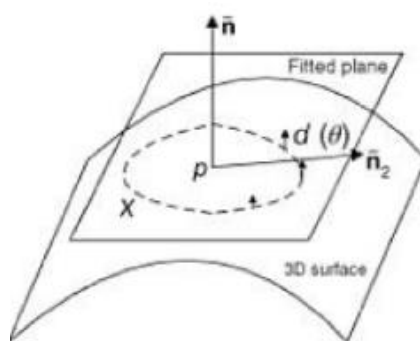


Figure 2.11: Point Signature (Chua and Jarvis, 1997).

2.3.3 Summary and Comparison between Histogram-Based and Signature-Based Local Surface Feature Description Methods

Table 2.2 summarizes all methods used to construct local descriptors by different authors. Histogram based methods are often used to construct a descriptor as they can handle noise and clutter very well. However, since the descriptor is built by encoding geometric measures computed at every neighbouring point, it has a more descriptor power compared to histogram descriptor. Both of these methods have their own advantages and disadvantages. A better solution can be further proposed which combines both ideas of histogram and signature to create a descriptor which can share both advantages and eliminate drawbacks.

Table 2.2: Summary of Methods for Local Surface Feature Description.

No	Method Category	Method Name	Data Type	Outcomes	Reference
1	Histogram	Spin Image	Mesh	It can handle noise and occlusion but it is easily affected by the changing mesh resolutions and non-uniform sampling.	Johnson and Hebert (1999)
2	Histogram	Feature Vector	Point Cloud	This method can detect high occluded objects even in insufficient and noisy scenes, however, it might result in high false-positive ratio.	Bielicki and Sitnik (2013)
3	Histogram	3D Shape Contexts and Harmonic Shape Contexts	Point Cloud	By comparing both shape contexts with spin image, both shape contexts performed better than spin image in noisy scenes while 3D shape contexts outperformed	Frome, et al. (2004)

				the rest in cluttered scenes.	
4	Histogram	THRIFT	Point Cloud	This descriptor is more robust to changes in sampling density.	Flint, Dick and Hengel (2014)
5	Histogram	Variable Dimensional Local Shape Descriptors (VD-LSDs)	Point Cloud	The experiment showed that this descriptor achieved better recognition rate compared to the spin image on a few data sets.	Taati, et al. (2007)
6	Histogram	3D Tensor	Mesh	Tensor can handle very well with the changing mesh resolutions and also noise. The experiment conducted had proved that it performed than the spin image.	Mian, Bennamoun and Owens (2006)
7	Signature	Point's Fingerprint	Mesh	This descriptor contains more feature descriptive information where it outperformed	Sun and Abidi (2001)

				both spin image and point signature.	
8	Signature	Point Signatures	Mesh	Although this method is able to recognize objects in simple and complex scenes, but, it is sensitive to mesh resolutions.	Chua and Jarvis (1997)
9	Signature	Depth Values	Point Cloud	The result of the experiment showed that this method has a better performance than the spin image.	Mian, Bennamoun and Owens (2010)

2.4 Surface Matching (Coarse Recognition and Localization)

There are two parts in this surface matching step which are feature matching and hypothesis generation. First, after the descriptors which contain local descriptive information are made from the detected unique keypoints, a set of feature correspondences between the model and the scene needs to be established by matching their descriptors. Once the correspondences are obtained, an algorithm is needed to identify the feature correspondence groups and use them to vote for candidate models that need to be recognized and determine transformation hypotheses.

2.4.1 Feature Matching

Feature matching aims to extract point relations which also known as feature correspondences between the desired model and the complex scene. The most common way to accomplish this is to carry out a brute-force search, where a comparison between the model features and scene feature is performed to find the correspondences.

Johnson and Hebert (1999) had proposed a surface matching engine or slicing based method to show how two surfaces are matched where spin image descriptors from points on two surfaces (model and scene) are compared to a best-match to establish point relation. When two spin images (model and scene) are found to have a high correlation, a feature correspondence between them is determined. This step is repeated until all the point correspondences are gathered together.

Besides, hash table from geometric hashing is one of the most popular ways to detect and store feature correspondences. According to Mian, Bennamoun and Owens (2006), they had created a correspondence algorithm known as hash table-based voting scheme which can automatically extract feature correspondences. First, they built a 4D hash table by using the tensor descriptors. Besides, since the tensor descriptors already served as the view of local surface areas, they enable the hash table and matching process to be less dependent on the resolution and surface sampling. Next, after filling all the tensors into the 4D hash table, the matching of the tensors is performed by utilizing a voting scheme to automatically establish the feature correspondences. Hashing method is efficient and of low polynomial complexity.

Frome, et al. (2004) also utilized the similar locality-sensitive hashing (LSH) technique to perform feature matching. Since there are a lot of 3D shape context descriptors need to be matched to establish feature correspondences, it might take a long time to complete the step. This LSH functions based on the principle that two points are identified to have the same hash if they are near to each other in feature space. This results in minimization of the search space by orders of magnitude which can speed up the matching process. A hash function is defined to hash the descriptors that located in the same hypercube to the identical hash score. Hash function that is used in this method targets to maximize collisions for similar points which desires to make identical items to

have a large probability of having the same hash value. LSH is a better choice compared to traditional hashing as it realizes efficiencies in memory and number of computations conducted.

Papazov and Burschka (2010) used the idea of the hash table to establish pairs of correspondences between the oriented model points and oriented scene points. Before this, the hash table is utilized by them to store the descriptors of pairs of oriented model points. Besides, descriptors of oriented scene point pair are also computed. By matching the scene point pair descriptor to the hash table that stores model point pair descriptors, a corresponding oriented model point pairs can be established. In contrast to Matei, et al. (2006), they constructed a hash table for indexing feature of the model to form a collection of geometry descriptor of single model points. By comparing these two methods, pairs of correspondences are better than single point correspondences as it can reduce the time used in the recognition phase and allow for a simple computation of the subsequent aligning rigid transformation.

According to Rodolà, et al. (2013), they used *kd*-tree (*k*-dimensional Tree) method to match two surfaces (model and scene) to establish means of point-wise correspondences. First, they determined an original group of so-called strategies where every scene point is linked with the *k*-nearest model points in the descriptor area. In the descriptor space, each scene sample has the possibility to match the model samples that show identical surface characteristics. To prevent overcrowding of matching, the total amount of “attempts” is limited to value of *k*. When the nearest model descriptor is located at a great distance from the data, clutter pre-filtering is performed to exclude the matching correspond scene point. If the model descriptor satisfies the condition of *k*, *kd*-tree is used to perform fast searching to match the scene to the model. This matching direction can help to minimize the false positive rate for the same number of strategies. Similarly, Guo, et al. (2013) used a pre-constructed *kd*-tree to match the scene features to every model feature to obtain feature correspondences. A feature correspondence from the scene and its nearest model is established if the ratio between the shortest and the second shortest distance is less than a threshold (eigenvalue). Although *kd*-trees are effective and efficient in low dimensions. However, it might not be so efficient when

encountering high dimensional data as it might require a longer time to backtrack through the tree to search for the ideal solution.

The method utilized by Zhong (2009) is called Locality Sensitive Trees (LST). This tree is defined as a randomly-distributed binary tree where the nodes or leaves represent the feature space's subdivision to be matched. Moreover, the inner part of each leaf nodes consists of a unique random test which is useful in grouping new feature vectors. In this paper, after the intrinsic shape signature descriptors which represent the 3D point clouds are constructed, the descriptors from two 3D point clouds are matched to establish the feature correspondences. Figure 2.12 shows the sequence of matching two points using intrinsic shape signature. The branch of the tree indicates various model descriptor that can be matched to the scene. Then, the established feature correspondences are associated with each of the tree's leaves.

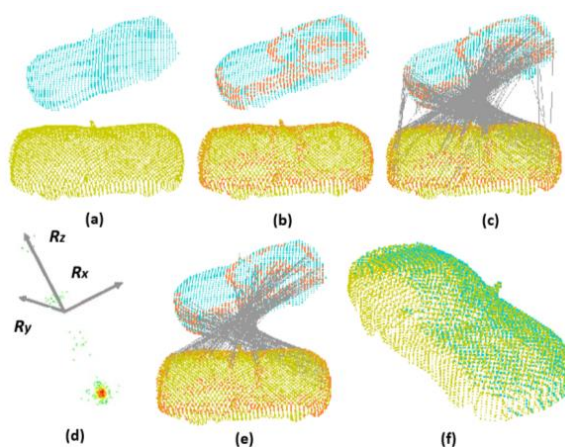


Figure 2.12: Sequence of Matching Two Points using Intrinsic Shape Signature (Zhong, 2009).

2.4.2 Summary of Feature Matching

Table 2.3 summarizes all methods used to match features by different authors. This step is very important as it is performed to establish point relations between the model and the scene. These feature correspondences will be used in the next step to vote for the candidate model. Based on research, it can be observed that the methods that often be used are hashing technique and tree-based method. Between these two methods, hashing technique might perform better than tree-based method because it has the ability to minimize the search space, making the computation time and complexity lower. For tree-based method, although it

can also perform fast searching which is efficient in low dimension data, but the efficiency decreases when it comes to high dimension data.

Table 2.3: Summary of Methods of Feature Matching.

No.	Feature Matching	Outcomes	Reference
1	Surface matching engine (Slicing based)	No further explanation.	Johnson and Hebert (1999)
2	Hash Table	In this paper, since the authors are using tensor descriptors that indicates local surface area of the views, they made the hash table and matching step less independent of the surface sampling and the resolution. From the experiment, they demonstrated that hashing matching time is not affected by the number of models in the data library, unlike the spin images.	Mian, Bennamoun and Owens (2006)
3	Locality-Sensitive Hashing (LSH)	This method realizes efficiencies in memory and number of computations conducted.	Frome, et al. (2004)
4	Hash Table	Hash table is utilized to store oriented model point pair descriptors. This enables them to detect doublets between model and scene faster and it	Papazov and Burschka (2010)

		makes the aligning rigid transform easier.	
5	Hash Table	The hash table stores geometry descriptors of single model points where it makes the computational time longer.	Matei, et al. (2006)
6	<i>kd-tree</i>	<i>kd-tree</i> is able to perform fast searching to match the scene to the model which can reduce the time required in the feature matching process.	Rodolà, et al. (2013)
7	<i>kd-tree</i>	This method is effective and efficient in low dimension data.	Guo, et al. (2013)
8	Locality Sensitive Trees (LST)	Based on one of the experiment, with LST indexing, only 0.4% of the feature matches are performed, other 99.6% of the pairwise feature comparisons are pruned away. Still, LST is able to recognize object accurately similar to how an exhaustive matching method does.	Zhong (2009)

2.4.3 Hypothesis Generation

For hypothesis generation, the tasks are to determine candidate models which are most likely to locate in the scene and to perform transformation hypotheses for them. By using the feature correspondences established from feature matching, candidate models which are most likely to be located in the complex scene are obtained. Then, each candidate models are utilized to further perform rigid transformations that align one surface with another. There are multiples

methods that have been utilized such as interpretation trees, game theory, geometric consistency, Hough transform, geometric hashing and Random Sample Consensus (RANSAC).

First of all, one of the methods is known as constrained interpretation trees where each branch in the tree represents a feature correspondence. It starts from the root of the tree where there is no feature correspondence. Then, it successively constructs the feature correspondence between model and scene to the leaf node. When the branches become too many, some can be said to be trimmed to ensure the tree follows the correspondence arranging conditions. The result of this technique is a tree that can perform consistent interpretations for the transformation hypotheses for each model. This technique is utilized by Bariya and Nishino (2010) to perform 3D object recognition where the nodes in the tree indicate correspondences established between a model and scene feature with each branch contains a hypothesis of whether the candidate model is present or absent in the scene. They searched for candidate model in the scene one at a time by using a constrained interpretation tree that extracts the rich descriptive information produced by the scale-dependent corners, as shown in Figure 2.13. Since each leaf node indicates a set of correspondences with its parent nodes, a rigid transformation is computed for each node in order to align pairs the scene and model corner points that establish that correspondences.

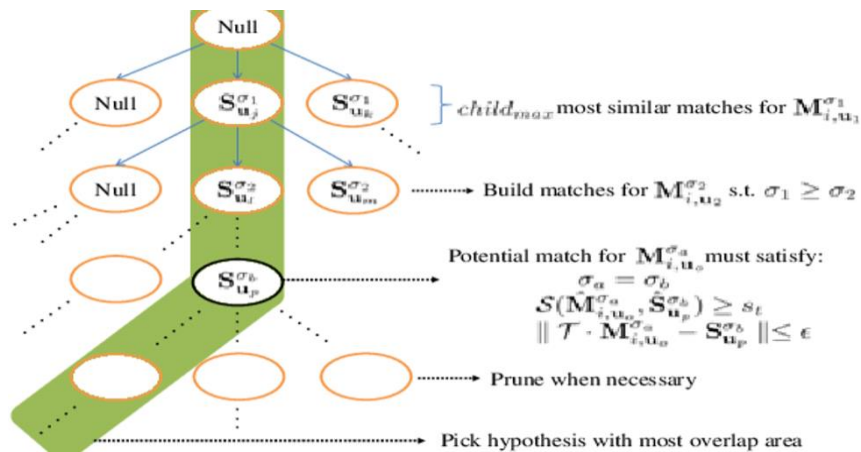


Figure 2.13: Schematic of Scale-Hierarchical Interpretation Tree (Bariya and Nishino, 2010).

According to Johnson and Hebert (1999), when they were matching features to establish and group correspondences, geometric consistency is

performed to eliminate outliers which will exhibit a major error when undergoing rigid transformation. The remaining correspondences that are geometrically consistent are utilized to compute a transformation hypothesis.

A similar method is also implemented by Chen and Bhanu (2007). Previously, they used hash table to store the descriptors which contain model descriptive information. Then, they compared the descriptors (local surface patches) between model and objects. Votes are casted and added to the hash table to know which model receives the highest votes. High quality corresponding descriptors can be identified as well. Since the hash table may consist of a lot of local surface patches, those with the maximum similarity and similar surface type are selected as the potential corresponding patch. Next, geometric consistency is carried out to group the consistent potential ones and filter the outliers. The largest group is likely to be the actual corresponding pairs. After voting, candidate models which gained the top three highest votes are obtained from the hash table entries. The following step is to apply rigid transformation which includes rotation matrix and translation vector to the candidate models. This geometric consistency method is useful because it can minimize the error of correspondence matching and further improve the efficiency of hypothesized transformations.

The other technique used to perform hypothesis generation is known as game theory. According to Rodolà, et al. (2013), they utilized game theoretic considerations to select the best surviving feature correspondences that satisfy a global rigidity constraint. They first defined a payoff function to measure the quality of a hypothesis that is backing up by another hypothesis with respect to the ultimate aim. The game contest starts by selecting sparse sets of liable correspondences to survive. The candidate subset then undergoes isometric transformation. This method is easy to implement and very efficient.

Hough transform or Hough voting is used to vote the feature correspondences to generate candidate models in 3D Hough space. By referring to Tombari and Stefano (2012), all votes will be inserted into an array whose dimensionality is the same as the number of unrecognized parameters of the shape class. Each Hough space point corresponds to the presence of a transformation between the scene and the model. In the end, the presence of the candidate model is obtained through the peaks in the Hough accumulator. Two

variants are developed to the standard Hough voting scheme which are Hough N-N (N represents Neighbours) and Hough N-C (C represents Central). A similar idea of Hough transform is utilized by Ashbrook, et al. (1998). They used a Hough voting scheme to perform the transformation of the local correspondences that aligns complete surfaces. The benefits of implementing this Hough voting is that it is able to model local correspondences transformation errors by using a probabilistic Hough transform.

Another useful approach of hypothesis generation is known as geometric hashing and it is described by Lamdan and Wolfson (1998). Normally, a hash table is built to store the model points' coordinates with their own reference basis. In the recognition phase, an ordered pair of scene points is chosen and all other scene points in this basis are expressed in this coordinate system. Then, the basis is used to vote for triplets which are the model, basis and angle for which these coordinates presented and is indexed into the hash table. The two-point basis of the triplet and the highest support is then utilized to compute the model hypothesis.

Random Sample Consensus (RANSAC) is another useful hypothesis generation technique which it enhances the geometric hashing technique by cancelling the voting part and further confirms the candidate models' position consistency using a minimal set of feature correspondences. The feature correspondence set will be utilized to generate a rigid transformation which aligns the model with the scene. All qualified point pairs that exhibit a high consistency with the transformation will be counted until the total amount reaches a pre-set threshold. According to Schnabel, Wahl and Klein (2007), they presented an efficient RANSAC algorithm to detect basic shapes like cylinders, cones, spheres, planes and tori in unorganized point clouds data even under adverse conditions where there are a lot of outliers and a high degree of noise.

The input of this method is point cloud with points inside and associated normal while the output is a group of primitive shapes with their corresponding disjoint points set and a set of remaining points. First, in their localized sampling strategy, they implemented an octree to form spatial proximity between samples where it can adapt the size of minimal sets to the density of outlier and shape size. A good cell which likely contains mostly points for the primitive shape needs to be chosen properly from any level of the octree. A cell will keep

generating new shape candidates and then collecting them into a candidate set. The implementation of RANSAC to detect basic shapes might not be useful in complex object localization. In object localization, it is possible to use this method to identify a simple chair which only consists of one flat surface and four cylinders. However, it could not efficiently detect an object with arbitrary shapes as these shapes are not included in their method. Figure 2.14 presents the algorithm of RANSAC to extract shapes in the point cloud in this method.

```

 $\Psi \leftarrow \emptyset$  {extracted shapes}
 $\mathcal{C} \leftarrow \emptyset$  {shape candidates}
repeat
   $\mathcal{C} \leftarrow \mathcal{C} \cup \text{newCandidates}()$  {see sec. 4.1 and 4.3}
   $m \leftarrow \text{bestCandidate}(\mathcal{C})$  {see sec. 4.4}
  if  $P(|m|, |\mathcal{C}|) > p_t$  then
     $\mathcal{P} \leftarrow \mathcal{P} \setminus \mathcal{P}_m$  {remove points}
     $\Psi \leftarrow \Psi \cup m$ 
     $\mathcal{C} \leftarrow \mathcal{C} \setminus \mathcal{C}_m$  {remove invalid candidates}
  end if
until  $P(\tau, |\mathcal{C}|) > p_t$ 
return  $\Psi$ 

```

Figure 2.14: Algorithm of RANSAC to extract Shapes in the Point Cloud (Schnabekl, Wahl and Klein, 2007).

