Autonomous Active Perception Framework for Large Object Recognition Tasks

A thesis submitted in partial fulfilment of the requirements

for the degree of

Master of Engineering (Research)

by

Thanh Long Vu

to

School of Mechanical and Mechatronic Engineering
Faculty of Engineering and Information Technology
University of Technology Sydney
NSW - 2007, Australia

July 2022

Certificate of Original Authorship

I, *Thanh Long Vu*, declare that this thesis is submitted in fulfilment of the requirements for the award of *Master of Engineering (Research)*, in the *Faculty of Engineering and IT* at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise referenced or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

This document has not been submitted for qualifications at any other academic institution.

This research is supported by the Australian Government Research Training Program.

Signed: Production Note: Signature removed prior to publication.

Date: 31/7/2022

Autonomous Active Perception Framework for Large Object Recognition Tasks

by

Thanh Long Vu

A thesis submitted in partial fulfilment of the requirements for the degree of Master of Engineering (Research)

Abstract

In recent decades, the technology development in hardware and software has stimulated robotics systems' implementation in various fields. Robots are most commonly deployed in dangerous or mundane working environments, significantly reducing accidents, injuries and casualties in the workforce. Ideally, robots would be designed to perform the tasks autonomously, conducting calculations and making decisions based on sensory data. However, although robotics research has continuously advanced throughout the last half a century, there are still many complicated tasks where robots cannot achieve full autonomy yet. In these scenarios, interaction from a human supervisor may be required to make control decisions either locally or remotely. The quality of the decisions made by the supervisor or operator depends heavily on the available sensory feedback from the robotics system, which helps the human perceive the environment where the robot is operating. Therefore, perception capabilities are required for all autonomous and semi-autonomous robotics systems to process and make sense of the received data, so the system or the operator can perform necessary actions.

This thesis focuses on developing an active perception framework for robots working remotely, where the human operator cannot directly perceive the surrounding environment. The specific modality of sensor data received from the remote system is coloured Three-Dimensional (3D) point cloud data obtained from sensing devices such as Light Detection and Ranging (LiDAR) or depth cameras. Additionally, this thesis investigates the practicality and benefits of utilising Virtual Reality (VR) as a tool to visualise the data obtained from a remote system.

This thesis firstly reviews multiple frontier detection algorithms for Two-Dimensional (2D) exploration, comparing these algorithms with other notable frontier detection algorithms. The algorithms are implemented to enable insight to be gained about the performance of the inspected algorithms compared with the other notable algorithms. A 3D interactive and active mapping framework for a mobile manipulator platform based on dynamic Gaussian Probabilistic Implicit Surfaces (GPIS) was investigated and implemented to validate its efficiency while simultaneously exploring and interacting with a large pile of objects. The framework is shown to perform near real-time map updates for a dynamically changing environment due to its probabilistic nature. Two perception systems are presented that employ the point cloud data in the framework to perform object detection, pose estimation and scene overlay annotation. Finally, a framework for sensor data visualisation in VR environments is presented, which acquires, transmits and renders real-time RGB-D sensor data with integrated automatic perception annotations in a VR environment. This enables various data modalities and limiting factors to be investigated and optimised to improve the subjective cognitive workload.

Experiments are conducted in both simulated and real-life scenarios. The two real robotic platforms used in the experiments are mobile manipulators. The first platform is composed of a 6 Degree of Freedom (DOF) manipulator and a commercial mobile base platform and was mainly used to validate exploration algorithms. The second mobile manipulator platform comprises a custom-designed mobile base and a 5 DOF manipulator, equipped with a camera system consisting of two RGB-D calibrated cameras and a real-time VR interface. This platform was used to conduct and investigate the optimal configuration for operator performance during collaborative autonomy tasks.

Contents

Declaration of Authorship i

Abstract ii

List of Figures viii

List of Tables xii

Glossary of Terms xv

1 Introduction 1

1.1 Background 2

1.2 Motivation 3

1.3 Scope 5

1.3.1 Aims 6

1.3.2 Objectives 7

1.4 Contributions 7

1.5 Methodology 8

1.5.1 Evaluation of Frontier-Based Exploration and Dynamic Mapping Algorithms for Autonomous Robotic Systems 8

1.5.2 Enhancing Human-Robot Collaboration through Processing and Presentation of 3D Sensor Data in VR 9

1.6 Publications 10

1.7 Thesis Outline 10

2 Review of Related Work 13

2.1 Mobile Robot Exploration and Mapping 13

2.1.1 Frontier-based Exploration 14

2.1.2 Localisation and Mapping 18

2.1.3 Active and Interactive Perception 19

2.2 3D Point Cloud Processing 21

2.2.1 Filtering Algorithms 23

2.2.2 Segmentation Algorithms 24

2.2.3 Clustering Methods 28

2.3 Virtual and Augmented Reality in Robotics . . . 30
2.3.1 Virtual and Augmented Applications in Training Programs . . . 31
2.3.2 Virtual and Augmented Reality Applications in Control Interfaces . 33
2.3.3 Dense 3D Data Transmission . . . 35
2.4 Summary . . . 36

3 Frontier Detection Algorithms **38**
3.1 Introduction . . . 38
3.2 Methodology . . . 39
3.2.1 Experiment 1 . . . 41
3.2.2 Experiment 2 . . . 42
3.2.3 Experiment 3 . . . 44
3.3 Experimental Results . . . 44
3.3.1 Discussion . . . 48
3.4 Conclusion . . . 51

4 Active and Interactive Mapping **52**
4.1 Introduction . . . 52
4.1.1 Active and Interactive Mapping Problem . . . 53
4.2 Methodology . . . 54
4.2.1 System Overview . . . 54
4.2.2 Experimental Setup . . . 56
4.2.3 Simulated Experiment . . . 57
4.2.3.1 Ablation Study . . . 57
4.2.3.2 Benchmark Test . . . 58
4.2.4 Real-life Experiment . . . 58
4.3 Experimental Result . . . 58
4.3.1 Simulated Experiment . . . 58
4.3.1.1 Ablation Study . . . 58
4.3.1.2 Benchmark Test . . . 59
4.3.2 Real-life Experiment . . . 60
4.3.2.1 Dynamic GPIS Accuracy . . . 60
4.3.2.2 Next Best View Test . . . 60
4.4 Conclusion . . . 62

5 Perception and Estimation **64**
5.1 Introduction . . . 64
5.1.1 Rectangular-shaped Object Pose Estimation . . . 65
5.1.2 Overlay Bounding Box Annotation . . . 67
5.2 Methodology . . . 68
5.2.1 Core Background Fundamentals . . . 68
5.2.1.1 Region Growing Segmentation . . . 68
5.2.1.2 Principal Component Analysis . . . 69
5.2.1.3 Rotation Averaging . . . 72
5.2.2 System 1: Rectangular-shaped Object Pose Estimation . . . 74

5.2.2.1 Filtering Strategies to Improve Object Detection in Noisy RGB-D Point Cloud Data . . . 74
5.2.2.2 Region Growing Segmentation using Normal Vectors 75
5.2.2.3 Rectangular Object Recognition and Pose Estimation using PCA . . . 75
5.2.2.4 Best Surface Selection for Rectangular Prism-Shaped Objects using Sigmoid Scores . . . 76
5.2.2.5 Rotation Averaging for Surface Pose Estimation 77
5.2.3 System 2: Overlay Bounding Box Annotation . . . 79
5.3 Experimental Result . . . 80
5.3.1 System 1: Rectangular-shaped Object Pose Estimation . . . 80
5.3.1.1 Comparative Study: System 1 and ICP-integrated System 81
5.3.1.2 Ratio Between Eigenvalue and Object Dimensions 83
5.3.1.3 Discussion . . . 85
5.3.2 System 2: Overlay Bounding Box Annotation . . . 86
5.3.2.1 Simulated Experiments . . . 86
5.3.3 System 2: Background Removal and Clustering Process . . . 87
5.3.3.1 Simulated Experiment . . . 88
5.3.3.2 Real-world Experiments . . . 90
5.4 Conclusion . . . 92

6 Using Virtual Reality for Collaborative Autonomy 93
6.1 Introduction . . . 93
6.1.1 User Study 1: The Influence of Sensor Data Visualisation Configuration on Users' Performance . . . 94
6.1.2 User Study 2: An Investigation of User Performance in Virtual Reality-based Annotation-assisted Remote Robot Control 94
6.2 Methodology . . . 95
6.2.1 The VR-ROS Framework For Collaboration Autonomy . . . 95
6.2.1.1 Hardware and High-level Overview . . . 95
6.2.1.2 Real-time Point Cloud Transmission for VR Rendering using Compressed Depth Images . . . 95
6.2.1.3 Point Cloud Reconstruction . . . 96
6.2.2 User Study 1: Experiment Setup . . . 99
6.2.2.1 Experiment Procedure . . . 101
6.2.2.2 Experimental Measurement 1: Objective Task Completion Time . . . 102
6.2.2.3 Experimental Measurement 2: Subjective Cognitive Workload . . . 102
6.2.3 User Study 2: Experimental Setup . . . 103
6.2.3.1 Hardware and High-level Overview . . . 103
6.2.3.2 Custom Robotic Manipulator . . . 104
6.2.3.3 Data Transmission and Point Cloud Rendering . . . 104
6.2.3.4 Experimental Procedure . . . 105
6.3 Experimental Result . . . 107

6.3.1 User Study 1 Results 107
6.3.1.1 Objective Task Completion Time 107
6.3.1.2 Subjective User Experience Results 108
6.3.1.3 Discussion 109
6.3.2 User Study 2 Results 110
6.3.2.1 Objective Task Completion Time 110
6.3.2.2 Objective Task Precision 111
6.3.2.3 Discussion 111
6.4 Conclusion 112

7 Conclusions **114**
7.1 Summary Of Contributions 115
7.1.1 Efficiency Evaluation of Frontier Detection Algorithms for Map Exploration 115
7.1.2 Validation of an Active and Interactive 3D Mapping Framework for Mobile Manipulator Platforms 115
7.1.3 Development of Perception Systems for Object Detection and Pose Estimation using Point Cloud Data 116
7.1.4 Enhancing Perception for Collaborative Autonomy: A Framework and User Studies for Sensor Data Visualisation in VR Environments 116
7.2 Discussion of Limitations 117
7.3 Future Work 118

Appendices **120**

A Human Ethics Application **120**

Bibliography **122**

List of Figures

1.1 Abseiling workers performing manual rock-scaling operations. 4

2.1 The rays that traverse the occupancy grid (grey cells are unknown space, and white cells are freespace). The dashed lines show the cells that Bresenham's line algorithm traverses when determining the cells that are on the line between a ray's endpoints. 16

2.2 (a) Frontier-Tracing Frontier Detection (FTFD); (b) Expanding-Wavefront Frontier Detection (EWFD). Thick red borders indicate the cells that are determined to be potential frontiers; dashed borders are the cells that are removed from the queue but are discarded before full evaluation. The blue cells are frontier cells from the previous timestep, which are also evaluated. The thick black lines denote the sensor Field of View (FOV) [1]. 18

2.3 Example of Point Cloud Library (PCL) applications [2]. (a) RangeImage display utilising PCL Visualisation (bottom) for a 3D data set (top). (b) PCL StaticalOutlierRemoval application. Left: Raw data. Middle: StaticalOutlierRemoval result. Right: The algorithm's rejected points. 22

2.4 Qualitative comparison of the filtering methods investigated by Moreno [3]. (a) - (d) The point cloud after the implementation of various filtering methods: (a) No filter, (b) Pass-through filter, (c) Voxel grid filter, (d) Approximate voxel grid filter. 24

2.5 GNG vs Voxel Grid Comparison. Top Left: Noisy model. Top Right: Original CAD model. Bottom left: filtered model using GNG method. Bottom right: filtered model using Voxel Grid [4]. 25

2.6 Segmentation of natural colour image. (a_n) Challenging problem initialisation. (b_n) Final result with the proposed algorithm in [5]. 25

2.7 Seeded region growing experimental results. (a) Artificial image with noise. (b) to (h) Results with different initial seeds sets [6]. 27

2.8 Unseeded region growing experimental results. (a)(c) Artificial images with different noise levels. (b)(d) Results of (a) and (c), respectively [7]. 27

2.9 Single-link vs Complete-Link on a data set containing two classes (1 And 2) connected by noisy patterns (*) [8]. 29

2.10 Results of shared nearest neighbor algorithm [9]. 30

2.11 Virtual Reality teleoperation. (a) Real-life implementation. (b) First-person view from inside VR [10]. 32

2.12 (a) A system based on the GAVRe2 framework. (b) First-person view from inside VR [11]. 33

2.13 The interfaces given to users. Left: 2D interface. Right: 3D VR interface [12]. 34
2.14 Virtual hand operating in VR, controlled through motion-capture glove [13]. 34

3.1 Freiburg lab environment with a preplanned trajectory for Experiment 1. . 41
3.2 The two map setups used in Experiment 2. (a) Small map top view. (b) Large map top view. (c) Small map perspective view. (d) Large map perspective view. (e) Scanned small map. (f) Scanned large map. 43
3.3 Experiment 2 setup. 44
3.4 Experiment 3 real-world platform: (a) The Neobotix MP-700 mobile robot [14]; (b) Robot and information diagram. 45
3.5 Experiment 3 setup. (a) Current real-world view of the robot. (b) Updating the map with the latest sensor observation (robot position indicated by frame annotation). (c) Newly detected frontiers (in red) in the constructed map. 45
3.6 Experiment 1 results. (a) The number of cells processed and evaluated by all algorithms. (b) The calculation time of the algorithms that consider the active area in relation to the maximum range of the simulated sensor (measured by the number of map cells). 46
3.7 Experiment 2 results for the small map case. (a) Average calculation time per iteration. (b) Calculation time as the small map is gradually explored for all algorithms. 46
3.8 Experiment 2 results for the large map case. (a) Average calculation time per iteration. (b) Calculation time as the large map is gradually explored for all algorithms. 47
3.9 Experiment 3 results. (a) Average calculation time per iteration with a horizontal line representing the updating rate of the map constructed by the Simultaneous Localisation and Mapping (SLAM) algorithm. (b) The calculation time as the map is explored for all algorithms. 47
3.10 Average calculation time in only the **first** iteration of each algorithm. . . . 48
3.11 Scanned map of Experiment 3. 50

4.1 The Active and Interactive Mapping cycle. The projection of GPIS mesh on the ground (red line) defines potential exploration segments, and the green bars indicate segments' information utilities. The Next Best View (NBV) is a red bar + arrow. Dark green dots are GPIS training points from the removed object [15]. 54
4.2 Active and interactive mapping framework overview. 55
4.3 An illustration of full formulation and single factor utilities. First, identify pile segments (red contour) and compute the utilities (green bars). Max utility gives NBV (red bar+arrow). (*b*) Shows samples (blue dots) from the manipulability annulus. (*c*) Shows heights (orange bars). Grey bars on the ground indicate the uncertainty for imaginary segments. 56
4.4 Three Gazebo simulation scenarios. 57
4.5 Comparison of task and map coverage for simulation case 2. 60
4.6 Real-life experiment: comparing map accuracy between ours, [16] and [17]. . 61

4.7 Real-life experiment, active and interactive mapping cycle. 62

5.1 System 1 overview and its demonstration. (*a*) System 1 overview. (*b*) Simulation of an arm-equipped robot detecting the best brick for picking from a pile. Chosen brick is labelled with RGB-axes. 66
5.2 System 2 overview. 67
5.3 Principal Component Analysis (PCA) reveals the natural distribution of normally distributed data [18]. Here eigenvectors are labeled as $\mathbf{u}_1$ and $\mathbf{u}_2$. The data mean is located at the centre of the surface. Standard deviations along each axis are the singular values $\lambda_1^{\frac{1}{2}}$ and $\lambda_2^{\frac{1}{2}}$. The ratio of length-to-height is $\lambda_1^{\frac{1}{2}} : \lambda_2^{\frac{1}{2}}$. 71
5.4 Rotation of a vector, $\mathbf{x}$ about the axis, $\mathbf{u}$ by an angle, ϕ [19]. 72
5.5 In unit 3-sphere manifold, quaternion $\mathbf{q}$ defines an angle, $\theta = \frac{1}{2}\phi$ with a unit quaternion, $\mathbf{q}_1$ [19]. 73
5.6 The five Experiment 1 scenarios. 82
5.7 Segmented point cloud results from Experiment 1. 83
5.8 Bounding boxes from the simulated experiments: (a), (b) individual objects experiment; (c), (d) multiple objects experiment with different cameras viewpoints. 86
5.9 System 2: Background removal and clustering process Gazebo simulated environment 88
5.10 System 2: Background removal and clustering process simulated environment view in RViz. 88
5.11 Results of the implemented ground removal and clustering processes. (a) The point cloud after ground plane removal through RANdom Sample And Consensus (RANSAC). (b) The point cloud after the removal of clusters below ground level and small clusters. (c) The position of clusters in (b) in the original point cloud. 89
5.12 System 2: Background removal and clustering process real-world experiments data set. (a) Dataset 1; (b) Dataset 2; (c) Dataset 3; (d) Dataset 4. 91
5.13 System 2: Background removal and clustering process real-world experiments result. (a)(b) Wall with minor roughness. (c)(d) Beach rock under sunlight. 91

6.1 Operator interacting with a remote robot in VR with the visual aid of processed point cloud data that is transmitted from RGB-D cameras. . . . 96
6.2 A flowchart of the lossless colour and depth image data compression and transmission process. 97
6.3 Point cloud reconstruction experimental results. (a) Raw point cloud data view in RViz. (b) Comparison of the reconstructed point cloud (white) and original point cloud (red). 98
6.4 Field of view comparison between depth and colour images. 99
6.5 Participant view in VR: (a) Participant view: colour point cloud (C-PC); (b) Participant view: mono-coloured point cloud (MC-PC); (c-d) Participant view: colour and depth image stream (C-D). 100

6.6 A participant assembling a puzzle while visualising it via RGB-D camera stream in a VR headset. Inset: The puzzle used in the experiment (left): Unassembled start configuration, (right) Successfully completed assembly. . 101
6.7 Hardware block diagram showing data passed from a robotic arm with cameras to a remote operator controlling the system via a VR rig that displays the virtual robot and the transmitted perception data. 103
6.8 Participant's view in VR with differing settings: (a) Setting 1 - Point cloud only; (b) Setting 2 - Point cloud and two image streams; (c) Setting 3 - Point cloud and annotations; (d) Setting 4 - Point cloud, two image streams, and annotations. 105
6.9 A participant manipulating the end-effector to the green area of the target while in VR. 106
6.10 Experimental results: (a) Completion time; (b)-(d) for statements 1 to 3 the participants' responses on the Likert scale. 108
6.11 Experimental results for each setting: (a) Completion time; (b) Precision quantified score; (c) Histogram of repetitions where participants missed the green target area. 111

List of Tables

2.1 Example iteration times for Naïve frontier detection if cell evaluations take 1ms, 1us, or 1ns. The first two rows are the 2D areas of the "FR-079 corridor" and "Freiburg campus" data sets, respectively. The second two rows are the full 3D volumes of the same data sets, [20]. 15

3.1 Properties and suggestions for using different frontier detection algorithms. Yes* is used to indicate where one algorithm performs markedly better than the others for a particular property. 49

4.1 Ablation analysis on Utility factors. 59
4.2 Benchmark Test 59

5.1 A comparison of the centroid estimation accuracy between the Iterative Closest Point (ICP)-integrated system and System 1. 84
5.2 A comparison of the orientation estimation accuracy between the ICP-integrated system and System 1. 84
5.3 A comparison of execution time (in seconds) between the ICP-integrated system and System 1. 84
5.4 A comparison of the brick's PCA ratio with size ratio. $(\frac{L}{H})^2$ refers to the length-to-height ratio squared, and λ_i refers to the i'th eigenvalue from PCA. 85
5.5 Bounding boxes centre and error values 87

6.1 Summary of experimental design information for the two user studies. . . . 99
6.2 Experimental trial visualisation settings. 100
6.3 Experimental visualisation settings (i.e., images, coloured point cloud (PC) and bounding box annotations) indicating which information is available "A" or not available "N/A" to participants. 105

Acronyms & Abbreviations

2D	Two-Dimensional
3D	Three-Dimensional
AR	Augmented Reality
BFS	Breadth-First Search
DOF	Degree of Freedom
EWFD	Expanding-Wavefront Frontier Detection
FFD	Fast Frontier Detection
FOV	Field of View
FTFD	Frontier-Tracing Frontier Detection
GDP	Gross Domestic Product
GP	Gaussian Process
GPIS	Gaussian Probabilistic Implicit Surfaces
GPOM	Gaussian Processes Occupancy Maps
HALO	High Access Localised Operations
ICP	Iterative Closest Point
IG	Information Gain
IoU	Intersection Over Union

LiDAR	Light Detection and Ranging
MBZIRC	Mohamed Bin Zayed International Robotics Challenge
NBV	Next Best View
PCA	Principal Component Analysis
PCL	Point Cloud Library
RANSAC	RANdom Sample And Consensus
ROS	Robot Operating System
SLAM	Simultaneous Localisation and Mapping
UTS	University of Technology, Sydney
VR	Virtual Reality
WFD	Wavefront Frontier Detector
WFD-INC	Incremental Wavefront Frontier Detector

Glossary of Terms

Autonomous	Without human intervention.
Active and passive perception	Active perception involves actively seeking and processing sensory information through exploration and interaction with the environment, while passive perception involves receiving sensory information without any active involvement on the part of the system.
Collaborative Autonomy	Refers to a mode of operation in which humans and robots work together as a team to achieve a common goal, with each entity contributing their unique strengths and abilities. This approach leverages the strengths of both humans and robots to enhance performance and efficiency beyond what either entity could achieve individually.
Human operator	A person who manipulates the robot's action at a remote location.
Human supervisor	A person who oversees the robot's operations and provides the robot with instructions when needed.
Manipulator	Robotic arm
Mobile base	A robot that exclusively navigates on horizontal planes.
Occupancy map	A discretised grid representing the environment is made up of cells with values that indicate the likelihood that a given point in space is obstructed.
Voxel	Volumetric Pixel represents a 3D cube-like volume in Euclidean space.