Papazov and Burschka (2010) also utilized RANSAC algorithm to sample a minimal point set from the scene. In this method, there are only two oriented points in a minimal set which are not sampled uniformly. Then, in order to generate oriented scene point pair from the two oriented points, normals of both points are computed using Principal Component Analysis. The descriptor for each scene point pair is created and used to retrieve all model pairs in model hash table which are identical to scene point pairs. A model corresponding to model pairs is restored and a rigid transformation that best aligns model pairs to scene pairs is performed. The location defined in the rigid transformation is considered to be the transformation hypothesis. Taati, et al. (2007) mentioned that the feature correspondences which established in the previous point matching step contain a large percentage of outliers. Therefore, they implemented a robust RANSAC algorithm to eliminate the outliers and align the model with its instance in the scene.

According to Mian, Bennamoun and Owens (2010), they used the technique of pose clustering to generate transformation hypotheses. They

calculated a transformation that aligns the model and the scene based on each k -feature correspondences. All transformations are grouped and the largest cluster indicates the actual transformation hypotheses. In addition, Zhong (2009) also performed pose transforms which includes translation and rotation between the matching descriptors then clustered them in a six-dimensional pose space close to the actual pose transform. Guo, et al. (2013) computed a rigid transformation by aligning the local reference frame of model feature to the local reference frame of scene feature. A single feature correspondence using their RoPS feature descriptor can be used to estimate the rigid transformation. It outperformed other algorithms which used point signatures and spin image descriptors where they need at least three correspondences to compute a transformation.

2.4.4 Summary of Hypothesis Generation

Table 2.4 summarizes all methods used to generate model hypotheses by different authors. For hypothesis generation, the technique must have a high ability to find the candidate models which have a high possibility locating in the rich scene and perform a rigid transformation for candidate models. Based on the research, there are many methods that have been proposed such as interpretation trees, game theory, geometric consistency, Hough transform, geometric hashing and Random Sample Consensus (RANSAC). Every method has its own benefits and drawbacks. Hough Transform might perform better than geometric consistency and pose space clustering as it has a voting scheme that can improve the accuracy. However, this voting scheme might result in a longer computational time. RANSAC can improve the performance by eliminating the voting part and it is easy to implement.

Table 2.4: Summary of Methods of Hypotheses Generation.

No.	Hypotheses Generation	Outcomes	Reference
1	Constrained Interpretation Trees	From the experimental results, hypotheses are generated effectively depending on the scale	Bariya and Nishino (2010)

		of the corresponding features, which able to achieve a recognition rate of 97.5% with up to 84% occlusion.	
2	Geometric Consistency	It eliminates correspondences that are not geometrically consistent to prevent transformation error.	Johnson and Hebert (1999)
3	Geometric Consistency	This method exhibits a much lower computational complexity and better performance.	Chen and Bhanu (2007)
4	Game Theory	The gameplay that performs the actual recognition step is able to produce reliable matches. Besides, this method is easy to implement and very efficient.	Rodolà, et al. (2013)
5	Hough Transform	Two variants that have been added to the Hough voting scheme are more robust to quantization effects. Based on the experiment, it showed that this method outperformed algorithms that reply on geometric consistency and pose space clustering.	Tombari and Stefano (2012)
6	Hough Transform	The benefit of using Hough voting is that it can model local correspondences transformation errors by utilizing a probabilistic Hough transform.	Ashbrook, et al. (1998)
7	Geometric Hashing	This method can operate in the presence of only partial information and it does not require domain-specific knowledge but	Lamdan and Wolfson (1998)

		only the location of the feature correspondences.	
8	Random Sample Consensus (RANSAC)	It improves the geometric hashing technique by eliminating the voting part.	Schnabekl, Wahl and Klein (2007)
9	RANSAC	This method is robust against outliers and it is easy to implement. However, it requires prior knowledge about data in order to able to calculate the number of iterations required to complete the whole sampling process.	Papazov and Burschka (2010)
10	RANSAC	One of the drawbacks of this method is that the number of iterations is strongly dependent on inlier percentage.	Taati, et al. (2007)
11	Pose Clustering	This method requires only a little memory and makes accurate clustering algorithms usage less costly.	Mian, Bennamoun and Owens (2010)
12	Pose Clustering	No further explanation concluded in this paper.	Zhong (2009)
13	Pose Clustering	The experiment showed that it performed better than other algorithms which implement point signatures and spin image descriptors where at least three correspondences are needed to compute a transformation.	Guo, et al. (2014)

2.5 Fine Localization (Verification)

The final step of 3D object recognition and localization is known as fine localization or verification where it improves the accuracy of the transformation hypotheses by distinguishing true hypotheses from the false hypotheses. There are normally two methods to complete this step which are the individual and global verification approaches.

2.5.1 Individual Verification Methods

After obtaining multiple transformation hypothesis from the previous step, each of them is used to align a candidate model with the scene. Next, an important step of measuring the accuracy of the alignment is performed to find the acceptable hypotheses.

Iterative Closest Point (ICP) is the most frequently implemented algorithm to measure the accuracy of the alignment nowadays. This method normally determines the best transformation hypothesis that minimized the total distance between the closest points in the model and the scene. Once the best hypothesis is obtained, the scene features that correspond to that model can be identified. According to Chen and Bhanu (2007), after they get the transformed data set from the model rigid transformation, they searched the closest point in the test image for every point in this data set.

Guo, et al. (2014) refined the transformation using ICP algorithm which results in a residual error. They then used the residual error and visible proportion together with their thresholds to accept the correct transformation hypothesis and to find the correct candidate model. One of the challenging parts in using this method is to determine the thresholds as they cannot be too strict or else it will eliminate the correct ones which are highly occluded in the scene and they cannot be too loose as well or else many false positives will be produced.

2.5.2 Global Verification Methods

The difference between this method and the individual verification method is that it examines the whole set of hypotheses instead of checking the candidate model one by one.

According to Aldoma, et al. (2012a), they presented a cost function to perform a global optimization to eliminate wrong active hypotheses. However, a global cost function consists of a high computational burden, so they used Simulated Annealing to minimize the cost function to retrieve accurate hypotheses within a limited amount of time and computational resources. The benefit of this technique is that it can recognize occluded models without a high number of false positives.

By referring to Papazov and Burschka (2010), their object recognition algorithm utilized the mean of acceptance function to save the useful hypotheses in solution list. There are two parameters in the acceptance function which are the support term and penalty term. In contrast to normal RANSAC, there is only support term (score function) which measures the quality of each hypothesis (number of transformed model points fall within ε -band of the scene). In this method, an extra penalty term exists to penalize hypothesis of occluding transformed model parts in the scene. Lastly, a conflict graph is created to filter the weak hypothesis in the solution list. By implementing non-maximum suppression or also known as local maximum search over the conflict graph, the final hypothesis is chosen.

According to Schnabekl, Wahl and Klein (2007), their algorithm also consists of a score function (lazy cost function evaluation scheme) which is used to measure the quality or to be said counting the number of compatible points of the shape candidates. The best candidate model is chosen if it has the highest score (highest compatible points). With this score function, it can help to significantly reduce the overall computational cost. Lastly, a least-squares approach serves as a refitting tool to optimise the geometric error of the chosen candidate shape.

2.5.3 Summary and Comparison between Local and Global Verification Methods

Table 2.5 summarizes all methods used to verify the hypotheses generated by different authors. For local verification method, since it examines each hypothesis one by one to align a candidate model, it might increase the overall computational cost. For global verification method, it examines the whole set of

hypotheses at once and some of the algorithms even have the extra acceptance or score function which allows it to find the correct hypothesis more accurately.

Table 2.5: Summary of Methods of Verification.

No.	Verification Method Type	Verification Method Name	Outcomes	Reference
1	Individual	Iterative Closest Point (ICP)	No further information.	Chen and Bhanu (2007)
2	Individual	Iterative Closest Point (ICP)	It is difficult to determine optimal thresholds to achieve a high recognition rate.	Guo, et al. (2014)
3	Global	Cost function	Based on the experiment, this method can recognize occluded models without a high number of false positives.	Aldoma, et al. (2012a)
4	Global	Acceptance function	In contrast to the normal algorithm which consists only score function, the extra penalty term can help to penalize hypothesis of occluding transformed model parts in the scene to further improve the accuracy.	Papazov and Burschka (2010)
5	Global	Score function (Cost function)	Overall computational cost is reduced by	Schnabekl, Wahl and

			introducing the score function in the algorithm.	Klein (2007)
--	--	--	--	--------------

2.6 Summary

In summary, the four steps to perform 3D object recognition and localization are 3D keypoint detection and extraction, construction of local surface feature descriptors, surface matching and fine localization. In this literature review section, various techniques and methods proposed by different authors for each step were summarized and analysed. Different recognition algorithms presented can be specifically designed to recognize and localize object in different range image types such as depth image, point cloud or polygonal mesh.

Based on this literature review, keypoint detection is always the first step for a capable and accurate 3D exploration of the environment. Keypoints are the salient points in the environment which contain high discriminative information. Therefore, it is essential to implement a fast, efficient and robust technique for an automatic extraction of keypoints in input data. Two methods to perform this keypoint detection are the fixed-scale and the adaptive-scale methods. After the keypoints have been detected and extracted, the important descriptive information of the keypoints are used to construct feature descriptors. There are mainly two categories of descriptors for interest feature points which are the histogram-based and the signature-based methods. For surface matching, feature matching and hypothesis generation are performed to establish a set of feature correspondences between the interested model and the complex scene needs with descriptors and use them to vote for candidate models that need to be recognized and determine transformation hypotheses. Last but not least, after generating multiple candidate model hypotheses, step of measuring the accuracy of the alignment is performed to find the final qualified hypotheses.

CHAPTER 3

METHODOLOGY AND WORK PLAN

3.1 Introduction

In this chapter, the methods used in the project of object localization in 3D point cloud are explained. It includes the flow of the steps and theories of each method used in this project. In order to develop an efficient algorithm for this project, many existing methods were studied to understand the principles behind those methods and to identify and analyse their potential drawbacks so that the project design could be further improved in certain advanced algorithm development. Parameters of the algorithms in each step were the most important factor that they could affect the performance of the process. While developing algorithms for each step of the object localization process, parameters were frequently adjusted as they play a vital role in producing desirable and the targeted results.

3.2 Point Cloud Library (PCL)

The full algorithm of this project was implemented using an open source library called Point Cloud Library (PCL). According to Aldoma, et al. (2012b), PCL is a powerful library that can process both 2D or 3D datasets and it contains a lot of readily available tools together with their source codes. In this project, the point cloud file format used was Point Cloud Data (PCD) which could be processed by PCL. Microsoft Visual Studio 2013 was used to develop the project's source codes as PCL mainly adopts C++ programming language.

3.3 Project System Overview

Object localization in 3D point cloud project consists of two main phases. The first phase is known as object recognition and the second phase is object localization. First, pre-processing of the point clouds was performed. There were two main types of input point clouds needed to be fed into the main algorithm, which were the reference point cloud that displayed the scene of multiple objects and the target point cloud that represented the target object. In order to obtain target point clouds, significant objects located in the scene point cloud were segmented from the scene.

In the following object recognition and localization phases, both target and reference point clouds were processed to extract keypoints/points of interest which were then used to construct descriptors. Descriptors constructed for each keypoint contained the local geometry which represented a description of its local neighbourhood. The next step was to perform feature matching. The descriptors of the target's keypoints and reference's keypoints were compared to establish point-to-point correspondences. Then, by using these feature correspondences, transformation model hypotheses were generated. Finally, a hypotheses verification method was developed to recognize and localize the final correct hypothesis in the reference point cloud, if there were more than one hypotheses found. The final hypothesis recognized in the scene was then localized by using bounding box and the information of the length, height and width of the object in the bounding box were found. The overview and flowchart of this project are shown in Figure 3.1 and Figure 3.2 respectively.

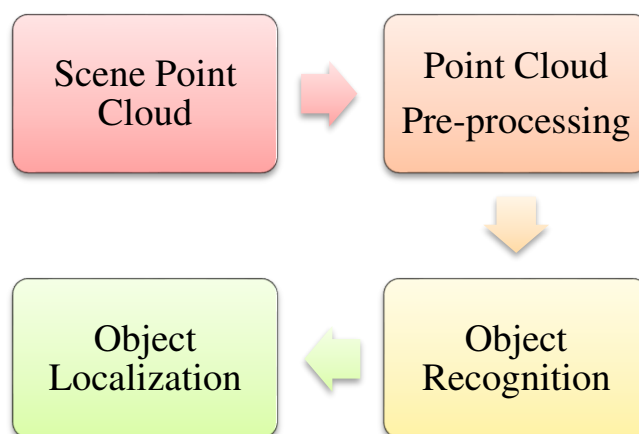


Figure 3.1: Overview of Object Localization in 3D Point Cloud.

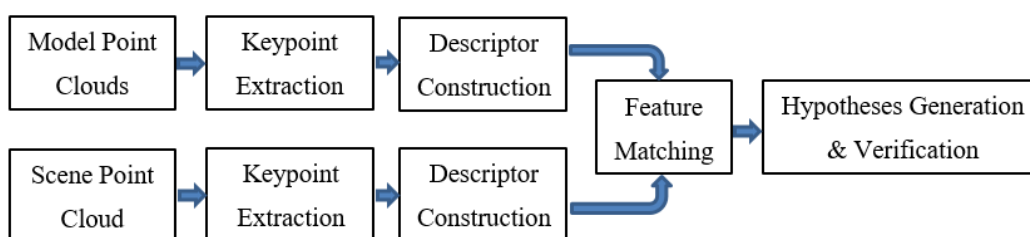


Figure 3.2: Flowchart of Object Localization in 3D Point Cloud.

3.3.1 Surface Normal Estimation

In this project, the point type in the main point cloud dataset used was PointXYZ type, where it only consisted of 3D point's x, y and z information. However, Point Cloud Library (PCL) (2018a) stated that the SIFT keypoint detector used in the keypoint detection step which will be explained in next section required a value, such as colour (RGB) or intensity from the point cloud in computing a scale space to extract keypoint. Since the point cloud dataset used did not have these information, surface normal of the points were computed and they acted as the input information to SIFT keypoint detection. A normal is defined as a vector perpendicular to a targeted point on the estimated plane. The surface normal were estimated and computed for each point in the point as described below.

Based on Point Cloud Library (PCL) (2012), the surface normal estimation was performed based on Principal Component Analysis (PCA) with a user-defined k -neighbourhood size by using `pcl::Feature::setKSearch` function. Surface normals were estimated and computed from a set of neighbours (k -neighbourhood) located at surrounding of a point. After selecting the size of the k -neighbourhood of a targeted point p_i , a covariance matrix C was generated for the point, as shown in Equation 3.1. PCA was then applied to obtain the eigenvalues, λ_j which were in a descending order and the eigenvectors, v_j of the covariance matrix built by the k -nearest neighbours. The final surface normal of a targeted point was defined as the eigenvector with the smallest eigenvalue. Equation 3.2 shows how the surface variation, σ was obtained from the relationship between each eigenvalue. Besides, since the input model and scene point cloud data were very large, an empty three dimensional kD tree (k-Dimensional tree) was created in the algorithm to increase the speed to search for k neighbours around a targeted point.

$$C = \frac{1}{k} \sum_{i=1}^k (p_i - \bar{p}) \cdot (p_i - \bar{p})^T, C \cdot \vec{v}_j = \lambda_j \cdot \vec{v}_j, j \in \{0,1,2\} \quad (3.1)$$

where

k = Number of nearest neighbours contained in neighbourhood of p_i

\bar{p} = Centroid of neighbours in neighbourhood

λ_j = j^{th} eigenvalue of computed covariance matrix

v_j = j^{th} corresponding eigenvector

$$\sigma = \frac{\lambda_0}{\lambda_0 + \lambda_1 + \lambda_2} \quad (3.2)$$

3.3.2 Keypoint Detection

The next step is known as keypoint detection. Since the input of the algorithm was a large quantity of points, both useful and poor points were mixed together. Informative keypoints were more desirable as they contained a rich 3D information and could be well-localized, for example, edge or corner points. Besides, keypoint detection helped to reduce potential sample points to minimize the computational time and complexity. Detection of the keypoint locations was important as it could further affect the success of the feature descriptors in the next step.

The method of keypoint detection used in this project was Scale Invariant Feature Transform (SIFT) which was proposed by Lowe (2004) for object recognition and the algorithm was implemented in PCL using `pcl::SIFTKeypoint` (Point Cloud Library (PCL), 2018a). The idea of SIFT was originally proposed for 2D images and it was further adapted and developed by the PCL teams to 3D point clouds. In PCL, SIFT keypoint detector searches through the Difference-of-Gaussian (DoG) clouds in the Gaussian scale space and the 3D keypoints are detected as the local extrema that locate in the scale space. The Gaussian scale space and its extrema were constructed in PCL by setting four parameters given in the algorithm: number of octaves, number of scale levels per octave, minimum scale size and minimum contrast (minimum DoG value required to mark a keypoint). The number of octaves refers to the octaves number to compute the keypoints. The number of scale levels per octave indicates the number of scales set to compute the keypoints within each octave. The minimum scale parameter is the standard deviation of the smallest scale contained in the Gaussian scale space. The minimum contrast parameter provides a threshold to limit the number of keypoints detected without adequate contrast.

After feeding both model and scene point clouds, $I(x,y,z)$ to the algorithm, the system created a Gaussian scale space using 3D density that was convolved with a series of Gaussian kernels to build a pyramid of density maps whose standard deviations, σ_j were differed by a fixed multiplicative factor, k for each point (Lowe, 2004). Besides, a blur filter was constructed by carrying out a radius search and the weighted average of neighbouring points was taken as a new local cloud intensity for each point. DoG clouds that contained different intensity values were obtained by subtracting two adjacent point clouds repeatedly, as shown in Equation 3.3. Eight nearest neighbours located at the current scale and nine nearest neighbours located at the neighbouring scale (above and below) were used to compare with every point in the DoG clouds (Hansch, Weber and Hellwich, 2014). Originally, SIFT detector requires a value from point cloud such as colour or intensity to compute keypoints. However, the point cloud used in this project does not contain the needed information. Therefore, the value of normal of each point computed in the previous step was used to compute keypoints rather than intensity variants. Besides, a kD tree was also built to speed up the neighbour searching step.

$$D(x, y, z, \sigma_j) = G(x, y, z, \sigma_{j+1}) - G(x, y, z, \sigma_j) \quad (3.3)$$

where

D = Difference-of-Gaussian (DoG) clouds

G = Convolution of Gaussian yield point cloud

3.3.3 Descriptor Construction

Now, all the important keypoints had been detected. Local descriptors were then constructed to describe the local geometry for each keypoint. In this project, there were two methods used to develop the descriptors: Point Feature Histogram (PFH) and Signature of Histograms of Orientations (SHOT).