Chapter 1

Introduction

Perception is an essential component of automated robotic systems. Most existing robots utilise data received from sensors in one or more modalities, such as Two-Dimensional (2D) Light Detection and Ranging (LiDAR) scans, Three-Dimensional (3D) point clouds, colour or depth images. A research question studied over decades is how to perceive environments using the information collected by sensors effectively. With the continuing development of technology, sensors' performance and accessibility have improved, providing extensive and detailed data sets in various scenarios. The advancement of sensors' performance and availability of different sensor types present a need for new and more advanced techniques and methods to collect and process the data to obtain the desired information.

Over recent years, robots more frequently need to operate in real-world environments without prior knowledge and in the presence of dynamic objects. Thus, there has been an increased interest in active perception problems. The complicated tasks and diversity of working environments require the robot to actively relocate itself to gather information and complete the task. However, the role of a human is also important due to the inadequacies of autonomous robot systems. Humans act as supervisors and operators to make high-level decisions and collaborate with robots, which were developed to execute precise, simple, repetitive tasks. Since human collaboration is part of the robotic system, useful information and sensor data must be collected and presented to the human supervisor to ensure their perception of the environment is clear and accurate.

Therefore, further research is required to devise a functional, low-level automated active perception framework that communicates and collaborates with human supervisors to perform a task. The collaborative autonomy between humans and machines is another aspect of the framework that needs to be addressed.

This chapter introduces the developed perception framework. It commences with a background of the targeted topic of active perception and details research questions and issues. The other sections of this chapter describe the scope and the contributions and outline the contents of other chapters of this thesis.

1.1 Background

The mining industry is a cornerstone of the Australian economy. In 2010, Australia's mineral sector contributed up to 8% of the country's Gross Domestic Product (GDP). The sector is among the world's top five producers of several key mineral commodities. Additionally, the resources sector's annual export earnings are in the hundreds of billions of dollars, which is significantly higher than exports from both the agricultural and manufacturing sectors [21].

Mining sites require extensive risk management plans to minimise dangerous hazards, ensuring the safety of workers and others. Hazards in mining include ground and strata failure, fire or explosion, and falls. Despite the risk management plans and safety methods actively practised, there are recorded casualties and accidents annually. The Queensland government documented the mining industry safety performance over the last few years [22], which showed there is at least one casualty, approximately one hundred serious accidents and thousands of potentially serious incidents yearly. Due to the serious accidents that occurred frequently and unpredictably, machines have increasingly been implemented in the mining industry to address safety issues and improve productivity [23].

At the beginning of the 20th century, the mining industry began extensively using mechanical machines, replacing human and animal power with diesel and electric machines to improve productivity. As expectations rose and requirements became increasingly ambitious, the machines needed to progressively become larger and more powerful. However, due to practical limits, different approaches to improve the machines' performance, such as automation, were investigated. The development of technology, changes in the workforce, and escalating health and safety concerns made automation a strong contender [24].

An approach to implementing automation in the mining industry is to utilise robotics systems to perform mining activities. Throughout the last few decades, the automation of multiple mining processes has become possible. Numerous robotics systems can perform low-level operations independently, allowing human operators to focus on high-level control, supervision and management activities. Therefore, less human power and intervention are required, significantly reducing the potential of incidents while increasing the productivity of the mining activities.

Previous works on the implementation of different technologies and methods into semi-autonomous remote control mining systems include autonomous path planning [25][26], wireless communication [27], and navigation localisation [28]. In the past decade, the research has focused more on the autonomous navigation of mobile mining robot bases. Hence, to further optimise the system, it is important to consider the perceptive functionality of the robots.

This thesis is originally motivated by an ongoing project at the University of Technology, Sydney (UTS) to develop a semi-autonomous remote-control robotic mobile manipulator platform to perform rock-scaling on walls of open pits and underground mines. Currently, human workers are performing rock-scaling tasks at heights with the potential for falling rocks (Figure 1.1). They risk overexertion injuries, falling, crushing and even death. The project's ultimate objective is to substitute the manual labour at the rock face for a High Access Localised Operations (HALO) robot platform and move the people into remote supervisory roles. Thus, eliminating the exposure to risks associated with the specified hazardous environment.

1.2 Motivation

A user's performance and experience, while remotely operating or supervising a robot, can be enhanced by removing the low-level operations, thus enabling the user to perform tasks

Figure 1.1: Abseiling workers performing manual rock-scaling operations.

more efficiently and comfortably by solely interacting with high-level control decisions. To allow for such capabilities, the robotic system must execute the actions inherent in high-level commands autonomously and independently of human interactions. To achieve the required level of autonomy while working in an uncertain, unstructured, and potentially dynamic environment, a system must be capable of collecting sufficient sensor data required to perform the operations and process the data to obtain necessary information. Afterwards, the robotic system has to determine the subsequent actions based on the received information. More importantly, the system must maintain continuous communication with the remote operator, displaying the process status, data collected and decisions made over time. Since every decision the low-level automated system makes is based on the data collected, robotic systems need to ensure accurate and sufficient data are accumulated during the operations.

The procedure for obtaining and processing data varies among different types and models of sensors. The sensing modality also differs between different data types, such as point clouds and image streams, the focused features, the surrounding effects, etc. On the other hand, data transmissions, storage and visualisation are also limitations to be considered. For this thesis, the chosen data modality is mainly 3D data generated by RGB-D cameras or LiDAR sensors. Additionally, the placement and viewpoint of sensors can drastically affect the quality and quantity of the amount of useful information collected by the system.

Commercial RGB-D cameras are becoming progressively more affordable and better. Researchers and engineers are utilising them for numerous projects both for study purposes and industrial product development. However, most commercial RGB-D cameras suffer significant performance deficiency under challenging lighting and exposure conditions (i.e. outdoor areas, intense direct sunlight). Moreover, the further away objects are from the camera, the higher the error of the objects' readings. For example, to execute rock scaling tasks at the mine site, an autonomous perception system, which utilises RGB-D cameras, is required to operate in a harsh environment to target large-sized objects. Because of the limited FOV and reliable camera range, the camera must be relatively close to the targets while also being able to observe the entire object or at least its critical features. Therefore, an autonomous active perception framework is required to actively accumulate information over time about the object from a focused viewpoint. An alternative sensing device is LiDAR. Although not as affordable as RGB-D cameras, LiDAR devices have improved accuracy, FOV and accessibility over the years. Unlike cameras, where the placement of the sensors must be mobile to improve the coverage, LiDAR has a much higher accuracy even at a distance but suffer from a smaller vertical FOV. In general, depending on the model of LiDAR, it can have a large FOV on one axis and a small FOV on the other two or worse resolution due to the limited number of rays available. Therefore, LiDAR may not be required to be mobile, but its placement and viewpoint must be carefully selected.

1.3 Scope

This research aims to develop an active perception framework that can operate in collaboration with a human operator. Unlike a passive perception framework, an active perception framework can actively observe the environment to collect meaningful information, focusing on specific targets to assist both the robot's and the human operator's perceptions. The framework will be responsible for low-level automated processes while collaborating with the user, who will make high-level decisions. In addition to the semi-autonomous system, the research also investigates possible and intuitive approaches for data presentation and human interaction, which is important for the collaboration between the user and the robotic system.

1.3.1 Aims

The aims of this research project are:

Develop an autonomous active perception framework

In recent years, technology development and various sensing devices have facilitated researchers and engineers with multiple approaches to allow robotic systems to perceive the surrounding environment. Among sensing devices, RGB-D cameras and LiDAR sensors can provide the most information in the form of point clouds. However, RGB-D cameras have substantial limitations in their FOV, range or accuracy. While LiDAR does not have identical limitations, the data generated from LiDAR are relatively sparse compared to RGB-D cameras. Sensor limitations impede a robot's ability to perceive the surrounding environment. Therefore, if robots can replicate the human's ability to actively "look" at objects instead of passively "observe", they can target specific locations and optimise the information collection process to obtain the most meaningful data for the perception of the environment.

Improve the communication of what a robot perceives, so the visualisation is useful for a human supervisor

Although a robot will ideally be able to complete tasks autonomously without the help of humans, as the task complexity increases, or for safety, a remote human supervisor will often have to intervene to provide instruction or authorisation. The quality of the supervisor's decision is based entirely on their perception of the robot's surrounding environment. Due to limits in network bandwidth and human cognition, care must be taken when transferring this data and presenting it to the human supervisor. Virtual Reality (VR) has proven to be a successful way to immerse a human into a virtual environment, allowing the user to interact with the robot intuitively. This visualisation modality, combined with image processing and annotation techniques, can be utilised to ensure that the perceptual data on which the supervisor bases their decision is up-to-date, accurate, relevant and comprehensible.

Implementation of point cloud processing algorithms to analyse and highlight important information

As discussed in the previous aim, the user performance in collaboration with the robotic system heavily depends on the human's perception of the world surrounding the robot. Since humans do not see the world similarly to robots, human operators can easily be overwhelmed by the large amount of data obtained from sensors. This can be avoided by presenting only relevant information to the current task. Therefore, it is necessary for the robotics system to autonomously analyse the raw sensor data before transferring them to the user. Since this thesis mainly focuses on 3D point cloud data, implementing point cloud processing algorithms is required as part of the automated system to detect features in the scene that are useful for the user.

1.3.2 Objectives

The objectives of the research project are to:

- Implement and produce experimental results of multiple algorithms to explore an unknown environment
- Experimentally evaluate an active perception framework that autonomously manipulates the camera to explore and interact with a dynamic environment
- Implement point cloud processing algorithms into the framework to analyse and detect meaningful features for human collaboration
- Implement methods that enable a robot's perceptual information to be communicated and visualised by a human supervisor in VR
- Design systems for conducting the experiments in both simulated and real-life environments, and draw conclusions from the generated results

1.4 Contributions

This thesis reviewed and implemented multiple frontier detection algorithms for 2D exploration, comparing these algorithms with other notable frontier detection algorithms. The algorithms are implemented in both simulations and real-life experiments to provide insightful conclusions about the performance of the inspected algorithms compared with the other notable frontier detection algorithms. Additionally, a 3D interactive and active mapping framework for a mobile manipulator platform based on dynamic GPIS was implemented to validate its efficiency through simulated and real-life experiments. Two perception systems are presented that employ the point cloud data in the framework to perform object detection, pose estimation and scene overlay annotation. The first system was published and nominated for best paper at Australian Conference on Robotics and Automation (ACRA) 2019, while the second system contributed to the best student paper at ACRA 2022. Lastly, a framework for sensor data visualisation in VR environments is presented. User studies were conducted to determine the optimal configuration for human supervision.

1.5 Methodology

1.5.1 Evaluation of Frontier-Based Exploration and Dynamic Mapping Algorithms for Autonomous Robotic Systems

All autonomous robotic systems need to perceive their environment. Robotic systems require knowledge of the environment to perform tasks and navigate themselves around. Therefore, when encountering an area with no prior knowledge, it is essential for the robot to perform exploration. For most exploration algorithms, the environment map is created and saved as a two-dimensional grid map through the information obtained from a 2D LiDAR. Many robotics exploration approaches that utilise 2D LiDAR sensor data also use the concept of frontier cells. Frontier-based exploration approaches repeatedly detect frontiers and move towards them until there are no more frontiers, and the map is completed. The speed of exploration approaches that employ frontier cells can be enhanced by improving the frontier detection algorithms. However, the detection algorithms' performance is situational and dependent on the environment. Therefore, reviewing and experimenting with multiple frontier detection algorithms is necessary to develop insightful conclusions for each algorithm.

On the other hand, due to the complexity of tasks, the autonomous robotic system usually employs platforms that combine mobile bases with robotic arms to enhance the robot's movement capabilities. Due to the implementation of manipulators on mobile platforms, 3D maps of the environment are required. In addition, the working environment is not static for most real-world applications. Therefore, the mapping framework must incorporate the environment's dynamic characteristics and generate a 3D representation of the surroundings. Selected mapping frameworks need to be implemented and experimented with in different settings to determine the performance and efficiency of dynamic mapping algorithms.

1.5.2 Enhancing Human-Robot Collaboration through Processing and Presentation of 3D Sensor Data in VR

Human supervision is often required to make high-level decisions in collaboration with the automated low-level processes that are handled by the robot. However, raw 3D sensor data is often dense, noisy and incomprehensible, especially when shown to humans. Therefore, the 3D data obtained directly from sensors must be processed before being presented to humans. Various algorithms were created to filter out the noise, extract features, and perform object detection on raw data. By exploiting the existing perception and estimation algorithms, systems can be developed that first attempt to minimise the noise, simplify the raw data, and then perform feature extraction for specific applications.

Leveraging VR to represent the real world is advantageous because users can immerse themselves in the scene. Due to the large amounts of information in 3D sensor data, challenges such as appropriate point cloud processing, data transmission, and presentation have not yet been thoroughly addressed in the literature. Since the raw 3D point clouds cannot be transferred between the robot system and VR, other data modalities, which can be utilised to generate the point cloud, need to be considered. The problem of data presentation in VR requires user studies to be conducted to derive an optimal setting for data visualisation and to include perception and estimation algorithms to enhance the comprehension of sensor data.

1.6 Publications

- **Vu, T.L.**, Liu, L., Paul, G. and Vidal Calleja, T., 2019, December. Rectangular-shaped object recognition and pose estimation. In Australasian Conference on Robotics and Automation (ACRA).

- Liu, L., Fryc, S., Wu, L., **Vu, T.L.**, Paul, G. and Vidal-Calleja, T., 2021. Active and Interactive Mapping With Dynamic Gaussian Process Implicit Surfaces for Mobile Manipulators. IEEE Robotics and Automation Letters, 6(2), pp.3679-3686. `http://dx.doi.org/10.1109/LRA.2021.3061324`

- Quin, P., Nguyen, D.D.K., **Vu, T.L.**, Alempijevic, A. and Paul, G., 2021. Approaches for Efficiently Detecting Frontier Cells in Robotics Exploration. Frontiers in Robotics and AI, 8, p.616470. `http://dx.doi.org/10.3389/frobt.2021.616470`

- **Vu, T.L.**, Le, D.T., Nguyen, D.D.K., Sutjipto, S. and Paul, G., 2021. Investigating the Effect of Sensor Data Visualization Variances in Virtual Reality. In the 27th ACM Symposium on Virtual Reality Software and Technology (pp. 1-5).`https://dl.acm.org/doi/10.1145/3489849.3489877`

- **Vu, T.L.**, Nguyen, D.D.K., Sutjipto, S., Le, D.T. and Paul, G., 2022. Investigation of User Performance in Virtual Reality-based Annotation-assisted Remote Robot Control. In the 28th ACM Symposium on Virtual Reality Software and Technology (pp. 1-2). `https://dl.acm.org/doi/abs/10.1145/3562939.3565687`

- **Vu, T.L.**, Nguyen, D.D.K., Sutjipto, S., Le, D.T. and Paul, G., 2022. Investigation of Annotation-assisted User Performance in Virtual Reality-based Remote Robot Control. In Australasian Conference on Robotics and Automation (ACRA).

1.7 Thesis Outline

This chapter presented the research question and provided the background for the thesis. Chapter 2 investigates the past research and study covering various methods, approaches and experimental results of the related topics. Chapter 3 to 6 presents the research activities conducted within the scope of this thesis. The detailed outline of each chapter is as follows:

Chapter 2 presents the related works on the different problems of an active perception system. The chapter covers the past work addressing the three main issues of active perception systems: sensor data processing, robotics mobile manipulator exploration and mapping, and data sensor visualisation utilising VR and Augmented Reality (AR).

Chapter 3 investigates multiple approaches to enhance the speed of the mobile base exploration process. This chapter focuses on improving exploration activities by speeding up frontier detection algorithms for 2D maps. Several algorithms were studied and compared with other state-of-the-art frontier detection algorithms. The results from experiments conducted in both simulated and real-life environments are also outlined in this chapter.

Chapter 4 implements an interactive probabilistic mapping framework for a mobile manipulator picking objects from a pile. The framework actively maps a dynamic environment while interacting with the environment simultaneously. The framework creates 3D occupancy grid maps by utilising an RGB-D camera. Experiments conducted to validate and compare the framework performance in real-world environments are also outlined.

Chapter 5 presents two separate point cloud-based processing systems. The first system aims to detect rectangular-shaped surfaces and estimate the surfaces' poses for precise grasping tasks. The second system aims to detect objects above ground level and annotate them with bounding boxes to distinguish the objects from the background. Simulated and real-world experimental results are provided for both systems.

Chapter 6 presents the work done to integrate Robot Operating System (ROS) and VR systems. This chapter outlines the basic integration setup that consists of two computers, one with ROS and the other with the VR environment. Two problems with the integration were reviewed, and an approach has been proposed to address each problem. Additionally, a study on the effects of sensor data configuration and modalities on human users is conducted and presented in the chapter.

Chapter 7 summarises the research work that was done for this thesis. It discusses the shortcomings of the frameworks and their applications. The research conclusions are then drawn, and potential future research directions are suggested.

Appendix provides details of the ethics approval relating to the user studies.

Chapter 2

Review of Related Work

This chapter presents the literature review of the related works for the different approaches to the active perception problem. The literature is divided to focus on the three most important and related issues of active perception systems: mobile robotic manipulator exploration and mapping, sensor data processing and data sensor visualisation utilising VR and AR. Regarding the robotic exploration and mapping topic, the review focuses on algorithms to speed up the exploration process and 3D dynamic mapping. This chapter also reviews sensor data processing to investigate the methods that are used to process and analyse 3D point cloud data, such as filtering, segmentation and clustering. Finally, the topic of data visualisation in VR and AR explores existing implementations in robotics, with a particular focus on the problem of rapidly transmitting large perception data sets in real-time.

2.1 Mobile Robot Exploration and Mapping

The navigation system of mobile robots requires information about the surrounding environment. Depending on the applications, the information can be collected from various sensors and utilised to generate maps with useful attributes. The organisation of this section is as follows. Section 2.1 focuses on frontier-based exploration, including the concept of frontier-based exploration and algorithms to speed up the exploration process. Section

2.1.2 studies the tools to gradually build the map over time and navigate within the given map. This section considers the navigation problems in both 2D and 3D environments. Section 2.1.3 focuses on active and interactive perception, where a robot needs to perceive a dynamic environment that is changed due to the robot's actions or other reasons.

2.1.1 Frontier-based Exploration

Maps are frequently depicted using occupancy grids, or connected regions, in which each cell represents a specific location in space ([29]). A cell can be in one of three different states: "unknown", meaning there may or may not be an obstruction there; "known freespace" meaning there is no obstacle there and it can be safely passed through; or "known occupied". The terms "known freespace" and "known occupied" are also known as freespace and occupied, respectively. This status is typically expressed with a value between 0 and 1, showing the possibility that the cell contains an obstruction. The concept of frontiers was first proposed by [30]. Frontiers are freespace cells of an occupancy grid that have at least one neighbouring cell that has an unknown state ([30]). Since then, a variety of robot exploration techniques have used frontiers, whether they are used by single robots ([31–37]) or teams of multiple robots ([38–40]). Although there are probabilistic methods for frontier detection ([15]), these methods cannot be directly compared to the developed algorithms since they use different types of fundamental data structures.

The Naïve algorithm detects frontiers by evaluating every cell in the robot's map and determines whether it is freespace and has at least one unknown neighbour. If the cell is freespace and at least one of its neighbours is unknown, the cell is considered to be a frontier. Table 2.1 demonstrates the drawback of this algorithm in terms of speed when implemented in large real-life environments. The table presents the experimental results from the implementation of the Naïve algorithm to perform frontier detection in the Freiburg datasets ([41]) in both 2D and 3D scenarios. The significant difference in the total calculation time is shown, depending on how quickly the frontiers can be detected (8.79 seconds compared to 2.25 weeks as the number of cells being evaluated increased).

Previous works have proposed faster algorithms such as Wavefront Frontier Detector (WFD) ([42]). The WFD algorithm begins with a Breadth-First Search (BFS) from the

Table 2.1: Example iteration times for Naïve frontier detection if cell evaluations take 1ms, 1us, or 1ns. The first two rows are the 2D areas of the "FR-079 corridor" and "Freiburg campus" data sets, respectively. The second two rows are the full 3D volumes of the same data sets, [20].

Dimensions (m)	cm^2 Cells / cm^3 Voxels	1ms	1us	1ns
43.8 x 18.2	8.79×10^6	2.44hrs	8.79s	0.01s
292 x 167	4.88×10^8	5.6days	8.13min	0.49s
43.8 x 18.2 x 3.3	2.90×10^9	4.8wks	48.35min	2.90s
292 x 167 x 28	1.37×10^{12}	43.41yrs	2.25wks	22.76min

robot's current location through freespace cells until frontier cells are encountered. This algorithm is more effective than the Naïve algorithm because it only evaluates the map's freespace subset, neglecting occupied and unknown cells. Another algorithm, Incremental Wavefront Frontier Detector (WFD-INC) [43], has a similar principle as WFD, but BFS is restricted to only the active area of the latest scans received since the last time WFD-INC was called. An active area refers to the area that was modified by the latest scan. By limiting BFS to the active area, WFD-INC processing time is proportional to the number of voxels in the restricted area rather than the size of the complete map. Incremental-Parallel Frontier Detector [43] is similar to WFD-INC, but parallel computation is exploited to improve the algorithm speed further.

Fast Frontier Detection (FFD) only evaluates the cells of each individual scan, particularly the cells at the perimeters of the scan range, along which any new frontier must necessarily lie [42, 43]. A bounding box is built around the scanned area. Frontiers detected from the previous timestep that reside in the bounding box are checked to see whether they are still frontiers after FFD is performed on the latest scan. Compared to WFD, this algorithm is faster, but it must be repeated after every new scan from the sensor is obtained. Additionally, FFD is less effective in detecting frontiers at the maximum range of sensors. Divergence in laser points at extreme ranges causes the line tessellation method implemented by FFD, Bresenham's line algorithm (see Figure 2.1), to pass through unknown space, which means it does not detect all the frontiers on the way.

Another algorithm called OBB-based Frontier Detector, presented by Senarathne et al. ([44, 45]), efficiently updates the set of frontier cells by keeping track of the cells being updated and their previous states. This algorithm suffers a drawback where cells are checked unnecessarily if the edges of the new observation have many updated cells.

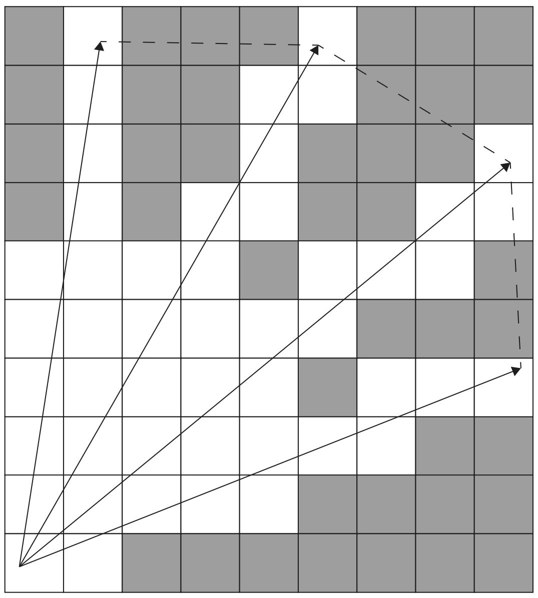


Figure 2.1: The rays that traverse the occupancy grid (grey cells are unknown space, and white cells are freespace). The dashed lines show the cells that Bresenham's line algorithm traverses when determining the cells that are on the line between a ray's endpoints.

A frontier detection algorithm introduced in [46] employs the idea behind a randomly exploring tree. A tree is constructed from the robot's current position to locate the frontiers in the map by using customised branching and selection rules. The robot passes through the area when a set of frontier cells is detected. Afterwards, the branches on the tree are removed to free up memory and enhance the speed of the calculation process. [47] proposed an algorithm to perform frontier detection utilising multiple robots. Each robot performs frontier detection and creates distinct local maps. By combining the information from all local maps, a global map can be constructed with all the frontiers detected.

The Naïve Active Area frontier detection (NaïveAA) algorithm was introduced in [48]. In this algorithm, only cells that are in the active area need to be assessed to see if they have changed into a frontier or are no longer a frontier, as per NaïveAA algorithm's principle. Thus, NaïveAA iterates through each cell, c, in the active area, A_t, of any scans performed since the last frontier detection step, including the immediately adjacent cells to the scanned areas. The list of frontier cells at time t, F_t, is updated to include cells that are new frontiers while subtracting cells that are no longer frontiers. Then, to discover which frontier cells are linked in $O(|F|)$ time, Kosaraju's series of Depth First Searches

(KOSARAJU_DFS) [49] is executed.

An additional algorithm called EWFD with an initial iteration process that is comparable to WFD is also suggested by [48]. Freespace cells are analysed using BFS to determine whether or not they are frontiers. A cell is added to the frontier set if it is next to an unknown cell. The adjacent cells of freespace cells are added to the queue. Each cell that BFS visits is labelled as visited. After the initial iteration, this labelling is left in place. Once a set of frontiers exists, an additional step is included in subsequent iterations of EWFD. The set of boundaries from the previous timestep that is located inside the active area of the most recent scan is determined before the start of BFS. These cells are added to BFS queue and labelled as "unvisited." Freespace cells are now evaluated as they are removed from the queue, considering their frontier status. They are removed from the frontier set if the previous timestep's frontier set includes them as frontiers, but the current timestep does not. The set of connected frontier groups is obtained using Kosaraju's connectivity algorithm [49] once the set of frontiers has been updated.

FTFD exploits the border of the current observation that has a high possibility of intersecting with existing frontiers [1]. The frontiers that reside in the observed area and the endpoints of sensor rays are used as initial points to start searching for new frontiers, which are part of the latest observation's perimeter. The task is to update the frontier set from the previous timestep, F_{t-1} to incorporate the information from the observation made at t, O_t. The set of frontiers at t, F_t is the updated F_{t-1}. Initially, F_t is equal to F_{t-1}. The active area frontiers, F_{aa} is a subset of the frontier cells in F_{t-1} that lie inside A_t. BFS is initialised with a queue consisting of F_{aa} and cells by the endpoints of the rays used to integrate the latest scan into the map. A cell, c is removed from the queue at each iteration of BFS. If c is an unvisited frontier, then it is added to F_t, and the adjacent freespace cells which belong to the map's active area are included in the queue. If c is not a frontier at t, but was previously a frontier at $t-1$, then it is removed from F_t. If c is occupied, its neighbouring freespace cells will be added to the queue. The resulting pattern of cells that is evaluated is shown in Figure 2.2(a). This pattern for FTFD is compared to that of EWFD in Figure 2.2(b).

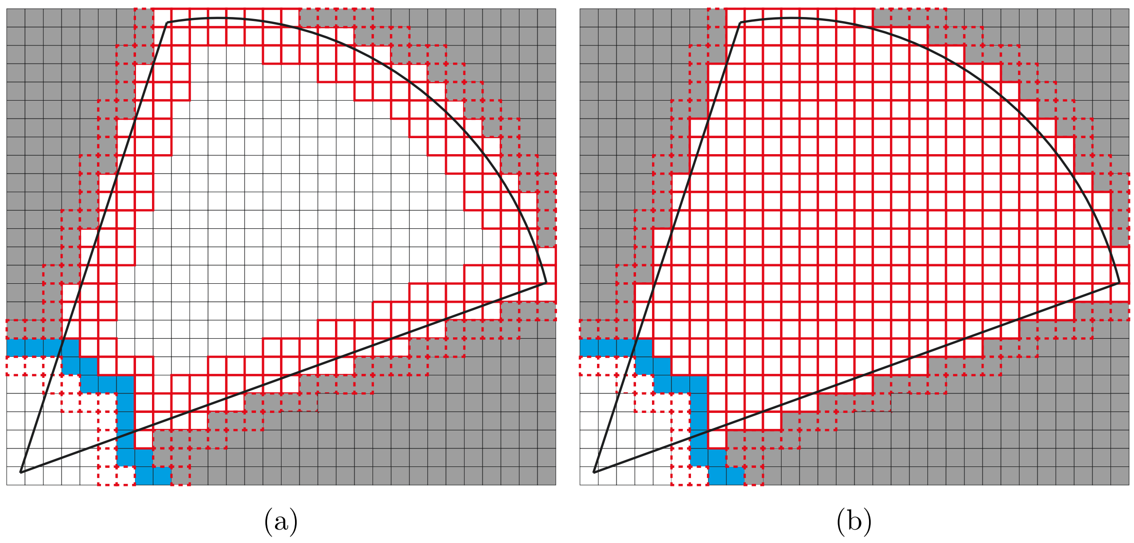


Figure 2.2: (a) FTFD; (b) EWFD. Thick red borders indicate the cells that are determined to be potential frontiers; dashed borders are the cells that are removed from the queue but are discarded before full evaluation. The blue cells are frontier cells from the previous timestep, which are also evaluated. The thick black lines denote the sensor FOV [1].

2.1.2 Localisation and Mapping

SLAM endows mobile robots with the ability to operate in an unknown environment autonomously. The SLAM problem raises the question of whether the robot can gradually build up information about the environment without prior knowledge while aware of its location within the environment [50]. Researchers have formulated the SLAM problem, and numerous solutions have been proposed practically and theoretically.

The combination of 2D LiDAR and SLAM is an efficient way to create a map of the working environment. The completed map is crucial for the navigation system of mobile robots to execute map-based localisation tasks [51]. However, the environment map obtained from SLAM and 2D LiDAR can be faulty due to environmental conditions, including loop closure, glass and transparent material. The error causes the loop closure problem when the mobile robot's relative pose is computed using scan-to-scan matching. The errors quickly accumulate over time and, eventually, prevent the SLAM algorithm from correctly connecting the loops in the map. The work in [52] presented a new approach addressing the loop closure problem of SLAM, which is competitive with well-known algorithms. The paper proposed a system that treats loop closure as a pose optimisation problem. Jiang [53] proposed an approach for transparent environment map building using 2D LiDAR. The proposed method implemented neural networks to classify glass objects and build a glass-confidence map. 3D localisation is also desirable, especially for robotic systems that require the robot to operate at different heights in the environment, such as mobile manipulators or drones. In these scenarios, 3D LiDAR and RGB-D cameras are preferred because they provide information in the form of 3D data. Fallon proposed a 3D localisation algorithm for indoor environments using an RGB-D camera [54]. The introduced algorithm generates a 3D model of the environment with large planes extracted from several consecutive frames captured by the camera. The particle filter then utilises the generated model for 3D localisation.

Navigating with a mobile robotic system given a map of an environment requires the robot to have the ability to localise, and recognise its current position and orientation within the map. Monte Carlo localisation (MCL) is a probabilistic approach to the localisation problem [55]. In the first phase of MCL, a set of random particles is generated, presenting the possible robot poses. In the second phase, for each time step, the motion model is applied to all particles to predict the current pose of each particle. Based on sensor data readings, particle filter algorithms are then implemented to eliminate particles with low likelihood. The two phases of MCL are repeated recursively. Adaptive Monte Carlo localisation (AMCL) is a variant of MCL commonly utilised in robotic navigation systems through ROS. The AMCL variants differ from the original algorithms because AMCL utilises the Kullback–Leibler distance (KLD) sampling to bound the approximation error generated by the particle filter of MCL [56]. By implementing KLD-sampling to adapt the estimated size of sample sets, the efficiency of the particle filter in AMCL is increased.

2.1.3 Active and Interactive Perception

The representation of the map is crucial for mobile manipulators interacting with dynamic environments. The mapping representation must efficiently reflect the changing environment. KinectFusion [57] is a mapping system that utilises RGB-D cameras. The map generated is volumetrically built as a Truncated Signed Distance Function (TSDF) data structure. The transient noise is smoothed out by averaging the information of each voxel over time. Therefore, the model is capable of handling dynamic environments. However, significant topological changes are only addressed after a delay. The 3D world can also be represented by occupancy maps, such as Octomap [20]. These maps present each voxel in space by its probability of occupancy. Octomap was implemented to create a robust multi-stage pipeline for bin-picking applications [17]. However, the resolution and precision of the Octomap are compromised because structural correlations between neighbouring cells are not addressed. Also, it is difficult to reason about object shape or perform detection [58]. Gaussian Processes Occupancy Maps (GPOM) is a method of combining Gaussian Process (GP) with occupancy mapping that is probabilistic and continuous but incurs a large computational complexity. Hilbert Maps utilises kernel approximation methods to form a faster and simpler occupancy map type [59]. The work was then extended to handle dynamic obstacles and model learning through regression in [60].

Despite all of the aforementioned benefits, GPIS exhibits $O(n^3)$ computational complexity, which limits its use [61, 62]. An online GPIS solution was described by Lee et al. [16] that incrementally combines consecutive scans from diverse views and saves data in small clusters for quick parallel processing. However, this approach does not address dynamic environments because once an object has been mapped, it stays in the GPIS data even after it has been physically removed from the scene. Here, the fundamental perspective of GP is implemented as a tool for instantaneous probabilistic conditioning of the immediate training data [63]. The map is demonstrated to be instantly updated in the pertinent local regions if a system is created to recognise and filter out training points that represent the removed object.

Other dynamic mapping research has concentrated on finding and monitoring moving elements in static settings [64] [65] or removing dynamic elements from the environment [66] to create accurate static maps.

An Information Gain (IG) metric can be used to determine the best course of action in robotic exploration. Applications for mapping and exploration frequently use occupancy maps and Octomaps because of their probabilistic occupancy representation [67, 68]. Recently, GP has also been employed for IG-based exploration tasks, such as mutual information maximisation [69], conditional entropy reduction [70], and entropy reduction [71].

Using restricted Variational Sparse GP and online kernel learning, [72] proposed an effective method for active shape modelling that maintains the precision of the reconstruction.

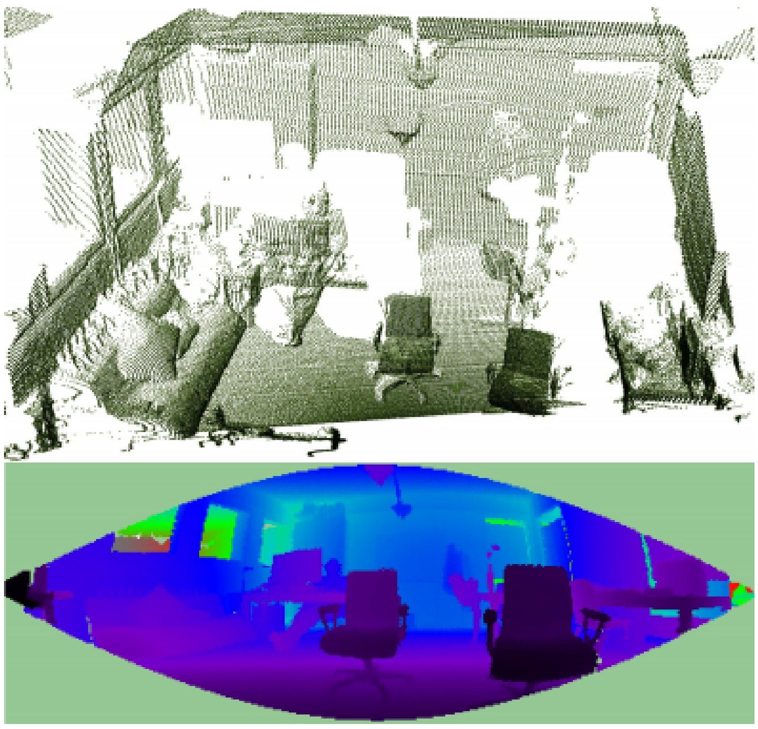
Another IG metric is the entropy gradient [73, 74]. The advantage is that it pulls the robot firmly towards the boundary between the explored and unknown areas by creating a rough gradient formulation for discretised 3D occupancy maps. Later, Jadidi et al. [69] produced continuous gradient frontier maps using a 2D GPOM variation, but an analytical expression was omitted because their map definition was not straightforward. The NBV candidates are identified using the gradient map, but they did not evaluate NBV utility.

Shape detection, classification, and validation in interactive applications have used GPIS with built-in surface normals [58]. A robotic arm fitted with tactile sensors is used by the GPIS-based architecture described in [75] to explore and map the environment. [70] proposed an approach to optimise a manipulator trajectory that is reconstruction-aware to accurately estimate the object's 3D shape for pick-and-place operations.