Point Feature Histogram (PFH) method was proposed by Rusu, et al. (2008). The algorithm was developed by using module `pcl::PFHEstimation` available in Point Cloud Library (PCL) (2020a). This method mainly required two input information which were the xyz data and surface normals computed in the previous section. Therefore, the surface normals computed must have a

high quality or else it would affect the performance of the descriptor. Basically, a PFH descriptor mainly uses the difference of surface normals' directions to collect the geometrical information of a targeted keypoint from its neighbourhood.

After feeding the input keypoint clouds of both model and scene and their computed surface normals to the algorithm, a k -neighbourhood of point p_i was created by setting the radius of a sphere r that used to search for neighbours. Next, point pairing was performed. According to Grupo De Robotica (2015a), a targeted point was not only paired with its k neighbours, but the neighbours were also paired among themselves. For each point pair p_s and p_t , a coordinate frame containing 3 unit vectors was computed by using their normals n_s and n_t at either point, as shown in Equation 3.4. By using the computed coordinate frame, the difference between the normals of the point pair was expressed as three angular features, as shown in Equation 3.5. Lastly, four main features, the three angular features and the point pair's Euclidean distance d were binned into a histogram with 125 bins. This means that each keypoint would have its own 125-bin histogram. The final PFH descriptor was the combination of all histograms with their own four features. An empty kD tree was built to perform the neighbour searching process.

$$u = n_s, v = u \times \frac{(p_t - p_s)}{\|p_t - p_s\|_2} \times u, w = u \times v \quad (3.4)$$

where

$$d = \|p_t - p_s\|_2$$

$$\alpha = v \cdot n_t, \phi = u \cdot \frac{(p_t - p_s)}{d}, \theta = \arctan(w \cdot n_t, u \cdot n_t) \quad (3.5)$$

As mentioned by Rusu, Blodow and Beetz (2009), PFH has a very high complexity of $O(k^2)$ in computing descriptors at real time. At first, PFH was used to generate descriptors for the keypoints. However, it required too much time, more than two hours to construct the descriptors for a larger scene point cloud. Therefore, in order to increase the efficiency of the descriptor

construction process, Signature of Histograms of Orientations (SHOT) was implemented by using readily available module *pcl::SHOEstimation* in PCL (Point Cloud Library (PCL), 2018b). This method was proposed by Tombari, Salti and Stefano (2010a) which combined both signatures and histograms.

In the algorithm, the descriptor first built a weighted covariance matrix, M for a targeted keypoint p by using a fixed radius for neighbouring keypoints p_i , as shown in Equation 3.6. Then, Local Reference Frame (LRF) or the coordinate system was computed for the keypoint by using eigenvector and eigenvalue decomposition of the matrix. Then, a 3D isotropic grid in a spherical form for the targeted keypoint was created as a signature structure and it was aligned with the corresponding LRF, almost similar like 3D shape context method proposed by Frome, et al. (2004). In PCL, the spherical grid consists of 32 bins creating from 2 radial divisions, 8 azimuth divisions and 2 elevations. Next, one dimensional histogram in each bin was obtained by accumulating the geometric details of the keypoint such as cosine angles between normals of the keypoints and of the neighbouring keypoints. Lastly, the final SHOT descriptor was developed by combining all histograms. In the algorithm, the main parameter to be set and adjusted was the radius defining of which neighbouring keypoints were involved and described. Besides, a kD tree was provided to help in radius search for nearest keypoints for both model and scene clouds.

$$M = \frac{1}{\sum_{i:d_i \leq R} (R - d_i)} \sum_{i:d_i \leq R} (R - d_i)(p_i - p)(p_i - p)^T \quad (3.6)$$

where

M = Weighted Covariance matrix

$d_i = \|p_i - p\|_2$

R = Radius of sphere

3.3.4 Feature Matching

Next, once the descriptors of keypoints for both model and scene were computed, the scene and model keypoints were then successively matched through their own descriptors to compute a set of point-to-point correspondences.

In this project, the reference cloud was the scene descriptors and model descriptor cloud was set as the input cloud for matching. Descriptors of the scene functioned as a key to match themselves to all the descriptors of the model in order to establish model-scene point pairs that were identical. The reason of doing this was to account for the existence of a few model hypotheses. If the model's descriptors matched themselves against the scene's descriptors, the model instances would not be found. Each descriptor in the scene cloud was matched and compared with the model descriptors by using *pcl::KdTreeFLANN* <*pcl::KdTreeFLANN*> module available in Point Cloud Library (PCL) (2013). The system then computed a Euclidean distance between the model and scene descriptor. A distance threshold d was set to determine the similarity between the scene's keypoints and the model's keypoints. Point-to-point correspondences were obtained and added to a correspondence group if the distances were within the threshold, meaning that they were similar. All poor correspondences which had a bigger distance more than the threshold were eliminated. The threshold should not be too big, or else the number of outliers or mismatched keypoints would be higher. On the other hand, if a very low threshold was set, the number of the correspondences would be too less to obtain the model instances in the hypotheses generation step.

3.3.5 Hypotheses Generation

Through the feature matching step, the point-to-point correspondences between the scene and the model were collected in a so-called "correspondences" database. Based on (Grupo De Robotica, 2015b), these correspondences cannot completely recognise and locate the model in the scene as the outliers due to errors from keypoint detection or noise might exist among the correspondences. Therefore, correspondence grouping was performed to retrieve those correspondences which had a high geometric consistency and eliminate other poor correspondences. Lastly, by clustering the final set of correspondences/inliers, the correct model instance was found and localized with the true model in the scene.

The method used to group the correspondences and generate model hypotheses was known as Hough Voting or Hough Transform which was proposed by Tombari and Stefano (2010). It was implemented in PCL by using

pcl::Hough3DGrouping module (Point Cloud Library (PCL), 2020b). First, three main parameters were set as the input details for the algorithm: *rf_rad*, *cg_size* and *cg_thresh*. *rf_rad* was the radius required to compute Local Reference Frame (LRF) for each keypoint. *cg_size* was defined as the bin size in the Hough space that formed the cluster size where *cg_thresh* was the threshold set for voting in Hough space, which also known as clustering threshold.

According to Tombari and Stefano (2010), Hough voting is based on a voting process in Hough space to find the qualified correspondences. Once the algorithm was fed with defined input parameters, LRF was computed for each pair of correspondences in the “correspondences” database C . The vector between each keypoint F_i^M and model’s centroid C^M was calculated as shown in Equation 3.7. The vector computed was in global reference frame (GRF). In order to ensure these vectors were rigid translation and rotation invariant, they were then transformed to local reference frame (LRF) as shown in Equation 3.8. Next, the vector was also computed corresponding to scene keypoint. Since the LRF of the scene keypoint was invariant to transformation, it was assumed that $V_{i,L}^S = V_{i,L}^M$. The final vector was transformed back into GRF of the scene as shown in Equation 3.9. By using these transformations, scene keypoints casted votes in the algorithm-constructed 3D Hough space through the final vector $V_{i,G}^S$.

Inliers/correct correspondences were found by recognizing the peak in the space. All the inliers were clustered in the peak, representing the presence of the model hypotheses. In this step, if there were more than one hypotheses generated, bin size in the Hough space and clustering threshold were adjusted until a correct model instance was recognized. The final set of correspondences was visualized and a bounding box was constructed to locate the model instance in the scene. The length, height and width of the model was printed in the command window with respect to the unit in the PCL.

$$V_{i,G}^M = C^M - F_i^M \quad (3.7)$$

where

$V_{i,G}^M$ = Vector between C^M and F_i^M

$$V_{i,L}^M = R_{GL}^M \cdot V_{i,G}^M \quad (3.8)$$

where

$R_{GL}^M = [L_{i,x}^M L_{i,y}^M L_{i,z}^M]^T$, model rotation matrix

$$V_{i,G}^S = R_{LG}^S \cdot V_{i,L}^S + F_j^S \quad (3.9)$$

where

$R_{LG}^S = [L_{j,x}^S L_{j,y}^S L_{j,z}^S]$, scene rotation matrix

$F_j^S =$ Scene keypoint

3.4 Project Timeline

In order to ensure that the project could be done with the fulfilled aim, objectives and other requirements within the deadline, a work plan was drafted as an outline for goals and processes and it was strictly followed. By referring to Figure 3.3, the work plan was planned in weeks.

During the first week, the software required to develop algorithms for the project and to analyse the results were installed. The basic idea and operation of the software were studied and learned. After finished installing the software, pre-processing of the point cloud was done to create a few model point clouds from the scene cloud. Once the input point clouds were ready, keypoint detection was carried out. The following weeks were used to perform the other processes such as descriptor construction, feature matching and hypotheses generation. Parameters were adjusted in each step in order to produce a desirable result. The algorithm of the project was run over and over again to collect results. After obtaining all the results, the final report was ready to be written.

Task	Task Description	w1	w2	w3	w4	w5	w6	w7	w8	w9	w10	w11	w12	w13	w14
1	- Download software and learn about the basic idea - Find a more reliable point cloud dataset														
2	- Pre-process the point cloud to obtain both model and scene - Perform keypoint detection														
3	- Perform descriptor construction (2 methods) - Perform feature matching														
4	- Perform hypotheses generation - Adjust parameters for each step														
5	- Collect and analyze results from each process														
6	-Write final report														

Figure 3.3: Timeline of Second Part of Project.

3.5 Summary

In summary, the algorithm developed in this project is capable of recognizing and localizing an object from a rich 3D point cloud. First, keypoint detection using SIFT keypoint detector was performed to mark the points of interest on both model and scene point clouds. Gaussian scale space of the detector was the main factor that determined the number of keypoint detected. Next, descriptors that store the geometrical content of the keypoints were computed. There were originally two types of descriptor computed: Point Feature Histogram (PFH) and Signature of Histograms of Orientations (SHOT). In the end, SHOT was selected due to its high efficiency which will be explained in Chapter 4. After the descriptors were computed for both model and scene, feature matching was carried out to establish point-to-point correspondences between the model and scene. Since the correspondences consisted of some outliers, Hough Voting scheme was used to perform a rigid transformation to only collect those good correspondences. All the good correspondences were clustered to generate model hypotheses. The parameters in the Hough Voting were adjusted until there was only one actual model hypothesis found. Next, a bounding box was computed to locate the model instance from the scene, with the corresponding lines connecting between the input model and the model instance.

In this project, the main point cloud used was a fountain scene with three different objects: crocodile, seal and basin. Two types of input model point clouds were used to test the functionality of the algorithms. The first input model point cloud was the non-rotated model point clouds. First, the algorithm tested the localization of only one non-rotated model from the scene per time. Then,

all three non-rotated models were localized from the scene at the same time. Next, the algorithm performed the localization of the second input model point cloud which was the transformed model (rotated by 20°). All results will be shown and discussed in Chapter 4.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 Introduction

There were total of five steps needed to perform to obtain the final object localization results in this project. Each step had their own parameters to be adjusted. The final results will be discussed and analysed in this chapter.

4.2 Dataset

The main multiple-object rich point clouds used in performing object localization in 3D point cloud was obtained from Artec Europe (2020). It is a fountain scene containing a crocodile, a seal and a basin. This point cloud's texture size was fixed to 16384×16384 and it contains total 1505600 points with XYZ information. All three single target models were segmented manually by using CloudCompare where there are 29531, 20859 and 4133 points in crocodile, seal and basin models respectively.

4.3 Normal Estimation

It is important to estimate surface normals for each point in the point cloud as it contained geometric surface's properties that were required to construct keypoint detector and descriptor. The number of k -nearest neighbours of a point was the main parameter and it was regularly adjusted in this algorithm. However, it is difficult to set a correct scale size for the k -neighbourhood. If the number of the nearest neighbours is too large, the estimated surface normal representation of a targeted point would be distorted as there are too many details computed from the surrounding neighbours. If the k -neighbourhood size is too small, the estimated normal of a targeted point might not get enough details from surrounding neighbours. Therefore, the scale of the k -neighbourhood should be adequate where it could sufficiently obtain details from surrounding neighbours to compute surface normals.

In this part, k -neighbourhood scale was set as $k = 2, 4, 6, 8, 10$ and the results of the surface normals computed for crocodile, seal, basin and scene point clouds were analysed and discussed. The visualizations of the point's

surface normals for each point cloud at $k = 2$, $k = 4$ and $k = 10$ were presented in Figure 4.1, Figure 4.2, Figure 4.3, Figure 4.4 and Figure 4.5. Besides, time taken to estimate the surface normals for each point cloud at different neighbourhood size was measured and tabulated in Table 4.1.

As shown in Figure 4.1, the surface normals computed at $k = 2$ for all point cloud could not be seen. There were not enough details to compute surface normals since only two nearest neighbours were selected. The result was not accurate. As the size of k -neighbourhood increased, the distribution of surface normals all over the point clouds could be observed clearly. Starting from scales $k = 4$, the details provided by the nearest neighbours were sufficient as the surface normals were mostly estimated at the outlines of the point clouds. This shows that the neighbourhood scales were able to capture small details from the point clouds. The surface normal results were accurate. Besides, by comparing the normals at $k = 4$ and $k = 10$, the surface normals at $k = 10$ were more parallel and had a more consistent orientation. Based on Rocha (2017), a small neighbourhood could cause noisy normals. The results showed inconsistent normals at $k = 4$. To evaluate the efficiency of the surface normal estimator, time taken for the surface normal computation was measured at different k -neighbourhood scale. From Table 4.1, the surface normal estimator required more computational time at a larger neighbourhood as there were more neighbours involved in the covariance estimation.

The selected scale for k -neighbourhood to compute surface normals was $k = 10$. Although it was more complicated, but the results were the most accurate as the surface normals computed were consistently oriented at the important part of the point cloud.

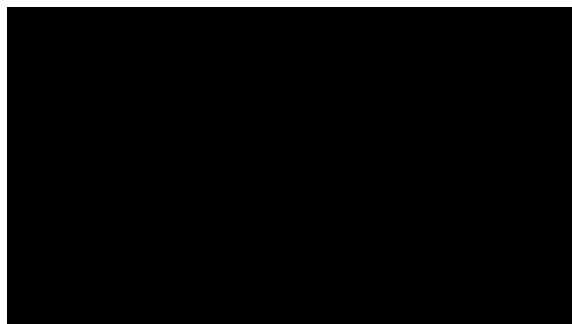


Figure 4.1: Visualization of Surface Normals for All Point Clouds at $k = 2$.

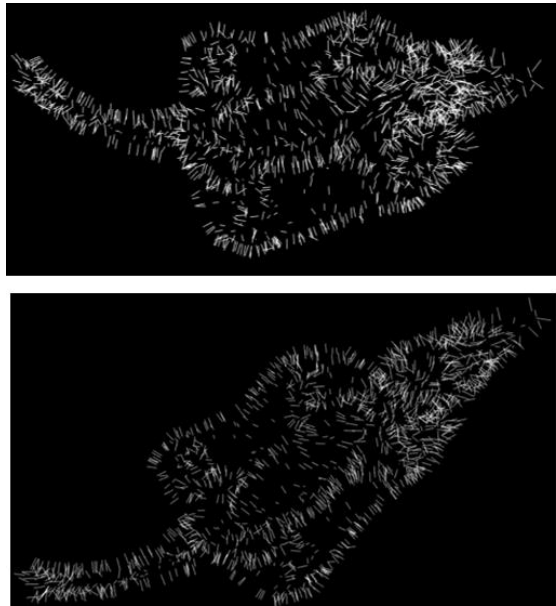


Figure 4.2: Visualization of Surface Normals for Crocodile (Top: $k = 4$; Bottom: $k = 10$).

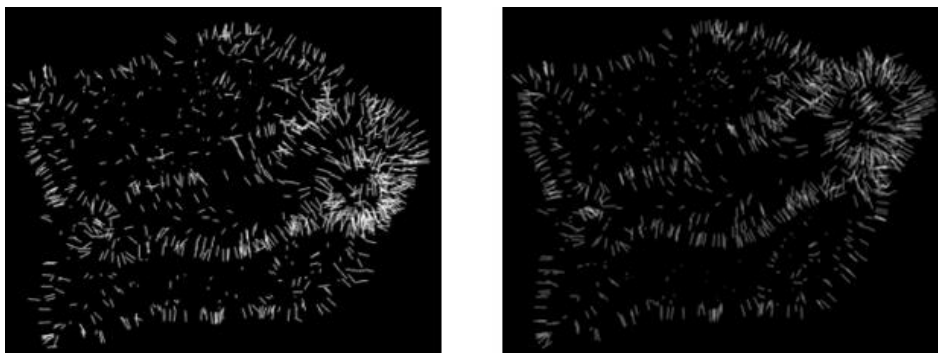


Figure 4.3: Visualization of Surface Normals for Seal (Left: $k = 4$; Right: $k = 10$).

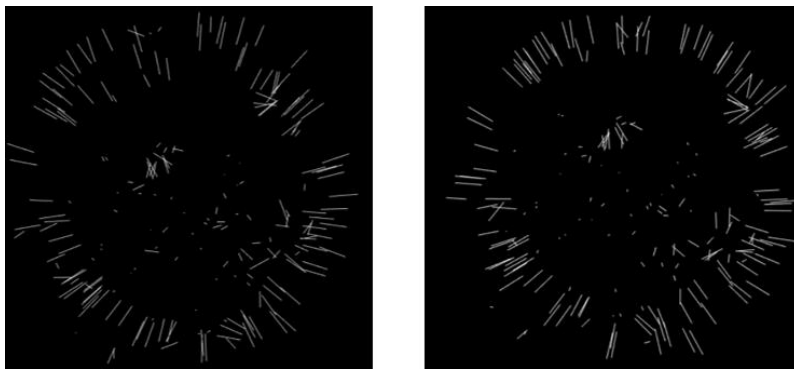


Figure 4.4: Visualization of Surface Normals for Basin (Left: $k = 4$; Right: $k = 10$).

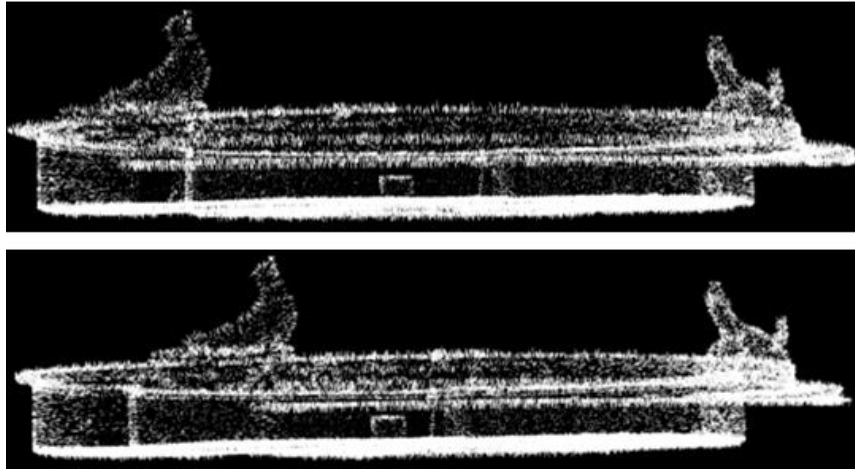


Figure 4.5: Visualization of Surface Normals for Scene (Top: $k = 4$; Bottom: $k = 10$).

Table 4.1: Time Taken for Surface Normal Estimation for Each Point Cloud at Different k .

k -Neighbourhood Size	Computational Time for Surface Normal Estimation (s)			
	Crocodile	Seal	Basin	Scene
2	0.421	0.346	0.187	27.154
4	1.202	0.883	0.292	85.678
6	1.346	0.960	0.311	94.665
8	1.415	1.042	0.340	102.560
10	1.528	1.135	0.370	105.999

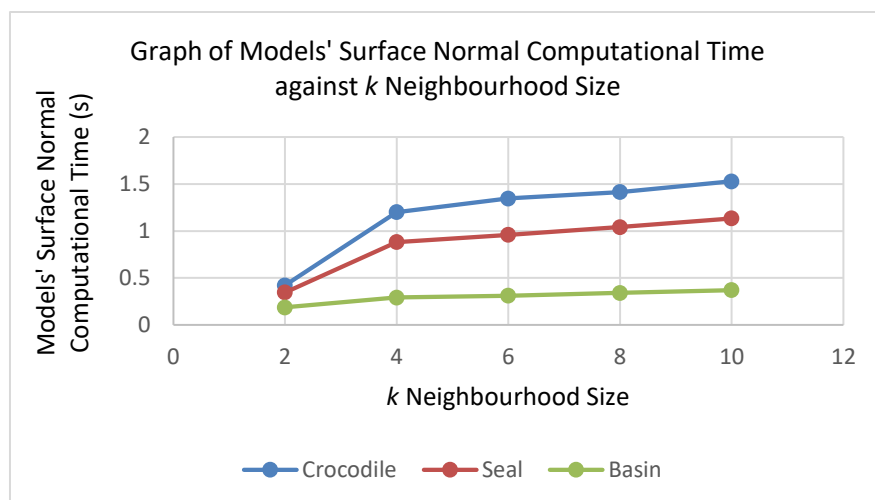


Figure 4.6: Graph of Models' Surface Normal Computational Time against k Neighbourhood Size.

4.4 Keypoint Detection

According to Tuytelaars and Mikolajczyk (2007), it is difficult to define how accurate a detected keypoint is. However, it is possible to analyse the properties of the keypoints. Good detected keypoints should consist of a few properties. The first property is quantity and quality. The number of keypoints found should be reasonable and sufficient. The number should not be too high or too low, but should be able to reflect the details of the model and scene point cloud. The second property is repeatability. In a single point cloud, the repeatability of keypoints detected should be low. For keypoint repeatability between model and scene, the number of duplicates should be high. Noise, changes in viewpoint, occlusion or any combination of the above may affect this property. The last property is efficiency / time performance. The efficiency of keypoint detection is related to the computation time. The shorter the time needed, the more efficient the detector is.

The keypoint detector used in this project was Scale Invariant Feature Transform (SIFT) keypoint detector and it was implemented by using SIFTKeypoint module available in Point Cloud Library (PCL). This detector has several parameters to be adjusted: number of octaves, number of scale levels per octave, minimum scale and minimum contrast. According to Point Cloud Library (PCL) (2018a), the first three parameters specify the range of scales to search and detect the keypoints.

This part tested the behaviour of two sets of the input models: non-rotated input models and rotated input models. The results of SIFT keypoint detector for non-rotated input models will be mainly discussed. In order to analyse the performance of the SIFT keypoint detector on how the Gaussian scale space sampling rates influenced the number of keypoints detected, different Gaussian scale spaces were created where all parameters were set to be the same except for the minimum scale of Gaussian scale space which was set to be 65, 70, 75, 80 and 82. The minimum contrast was set to 0 to make sure that every part of the point cloud could be scanned to detect all possible keypoint. After analysing the behaviour of the detector, the best minimum scale parameter would be selected for the final SIFT detector. The SIFT keypoint detector was set as following:

- $min_scale = 65, 70, 75, 80, 82.$

- $n_octaves = 70$.
- $n_scales_per_octave = 90$.
- $min_contrast = 0$.

4.4.1 Keypoint Quantity and Quality

The number of keypoints detected is one of the most important factors that can affect the performance of the whole object localization process. The number of keypoints marked should be reasonable and adequate, where it cannot be too many or too few. For keypoint quality, the keypoints found must be informative enough to be able to sketch out the model point cloud's approximate pattern. The results of the number of detected keypoints for three different input models (crocodile, seal and basin) and scene point clouds at different minimum scale of Gaussian scale space were tabulated, as shown in Table 4.2.

In Point Cloud Library, SIFT keypoint detector goes through a series of Gaussian filters based on different scales which were then subtracted. The local maxima were selected as keypoints. Based on Table 4.2, it can be observed that the bigger the minimum scale of the Gaussian scale space in the SIFT keypoint detector, less number of keypoints was detected. This means that when the standard deviation of the smallest scale contained in the Gaussian scale space or also known as scale of the keypoints increased, a coarser representation was produced. As the coarser scale of keypoints got higher, some feature went missing, the amount of maxima would not increase. By comparing the numbers of keypoints detected in each point cloud to the numbers of total points in their own point cloud, the number of keypoints found was considered sufficient.

The visualizations of the keypoints detected in each point cloud at the largest minimum scale (80) and the smallest minimum scale (65) were shown in Figure 4.7, Figure 4.8, Figure 4.9 and Figure 4.10. The detected keypoints were clearly located at the side of the models and they were able to outline the shape of the models. This shows that both detectors were capable in finding informative keypoints.

Table 4.2: Number of Detected Keypoints for Crocodile, Seal, Basin Models and Scene Point Cloud at Different Minimum Scale.

Minimum Scale of Gaussian Scale Space	Crocodile Model	Seal Model	Basin Model	Scene
65	915	861	166	37001
70	830	650	161	31773
75	779	638	116	27615
80	644	496	99	24526
82	629	444	87	22904

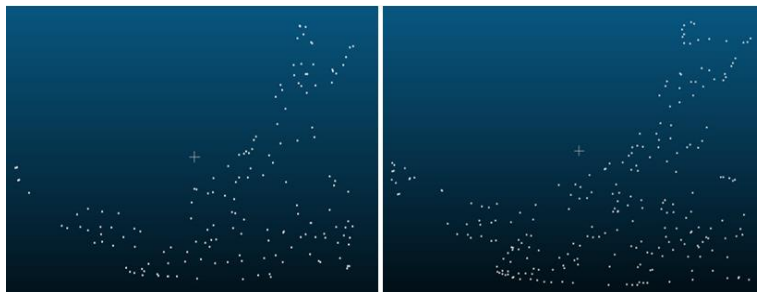


Figure 4.7: Visualization of Crocodile's Detected Keypoints (Left: *Min Scale* 82; Right: *Min Scale* 65).

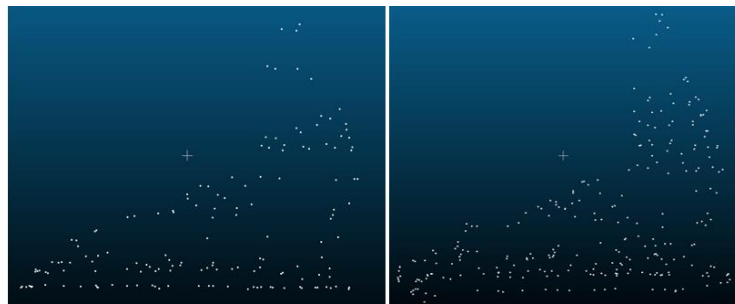


Figure 4.8: Visualization of Seal's Detected Keypoints (Left: *Min Scale* 82; Right: *Min Scale* 65).

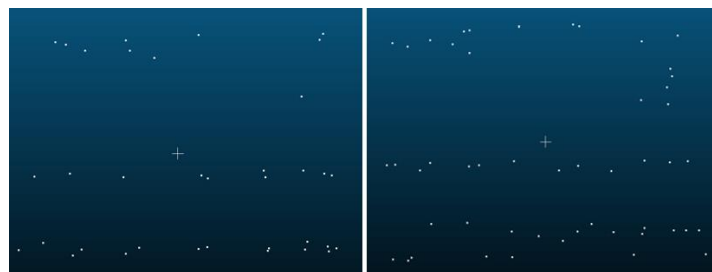


Figure 4.9: Visualization of Basin's Detected Keypoints (Left: *Min Scale* 82; Right: *Min Scale* 65).

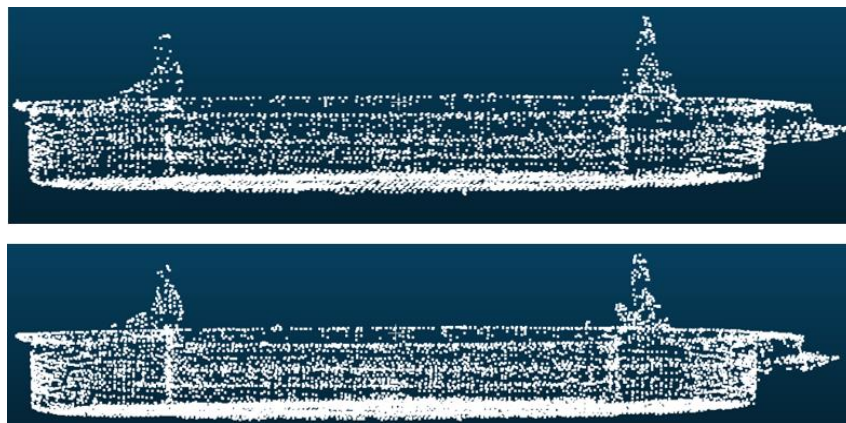


Figure 4.10: Visualization of Scene's Detected Keypoints (Above: *Min Scale* 82; Below: *Min Scale* 65).

4.4.2 Keypoint Repeatability

In evaluating keypoints, one of the most crucial properties of SIFT keypoint detector is its repeatability. In this project, few tests were done to evaluate the repeatability of the points-of-interest: keypoint repeatability results in own model and scene point clouds, keypoint repeatability results between model and scene point clouds and keypoint repeatability results in own model point clouds under different conditions (Minimum scale of Gaussian scale space in SIFT detector & rotation).

First of all, keypoint repeatability resulted in own models and scene point clouds was analysed. The keypoints of the models and scene at different minimum scale which were numerically identical were found and tabulated, as shown in Table 4.3. The number of original keypoint was also tabulated to ease the comparison. Figure 4.11, Figure 4.12, Figure 4.13 and Figure 4.14 show the relationships between the changing minimum scale of the SIFT keypoint detector with the number of total detected and repeated keypoints for both models and scene. It can be observed that all SIFT keypoint detectors created resulted in finding a lot of keypoints that were located at similar position. Besides, with more detected keypoints, there was a higher chance where repeated keypoints appeared more frequently which resulted in low efficiency. To further analyse the remaining unique keypoints in each point cloud, the unique keypoints at the minimum scale of 65 and 80 for the crocodile point clouds were visualized as presented in Figure 4.15. As the results show, the location of the unique keypoints was the same as the location of the total

keypoints. The brief pattern of the point cloud was still clear even after removing all the repeated keypoints. This means that the detected keypoints were actually descriptive enough.

Table 4.3: Table of Number of Original and Repeated Keypoints for Input Crocodile, Seal, Basin Models and Scene Point Clouds at Different Minimum Scale of Gaussian Scale Space.

Minimum Scale of Gaussian Scale Space		65	70	75	80	82
Crocodile Model	No. of Original Keypoints	915	830	779	644	629
	No. of Repeated Keypoints	674	603	560	466	470
Seal Model	No. of Original Keypoints	861	650	638	496	444
	No. of Repeated Keypoints	583	398	413	328	281
Basin Model	No. of Original Keypoints	166	161	116	99	87
	No. of Repeated Keypoints	115	112	87	67	52
Scene	No. of Original Keypoints	37001	31773	27615	24526	22904
	No. of Repeated Keypoints	24153	20565	18090	16036	14894

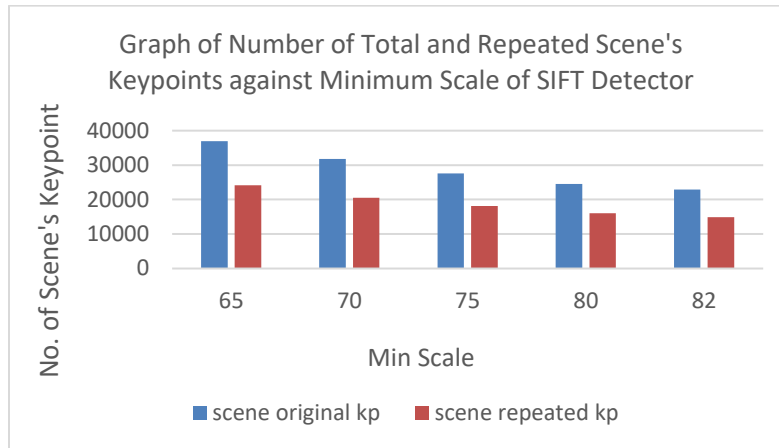


Figure 4.11: Graph of Number of Total and Repeated Scene's Keypoints against Minimum Scale of SIFT Keypoint Detector.

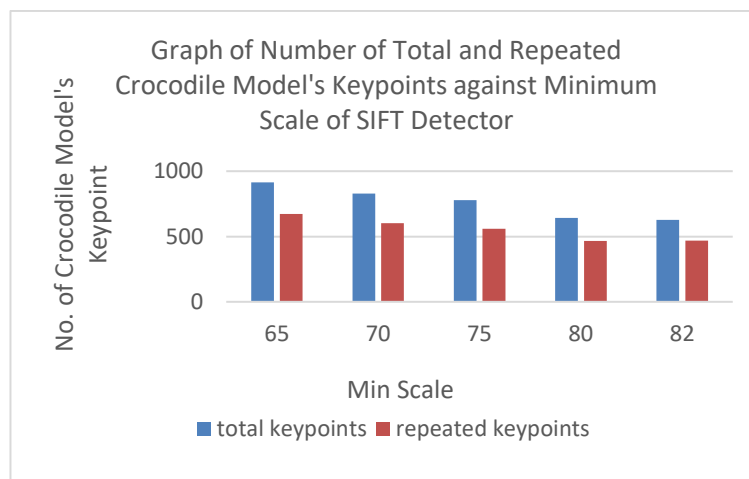


Figure 4.12: Graph of Number of Total and Repeated Crocodile's Keypoints against Minimum Scale of SIFT Keypoint Detector.

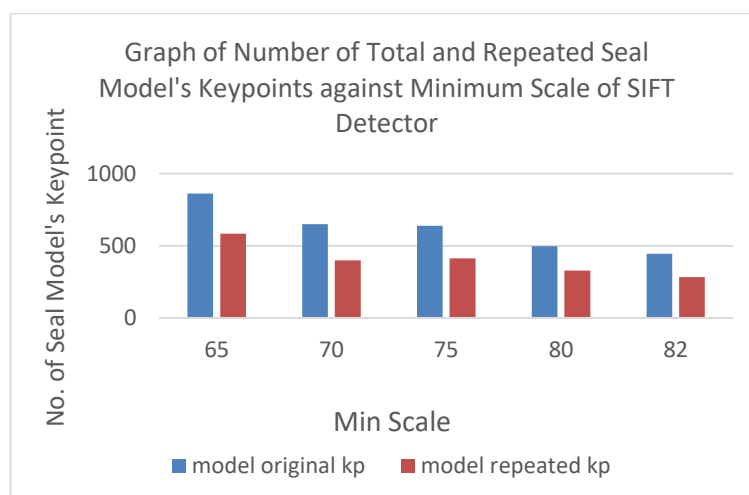


Figure 4.13: Graph of Number of Total and Repeated Seal's Keypoints against Minimum Scale of SIFT Keypoint Detector.

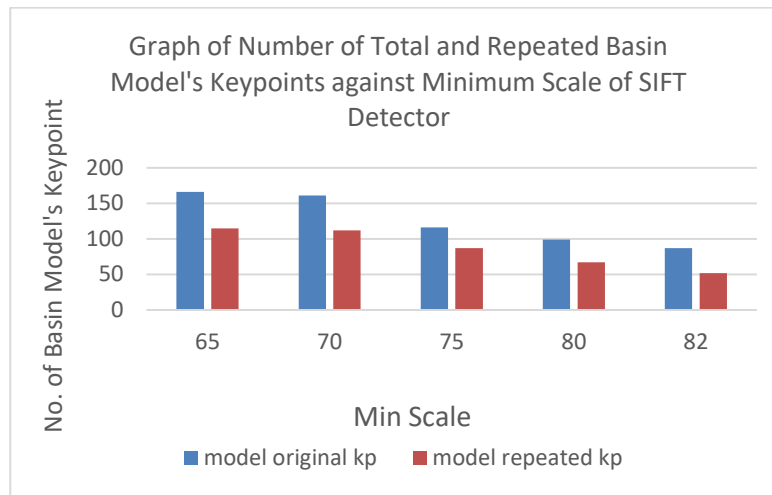


Figure 4.14: Graph of Number of Total and Repeated Basin's Keypoints against Minimum Scale of SIFT Keypoint Detector.