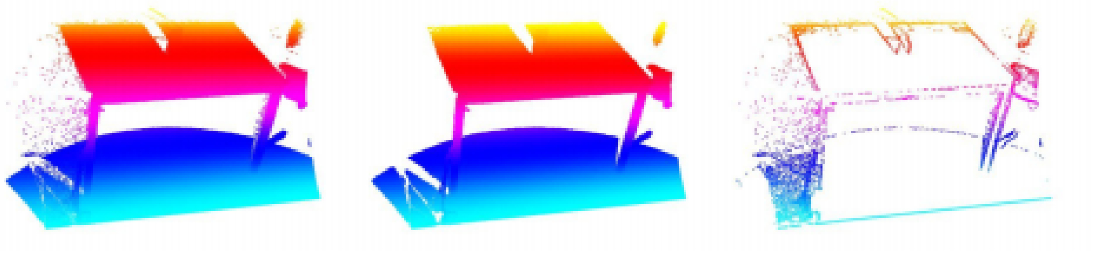
2.2 3D Point Cloud Processing

Point Cloud Library (PCL) is a library introduced in 2011 that includes algorithms for 3D point cloud processing [2]. The library is still one of the most widely known and implemented libraries when working with the 3D point cloud. Figure 2.3 presents three implementation examples of the library's built-in algorithms. The library's original algorithms are often used as a benchmark for validating the results of other research. More recent works that implemented functions of the PCL include adopting the voxel grid filter module to ensure evenly distributed input data by Zhang et al.[76]. Whelan et al. developed the Kintinuous extension of the Microsoft Kinect Fusion algorithm based on the PCL variant [77]. The author later published another paper that utilised PCL's colour integration module to the Kintinuous extension to create 3D mesh-based maps [78].

The organisation of this section is as follows. Section 2.2.1 focuses on the filtering problem when engaging with a 3D point cloud. The importance of the filtering process is described, and then several filtering algorithms and their related works are examined. Section 2.2.2 focuses on the 3D point cloud segmentation problem. Similar to the previous section, the



(a)



(b)

FIGURE 2.3: Example of PCL applications [2]. (a) RangeImage display utilising PCL Visualisation (bottom) for a 3D data set (top). (b) PCL StaticalOutlierRemoval application. Left: Raw data. Middle: StaticalOutlierRemoval result. Right: The algorithm's rejected points.

importance of the segmentation process is discussed, followed by the concepts, ideas and basic algorithms that address the segmentation problem. Related works on the variants of the basic algorithms are also inspected. Section 2.2.3 focuses on the data clustering problem in general. The data clustering problem is a multi-disciplinary topic. Therefore, the early algorithms were not specifically designed for the computer vision field. However, these algorithms are later implemented to solve precisely the 3D point cloud clustering problems. Section 2.2.3 looks into the original algorithms for data clustering and their variants.

2.2.1 Filtering Algorithms

In robotic perception, with extremely large datasets and limited processing time, it is necessary to pre-process the input data to minimise the computational time. It is ideal that the sensor data can be processed before the next set of data arrives. Although the robotic systems might not require the processed sensor information for every data set, it is optimal to have the data handled and ready when needed. Over the past two decades, computing devices have developed significantly, improving both the processing power and available memory size. However, the typical RGB-D camera sensor has a resolution of up to 1920x1080 and operates at 30Hz. The camera produces a point cloud with over a million 3D coloured points, updated 30 times every second. Therefore, filtering algorithms to optimise the initial point cloud are needed to eliminate redundant information while maintaining valuable dataset features.

Moreno et al. published a comparison of multiple well-known 3D point cloud filtering algorithms [3]. The work presented was interested in the performance of three filtering algorithms in real-time video streaming. The final result concludes that, individually, none of the algorithms was able to produce a close to real-time performance. The filtering algorithms considered include pass-through filter, voxel grid filter and approximate voxel grid filter (Figure 2.4).

Similar to Moreno's work, Dziubich also investigated different algorithms' efficiency as a pre-processing step for a system aimed to process and stream point cloud [79]. A previous publication proposed an improved pass-through filter algorithm with further optimisation. This optimised pass-through filter processing speed can be enhanced more than two times compared to the original PCL pass-through filter algorithm [80]. Voxel grid filter was a noise removal tool for 3D point cloud data. It was utilised to evaluate and compare more recent algorithms such as the Growing Neutral Gas (GNG) noise filtering network [4], as shown in Figure 2.5. With multiple filtering layers, it is possible to focus on the object of interest and eliminate all the unnecessary surrounding data. However, this effect and the filtering layers depend mainly on the environment and unique features of the targets.

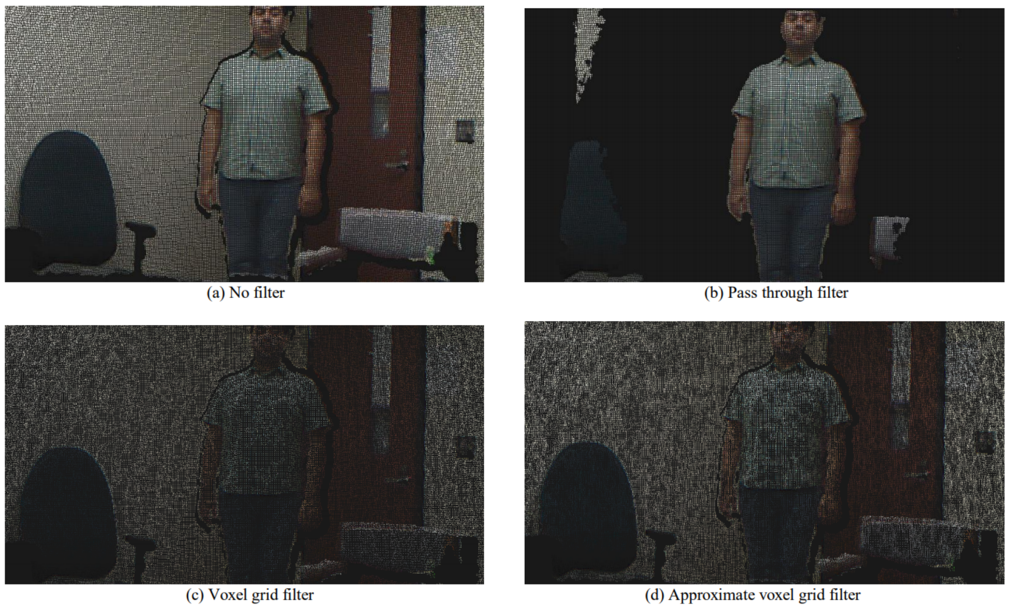


Figure 2.4: Qualitative comparison of the filtering methods investigated by Moreno [3]. (a) - (d) The point cloud after the implementation of various filtering methods: (a) No filter, (b) Pass-through filter, (c) Voxel grid filter, (d) Approximate voxel grid filter.

Three layers of filtering consisted of two pass-through filters, and one voxel grid filter was utilised in [81] to obtain a sparse 3D point cloud of specific colour and distance. Existing 3D point cloud filtering algorithms were reviewed and categorised into seven groups based on their traits in 2017 [82]. Han et al. also proposed an algorithm for 3D point cloud filtering inspired by the 2D guided image filter technique [83].

2.2.2 Segmentation Algorithms

Point cloud segmentation is a fundamental step in 3D point cloud processing [84]. The segmentation algorithms aim to classify the point cloud data into multiple homogeneous regions. Each represents a set of points with similar attributes. The set of points is isolated from the remaining points in the dataset and can be treated as an individual point cloud. The segmented and analysed point cloud is the prerequisite for other processes, such as object recognition, classification, and template matching, where each separate region is the input. Previous work demonstrated a segmentation process followed by a template matching problem [5]. The paper presents an algorithm to segment a rigid object in a 2D image using region-based segmentation and estimating the object's 3D pose. The pose

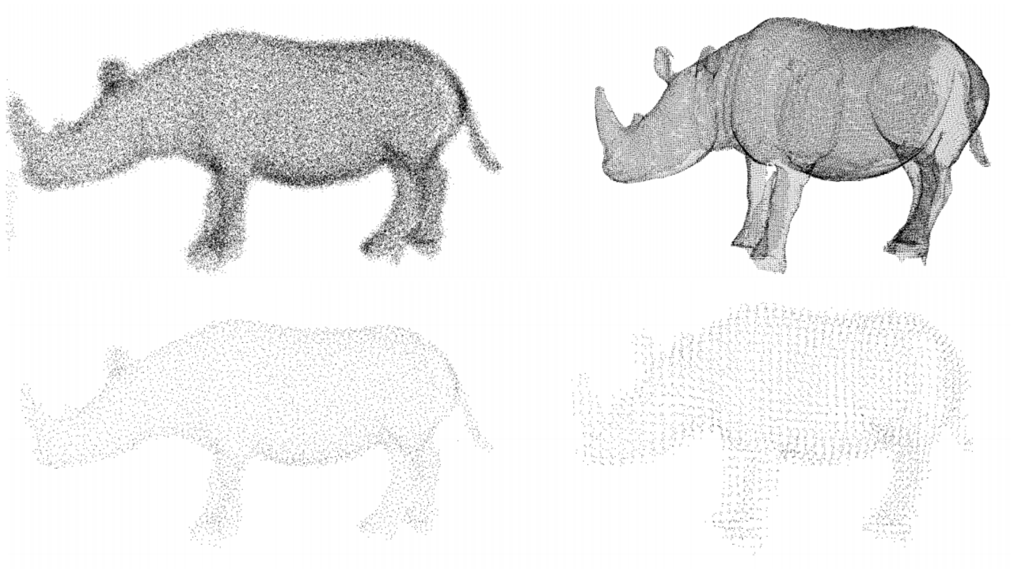


FIGURE 2.5: GNG vs Voxel Grid Comparison. Top Left: Noisy model. Top Right: Original CAD model. Bottom left: filtered model using GNG method. Bottom right: filtered model using Voxel Grid [4].

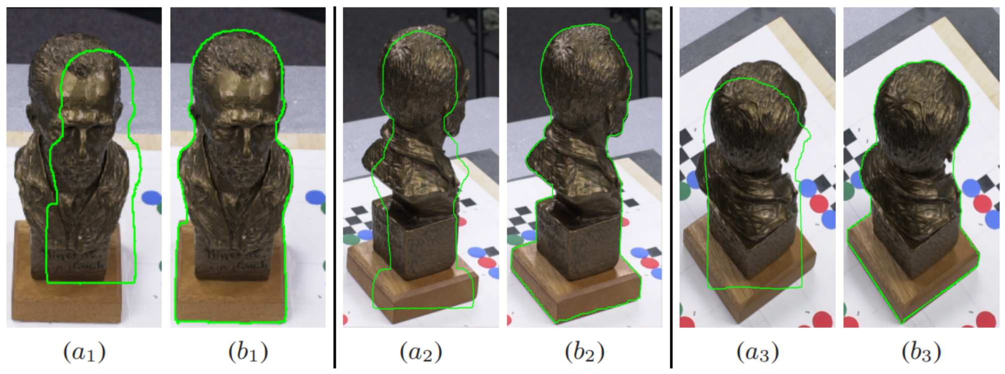


FIGURE 2.6: Segmentation of natural colour image. (a_n) Challenging problem initialisation. (b_n) Final result with the proposed algorithm in [5].

was calculated using prior knowledge of the target object's 3D model. By combining 2D and 3D information, the presented algorithm was robust to noise, occlusions and missing information (Figure 2.6).

Oehler proposed a coarse-to-fine point cloud segmentation algorithm [85]. The author's method executes surface segmentation of the scene on multiple resolutions, starting from the coarsest data. The final result is obtained by refining and merging segments from coarser resolutions. Trevor et al. [86] introduced an algorithm for segmentation targeting specifically organised point clouds. The paper also presented an edge refinement algorithm and discussed the tabletop object segmentation application of the proposed algorithm.

Edge-based segmentation algorithms focus on the boundaries of multiple regions in an image. Iannizzotto et al. presented a 2D edge-based segmentation process utilising active contour [87]. The active contour proposed is fast due to the low computation complexity. The result from the process introduced is a closed chain of points which is an advantage for the subsequent representation process. Patil implemented the edge-based segmentation module created by PCL to extract the edges of two halves of welding materials [88]. The shared edge of the halves is detected through 3D point clouds by utilising a variant of Canny Edge Detection.

Region-based segmentation algorithms examine the neighbours of a point. The nearby points are combined if they have similar properties or separate when dissimilarities are observed. Region-based segmentation algorithms are categorised as seeded and unseeded. The main difference between the two algorithms is the required inputs. Seeded algorithms expect a set of points, while unseeded algorithms only request the condition for subdivision. Seeded region growing was introduced by Adams et al. [6]. The original algorithm was designed for grey-scale images (Figure 2.7) but can be extended to colour or point cloud data. This algorithm is robust and independent of tuning parameters but heavily affected by the selection of the initial seeds. The unseeded region growing was designed to address the drawbacks of the seeded algorithm. Unseeded region growing [7] is robust and independent of tuning parameters designed for 3D segmentation and does not require the initial set of seeds (Figure 2.8). However, the unseeded region growing algorithm was affected mainly by the noise in the input data. Multiple layers of noise filtering can be implemented to negate the disadvantage of this region growing algorithm. Xu presented a robust region growing segmentation algorithm for planar surfaces with distinctive geometric features compared to surrounding objects [89]. The mentioned method combines multiple methods, including voxel structure, region growing strategy and robust principal component analysis (RPCA).

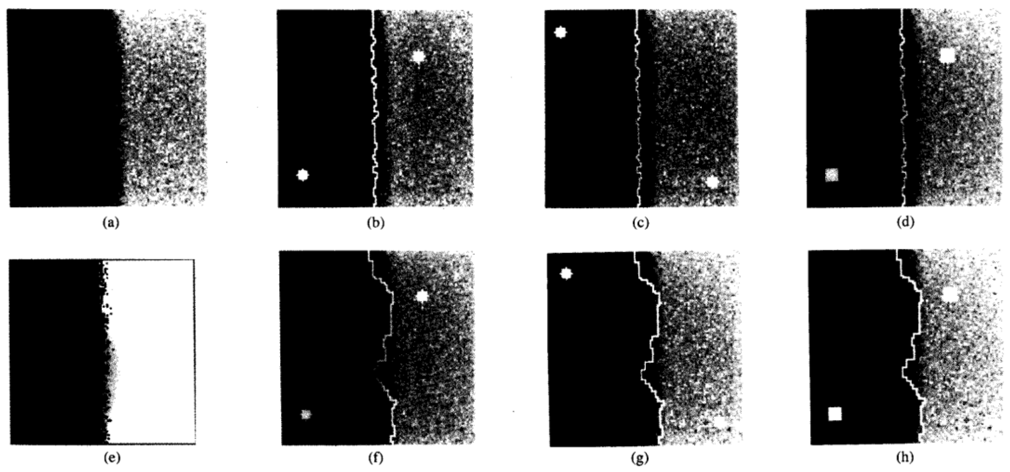


FIGURE 2.7: Seeded region growing experimental results. (a) Artificial image with noise. (b) to (h) Results with different initial seeds sets [6].

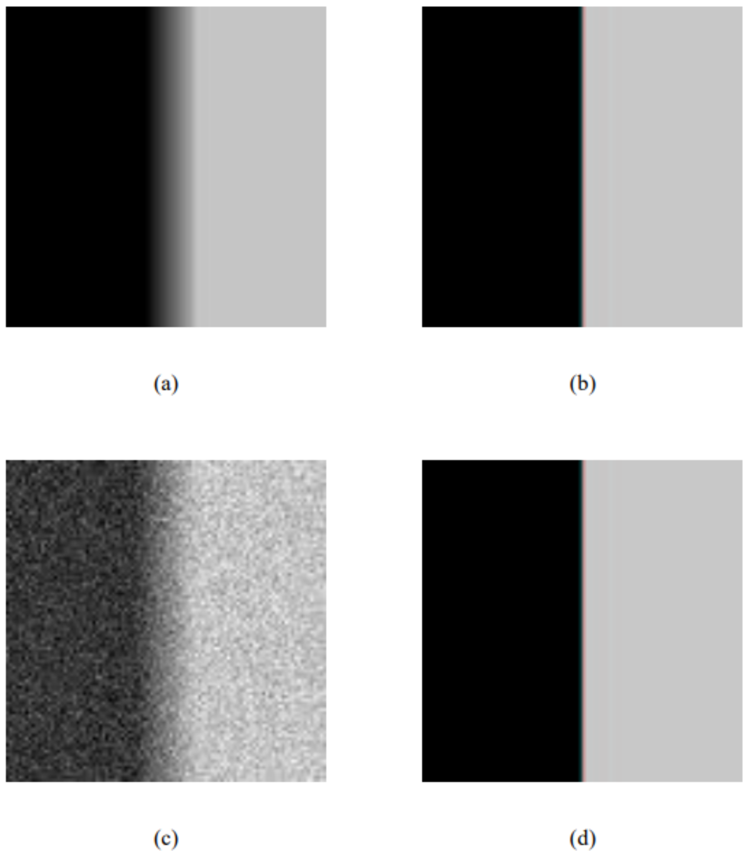


FIGURE 2.8: Unseeded region growing experimental results. (a)(c) Artificial images with different noise levels. (b)(d) Results of (a) and (c), respectively [7].

2.2.3 Clustering Methods

Clustering is the unsupervised classification process that aims to distribute patterns into groups or clusters [8]. Clustering is one of the preliminary steps often encountered in data analysis used by researchers in many disciplines. Das et al. [90] combined ground plane segmentation with a clustering process to create a variant of the Normal Distributions Transform (NDT) scan registration algorithm. Das et al. proposed a segmented greedy cluster NDT (SGC-NDT) variant, which segments the ground and clusters the remaining data before performing NDT to guarantee the convergence of the optimisation function. Tazir utilised a clustering algorithm to represent a cluster of points of the same surface as one point [91]. This one point is then provided as an input for the ICP matching process. The proposed algorithm aims to tackle the difficulty of working with multiple 3D point cloud sensor sources, each with a different resolution. Paul et al. clustered a generated grid-based point cloud based upon the grid's normal vector to locate the normal dissimilarity of a surface [37]. The grid-based point cloud was created by triangulating a 3D point cloud received from an RGB-D camera. Paul et al. identify the normal dissimilarities on the surface of steel bridge structures and designates them as possible locations of rivets. The clustering step was a back-end solution to detect voxels containing rivets, which the previous process might have missed or filtered in a steel structural environment.

Hierarchical clustering algorithms generate a tree diagram representing the grouped patterns and the similarity levels, which determine the grouping condition of patterns [8]. The two fundamental hierarchical clustering algorithms are single-link and complete-link. The basic concept of the single-link clustering algorithm considers all patterns as vertices of a graph [92]. If the distance between any two vertices is less than a user-defined threshold (Δ) then the two vertices are connected with an edge. The clusters with similarity level Δ are groups of connected patterns. The limitation of the single-link algorithm was due to the method by which the distance between two clusters is computed. The algorithm regards the minimum distance between any two points belonging to different clusters to be the distance between clusters. This results in the chaining effect in which two random clusters can be considered as one cluster if a string of noisy data exists at the centre, acting as the connection (Figure 2.9). The complete-link clustering algorithm was developed to form

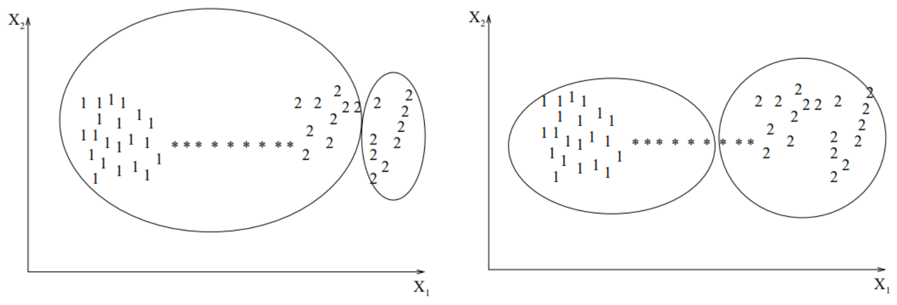


Figure 2.9: Single-link vs Complete-Link on a data set containing two classes (1 And 2) connected by noisy patterns (*) [8].

compact clusters that avoid the extremes of the single-link algorithm. The work presented by Hansen et al. [93] points out and discusses the difference between the two algorithms. The complete-link algorithm differs from the single-link one since the distance between 2 clusters is the maximum distance between any two points, each belonging to a distinct cluster, as opposed to the minimum distance in the case of the single-link algorithm. Due to how the distance between clusters is calculated, the complete-link algorithm results in overly separated clusters, which is undesirable. However, overall, the complete-link algorithm is more likely to produce useful clusters compared to the single-link algorithm [8].