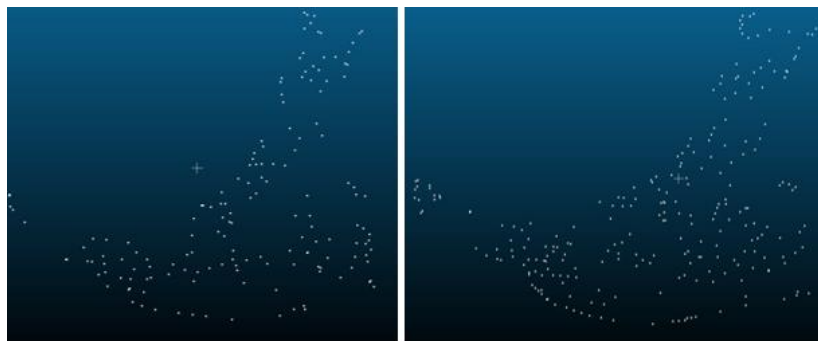


Figure 4.15: Visualization of Unique Crocodile's Keypoints (Left: Min Scale 82; Right: Min Scale 65).

Next, keypoint repeatability resulted between models and scene point clouds was found and evaluated. The keypoints analysed here were the unique/non-repeated keypoints remaining after removing all repeated keypoints from the original detected interest points. The keypoints which were numerically identical between models and scene were found, tabulated in Table 4.4 and plotted in Figure 4.16.

High repeatability of keypoints between the input models and scene is crucial as it could affect the efficiency performance of object matching and localization in further steps. The results show that the number of keypoints in models that actually located at the exact same place in the scene was not many. Keypoint repeatability did not improve for all input models as the minimum scale reduced where there was more keypoints. To have a better localization

performance, the minimum scale which resulted in a higher number of repeated keypoint was desirable and it can be seen that each model had their own preferable minimum scale parameter.

Table 4.4: Number of Repeated Keypoints between Input Models and Scene at Different Minimum Scale.

Min Scale	Crocodile	Seal	Basin
65	13	25	6
70	13	20	5
75	17	19	0
80	8	8	0
82	9	6	1

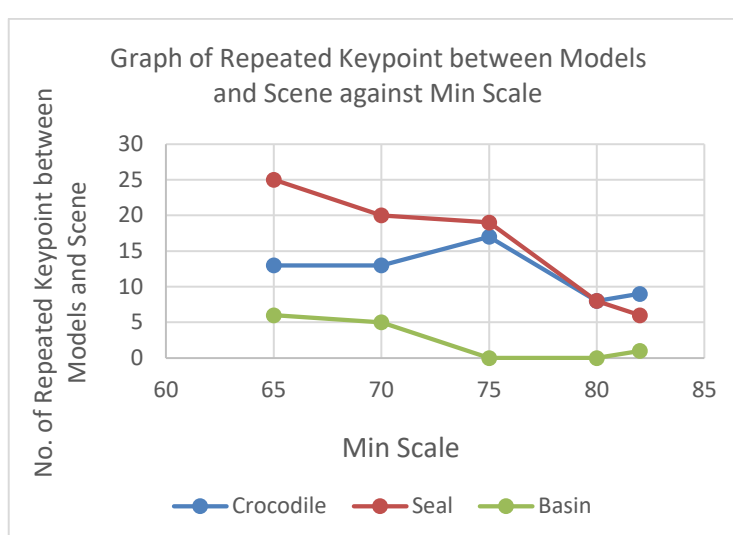


Figure 4.16: Graph of Number of Repeated Keypoints Between Three Input Models and Scene at Different Min Scale.

Besides, matching of keypoints between models and scene was also conducted by using CloudCompare to evaluate the repeatability of the keypoints. Cloud-to-Cloud distance (C2C absolute distance) with octree level 8 (octree's subdivision level where the cloud distance calculation will be executed) was computed after matching each model's keypoints to the scene's keypoints at different minimum scale of detector (CloudCompare, 2015a). The mean distance and standard deviation of keypoints in each model matching to the scene's keypoints were recorded as shown in Table 4.5. Figure 4.17, Figure 4.18,

Figure 4.19, Figure 4.20, Figure 4.21 and Figure 4.22 show the visualization of C2C absolute distance display range of three input models at minimum scale of 70.

In CloudCompare, the points' distances between two clouds are calculated by setting the scene point cloud as reference cloud and model point clouds as the compared one. According to CloudCompare (2015b), the distances between points are computed by implementing the Nearest Neighbour Distance method. For every point in the model point cloud (compared cloud), CloudCompare will examine the nearest point in the scene point cloud (reference cloud). Then, the Euclidean distances between the points are found. After computing all C2C distances, the mean distance and standard deviation values were calculated.

Table 4.5 shows that the minimum parameter of 70 for crocodile and basin models and of 65 for seal model resulted in the lowest mean distance and standard deviation values. Based on the colour scale of the value of the C2C distance computation, the colour range changes from blue to red when the distance becomes bigger. If the point in the model cloud is at the exact location in the scene cloud, the C2C distance computed will be zero and the colour shown will be blue. The results show that the blue point had a bigger distribution and the histograms of normal distribution were clearly shifted towards left/zero for all input models. Since there was not many exact same keypoints detected between models and scene, a threshold C2C distance of approximately 15 or within the first four classes was set. All points that were within this threshold were considered as repeated keypoints, as shown in Table 4.6.

Table 4.5: Mean Distance and Standard Deviation Between Model and Scene Point Cloud After Matching at Different Minimum Scale.

Min Scale	Crocodile		Seal		Basin	
	Mean Distance	Standard Deviation	Mean Distance	Standard Deviation	Mean Distance	Standard Deviation
65	10.13918	20.78699	16.87797	24.92541	15.80206	28.06951
70	9.17489	18.30128	16.90226	25.83518	9.96979	18.08997
75	14.53731	25.18010	18.41618	27.33734	18.40606	28.61948

80	14.45363	24.43072	21.71080	28.56758	19.44442	28.54013
82	17.31757	27.98655	21.33786	26.64655	15.14331	23.72197

Table 4.6: Percentage of Keypoints Within Threshold Between Model and Scene Point Cloud After Matching at Different Minimum Scale.

Min Scale	Crocodile (%)	Seal (%)	Basin (%)
65	79.781	67.401	74.405
70	80.723	66.006	74.233
75	70.277	62.188	70.339
80	71.318	57.515	69.307
82	67.987	56.152	68.539

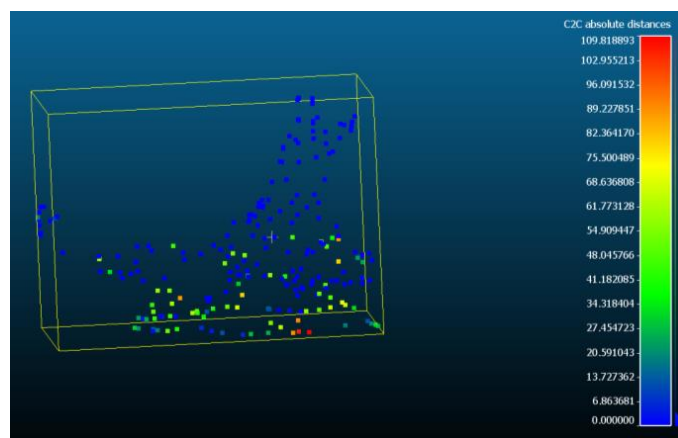


Figure 4.17: C2C Absolute Distance Display Range of Crocodile Model at Minimum Scale of 70.

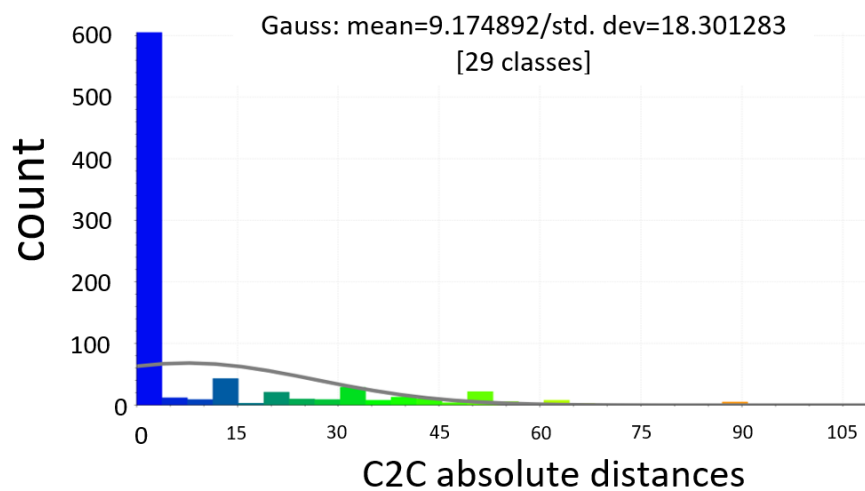


Figure 4.18: Normal distribution of Crocodile Model at Minimum Scale of 70.

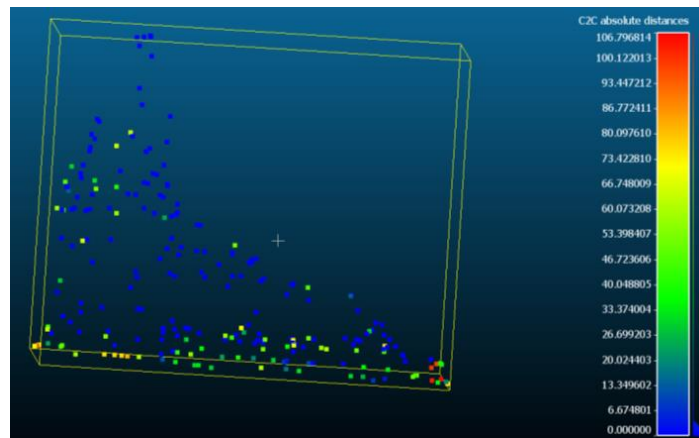


Figure 4.19: C2C Absolute Distance Display Range of Seal Model at Minimum Scale of 70.

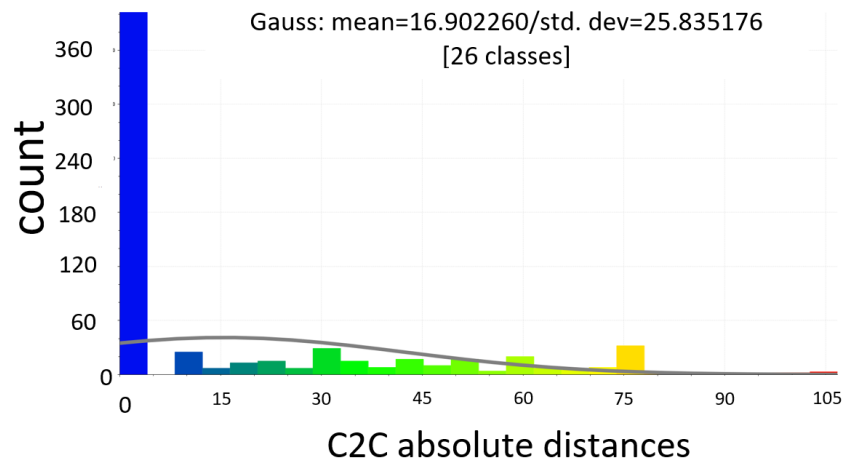


Figure 4.20: Normal distribution of Seal Model at Minimum Scale of 70.

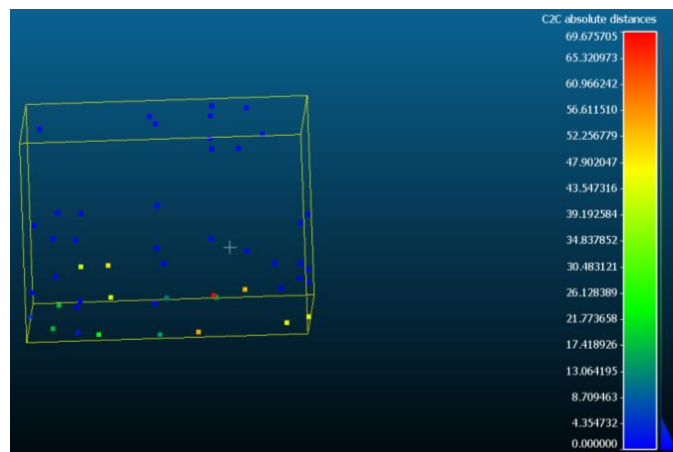


Figure 4.21: C2C Absolute Distance Display Range of Basin Model at Minimum Scale of 70.

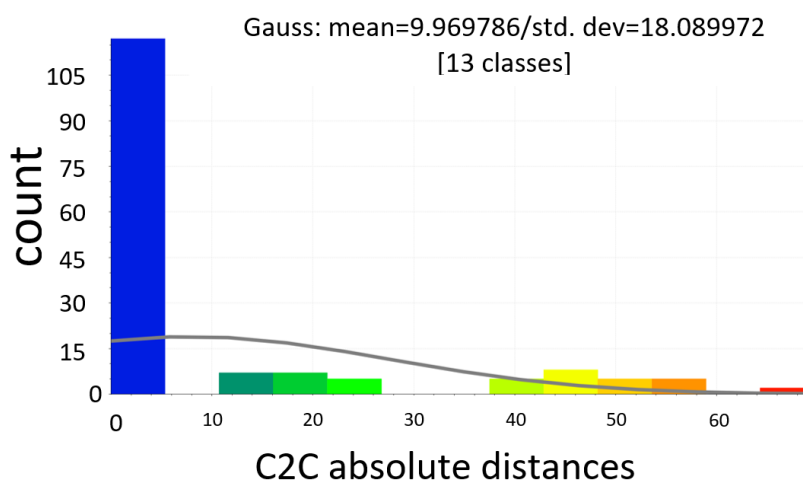


Figure 4.22: Normal distribution of Basin Model at Minimum Scale of 70.

The last keypoint repeatability to be analysed was the keypoints resulted between own point clouds under different conditions. This was to evaluate the ability of the detector in identifying the same keypoints even if the environment changed. The first varying condition was the changing minimum scale of Gaussian scale space in SIFT detector. Minimum scale of 65 was taken as the reference for comparison and all the keypoints analysed here were the unique ones. The numbers of exact repeated keypoints for the models were tabulated in Table 4.7. The results shown were poor. Similarly, a C2C distance threshold of approximately 15 or first four classes (bins) was set to collect keypoints which distances were within the threshold as presented in Table 4.8. Figure 4.23, Figure 4.24 and Figure 4.25 show the visualization of histogram of C2C absolute distance of three input models at minimum scale of 70.

Next, the evaluation of the repeatability for models' keypoints under transformation was done. The keypoints for each model at minimum scale of 65 were rotated at 20° , as displayed in Figure 4.26, Figure 4.27 and Figure 4.28. Table 4.9 shows that the numbers of the exact repeated keypoints and the percentage of repeated keypoints under C2C absolute distance threshold of 15 (first four classes) between the original and rotated keypoints for each model.

Based on the results above, SIFT keypoint detector was actually lack of capability in detecting same set of keypoint under changing circumstances as the amount of keypoints that located at exact same place was low. However, with a given distance threshold, the results of repeated keypoints were acceptable.

Table 4.7: Numbers of Repeated Keypoints between Own Models Point Cloud Under Different Minimum Scale.

Min Scale	Crocodile	Seal	Basin
65			
70	0	0	0
75	0	0	0
80	2	0	0
82	0	0	0

Table 4.8: Percentage of Keypoints Within Threshold for Input Models at Different Minimum Scale.

Min Scale	Crocodile (%)	Seal (%)	Basin (%)
65			
70	28.434	27.591	15.951
75	16.877	33.750	32.203
80	34.574	29.659	30.693
82	29.002	38.255	24.719

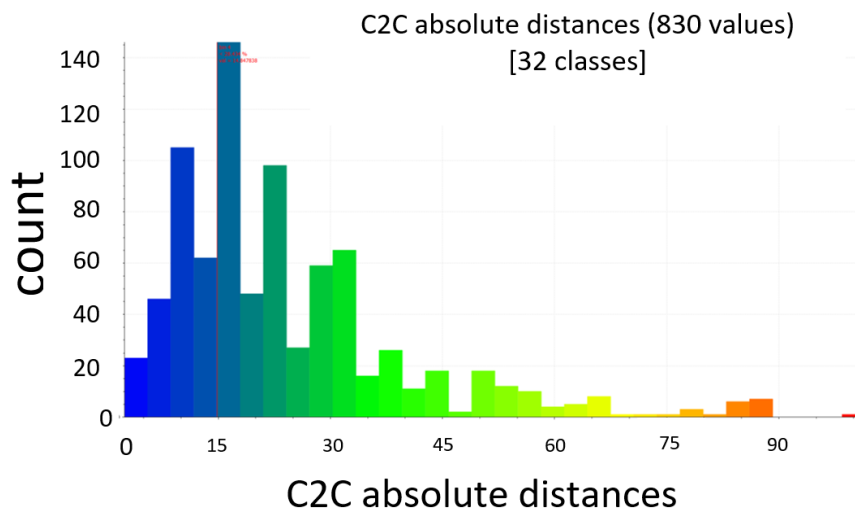


Figure 4.23: Histogram of C2C Absolute Distance of Crocodile Model at Minimum Scale of 70.

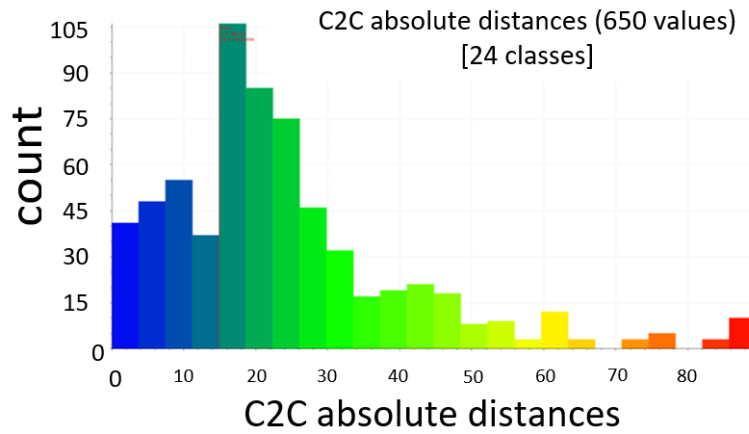


Figure 4.24: Histogram of C2C Absolute Distance of Seal Model at Minimum Scale of 70.