Partitional clustering algorithms have a relatively low computational cost, which makes these algorithms advantageous when processing large datasets. The partitional clustering algorithms generate clusters through the optimisation process of a criterion function. The most common and simple algorithm is the k-means algorithm, which utilises the squared error criterion[8]. In this algorithm, k initial patterns are selected as centres for k clusters. All the remaining patterns are then distributed to their closest cluster. After all the points have been allocated, the centre of each cluster is recomputed. Even if the criterion function does not converge, a new set of k cluster centres can be selected before repeating the process. Zhao et al. evaluated the performance of partitional algorithms on large document datasets compared to agglomerative clustering algorithms [94]. The paper also described a variant of the original k-means algorithm, which performs k-way partitioning via repeated bisections. This variant obtains k clusters by performing the 2-way partition clustering $k-1$ times.

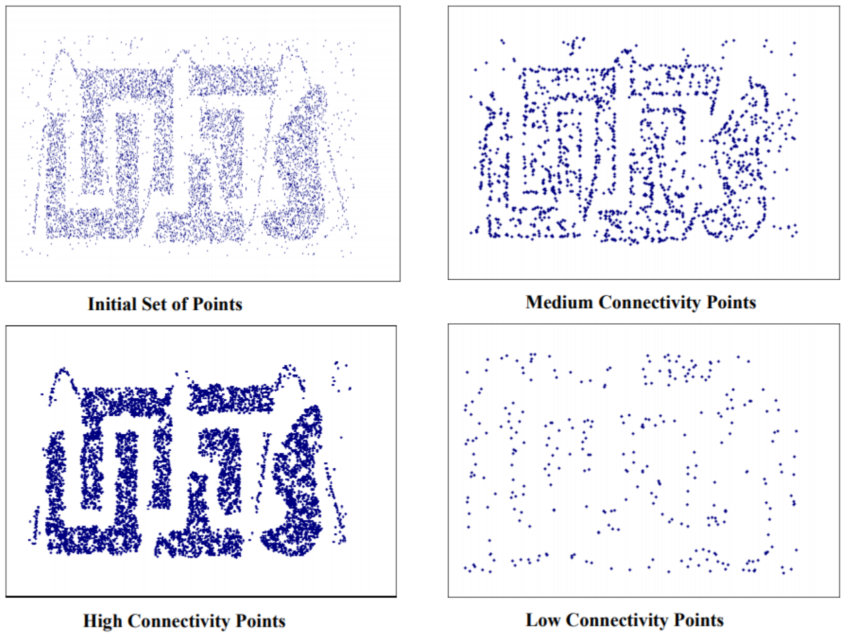


FIGURE 2.10: Results of shared nearest neighbor algorithm [9].

Nearest neighbour clustering algorithms assign a pattern to the same cluster of its nearest neighbour if the distance to the nearest neighbour is below a threshold [8]. Another approach for nearest neighbour clustering algorithms suggested considering the shared nearest neighbours instead of the distance to these neighbours [9]. The shared nearest neighbours approach addresses the difficulties when working with high-dimensional data. The basis of the shared nearest neighbours algorithm defines similarities between patterns by their mutual neighbour patterns. Due to the nature of this algorithm, it is also highly effective in eliminating noise from the dataset (Figure 2.10). Noise elimination is achieved by targeting elements in the dataset with a limited number of nearby neighbours.

2.3 Virtual and Augmented Reality in Robotics

In recent years, 3D point cloud technology has proven capable of representing both static and dynamic 3D objects at various scales ranging from blood vessels to buildings or even cities. Furthermore, researchers have shifted their focus towards point cloud data visualisation in VR environments. Numerous methods were considered and investigated, such as estimating the current frame, utilising the data from the previous frame, and filling in the difference with the latest measurements [95]. An alternative approach involves rendering the point cloud data by combining various techniques, from good data structures to complex optimisation and sampling algorithms [96]. Another method proposed overlaying the 3D point cloud with an existing CAD model, which can be displayed in a VR environment [97].

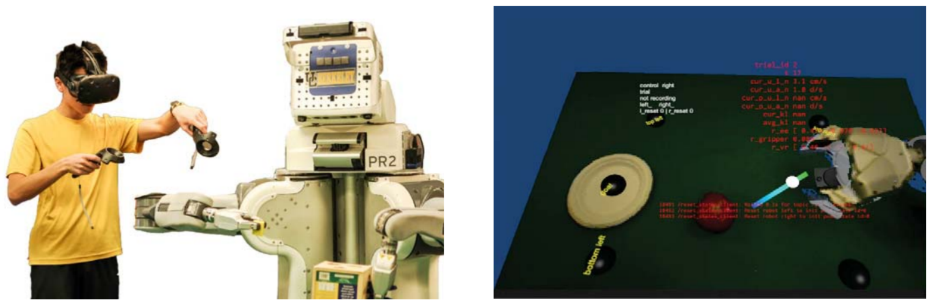
The VR environment also enables the display and annotation of meaningful information from the raw point cloud data. Works have been conducted regarding providing an intuitive and straightforward tool for annotating point cloud data in VR [98][99], where it has been shown to be effective while also remaining relatively intuitive to use.

Section 2.3.1 focuses on the applications of VR technology in training programs in various fields, including industrial assembly, robotics surgery, human-robot collaboration, and many others. Section 2.3.2 focuses on the applications of AR technology in control interfaces. The implementation of AR aimed to improve the users' performance by providing additional information about the task on screen. Section 2.3.3 focuses on data transmission between VR and ROS in remote operations.

2.3.1 Virtual and Augmented Applications in Training Programs

Currently, there is a large body of literature investigating the application of VR and AR in training for industrial assembly tasks. VR and AR environments can be utilised in assembly tasks training platforms and have been demonstrated to outperform traditional methods [100]. As they become widely applied in industrial assembly, training, and remote operations, assessment systems are required to evaluate the simulations and user performance therein.

In addition to industrial assembly training, VR and AR platforms are also implemented to assist the training and work experience in various industries, such as over ten years of VR training for robotics surgery [101] and flight simulation [102]. In education, learning systems utilised VR and haptic technology to learn handwritten Japanese characters [103]. The combination of haptic technology and VR allows the learner to see and learn through the sense of touch.



(a) (b)

FIGURE 2.11: Virtual Reality teleoperation. (a) Real-life implementation. (b) First-person view from inside VR [10].

The design and assessment of VR simulators for diverse fields, including human-robot collaboration and industrial training, have been extensively researched. Previous work demonstrated the design of a VR human-robot collaborative environment for industrial training purposes [104]. Other researchers investigated an evaluation procedure to empirically assess VR and AR training simulator's performance and effectiveness for industrial maintenance and assembly tasks [105]. A prototype system for aircraft maintenance has been presented [106], which addressed the limitations of AR technology when building user interfaces to display documentation as digital and graphical records.

VR has also been used in various robotics systems to assist with manipulation tasks, such as [10] implementing VR to teleoperate a 6DOF robotic arm performing complex tasks (Figure 2.11). Here, VR assists in the process of obtaining demonstrations required by a deep imitation learning network. In a different study, VR was utilised to program a grasp recognition algorithm, implementing programming by demonstration [107]. The VR environment gave the system essential information, including hand postures, contact points and computed normals. This information can then be implemented into a pre-grasp planning algorithm.

Upper limb rehabilitation can be augmented by VR gamification and haptic robotic manipulator feedback [11], as shown in Figure 2.12, to improve the connection between patients and therapists, increasing the productivity of therapists administering rehabilitation. It enabled remote routine customisation for the patient by providing an avenue for rehabilitation in a domestic setting. However, this work did not provide the VR user with any way

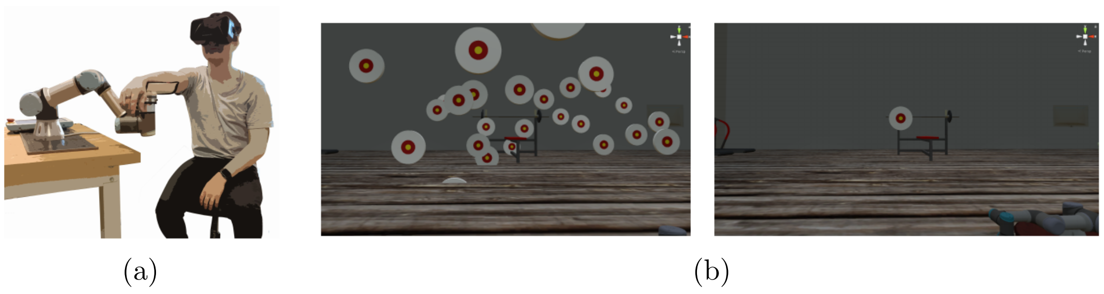


FIGURE 2.12: (a) A system based on the GAVRe2 framework. (b) First-person view from inside VR [11].

to incorporate real-world sensor data to enhance their experience. The real-world sensor data, such as a video feed textured onto a live geometric representation of the therapist, is hypothesised to improve the experience for everyone involved.

In a previous work [12], it was shown that VR could interface with the control system of a real-world mobile manipulator, which facilitated a natural and intuitive method for human-robot interaction. The technology used in most commercial VR platforms has the ability to track movements in 3D space and translate performed motions, coupled with intuitive controls, such as grabbing and pointing, which provides a compelling approach to resolve the limitations of traditional remote robot teleoperation methods (Figure 2.13). The proposed future work was to integrate live real-world sensor data and investigate the influence of the data limitations on a user's ability to perform manipulation tasks. To remove any controlling variables and isolate the sensing aspect, the experimental tasks presented were directly enacted by the user's own hands rather than via the previously presented VR-based remote control interface.

2.3.2 Virtual and Augmented Reality Applications in Control Interfaces

In recent years, the applications of VR technology in robotics systems have been a popular research topic. VR helps create an intuitive and effective control interface for remote control robotics systems while enhancing the user's perception ability through 3D data visualisation. Previous work has implemented VR to create the control interface for underwater undetonated munitions removal [108]. The work presented is also designed as

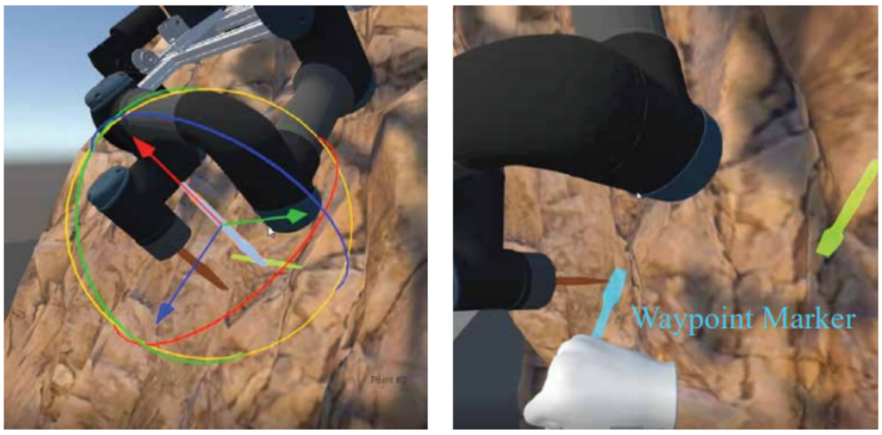


FIGURE 2.13: The interfaces given to users. Left: 2D interface. Right: 3D VR interface [12].

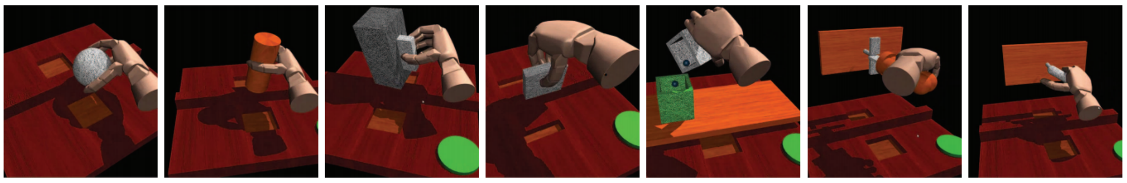


FIGURE 2.14: Virtual hand operating in VR, controlled through motion-capture glove [13].

a ROS package for straightforward integration into other applications. A different ROS-VR teleoperation package was introduced in [109]. The developed framework is designed to be implemented with any VR headset that is compatible with Unity. The paper also conducted a user study to obtain insights into the framework's performance in various manipulation tasks. In other research, VR was utilised in a control interface for real-time fine-grained dexterous control of the robot remotely [110]. Additionally, VR improves the users' awareness of the environment and the contact surface by bestowing the user with the manipulability of the viewpoint. An alternative approach to interacting with a VR environment is through direct motions of a teleoperated virtual hand [13]. The article illustrated a VR system that combines real-time motion capture, physics simulations, stereoscopic visualisation and a motion-capture glove to control the virtual hand (Figure 2.14).

The applications of VR in robotics systems also involve training simulations for multiple processes, such as [111] implementing VR to create an industrial robot operation training system. The presented system was not limited to simulation training but could also act as the control interface for real-world industrial manipulators. A prior study investigated the applications of VR in human-robot collaborative systems [112]. The author developed a VR-based framework integrating human responses into the system design process. The human-robot collaboration system layout design and the control program can also be optimised by utilising an event-driven simulation created with VR. Ruckert et al. proposed a systematic approach to adopt VR into collaborative assembly processes [113]. Additionally, the work introduced a specific VR system derived from an assembly process, achieving the highest possible immersion.

AR is widely implemented in control interfaces. However, unlike VR, AR focuses on providing users with additional overlay information. A user study was conducted to compare the performance of AR and VR in selection and transformation tasks on a 9 DOF object [114]. The study indicated that better users' performance, which was measured by completion time, occurred when using AR instead of VR. The overlay information AR provides can be documentation and guidance on the ongoing task [115] [116], or processed sensor data. A novel robot operation interface for a system that composes a self-tracking device and a depth camera with AR support was presented in [117]. The operation interface renders information, such as basic robotics manipulation commands and object detection results from the depth camera. In a different paper, AR was utilised to display the 3D information collected from a remote area interactively [118]. Furthermore, the paper introduced a gesture operating interface that converts the operator's movements into control commands in real-time. Alternatively, the remote control system can receive inputs from a haptic feedback device [119]. The article proposed an intuitive interface for remote welding robots programming that involves AR and haptic feedback. The proposed system reconstructs the surfaces of the workpieces and utilises AR to visualise the welding torch, assisting the operator with the planning phase of the welding task.

2.3.3 Dense 3D Data Transmission

Point clouds are a source of feedback that enhances the visualisation of depth information implemented in various algorithms. A point cloud is a type of data that is generally constructed of a set of unconnected $[X, Y, Z]$ geometric coordinates representing a single point in $\mathbf{R}^3$ space. Although they are a good way to represent objects in space, they are an inherently inefficient way to transmit or store data [120]. This has led researchers to present a data compression approach aimed at solving bottlenecks in 3D point cloud data transmission. It involved mapping the point cloud data onto panoramic images, which can be encoded using conventional image compression techniques. Another approach by [121] suggested removing unneeded features to reduce the amount of data transmitted. The authors could transmit point cloud data in near real-time by removing unnecessary noise and information through filtering and segmentation algorithms. A more recent paper has shown successful preliminary experimental results in real-time 3D point data transmission [122]. The researchers achieved this by implementing the Octree-based compression technique provided by the PCL and encoding the point cloud data into a text file before transmission.

2.4 Summary

Robotic perception and map-based mobile robot navigation problems have been investigated by various researchers in the past. However, there remain unanswered problems on the topics. In robotic perception, numerous variants of the original concepts and methods were developed and proved to be applicable to certain tasks. However, in most cases, the variant was established to address the specific problem or environment it was created to solve. Therefore, new problems and questions are introduced for mobile robots based on their operating environment.

Active perception is the combined problem of NBV selection, robotic perception and robotic manipulation. While there has been a substantial amount of research done for robotic perception and mobile robots, the NBV problem is highly situational, and possible solutions still need to be investigated. The answer to this problem largely depends on the specific task the robot aims to solve. Human collaboration is a solution proposed in several scenarios in which the robot is responsible for completing low-level tasks while the human makes high-level decisions. For remote control systems, the chosen NBV is heavily dependent on the information provided by the robot. This leads to the problem of data visualisation and human-robot interaction and collaboration.

The development of VR hardware and software has enabled the representation of the real world in a virtual environment. Additionally, when combining VR with sensors, such as LiDARs and RGB-D cameras, it is possible to capture the state of real-world objects in real-time within a VR environment. VR also enables users to immerse themselves into the environment, creating an intuitive environment when interacting with the robot while operating remotely. Therefore, VR was selected herein as the data visualisation tool for collaborative autonomy. However, user studies are required to determine the optimised configurations for displaying data and whether assistive annotation information is practical and effective.

Chapter 3

Frontier Detection Algorithms

3.1 Introduction

Environment perception is compulsory for every robotic system regardless of its task and purpose. One of the most researched questions in the robotic field over the last few decades is how to build a map to assist the navigation of a mobile robot. Usually, the navigation map is a two-dimensional grid map created utilising data obtained from range sensors. The process of perceiving the environment to generate the grid map involves distinct problems such as exploration, localisation, saving or updating the map through time. Modern robotic systems are often designed to operate in various environments, most with minimal to no prior knowledge. Thus, the ability to perform exploration is required.

Many robot exploration algorithms that utilise LiDAR sensor data rely on the concept of frontier cells. Frontier cells form the edge between the known and the unknown space. Frontier-based exploration involves repeatedly detecting the frontiers while moving towards them. Depending on the computation time of each iteration, one or multiple scans can be performed from the LiDAR viewpoints as the robot moves. The map frontiers and the next action is then determined based on the obtained information. The exploration is finished when no frontiers and unknown regions remain. This chapter presents the implementation of six algorithms for detecting frontiers in simulated and real-world scenarios and the findings of a comparative study.

Since frontier detection is a component of exploration algorithms, the speed of the exploration process can be enhanced by improving the frontier detection speed. The advance in frontier detection speed may also improve the quality of the decisions made by the exploration algorithms since faster decisions can be made with more recent data. In addition, with the technology development, frontier detection is more likely to be the bottleneck to the speed of the exploration algorithm instead of the sensors' data acquisition speed. Therefore, it is necessary for frontier detection algorithms to be efficient.

3.2 Methodology

This chapter presents the implementation and evaluation of six frontier detection algorithms in simulations and real-world scenarios. The implemented algorithms are: Naïve, WFD [42], WFD-INC [43], NaïveAA and EWFD[48], FTFD [1]. The details of these algorithms have been presented and reviewed in Chapter 2. However, they must be compared in similar environments to ascertain their strengths and weaknesses. The Naïve algorithm evaluates all cells to determine if they are frontiers based on the definition of a frontier. WFD differs from the Naïve algorithm because it targets the freespace cells subset. WFD-INC use BFS, similar to WFD, to detect frontiers among the freespace cells in the active area. The NaïveAA algorithm only evaluates cells that have recently become a frontier or are no longer a frontier. The initial iteration of EWFD is identical to WFD, but it incorporates the labelling process to speed up the frontier detection process. FTFD, on the other hand, exploits the fact that the border of the current observation has a high possibility of intersecting with existing frontiers. The experimental results, conclusions and insights for all algorithms are discussed and presented in this section. The experiments and analysis are performed for various 2D environments, including pre-recorded Freiburg maps, Gazebo simulation and real-world indoor hallways. The run-time of all six algorithms is determined theoretically and through simulations and compared to each other.