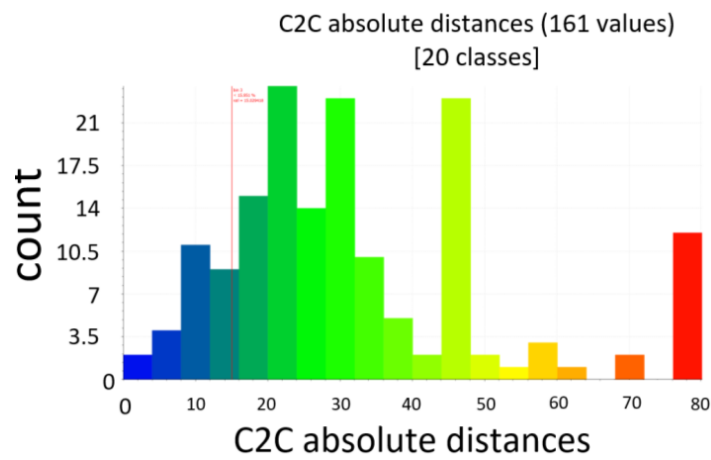


Figure 4.25: Histogram of C2C Absolute Distance of Basin Model at Minimum Scale of 70.



Figure 4.26: Crocodile's Keypoints Before and After 20° Rotation (Left: Rotated; Right: Original).

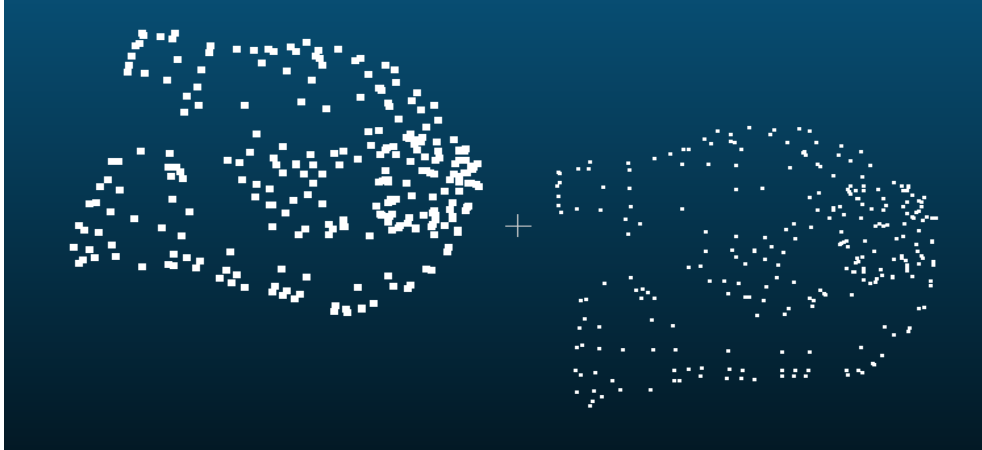


Figure 4.27: Seal's Keypoints Before and After 20° Rotation (Left: Rotated; Right: Original).

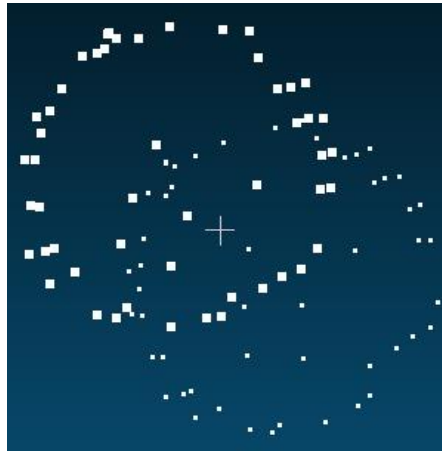


Figure 4.28: Basin's Keypoints Before and After 20° Rotation (Left: Rotated; Right: Original).

Table 4.9: Numbers of Exact Repeated Keypoints and Percentage of Repeated Keypoints within Threshold between Original and Rotated Keypoints for Each Model.

Models (After 20° Rotation)	Number of Exact Repeatability	Percentage (%) of Repeated Keypoints (Under C2C Threshold)
Crocodile	0	2.295
Seal	0	6.736
Basin	0	24.096

4.4.3 Keypoint Detector's Efficiency / Time Performance

In order to analyse the efficiency of the SIFT keypoint detector based on different minimum scales, time taken to detect the keypoints was computed, as shown in Table 4.10. Besides, Figure 4.29 compares the relationship of the total keypoint computation time for input crocodile, seal and basin model point clouds at different minimum scale of Gaussian scale space. The runtime experiments were done on an Intel Core i5 with 8GB RAM and noted that the time complexity was not 100% accurate as it depended on CPU performance.

Based on Table 4.10 and Figure 4.29, the SIFT keypoint computation time decreased with the increasing minimum scale of the Gaussian scale space. If smaller scales for keypoints detection were used, SIFT detector was actually using more time to detect keypoints that were located at same position and location. Therefore, in term of efficiency, the minimum scale of 82 was the ideal parameter that showed an adequate ratio between the amount of keypoint detected and execution time.

After analysing the results of keypoint quantity and quality, keypoint repeatability and time efficiency, the final minimum Gaussian scale selected for SIFT keypoint detector was 82 as it produced an adequate number of keypoints with a high quality. Besides, since the results of the keypoint repeatability rate were almost the same for every minimum scale, 82 was chosen as it required only a short period of time to detect the keypoint. This scale produced the highest efficiency for keypoint detection.

Table 4.10: Model's Keypoint Computation Time at Different Minimum Scale.

Min Scale	Computation Time (second)		
	Crocodile	Seal	Basin
65	154.588	157.578	150.949
70	153.047	154.510	150.328
75	150.011	154.370	150.083
80	142.779	154.057	138.921
82	140.764	151.502	135.045

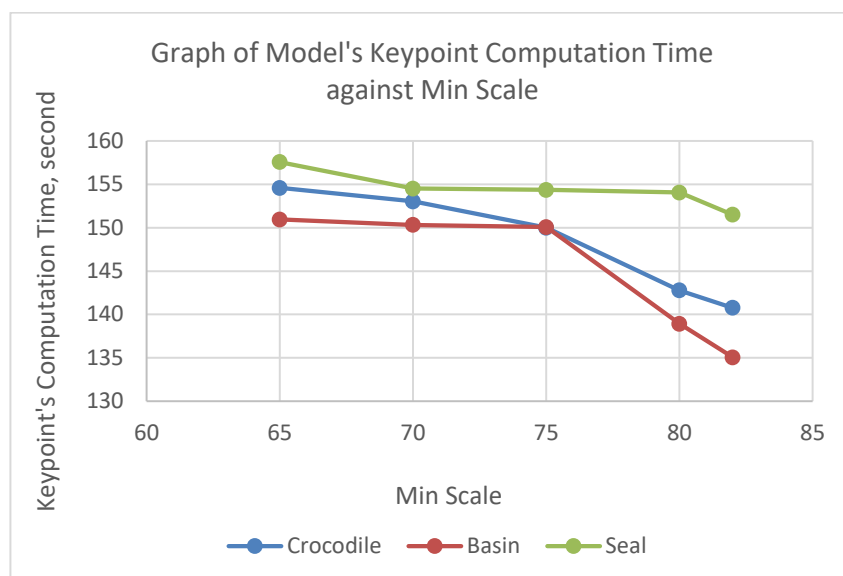


Figure 4.29: Graph of Model's Keypoint Computation Time at Different Min Scale.

4.5 Descriptor Construction

There were two main parameters that could affect the qualities of the descriptors computed. The first parameter was the surface normals estimated for each point in the very first step as both methods of descriptor construction (PFH and SHOT) used in this project required them to obtain the local geometry properties of the keypoints. From the previous results, the quality of the surface normals computed was high. Therefore, the surface normal would not cause much effect on the results of the descriptor construction. In this part, the main parameter that could affect the performance of descriptor construction was the radius of sphere set for searching neighbouring keypoints. The radius r set to search for the neighbours in PFH and SHOT should be adequate in collecting enough information from surrounding keypoints. According to Grupo De Robotica (2015a), the radius cannot be too large or else the information collected from the k nearest neighbours may be cluttered. Besides, it cannot be too small or else there is not enough keypoints to compute the local geometry.

For both PFH and SHOT descriptors, the radius of sphere was set as $r = 20, 40, 60$ and the output of histograms of the 50th keypoint at $r = 20$ and $r = 60$ were plotted as shown in Figure 4.30, Figure 4.31, Figure 4.32, Figure 4.33, Figure 4.34, Figure 4.35, Figure 4.36 and Figure 4.37. The y-axis represented the histogram values which formed by the percentage of the points storing in

each bin while the x -axis represented the number histogram bins. There were total 125 bins in the PFH histogram and 352 bins in the SHOT histogram. For both descriptors, when the radius was set larger, the percentage of the points storing in each bin was higher. $r=20$ was selected as final scale as all histograms show an adequate information and it has the highest efficiency in computing the descriptors.

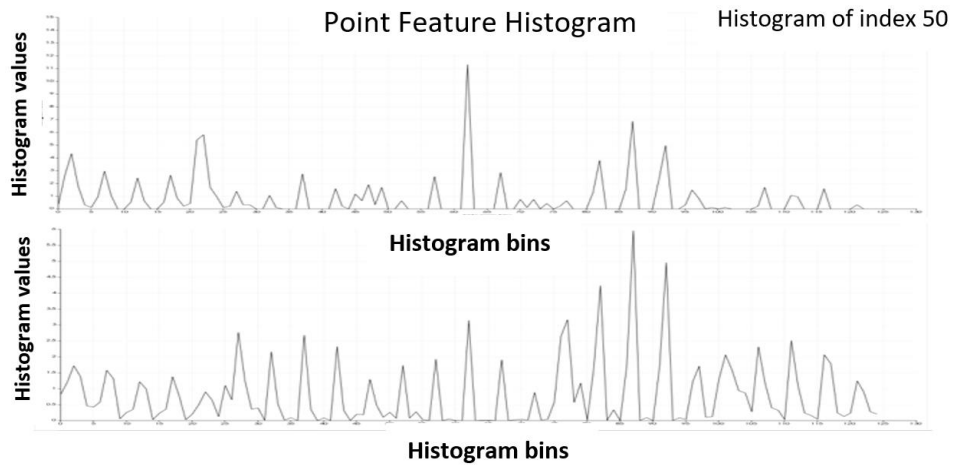


Figure 4.30: Visualization of PFH Output Histogram for Crocodile (Top: $r=20$; Bottom: $r=60$).

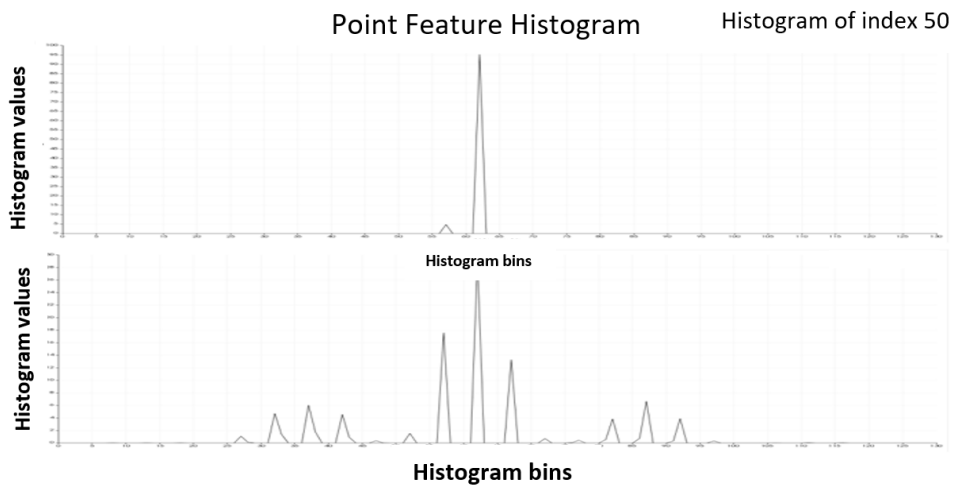


Figure 4.31: Visualization of PFH Output Histogram for Seal (Top: $r=20$; Bottom: $r=60$).

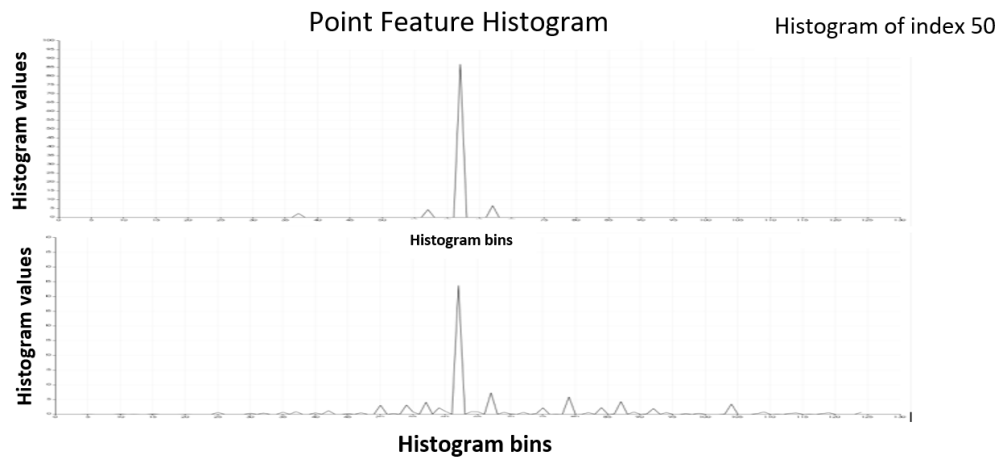


Figure 4.32: Visualization of PFH Output Histogram for Basin (Top: $r = 20$; Bottom: $r = 60$).

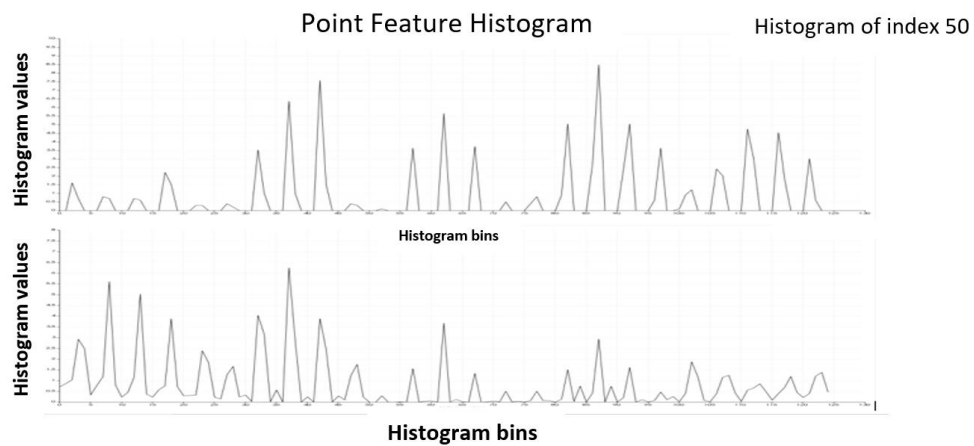


Figure 4.33: Visualization of PFH Output Histogram for Scene (Top: $r = 20$; Bottom: $r = 60$).

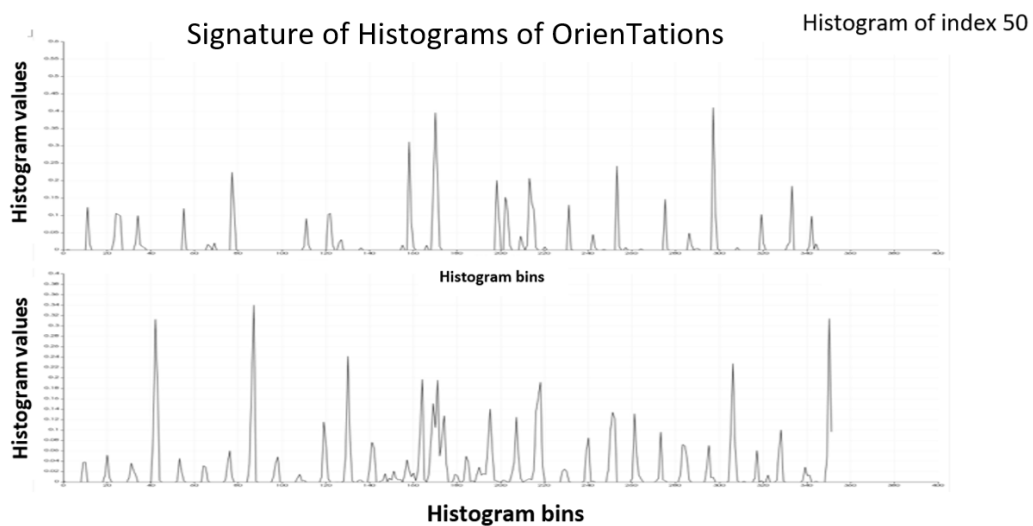


Figure 4.34: Visualization of SHOT Output Histogram for Crocodile (Top: $r = 20$; Bottom: $r = 60$).

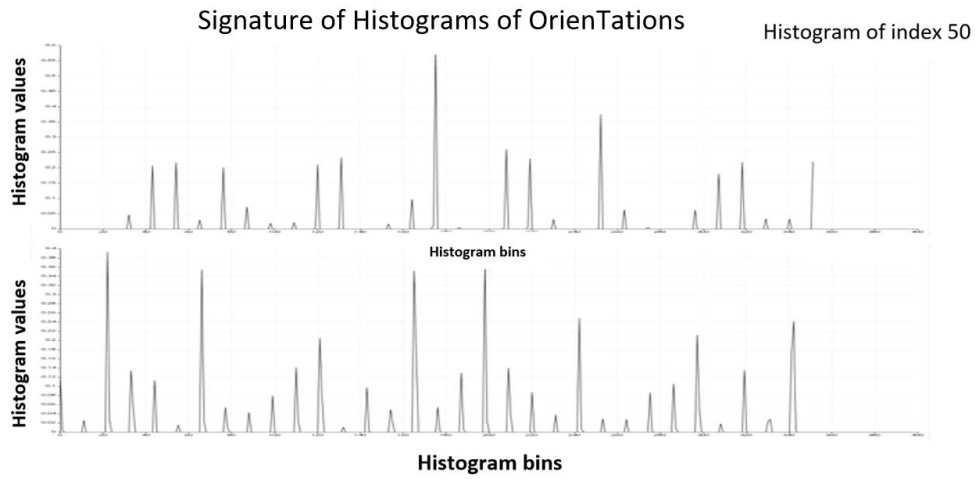


Figure 4.35: Visualization of SHOT Output Histogram for Seal (Top: $r = 20$; Bottom: $r = 60$).

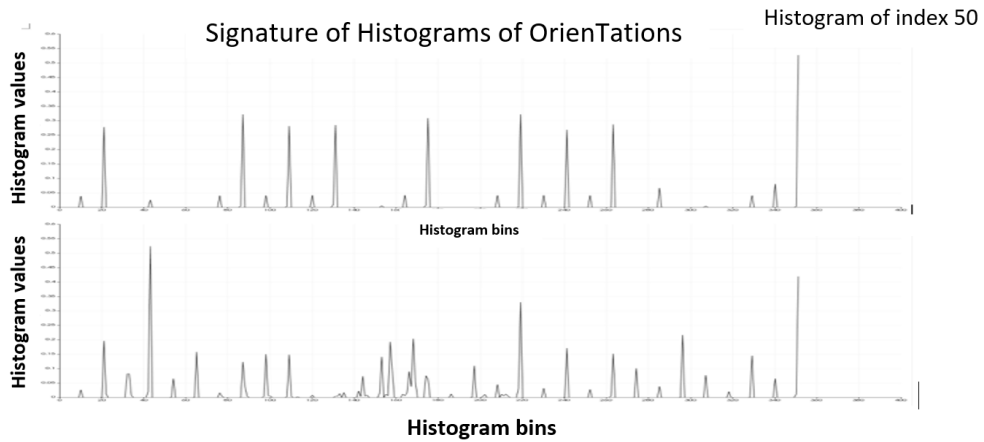


Figure 4.36: Visualization of SHOT Output Histogram for Basin (Top: $r = 20$; Bottom: $r = 60$).

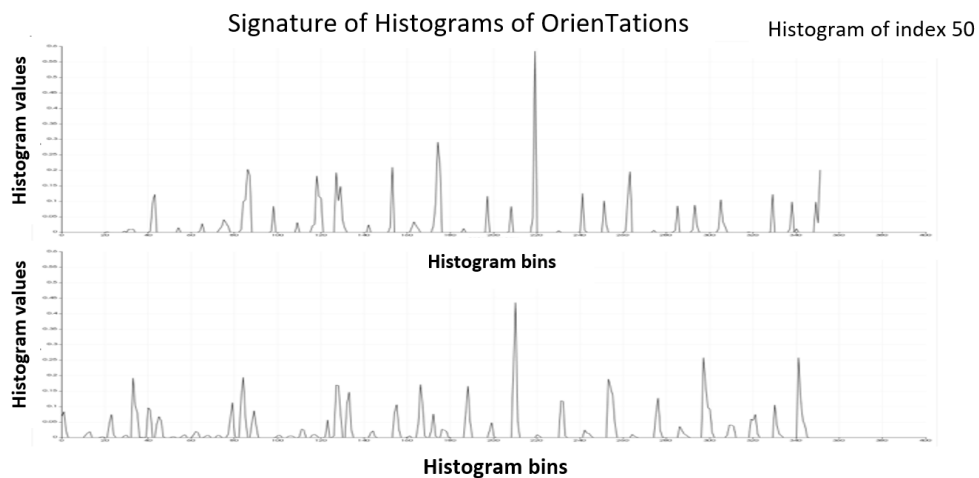


Figure 4.37: Visualization of SHOT Output Histogram for Scene (Top: $r = 20$; Bottom: $r = 60$).

The efficiency of the PFH and SHOT descriptors construction process was analysed by measuring their computing time. Based on Table 4.11, with a larger radius, both methods required a longer calculation time to compute their descriptors as there were more nearest neighbours involved. By comparing between PFH and SHOT, PFH spent a much longer time to compute the descriptors than SHOT as it has a complexity of $O(k^2)$. This drawback of PFH could be later seen when the descriptors were generated for a very large point cloud. From Table 4.11 and Figure 4.38, SHOT only needed a short time to compute the descriptors for scene point cloud. However, for PFH, it used more than 4 hours to compute its descriptors. Therefore, SHOT was more efficient and it was selected as the main descriptor construction method. The ability of SHOT descriptor in finding high quality correspondences was determined in the process of feature matching. The results will be shown and discussed in the next section.

Table 4.11: Comparison of Computational Time for PFH and SHOT Descriptors at Different Radius for Neighbour Searching.

Radius, r			20	40	60
Descriptor Computational Time, s	Crocodile	PFH	3.237	58.454	318.369
		SHOT	0.618	1.269	2.209
	Seal	PFH	1.074	21.542	102.912
		SHOT	0.371	0.378	0.679
	Basin	PFH	0.227	2.509	18.094
		SHOT	0.044	0.076	0.131
	Scene	PFH	220.557	3253.480	15183.000
		SHOT	32.649	39.607	87.227

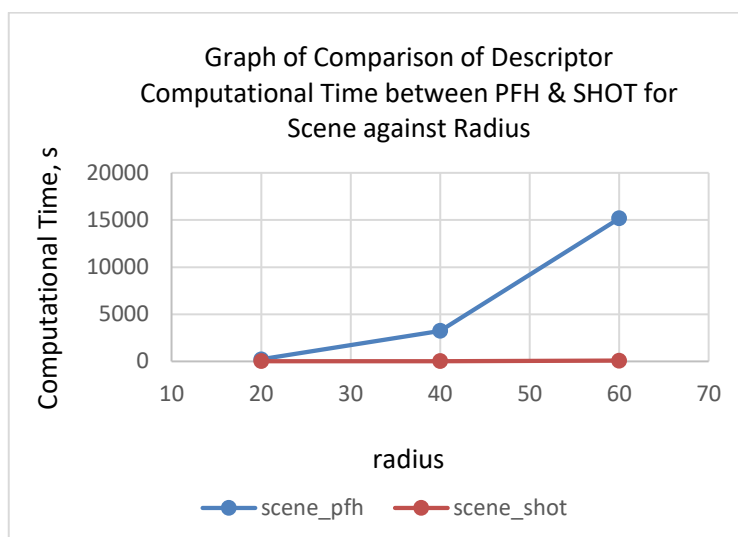


Figure 4.38: Graph of Comparison of Descriptor Computational Time between PFH and SHOT for Scene against Radius.

4.6 Feature Matching

Matching of the scene's descriptors and model's descriptor was performed to obtain point-to-point correspondences. In this step, there was only one parameter which was the distance threshold, d needed to be set to decide whether the scene's keypoints were similar to the model's keypoints. Three distance thresholds, $d = 0.15$, $d = 0.20$ and $d = 0.25$ were set to test the effect of the threshold on the quality of the correspondences found. The thresholds set were less than 1 as the SHOT descriptors were designed between 0 and 1 (Grupo De Robotica, 2015b). The quantity of the original correspondences and the correct correspondences detected between three models and the scene at different thresholds were tabulated in Table 4.12. Noted that the k nearest neighbours set for normal estimation was 10, the minimum Gaussian scale space set for SIFT keypoint detector was 82 and the radius set for constructing SHOT descriptor was 20.

Accuracy rate of the feature matching process was also calculated using the quantity of the original correspondences and the correct correspondences, as shown in Equation 4.1. The results were recorded in Table 4.12 as well. The accuracy rate of feature matching also represented the descriptiveness of the SHOT descriptors computed. Descriptiveness of the SHOT descriptors was defined as the ability of the descriptors in detecting inliers out of total correspondences.

$$Accuracy\ Rate\ (AR) = \frac{Correct\ Correspondences}{Total\ Correspondences} \quad (4.1)$$

Table 4.12: Quantity of Original and Correct Correspondences and Accuracy of Feature Matching from Different Thresholds.

Distance Threshold, d			0.15	0.20	0.25
Number of Correspondences	Crocodile	Total	608	1125	2060
		Correct	375	375	375
		AR	0.617	0.333	0.182
	Seal	Total	387	811	1598
		Correct	227	227	230
		AR	0.587	0.280	0.144
	Basin	Total	93	223	762
		Correct	60	60	60
		AR	0.645	0.269	0.079

Based on Table 4.12, when a smaller threshold was set, there were less correspondences found. It clearly shows that the accuracy rate of the feature matching process was higher with a smaller threshold. Besides, it can be observed that the number of correct correspondences was almost the same for all three models even at the different thresholds. This proved that the SHOT descriptor had a high capability in determining same set of inliers from different sets of total correspondences found. Distance threshold $d = 0.15$ was selected as it produced the highest accuracy of the feature matching process among other threshold. The accuracy rate for this threshold shows that the SHOT descriptor had a high descriptiveness as it was able to generate the correct correspondences of more than half from the total correspondences for all models.

4.7 Hypotheses Generation

Hypotheses generation was the final step to recognize and locate the input models from the scene point clouds. It filtered all low quality correspondences and clustered the remaining inliers to generate a model instance. As mentioned before, there were three parameters needed to be set: rf_rad , cg_size and cg_thresh . In this part, only parameter cg_thresh (clustering threshold) was

adjusted in order to recognize and localize only one actual model instance from the scene. It was set as $cg_thresh = 1, 5$ and 20 . The number of the model instances generated was recorded in Table 4.13. The other two parameters were set as:

- $rf_rad = 50$.
- $cg_size = 30$.

Noted that the k nearest neighbours set for normal estimation was 10, the minimum Gaussian scale space set for SIFT keypoint detector was 82, the radius set for constructing SHOT descriptor was 20 and the distance threshold d set for feature matching was 0.15. The accuracy rate of the model instances generated or also known as the model localization accuracy was calculated using Equation 4.2. The number of correct model instance was defined as the final model instance which was localized correctly with a set of correct correspondences and it always equalled to 1.

$$Accuracy\ Rate = \frac{Number\ of\ Correct\ Model\ Instance}{Number\ of\ Total\ Model\ Instances} \quad (4.2)$$

Table 4.13: Number of Model Instances Generated from Different cg_thresh .

cg_thresh			1	5	20
Number of Model Instances Generated	Crocodile	Total	29	3	1
		Correct	1	1	1
		AR	0.034	0.333	1
	Seal	Total	26	2	1
		Correct	1	1	1
		AR	0.038	0.5	1
	Basin	Total	6	1	1
		Correct	1	1	1
		AR	0.167	1	1

From Table 4.13, when the clustering threshold set to form a cluster was small, more model instances were generated as they only required a few correspondences to form a cluster. Since the correspondences was too less, the

model instances were mostly matched wrongly with the actual model in the scene. The accuracy rate of the model localization increased with the increment of cg_thresh . By adjusting the parameter, it was proved that all models were recognized and localized correctly when $cg_thresh = 20$.

4.8 Overall Parameter Set and Final Results

The algorithm first tested the localization of crocodile, seal and basin from the scene. The parameters set for each processes were set as following:

Table 4.14: Parameters Set for Each Step in Model Localization.

Normal Estimation	k	10
SIFT Keypoint Detection	min_scale	82
	$n_octaves$	70
	$n_scales_per_octave$	90
	$min_contrast$	0
SHOT Descriptor Construction	r	20
Feature Matching	d	0.15
Hypotheses Generation	rf_rad	50
	cg_size	30
	cg_thresh	20

Table 4.15: Dimension of Models Localized from Scene.

Dimension	Crocodile	Seal	Basin
Length	1338.49	1037.14	400.716
Height	851.776	750.881	364.901
Depth	662.133	746.483	331.603

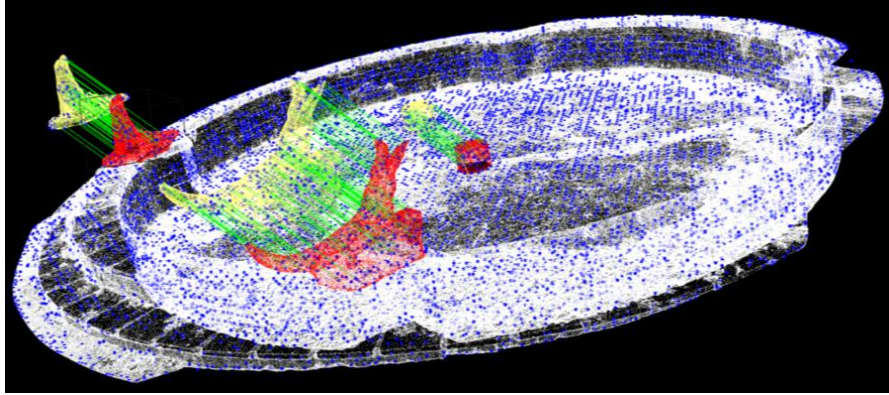


Figure 4.39: Localization of Three Models from Scene.

Figure 4.39 shows the localization of the crocodile, seal and basin models from the scene at the same time and Table 4.15 shows the dimensions of all three models that were localized from the scene. The yellow models represented the input models, the red models were the model instances generated from Hough Voting and the green lines were the correct correspondences matching between the input models and the model instances. Besides, the algorithm was used to localize the transformed models (20° rotation). The parameters set were shown in Table 4.16. Figure 4.40 and Figure 4.41 show the localization of the rotated crocodile and seal from the scene. Since the input rotated models were accurately localized from the scene, the algorithm was invariant to transformation.

Table 4.16: Parameters Set for Rotated Crocodile and Seal Localization.

Normal Estimation	k	10
SIFT Keypoint Detection	min_scale	82
	$n_octaves$	70
	$n_scales_per_octave$	90
	$min_contrast$	0
SHOT Descriptor Construction	r	60
Feature Matching	d	0.15
Hypotheses Generation	rf_rad	50
	cg_size	60
	cg_thresh	10

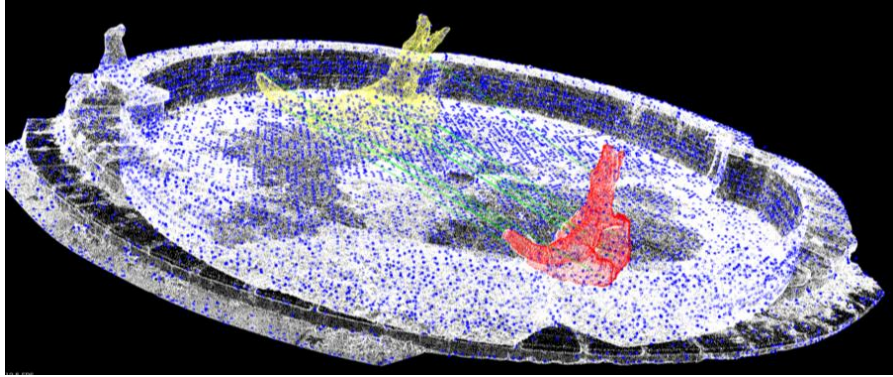


Figure 4.40: Localization of Rotated Crocodile Model from Scene.

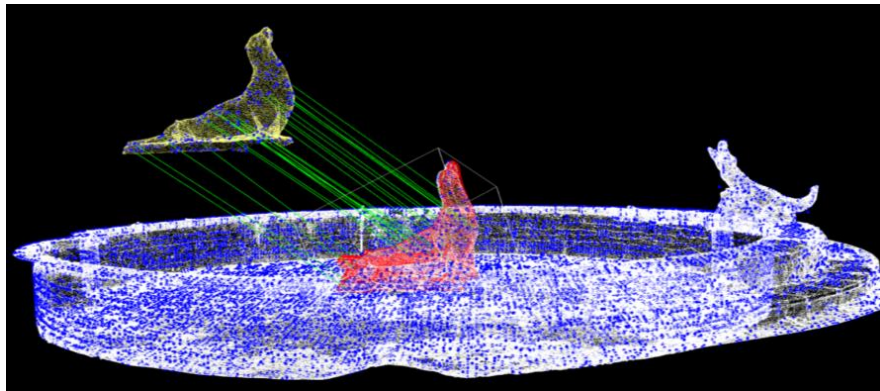


Figure 4.41: Localization of Rotated Seal Model from Scene.

4.9 Summary

There were total five processes to perform the object localization in 3D point cloud. Adjustment of the parameters in each process was very crucial as it could affect the final localization results. In normal estimation, the size of the k -neighbourhood was set as $k=10$ as it produced a group of consistently oriented surface normals. Then, $min_scale = 82$ was set for the SIFT keypoint detector as the number of the keypoints was adequate and the keypoints computed were descriptive. Next, radius of sphere $r = 20$ in SHOT descriptor construction was selected as it produced the highest efficiency in constructing the descriptors. In feature matching step, the distance threshold to compute the feature correspondences was set as $d = 0.15$ as it had the highest accuracy rate in detecting inliers out of total correspondences. Lastly, $cg_thresh = 20$ was set in Hough Voting as all three models were recognized and localized correctly from the scene. Besides, the algorithm was invariant to transformation as it was able to localize rotated models from the scene by simply adjusting the parameters.

CHAPTER 5

CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusions

In conclusion, an efficient algorithm that is invariant to transformation and is able to recognize and localize multiple models simultaneously from a rich 3D scene point cloud has been developed for this project. The first and second objectives were achieved as all targeted models were successfully localized from a rich fountain scene point cloud which consisted of multiple objects. The algorithm consists of a total of four major steps to perform the object localization: keypoint detection, descriptor construction, feature matching and hypotheses generation. Parameters in each of the steps play a crucial role in producing desirable and the targeted results. The parameters were carefully adjusted to ensure a 100% localization accuracy rate. A comprehensive comparison and evaluation of the results were performed to investigate the behaviour of the individual methods used. The localization of an object from a rich 3D point cloud came along with a 3D bounding box. The third objective was achieved as the length, width and height of the object were well-calculated from the bounding box. The algorithm was further tested on the transformed models and the results showed that the models were accurately localized from the scene. In short, the aim and objectives of this project were successfully accomplished.

5.2 Recommendations for Future Work

The algorithm provided in this project requires a frequent adjustment on the parameters and the users have to decide the final parameter manually by analysing the results. It is very time consuming in finding the perfect parameter for each process. Therefore, the future work will be focusing on the research of the object localization in 3D point cloud using machine learning or deep learning. Training dataset will be provided to the algorithm to allow it to learn and compute the best parameter. Then, testing dataset will be used to test the ability of the algorithm in locating objects from a rich point cloud. Perhaps the

optimization functions that are provided in machine learning or deep learning could produce an even more accurate result and a shorter computational time.