Three experiments have been conducted using the same algorithmic implementation code, developed in MATLAB. Experiment 1 uses simulated data to test the relative efficiency of the algorithms when detecting exploration frontiers in a controlled, known, simulated environment. An image of a known map was used to emulate the SLAM process as a map is gradually constructed from a sequence of sensor observations. Experiment 2 uses Gazebo as the simulation engine environment and a ROS node in MATLAB to handle the calculation of the algorithms. In this experiment, the aim is to investigate the possible overhead in this system setup and to test the stability and repeatability of the algorithms when they are fed simulated sensor data from Gazebo that mimics real-world scenarios. Experiment 3 was conducted in a 60x60m real-world office environment at the UTS, where a MP-700 [14] mobile robot equipped with a SICK-S300 laser scanner was manually controlled to construct a map. The software and system setup was similar to Experiment 2, with most processes handled in ROS. The MATLAB ROS node handles the frontier detection calculations for each algorithm. The main difference is that in Experiment 3, a real robot replaced the Gazebo simulator used in Experiment 2. For all scenarios, the calculation times per scan have been recorded alongside the total number of cells evaluated per scan. The algorithms compared are the Naïve, NaïveAA, WFD, WFD-INC, EWFD, and FTFD.

The experiments have been conducted to investigate and validate the following ideas:

- Given a set of different frontier detection algorithms, investigate which algorithms perform more efficiently and effectively in several typical mobile robot scenarios, then highlight the differences, strengths, and weaknesses of all algorithms;
- As a robot explores an environment, the number of frontier cells tends to increase. Therefore, it is necessary to investigate the relationship between each algorithm's calculation time and the number of frontier cells;
- Given that the experiments are conducted in both simulation and on a real robot, investigate if there are relationships or contradictions between the results obtained from the simulated experiments and the real-world cases;
- Determine whether the MATLAB ROS node implementation is fit for handling the task of frontier detection in real-time (i.e. faster than sensor data can arrive) for all setups and with all algorithms in both real-world and simulation.

In all scenarios, the experiments were conducted 10 times for each algorithm, and the median time taken is shown. Since the algorithms are not running on a real-time operating system, occasionally (but very infrequently), Linux system processes will cause one-off non-repeatable spikes in processing time. The spikes in processing time caused by Linux are in the order of tens of milliseconds. Using the median values of 10 runs, this noise is filtered out. Finally, to display the graphs clearly without unnecessary empty spaces, the processing time for the first iteration, in which the system is being set up, was excluded for all algorithms. The details will be discussed further in the results section.

3.2.1 Experiment 1

The first experiment was conducted purely in MATLAB using a ground truth map image of a Freiburg lab environment of size 1242 x 447, as shown in Figure 3.1, with a preplanned trajectory for a simulated robot to move through. At each point of the trajectory, a simulated laser scan was generated and ray-traced into the ground truth image. From that sensor observation, a local map was constructed to represent the current view of the simulated robot in the environment. The newly constructed map and information about the active area of the latest sensor observation (i.e. the bounding box of the latest scan), if relevant, were then put through each frontier detection algorithm to calculate and determine the frontier cells grouping.

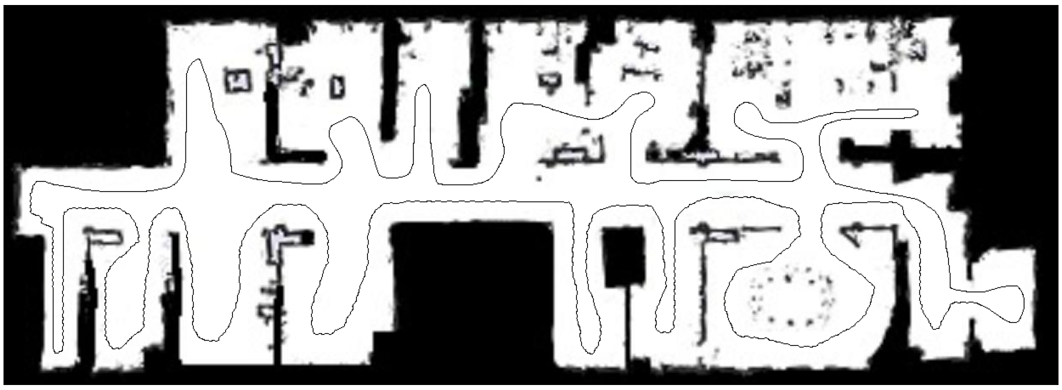


FIGURE 3.1: Freiburg lab environment with a preplanned trajectory for Experiment 1.

The main purpose of this experiment is to verify and analyse the performance of each algorithm in perfect conditions with no delay due to information transmission between multiple computers and no sensor noise.

The metrics measured for each algorithm were the average calculation time of each iteration, the total number of cells processed, and the total number of cells evaluated. The number of cells processed is defined as the number of cells being queried in some way, while evaluated means each time the algorithm checks whether a cell is a frontier or not.

Additionally, the NaïveAA, WFD-INC, EWFD, FTFD algorithms all use the active area as part of the frontier detection. Therefore, the relationship between the size of the active area and the frontier calculation time is also evaluated by changing the maximum measuring range of the simulated laser scan. This value was increased gradually from 100 pixels to 500 pixels, with other parameters set to be the same for the 5 cases. The computational time for each case was then recorded and is shown in Figure 3.6.

3.2.2 Experiment 2

The second experiment was conducted in two different environments, as shown in Figure 3.2. Two different computers were used in the experimental system (Figure 3.3). The Gazebo simulator with the Turtlebot3 and the SLAM algorithm operated on the first computer. Meanwhile, the other computer subscribed to the data on the first one via a ROS network and performed frontier detection.

The Turtlebot3 was manually controlled to move around the simulated environment while the SLAM algorithm was also active. The map constructed by the SLAM algorithm for this experiment has a size of 20 by 20 meters with a resolution of 0.05 meters. The first computer synchronised the generated map and the necessary parameters, such as the current robot pose, active area, and laser scan data. Then it published them through the ROS network. The second computer then collected the data and implemented them for each of the frontier detection algorithms in turn. To ensure a repeatable data set over multiple attempts in this experiment, the data from the first computer was recorded in a rosbag and then played back to all the frontier detection algorithms.

The main purpose of this simulated experiment is to validate the efficiency of the algorithms in a more realistic scenario where delays caused by the SLAM algorithms and information transmission between different machines exist, and the sensor data and robot

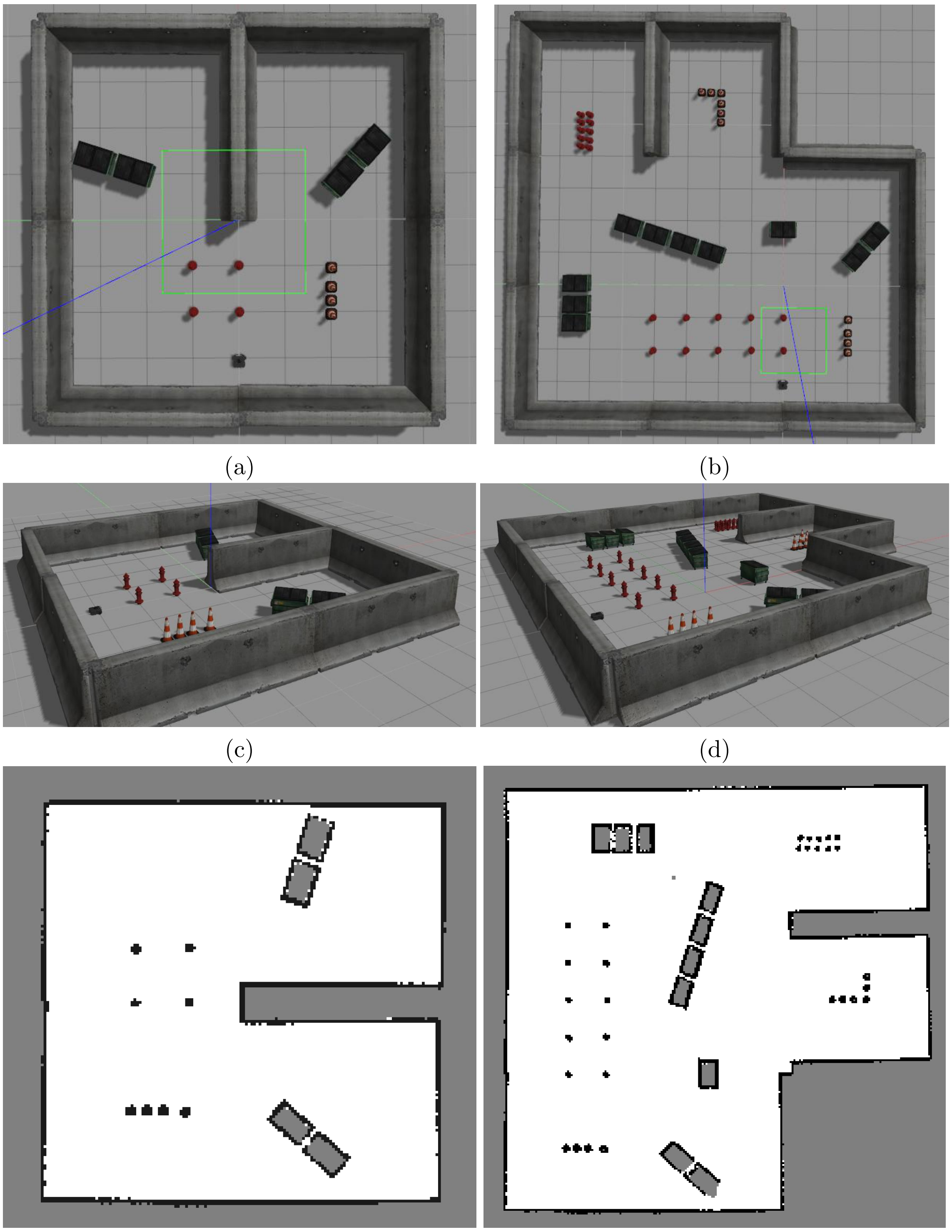


FIGURE 3.2: The two map setups used in Experiment 2. (a) Small map top view. (b) Large map top view. (c) Small map perspective view. (d) Large map perspective view. (e) Scanned small map. (f) Scanned large map.

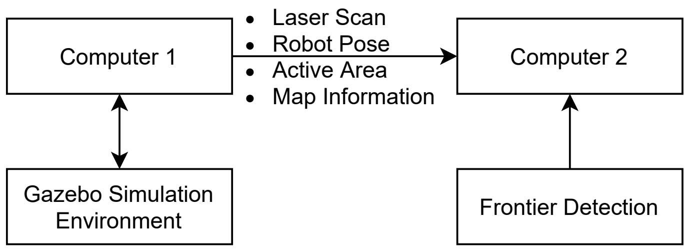


FIGURE 3.3: Experiment 2 setup.

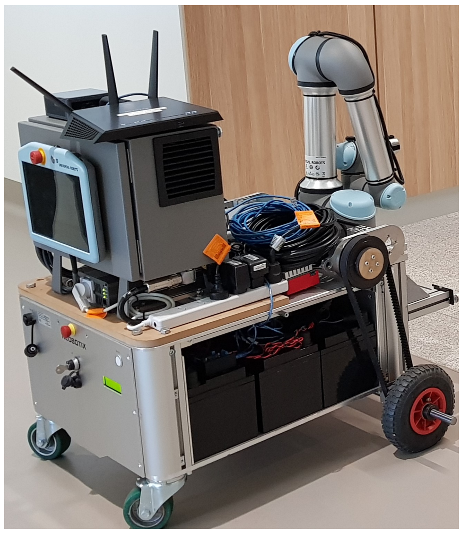
movement is similar to reality. This scenario also provides a basis for comparison with Experiment 3, which will be mentioned in the following section, to validate the relationship between the simulation and real-world results.

3.2.3 Experiment 3

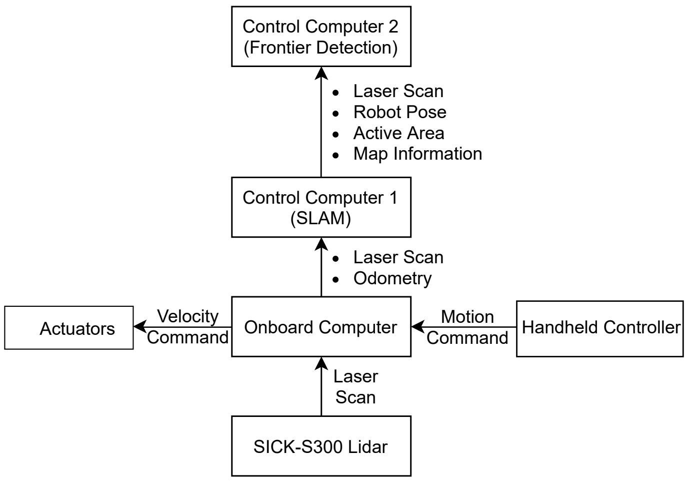
The third experiment was conducted in the real world using a Neobotix MP-700 [14] mobile robot shown in Figure 3.4. The robot was driven manually by a human operator around a large office environment inside UTS as shown in Figure 3.5. Laser scan data from the SICK-S300 LiDAR was collected and fused with odometry data from wheel encoders to incrementally build a map of the environment using ROS SLAM packages. Each scan was sent to the MATLAB ROS node to calculate the exploration frontiers by sequentially using the suite of detection algorithms. In order to have a consistent movement trajectory and a repeatable data set, the exploration frontiers calculation of each algorithm was performed on a previously recorded rosbag containing all the required parameters. The sensor mounted on the mobile platform has a field of view of 180 degrees and a maximum range of 30 meters. The map constructed by the SLAM algorithm for this scenario was limited to 200 by 200 meters with a resolution of 0.05 meters.

3.3 Experimental Results

First, from the Experiment 1 average calculation time shown in Figure 3.6(a), it is clear that the number of cells processed and evaluated by both Naïve and WFD algorithms are significantly higher than for the other algorithms. On the other hand, EWFD and FTFD are both drastically faster as they evaluate fewer cells than the other algorithms. This

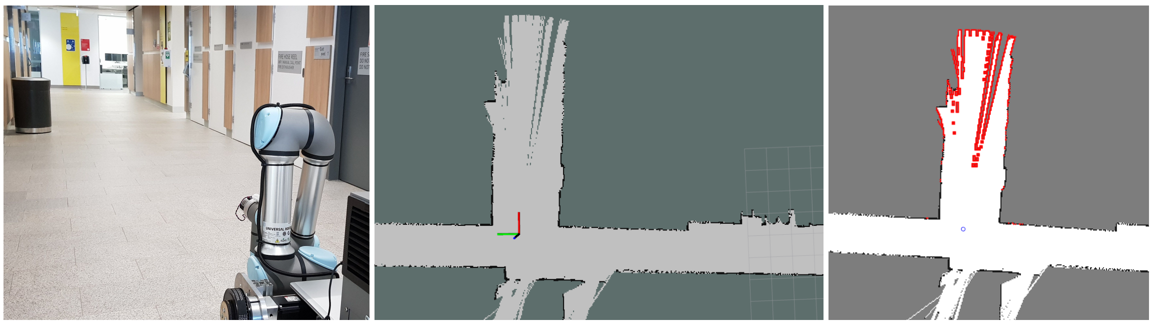


(a)



(b)

FIGURE 3.4: Experiment 3 real-world platform: (a) The Neobotix MP-700 mobile robot [14]; (b) Robot and information diagram.

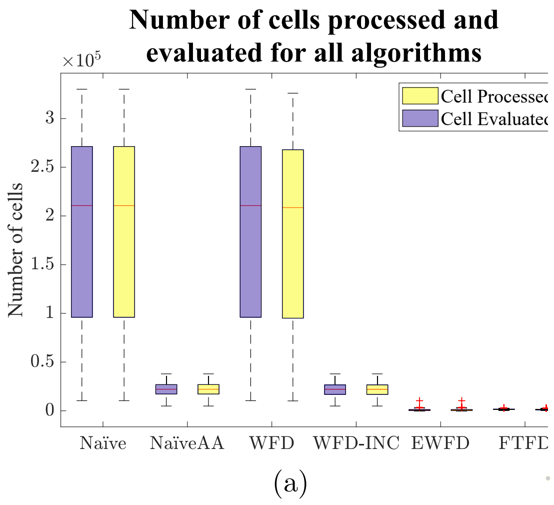


(a) (b) (c)

FIGURE 3.5: Experiment 3 setup. (a) Current real-world view of the robot. (b) Updating the map with the latest sensor observation (robot position indicated by frame annotation). (c) Newly detected frontiers (in red) in the constructed map.

performance advantage will eventually result in a considerably lower overall processing time, as shown in the Experiment 2 results for the small map (Figure 3.7) and the large map (Figure 3.8).

The Experiment 1 comparative plot of the sum of frontier detection time over all steps in the exploration (Figure 3.6(b)) demonstrates a strong relationship between the size of the active area and the total processing time for each of the algorithms in which the active area is considered. However, this relationship effect is smaller for EWFD and FTFD compared to the NaïveAA and WFD-INC.



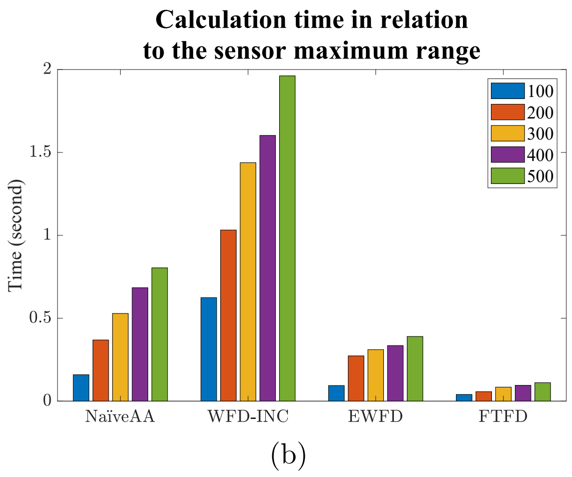
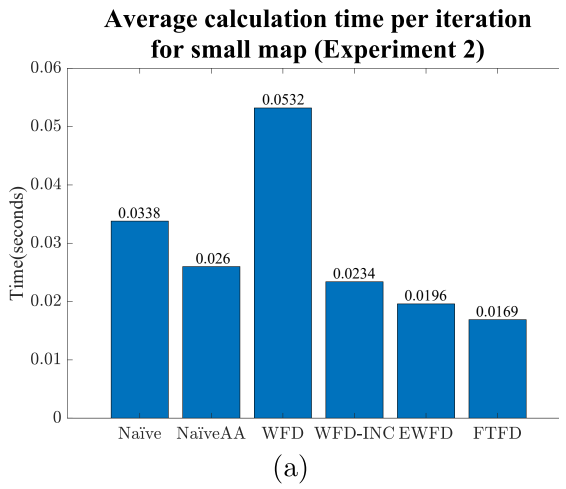


Figure 3.6: Experiment 1 results. (a) The number of cells processed and evaluated by all algorithms. (b) The calculation time of the algorithms that consider the active area in relation to the maximum range of the simulated sensor (measured by the number of map cells).



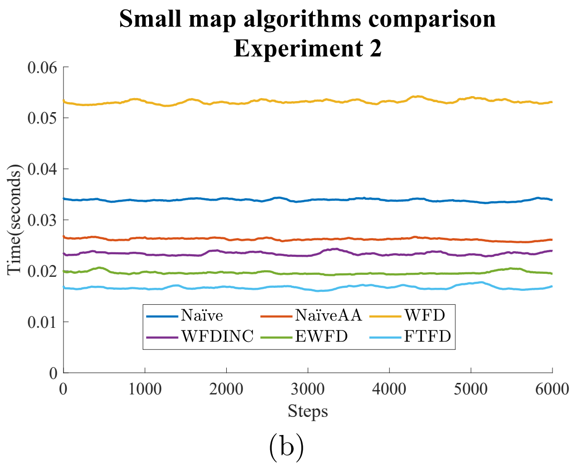
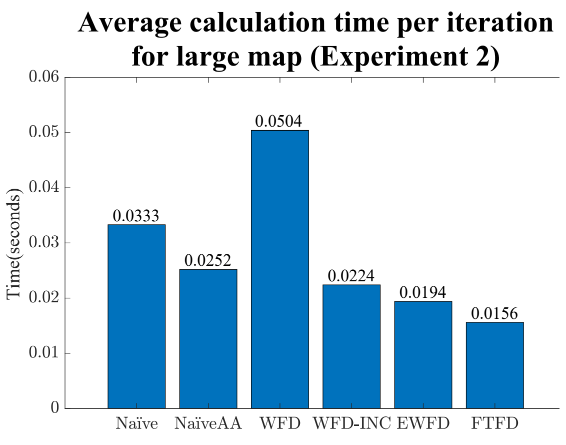
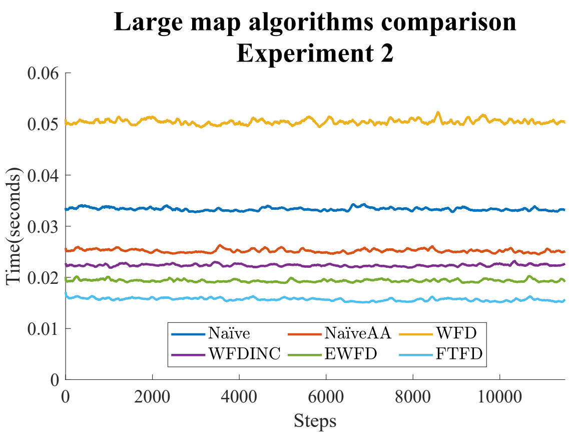


Figure 3.7: Experiment 2 results for the small map case. (a) Average calculation time per iteration. (b) Calculation time as the small map is gradually explored for all algorithms.

Both Figure 3.7(a) and Figure 3.8(b) indicate that the average calculation time of each iteration of NaïveAA, WFD-INC, EWFD, and FTFD algorithms is lower than the update rate of 5Hz used by the simulated SLAM algorithm. It is noted that 5Hz is not the maximum rate possible, but it is adequate for the robot speed in the experiments and can be guaranteed in all cases for all experiments. Since this research focuses on the frontier detection calculation time, the SLAM update and publish rate do not affect the results.

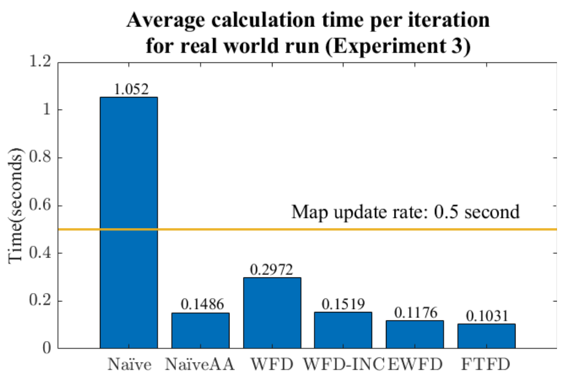


(a)

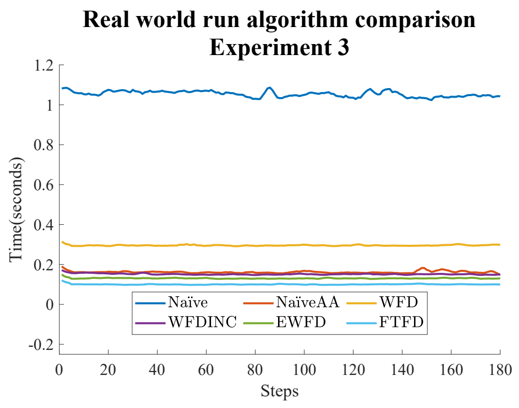


(b)

FIGURE 3.8: Experiment 2 results for the large map case. (a) Average calculation time per iteration. (b) Calculation time as the large map is gradually explored for all algorithms.



(a)



(b)

FIGURE 3.9: Experiment 3 results. (a) Average calculation time per iteration with a horizontal line representing the updating rate of the map constructed by the SLAM algorithm. (b) The calculation time as the map is explored for all algorithms.

The processing time of both Naïve and WFD algorithms are noticeably longer than other algorithms. Moreover, the novel algorithm, FTFD, produces a better overall result than the other algorithms in terms of time, processed cells and evaluated cells. Results from both the small and large maps in Experiment 2 (Figure 3.7(b) and Figure 3.8(b)) suggest that there is no relationship between the number of explored cells and the calculation time of each iteration on all algorithms since the calculation time for both map cases remains relatively stable as more sections of the map are revealed.

Figure 3.9 and Figure 3.7 illustrate the similarity between the results of the second and third experiments. In both experiments, EWFD and FTFD have the best performance,

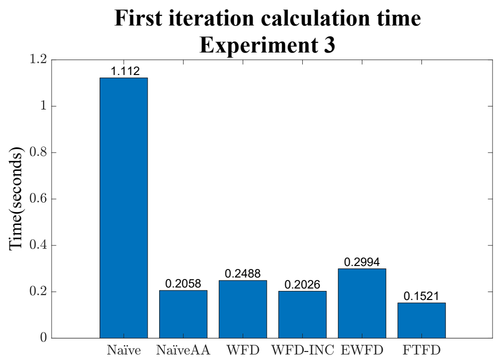


FIGURE 3.10: Average calculation time in only the **first** iteration of each algorithm.

with Naïve and WFD being the most time-consuming of all of the algorithms. In the real-world Experiment 3, as the map size is 200 by 200 meters with a resolution of 0.05 meters, the image of the map reaches 4000 by 4000 pixels, and all algorithms exhibit a drop in performance compared to the Gazebo simulated Experiment 2, where the maps are smaller. However, most of the algorithms are significantly faster than the update rate of the SLAM algorithm, which is 2 Hz (0.5 seconds). The only exception is the Naïve algorithm, with an average calculation time of more than 1 second, which means it cannot keep up with the update rate.

Finally, the average calculation time for the first iteration of each algorithm is shown in Figure 3.10. Comparing with the results from Figure 3.9(a), it is clear that most algorithms (except Naïve and WFD) required a relatively higher amount of time to set up the initial variables and, in the case of both EWFD and FTFD, to locate the first set of frontier cells.

3.3.1 Discussion

The results obtained from the different experiments provide insights into the overall performance of the different frontier detection algorithms. Importantly, it is clear that the results from both simulation Experiments (1 and 2) and real-world Experiment 3 are in agreement. Therefore, it is appropriate to synthesise these results and comparatively discuss and highlight the algorithms' strengths and weaknesses. A summary of properties and suggestions for each algorithm is shown in Table 3.1. It is noted that while all the algorithms can be run in 3D, their relative effectiveness has not been tested on 3D data within the scope of this work.

Table 3.1: Properties and suggestions for using different frontier detection algorithms. Yes* is used to indicate where one algorithm performs markedly better than the others for a particular property.

Properties	Naïve	NaïveAA	WFD	WFD-INC	EWFD	FTFD
Performance stable as map size grows	No	No	Yes	Yes	Yes	Yes
Faster when run after no map change	No	No	No	No	Yes	Yes
Performance stable as active area increases	N/A	No	N/A	No	Yes	Yes*
Capable of detecting pockets	Yes	Yes	Yes	Yes	No	No
Ease of software implementation	Yes*	Yes	No	No	No	No
Must run after every scan	No	No	No	No	No	Yes
High update rate	No	No	No	No	Yes	Yes*
Suitable for open-spaced map	No	Not ideal	No	No	Not ideal	Yes
Suitable for maps with multiple narrow paths or corridors	No	Not ideal	No	No	Yes	Yes
Ideal for maps with low resolutions	No	Yes	No	Yes	Yes	Yes
Suitable for maps with high resolutions	No	No	No	No	Yes	Yes*

From the simulations and the experimental results, it is observed that there is a close relationship between the number of cells processed and evaluated and the computational time. Therefore, as the map either becomes more detailed (i.e. high-resolution maps) or the sensor covers more area at the same time (i.e. large active area), algorithms that evaluate the smallest number of cells should be prioritised, such as FTFD or EWFD.

However, Naïve Active Area should be considered in scenarios where the resolution of the constructed map is relatively low, or the sensor's maximum range is short. In these scenarios, Naïve Active Area outperforms both EWFD and FTFD due to its simplicity in implementation while also being able to yield similar performance compared with EWFD and FTFD.

Secondly, FTFD outperforms EWFD when the map contains large open areas (e.g. Experiment 2 maps - Figure 3.2) since FTFD only evaluates the perimeter of each laser scan while EWFD processes the area. However, suppose the map contains multiple corridors or

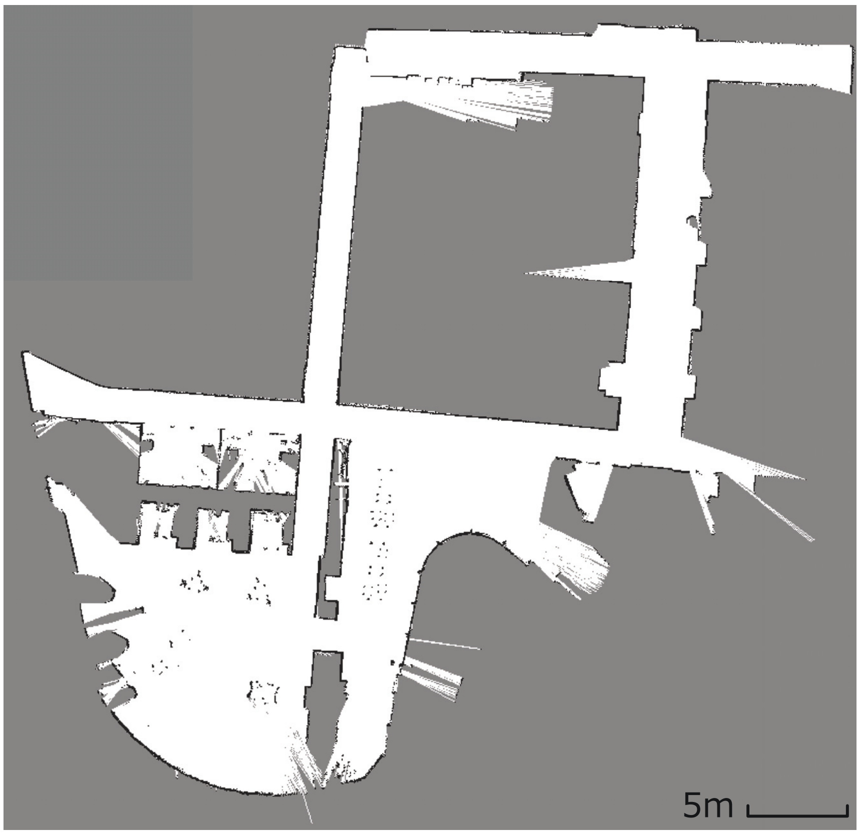


FIGURE 3.11: Scanned map of Experiment 3.

narrow paths (e.g. Experiment 3 map - Figure 3.11). In this case, EWFD is comparatively similar to FTFD, as the number of cells covered by the scan area and perimeter are now closer to equal.

Thirdly, there is no significant relationship between frontier detection calculation time and the expansion of the map (i.e. the discovery of more free space) as shown in Figure 3.7(b) and Figure 3.8(b). However, in theory, more free cells should mean an increase in processing time as more cells are to be evaluated.

Finally, the results confirm that Naïve Active Area should be used as a benchmark for future frontier detection algorithms in 2D static environments. As a benchmark, Naïve Active Area is simple, robust, and efficient compared to Naïve frontier detection, whose performance degrades with the size of the map. Furthermore, when it comes to performing frontier detection after each observation, which is common practice in exploration tasks, FTFD should be considered, as it functions most effectively after each laser scan and is also the algorithm with the highest efficiency among those evaluated. Additionally, as shown in Figure 3.9, it is reasonable to conclude that the MATLAB ROS Node is suitable for handling the task of frontier detection. The nodes were able to calculate the frontier cells within the map update rate of the SLAM algorithm for all algorithms in the experiments other than Naïve in the largest real-world map. The results also indicate that a MATLAB ROS Node can be implemented side-by-side with ROS to perform near real-time processes because the connection delay between the two programs is negligible.

3.4 Conclusion

This chapter reviewed multiple algorithms for the problem of frontier detection while also providing comparisons between the noteworthy frontier detection algorithms. Simulation and real-world experiments were conducted to provide insightful conclusions about the performance of the inspected algorithms. To validate the efficiency of the studied algorithms, three experiments were implemented with relevant data collection for the overall computational time, the number of cells processed and evaluated, etc. Though the presented algorithms will work in 3D, it is not immediately obvious how much advantage EWFD and FTFD would provide compared to other algorithms when used in 3D environments. Further experiments using 3D data still need to be performed in future. In the next chapter, an interactive probabilistic mapping framework for a mobile manipulator platform that utilises 3D sensor data is presented.

Chapter 4

Active and Interactive Mapping

4.1 Introduction

Exploration is done to gain information regarding the surrounding environment, which are necessary for robotics applications such as manipulator or mobile base trajectory planning, estimating the area of coverage and optimising sensor viewpoints. The frontier detection algorithms presented in Chapter 3 provide multiple approaches to increase the efficiency of exploration. However, the algorithms utilised 2D sensor data to generate a static 2D map of the world. These algorithms are limited to mobile bases that navigate on a surface. They are insufficient when manipulators are involved or when a 3D representation of a dynamic environment is required. This chapter presents the experimental results of an interactive probabilistic mapping framework for a mobile manipulator picking objects from a pile which was first introduced in [15].

The framework was designed to perform active exploration and mapping activity while simultaneously updating the map as the mobile manipulator interacts with the environment. The framework being implemented utilises a novel dynamic Gaussian Probabilistic Implicit Surfaces (GPIS) method to continuously create and update the map, reflecting the dynamic environment. In addition, the framework is able to actively choose the Next Best View (NBV). The NBV selection scheme balances a few factors: the reachability of picking objects and the IG for fidelity and coverage of the exploration process. To ensure the framework prioritised bounding segments over unknown regions, the IG formulation includes an uncertainty gradient-based frontier score by exploiting the GP kernel derivative. This addition results in a strategy that can efficiently address the conflict between the requirement of unknown environment exploration and object-picking exploitation, given a limited execution horizon. The contribution of this chapter is an experimental evaluation that demonstrates the effectiveness of the implemented framework in both simulated and real-world scenarios.

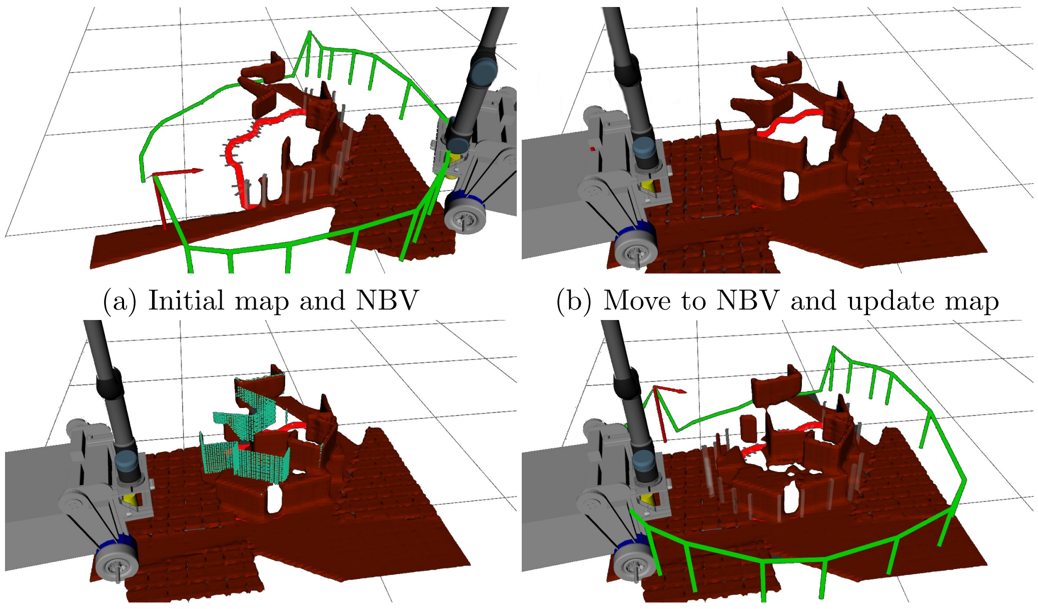
4.1.1 Active and Interactive Mapping Problem

Autonomous mobile manipulation applications have seen significant advancement in recent years. Examples of existing applications include pick and place for construction sites [17], logistics, and warehousing [123]. These applications demand interactive robotic systems that can perform exploration and mapping while also interacting, navigating and updating the changing environment.

Scene exploration approaches often adopt an information-theoretic strategy that aims to choose the next action to maximise the IG in an active mapping framework [71, 124–126]. However, the performance of mobile manipulator tasks can be enhanced by combining mapping and manipulation activities into one phase. Thus, the task becomes an active and interactive mapping problem, i.e. to select and pick the “best” objects with mapping aiding the selection, and where the next movement expands the knowledge of the scene.

This chapter implements and conducts experiments on an environment mapping framework for a mobile manipulator designed for bin-picking applications, as shown in Figure 4.1. The framework aims to address the problems of dynamic scene mapping, exploiting the map’s frontier and checking manipulability to select the next best view for efficient mobile base placement and manipulator motion planning. Within this work, dynamic mapping refers to the accurate capture of environment changes as objects are discovered (scene exploration) and later removed from the scene (object picking) by a mobile manipulator.

The proposed mapping framework is based on GPIS [61], which offers a probabilistic yet accurate map representation of the world in continuous form. GPIS is exploited to check


(a) Initial map and NBV (b) Move to NBV and update map

(c) Remove object (d) Update map and NBV

FIGURE 4.1: The Active and Interactive Mapping cycle. The projection of GPIS mesh on the ground (red line) defines potential exploration segments, and the green bars indicate segments' information utilities. The NBV is a red bar + arrow. Dark green dots are GPIS training points from the removed object [15].

if a mapped object is in the robot's workspace. Also, the probabilistic formulation makes the framework amenable to active mapping based on IG to analytically search for the NBV and optimal motion.

4.2 Methodology

4.2.1 System Overview

This section presents a brief overview of the framework originally introduced in [15]. As illustrated in Figure 4.2, the implemented framework consists of a GP-based dynamic mapping component that maps and updates the environment and a NBV selection module that spawns recommendations based on the map information. The mobile manipulator relies on this framework to perform trajectory planning for both its arm and mobile base and to modify the environment.