REFERENCES

- Ashbrook, A. P., Fisher, R. B., Robertson, C. and Werghi, N., 1998. *Finding surface correspondence for object recognition and registration using pairwise geometric histograms*. In: Burkhardt H., Neumann B., Computer Vision — ECCV'98, 5th European Conference on Computer Vision. Freiburg, Germany, 2-6 June 1998. Springer, Berlin, Heidelberg.
- Aldoma, A., Tombari, F., Stefano, L.D. and Vincze, M., 2012a. *A Global Hypotheses Verification Method for 3D Object Recognition*. In: Computer Vision – ECCV 2012: 12th European Conference on Computer Vision. Florence, Italy, 7-13 October 2012. ResearchGate.
- Aldoma, A., Marton, Z. C., Tombari, F., Wohlkinger, W., Potthast, C., Zeisl, B., Rusu, R. B., Gedikli, S. and Vincze, M., 2012b. Tutorial: Point Cloud Library: Three-Dimensional Object Recognition and 6 DOF Pose Estimation. *IEEE Robotics & Automation Magazine*, [e-journal] 19(3), pp.80-91. Available through: Universiti Tunku Abdul Rahman Library website <<http://library.utar.edu.my/>> [Accessed 16 April 2020].
- Artec Europe, 2020. *Fountain Basin*. [point cloud] Available at: <<https://www.artec3d.com/3d-models/fountain-basin>> [Accessed 26 March 2020].
- Bariya, P. and Nishino, K., 2010. *Scale-hierarchical 3D object recognition in cluttered scenes*. In: 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. San Francisco, CA, USA, 13-18 June 2010. IEEE.
- Bariya, P. and Nishino, K., 2010. *Schematic of Scale-Hierarchical Interpretation Tree*. [image] Available at: <<https://ieeexplore.ieee.org/document/5539774>> [Accessed 17 August 2019].
- Bielicki, J. and Sitnik, R., 2013. A method of 3D object recognition and localization in a cloud of points. *EURASIP Journal on Advances in Signal Processing*, [e-journal] 2013(1). Available through: Universiti Tunku Abdul Rahman Library website <<http://library.utar.edu.my/>> [Accessed 9 August 2019].
- Bielicki, J. and Sitnik, R., 2013. *Histogram of 2D Distribution of Parameters C1 versus C2*. [image] Available at: <<https://link.springer.com/article/10.1186/1687-6180-2013-29#citeas>> [Accessed 9 August 2019].
- Bielicki, J. and Sitnik, R., 2013. *Building of Reference Object Descriptor*. [image] Available at: <<https://link.springer.com/article/10.1186/1687-6180-2013-29#citeas>> [Accessed 9 August 2019].
- Brownlee, J., 2019. A Gentle Introduction to Object Recognition With Deep Learning. *Deep Learning for Computer Vision*, [blog] 22 May. Available at: <<https://machinelearningmastery.com/object-recognition-with-deep-learning/>> [Accessed 18 August 2019].

Chen, H. and Bhanu, B., 2007. 3D free-form object recognition in range images using local surface patches. *Pattern Recognition Letters*, [e-journal] 28(10). Available through: Universiti Tunku Abdul Rahman Library website <<http://library.utar.edu.my/>> [Accessed 17 August 2019].

Chua, C. S. and Jarvis, R., 1997. Point Signatures: A New Representation for 3D Object Recognition. *International Journal of Computer Vision*, [e-journal] 25. Available through: Universiti Tunku Abdul Rahman Library website <<http://library.utar.edu.my/>> [Accessed 15 August 2019].

Chua, C. S. and Jarvis, R., 1997. *Point Signature*. [image] Available at: <<https://link.springer.com/article/10.1023/A:1007981719186#citeas>> [Accessed 15 August 2019].

CloudCompare, 2015a. *Cloud-to-Cloud Distance*. [online] Available at: <https://www.cloudcompare.org/doc/wiki/index.php?title=Cloud-to-Cloud_Distance> [Accessed 7 April 2020].

CloudCompare, 2015b. *Distances Computation*. [online] Available at: <https://www.cloudcompare.org/doc/wiki/index.php?title=Distances_Computation> [Accessed 7 April 2020].

Filipe, S. and Alexandre, L. A., 2014. *A comparative evaluation of 3D keypoint detectors in a RGB-D Object Dataset*. In: 2014 International Conference on Computer Vision Theory and Applications (VISAPP). Lisbon, Portugal, 5-8 January 2014. IEEE.

Flint, A., Dick, A. and Hengel, A. V. D., 2014. *Thrift: Local 3D Structure Recognition*. In: 9th Biennial Conference of the Australian Pattern Recognition Society on Digital Image Computing Techniques and Applications (DICTA 2007). Glenelg, Australia, 3-5 December 2007. IEEE.

Flint, A., Dick, A. and Hengel, A.V. D., 2014. *A Keypoint with Two Least Squares Planes and their Related Normals for One Support Point on a Local Surface*. [image] Available at: <<https://ieeexplore.ieee.org/document/4426794>> [Accessed 11 August 2019].

Frome, A. Huber, D., Kolluri, R., Bülow, T. and Malik, J., 2004. *Visualization of Histogram Bins of the 3D Shape Context*. [image] Available at: <https://link.springer.com/chapter/10.1007/978-3-540-24672-5_18#citeas> [Accessed 10 August 2019].

Frome, A. Huber, D., Kolluri, R., Bülow, T. and Malik, J., 2004. *Comparison of Recognition Rate between Different Descriptors in Noise Experiments*. [image] Available at: <https://link.springer.com/chapter/10.1007/978-3-540-24672-5_18#citeas> [Accessed 10 August 2019].

Frome, A. Huber, D., Kolluri, R., Bülow, T. and Malik, J., 2004. *Comparison of Recognition Rate between Different Descriptors in Clutter Experiments*. [image] Available at: <https://link.springer.com/chapter/10.1007/978-3-540-24672-5_18#citeas> [Accessed 10 August 2019].

Frome, A., Huber, D., Kolluri, R., Bülow, T. and Malik, J., 2004. *ECCV 2004: Computer Vision - ECCV 2004*. [e-book] Springer, Berlin, Heidelberg. Available through: Universiti Tunku Abdul Rahman Library website <<http://library.utar.edu.my/>> [Accessed 18 April 2020].

Glomb, P., 2009. *Detection of Interest Points on 3D Data: Extending the Harris Operator*. [e-book] Springer, Berlin, Heidelberg. Available at: Springer Link <https://link.springer.com/chapter/10.1007/978-3-540-93905-4_13#citeas> [Accessed 26 July 2019].

Grupo De Robotica., 2015a. *PCL/OpenNI tutorial 4: 3D object recognition (descriptors)*. [online] Available at: <[http://robotica.unileon.es/index.php/PCL/OpenNI_tutorial_4:_3D_object_recognition_\(descriptors\)#PFH](http://robotica.unileon.es/index.php/PCL/OpenNI_tutorial_4:_3D_object_recognition_(descriptors)#PFH)> [Accessed 18 April 2020].

Grupo De Robotica., 2015b. *PCL/OpenNI tutorial 5: 3D object recognition (pipeline)*. [online] Available at: <[http://robotica.unileon.es/index.php/PCL/OpenNI_tutorial_5:_3D_object_recognition_\(pipeline\)#Correspondence_grouping](http://robotica.unileon.es/index.php/PCL/OpenNI_tutorial_5:_3D_object_recognition_(pipeline)#Correspondence_grouping)> [Accessed 20 April 2020].

Guo, Y. L., Sohel, F., Bennamoun, M., Lu, M. and Wan, J. W., 2013. Rotational Projection Statistics for 3D Local Surface Description and Object Recognition. *International Journal of Computer Vision*, [e-journal] 105(1), pp.63-86. 10.1007/s11263-013-0627-y.

Guo, Y. L., Sohel, F., Bennamoun, M., Lu, M. and Wan, J. W., 2014. 3D Object Recognition in Cluttered Scenes with Local Surface Features: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, [e-journal] 36(11), pp. 2270 – 2287. Available through: Universiti Tunku Abdul Rahman Library website <<http://library.utar.edu.my/>> [Accessed 23 July 2019].

Hansch, R., Weber, T. and Hellwich, O., 2014. *Comparison of 3D Interest Point Detectors and Descriptors for Point Cloud Fusion*. In: PCV 2014. Zurich, Switzerland, September 2014. IEEE.

Huang J. and You, S., 2013. *Detecting Objects in Scene Point Cloud: A Combinational Approach*. In: 2013 International Conference on 3D Vision - 3DV 2013. Seattle, WA, USA, 29 June-1 July 2013. IEEE.

Johnson, A. E. and Hebert, M., 1999. Using spin images for efficient object recognition in cluttered 3D scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, [e-journal] 21(5), pp. 433 - 449. Available through: Universiti Tunku Abdul Rahman Library website <<http://library.utar.edu.my/>> [Accessed 7 August 2019].

Johnson, A. E. and Hebert, M., 1999. *Oriented Point Basis*. [image] Available at: <<https://ieeexplore.ieee.org/document/765655>> [Accessed 7 August 2019].

Lamdan, Y. and Wolfson, H. J., 1998. *Geometric Hashing: A General And Efficient Model-based Recognition Scheme*. In: [1988 Proceedings] Second International Conference on Computer Vision. Tampa, FL, USA, 5-8 December 1988. IEEE.

Lowe, D. G., 2004. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, [e-journal] 60(2), pp.91–110. <https://doi-org.libezp2.utar.edu.my/10.1023/B:VISI.0000029664.99615.94>

Marr, B., 2019. *7 Amazing Examples Of Computer And Machine Vision In Practice*. [online] Available at: <https://www.forbes.com/sites/bernardmarr/2019/04/08/7-amazing-examples-of-computer-and-machine-vision-in-practice/#42cde3851018> [Accessed 18 August 2019].

Matei, B., Shan, Y., Sawhney, H. S., Tan, Y., Kumar, R., Huber, D. and Hebert, M., 2006. Rapid Object Indexing Using Locality Sensitive Hashing and Joint 3D-Signature Space Estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, [e-journal] 28(7), pp. 1111 – 1126. Available through: Universiti Tunku Abdul Rahman Library website <http://library.utar.edu.my/> [Accessed 25 July 2019].

Mian, A., Bennamoun, M. and Owens, R., 2006. Three-Dimensional Model-Based Object Recognition and Segmentation in Cluttered Scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, [e-journal] 28(10). Available through: Universiti Tunku Abdul Rahman Library website <http://library.utar.edu.my/> [Accessed 16 August 2019].

Mian, A., Bennamoun, M. and Owens, R., 2006. *Tensor Computation*. [image] Available at: <https://ieeexplore.ieee.org/document/1677516/versions> [Accessed 16 August 2019].

Mian, A., Bennamoun, M. and Owens, R., 2010. On the Repeatability and Quality of Keypoints for Local Feature-based 3D Object Retrieval from Cluttered Scenes. *International Journal of Computer Vision*, [e-journal] 89(2-3), pp. 348–361. Available through: Universiti Tunku Abdul Rahman Library website <http://library.utar.edu.my/> [Accessed 16 August 2019].

Mian, A., Bennamoun, M. and Owens, R., 2010. *Detected Keypoints (Red Dots)*. [image] Available at: <https://link.springer.com/article/10.1007/s11263-009-0296-z> [Accessed 16 August 2019].

Nicholson, C., 2019. *Object Recognition and Localization using Convolutional Neural Networks*. [image] Available at: <https://skymind.com/wiki/autonomous-vehicle> [Accessed 18 August 2019].

Papazov, C. and Burschka, D., 2010. *An Efficient RANSAC for 3D Object Recognition in Noisy and Occluded Scenes*. In: Kimmel R., Klette R., Sugimoto A., Computer Vision – ACCV 2010, Asian Conference on Computer Vision. Queenstown, New Zealand, 8-12 November 2010. Springer, Berlin, Heidelberg.

Point Cloud Library (PCL), 2012. *pcl::Feature<PointInT, PointOutT> Class Template Reference*. [online] Available at: <http://docs.pointclouds.org/1.5.1/classpcl_1_1_feature.html#a50129bc51cb240eca42df9963f7ac0c0> [Accessed 16 April 2020].

Point Cloud Library (PCL), 2013. *pcl::KdTreeFLANN< PointT, Dist > Class Template Reference*. [online] Available at: <http://docs.pointclouds.org/1.7.1/classpcl_1_1_kd_tree_f_l_a_n_n.html> [Accessed 18 April 2020].

Point Cloud Library (PCL), 2018a. *pcl::SIFTKeypoint<PointInT, PointOutT> Class Template Reference*. [online] Available at: <http://docs.pointclouds.org/1.8.1/classpcl_1_1_s_i_f_t_keypoint.html> [Accessed 16 April 2020].

Point Cloud Library (PCL), 2018b. *pcl::SHOTEstimation<PointInT, PointNT, PointOutT, PointRFT> Class Template Reference*. [online] Available at: <http://docs.pointclouds.org/1.8.1/classpcl_1_1_s_h_o_t_estimation.html> [Accessed 18 April 2020].

Point Cloud Library (PCL), 2020a. *pcl::PFHEstimation<PointInT, PointNT, PointOutT> Class Template Reference*. [online] Available at: <http://docs.pointclouds.org/trunk/classpcl_1_1_p_f_h_estimation.html> [Accessed 18 April 2020].

Point Cloud Library (PCL), 2020b. *pcl::Hough3DGrouping<PointModelT, PointSceneT, PointModelRfT, PointSceneRfT> Class Template Reference*. [online] Available at: <http://docs.pointclouds.org/trunk/classpcl_1_1_hough3_d_grouping.html> [Accessed 20 April 2020].

Rocha, L. C. G., 2017. *A Study on Local Feature Descriptors for Point Clouds*. Bachelor. Federal University of Rio Grande do Norte Center for Earth and Exact Sciences. Available at: <<https://www.semanticscholar.org/paper/A-study-on-local-feature-descriptors-for-point-Rocha/6ce34d8a3ea5e2ba9a069a8ca86aae63569d905b>> [Accessed 18 April 2020].

Rodolà, E., Albarelli, A., Bergamasco, F. and Torsello, A., 2013. A Scale Independent Selection Process for 3D Object Recognition in Cluttered Scenes. *International Journal of Computer Vision*, [e-journal] 102(1-3). Available through: Universiti Tunku Abdul Rahman Library website <<http://library.utar.edu.my/>> [Accessed 17 August 2019].

Rusu, R. B., Marton, Z. C., Blodow, N. and Beetz, M., 2008. *Intelligent Autonomous Systems 10: IAS-10*. [e-book] IOS Press. Available at: Google Books <<https://books.google.com/>> [Accessed 18 April 2020].

Rusu, R. B., Blodow, N. and Beetz, M., 2009. *Fast Point Feature Histograms (FPFH) for 3D registration*. In: 2009 IEEE International Conference on Robotics and Automation. Kobe, Japan, 12-17 May 2009. IEEE.

- Rusu, R. B., Marton, Z. C., Blodow, N. and Beetz, M., 2008. *Aligning Point Cloud Views using Persistent Feature Histograms*. In: Proceedings of the 21st IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Nice, France, 22-26 September 2008.
- Rusu, R. B., Marton, Z. C., Blodow, N. and Beetz, M., 2008. *Learning Informative Point Classes for the Acquisition of Object Model Maps*. In: Proceedings of the 10th International Conference on Control, Automation, Robotics and Vision (ICARCV). Hanoi, Vietnam, 17-20 December 2008.
- Rusu, R. B., 2009. *Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments*. PhD. Technischen Universität München. Available at: <<http://mediatum.ub.tum.de/doc/800632/941254.pdf>> [Accessed 18 April 2020].
- Schnabel, R., Wahl, R. and Klein, R., 2007. Efficient RANSAC for Point-Cloud Shape Detection. *Computer Graphics Forum*, [e-journal] 26(2), pp. 214-226. 10.1111/j.1467-8659.2007.01016.x.
- Schnabel, R., Wahl, R. and Klein, R., 2007. *Algorithm of RANSAC to extract Shapes in the Point Cloud*. [image] Available at: <https://www.researchgate.net/publication/220505939_Efficient_RANSAC_for_point-cloud_shape_detection> [Accessed 17 August 2019].
- Sipiran, I. and Bustos, B., 2011. Harris 3D: a robust extension of the Harris operator for interest point detection on 3D meshes. *The Visual Computer*, [e-journal] 27(11), pp. 963-976. <https://doi.org/10.1007/s00371-011-0610-y>.
- Sun, Y. and Abidi, M. A., 2001. *Surface matching by 3D point's fingerprint*. In: Proceedings Eighth IEEE International Conference on Computer Vision. Vancouver, BC, Canada, 7-14 July 2001. IEEE.
- Sun, Y. and Abidi, M.A., 2001. (a) *Local Coordinate System*. (b) *Local Fingerprint of the Same Point from Different Views*. [image] Available at: <<https://ieeexplore.ieee.org/document/937634>> [Accessed 15 August 2019].
- Taati, B., Bondy, M., Jasiobedzki, P. and Greenspan, M., 2007. *Variable Dimensional Local Shape Descriptors for Object Recognition in Range Data*. In: 2007 IEEE 11th International Conference on Computer Vision. Rio de Janeiro, Brazil, 26 December 2007. IEEE.
- Tombari, F., Salti, S. and Stefano, L. D., 2010a. *Computer Vision – ECCV 2010*. [e-book] Springer, Berlin, Heidelberg. Available through: Universiti Tunku Abdul Rahman Library website <<http://library.utar.edu.my/>> [Accessed 18 April 2020].
- Tombari, F. and Stefano, L. D., 2010b. *Object Recognition in 3D Scenes with Occlusions and Clutter by Hough Voting*. In: 2010 Fourth Pacific-Rim Symposium on Image and Video Technology. Singapore, Singapore, 14-17 November 2010. IEEE.
- Tombari, F. and Stefano, L.D., 2012. Hough Voting for 3D Object Recognition under Occlusion and Clutter. *IPSN Transactions on Computer Vision and Applications*, [e-journal] 4, pp. 20-29. <https://doi.org/10.2197/ipsjtcva.4.20>.

Tuytelaars, T. and Mikolajczyk, K., 2007. Local Invariant Feature Detectors: A Survey. *Foundations and Trends® in Computer Graphics and Vision*, [online] Available at: <https://www.researchgate.net/publication/220427977_Local_Invariant_Feature_Detectors_A_Survey> [Accessed 29 March 2020].

Zhong, Y., 2009. Intrinsic shape signatures: *A shape descriptor for 3D object recognition*. In: IEEE (Institute of Electrical and Electronics Engineers), 2009 IEEE 12th International Conference on Computer Vision Workshops. Kyoto, Japan, 27 September-4 October 2009. IEEE.

Zhong, Y., 2009. *Sequence of Matching Two Points using Intrinsic Shape Signature*. [image] Available at: <<https://ieeexplore.ieee.org/document/5457637>> [Accessed 17 August 2019].