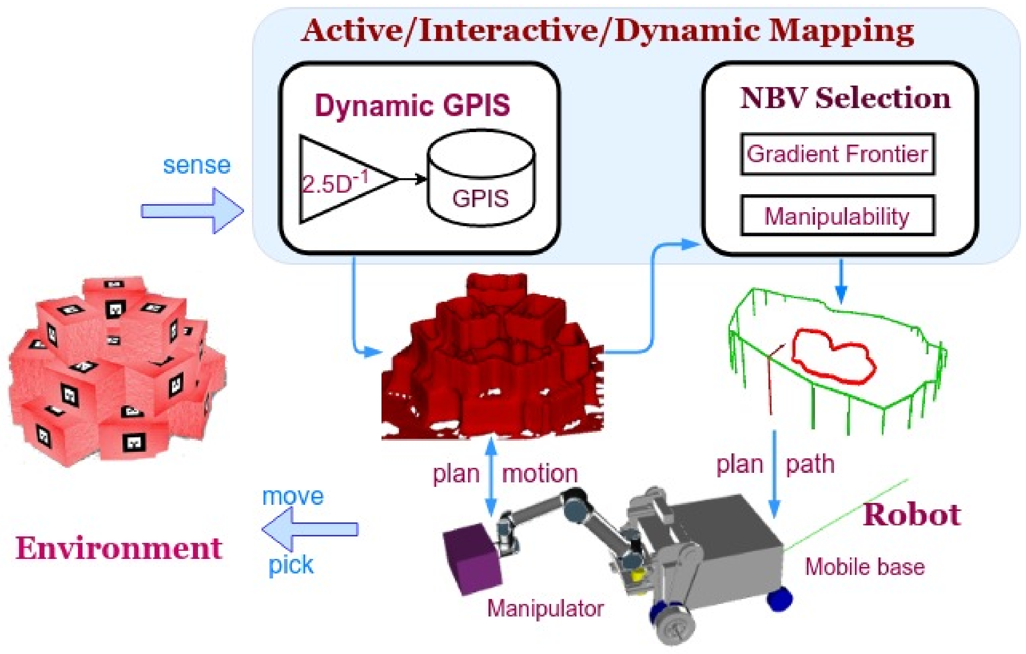
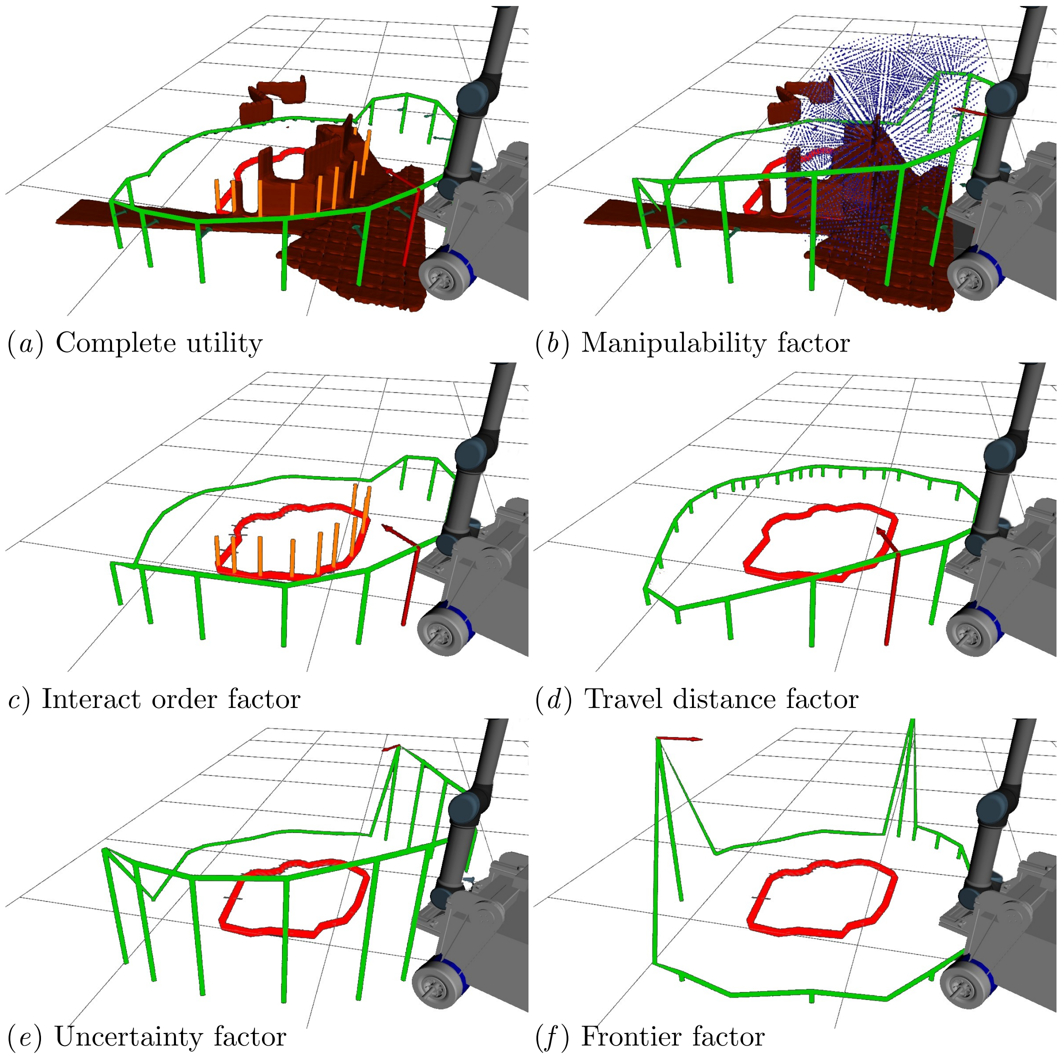


FIGURE 4.2: Active and interactive mapping framework overview.

Figure 4.1 shows a robot's exploration and interactivity cycle. First, an initial map containing a pile of multiple objects is built, and the NBV to obtain the most map information is computed. Consequently, the mobile base navigates to the recommended NBV, and the map is updated. Then, an object is picked up, and the map is updated again to reflect the changing environment. Based on the most recent map, a new NBV is computed, and the mobile manipulator moves towards it. The inspection cycle is repeated when the mobile base arrives at the new NBV.

The implemented framework's dynamic mapping representation is based on the online GPIS fusion introduced in [16]. In a similar way, the mapping module of the framework consists of two GP phases. A GPIS accumulated from multiple scans, and a frame-level 2.5D^{-1} map as a detector for points removed from the scene. It also exploits independent overlapped clusters [127] to store GPIS training points for parallel processing. The main difference between the framework being implemented with [16] is the handling of dynamic scenes and the introduction of a virtual wall, as described in detail in [15].

To determine the NBV, a set of candidate poses are identified, and then the set's utility function (see Figure 4.3) is computed. Afterwards, various GPIS-based metrics, including gradient frontier and manipulability, are incorporated into the selection scheme.



(*a*) Complete utility (*b*) Manipulability factor

c) Interact order factor (*d*) Travel distance factor

(*e*) Uncertainty factor (*f*) Frontier factor

FIGURE 4.3: An illustration of full formulation and single factor utilities. First, identify pile segments (red contour) and compute the utilities (green bars). Max utility gives NBV (red bar+arrow). (*b*) Shows samples (blue dots) from the manipulability annulus. (*c*) Shows heights (orange bars). Grey bars on the ground indicate the uncertainty for imaginary segments.

4.2.2 Experimental Setup

Extensive simulated and real-world experiments were conducted to evaluate the performance of the implemented framework [1]. The mobile manipulator utilised in the experiments consists of a Neobotix MP700 [14] as the mobile base, and a UR5e [128] as the manipulator. An RGB-D camera is mounted at the front of the mobile base, under the manipulator. The framework was implemented in ROS and operated on a 6-core laptop.

[1]A video demonstrating the experimental results can be found at `https://youtu.be/DYbNH5Z6zsk`.

The test environment comprises multiple piles of bricks located on flat ground. Each pile of bricks is roughly 4 m^2 in size with ∼50 segments considered in the optimisation, which is trivially performed by finding the maximum amongst all candidate utility scores. The bricks have AR tags labelling for easy pose detection and are picked up through magnetic grasping with a magnetic contact end-effector attached to the UR5e.

4.2.3 Simulated Experiment

A simulated Gazebo environment containing brick piles on flat terrain was created, as shown in Figure 4.4. Gazebo models of the base and arm were also designed with accurate mechanical properties to mimic the real-life system. An ablation study and benchmark tests were conducted in the simulated environment.

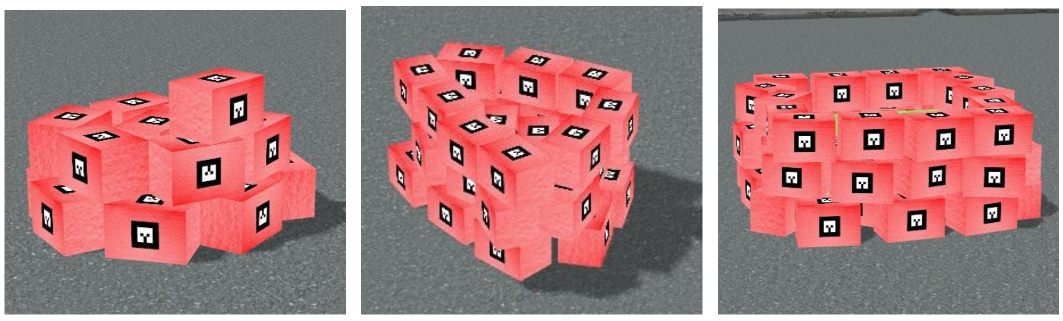

(*a*) Simulation case 1 (*b*) Simulation case 2 (*c*) Simulation case 3

Figure 4.4: Three Gazebo simulation scenarios.

4.2.3.1 Ablation Study

The importance of each factor in the NBV selection scheme was analysed. For each trial, a factor is removed from the framework. The performance of the updated framework is then recorded and shown in Table 4.1. Each variant is run in a fixed time interval, and the average number of picked bricks, map coverage percentage, number of objects which fell, and collisions between robot and environment were collected.

4.2.3.2 Benchmark Test

Benchmark testing was performed by comparing task throughput and map coverage between the implemented framework and two other frameworks. Three test scenarios were created, as shown in Figure 4.4. For each scenario, the framework and two other frameworks are executed five times each. The other frameworks are: *(1)* Octomap + [17] (dynamic) + [73] (discrete gradient frontier), and *(2)* Octomap + [17] (dynamic) + random (RDM) strategy.

4.2.4 Real-life Experiment

In the real-life experiment, two tests were performed to evaluate the accuracy of the dynamic GPIS mapping component and the effectiveness of NBV selection in the implemented framework.

4.3 Experimental Result

4.3.1 Simulated Experiment

4.3.1.1 Ablation Study

Table 4.1 shows the result of the ablation study. The "Pick Order" is important for maintaining a balanced pile shape. When this factor is removed from the framework, the brick pile is more likely to collapse, as seen in the high collapse count in the "Falls" column. The "Travel Distance" and "Frontier" are two factors related to the number of collisions when the mobile manipulator operates in the environment. The "Frontier" factor also affects the map coverage. Without it, the time required to achieve 100% map coverage is extended. The "Failure Penalty" factor prevents the mobile base from being stuck indefinitely. From Table 4.1, it is concluded that the complete utility function results in the best performance in all test categories.

Table 4.1: Ablation analysis on Utility factors.

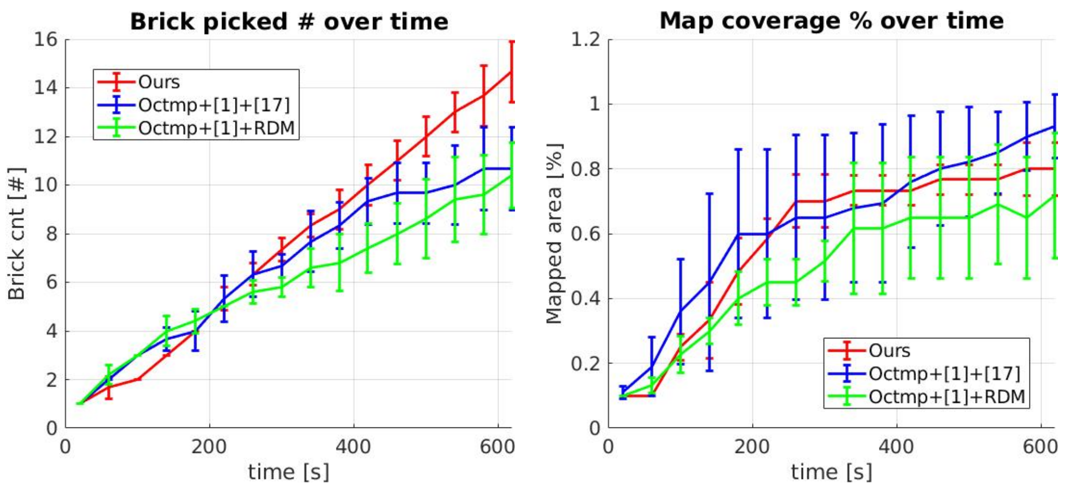
Part removed	Output			
	Bricks #	Map %	Falls #	Collision ?
NONE	**17**	**100**	**1**	**N**
Manipulability	14.5	100	2	N
Pick order	13	100	6	N
Travel distance	11	80	3	Y
Uncertainty	15	100	1	N
Frontier	16	90	4	Y
Failure penalty	7	60	1	N

4.3.1.2 Benchmark Test

Table 4.2 shows the results of the experiment, including the percentage of bricks picked and map coverage. The results indicate that the framework being implemented outperforms *(1)* and *(2)* in terms of task completion rate. It is observed in *(1)* and *(2)* that the robot frequently arrives at locations that have no pickable objects. This is explained by the GPIS-based manipulability factor being used in the framework, resulting in more promising locations for picking being chosen. Table 4.2 also shows that *(1)* has the best map coverage. However, *(1)* can occasionally navigate the mobile base into obstacle zones. On the other hand, since *(2)* tends to stay in explored areas, it has the lowest map coverage. The execution progress of the three frameworks for test case 2 is illustrated in Figure 4.5. The implemented framework's approach has the best performance for the object picking task and equivalent performance for map coverage, compared to Octomap+[17]+[73], but with smaller variations over the different runs.

Table 4.2: Benchmark Test

Simulation case	Implemented Framework [15]		*(1)* Octomap+[17]+[73]		*(2)* Octomap+[17]+RDM	
	Bricks%	*Map%*	*Bricks%*	*Map%*	*Bricks%*	*Map%*
#1	**95**±5.0	**100**±10	63±17	**100**±5.0	45±12	90±10
#2	**100**±6.0	82±8.5	72±8.0	**85**±9.0	61±12	71±19
#3	**100**±3.0	61±3.5	66±13	**70**±16	57±6.8	57±12



(a) Objects picked (b) Map coverage %

FIGURE 4.5: Comparison of task and map coverage for simulation case 2.

4.3.2 Real-life Experiment

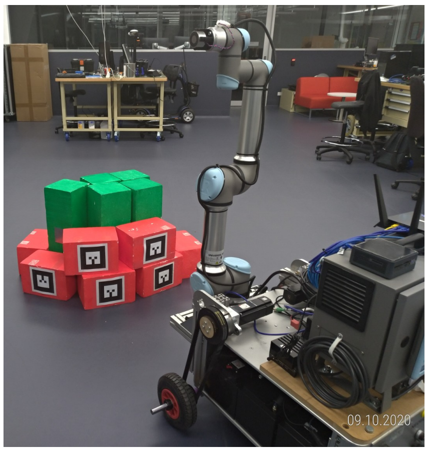
4.3.2.1 Dynamic GPIS Accuracy

A real-life scene Figure 4.6(a) is set up to evaluate the accuracy of dynamic GPIS by comparing with the methods of [16] and [17]. A map is initially generated using the Dynamic GPIS with a 2-step process: initialise using the original scene (b), then update after a brick is removed (c). For comparison purposes, a map is generated by feeding the depth image in step 2 directly to [16]. The signed distance values in the two maps are compared using CloudCompare [2] (d), and the error was found to be insignificant (e). Further, the same procedure is applied for the Octomap variant [17], and results from the implemented framework are shown to be superior (e). For the fairness of comparison, the leaf size of Octomap is chosen to be 0.03m, which most resembles the configuration of the Octree in this Dynamic GPIS.

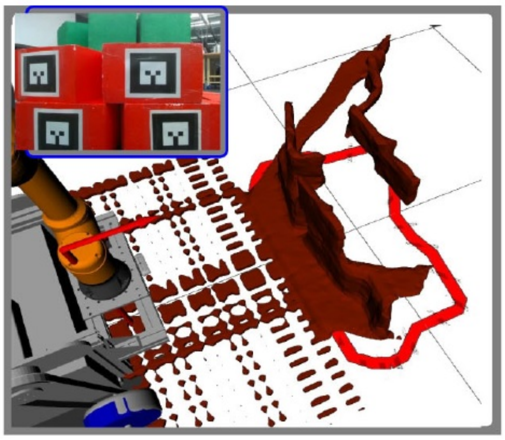
4.3.2.2 Next Best View Test

In the real-world test scene 2 (Figure 4.7(a)), the effectiveness of the NBV selection is evaluated. The scene is set to have two rows of bricks with the top two lying side by

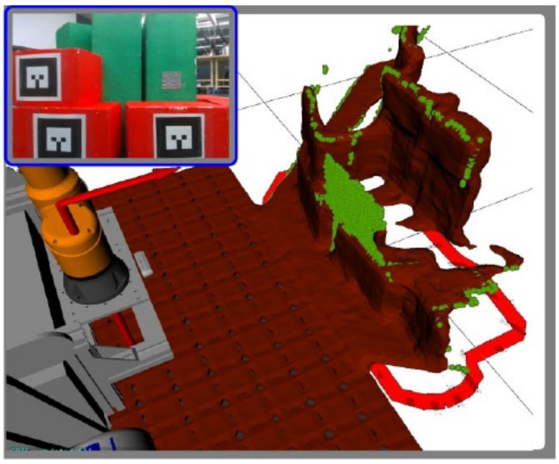
[2]CloudCompare project official website: `https://https://www.cloudcompare.org/`.



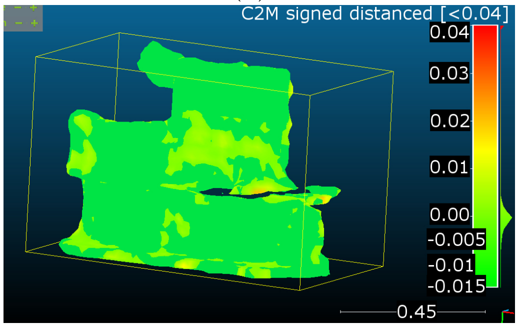
(a) Scene set up



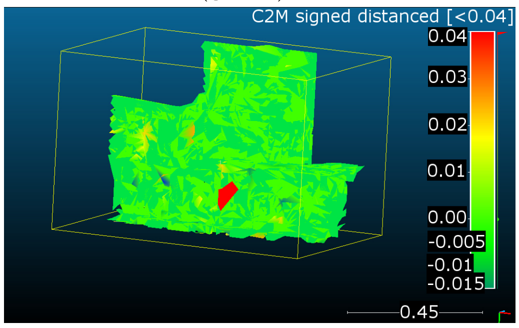
(b) Initial GPIS



(c) Remove brick, delete GPIS samples (green)



(d) Dynamic GPIS map vs [16], (note the smooth surface), obtain distance errors: $\mu_{dist} = 0.005$ cm, $\sigma_{dist} = 0.012$



(e) Apply [17], obtain before and after Octomaps, (note the artifacts), obtain distance errors: $\mu_{dist} = 0.033$ cm, $\sigma_{dist} = 0.015$

FIGURE 4.6: Real-life experiment: comparing map accuracy between ours, [16] and [17].

side. The robot's starting position is set to face the top two bricks (b). The robot then proceeded to detect and pick the top left brick and update the map. The NBV marker is

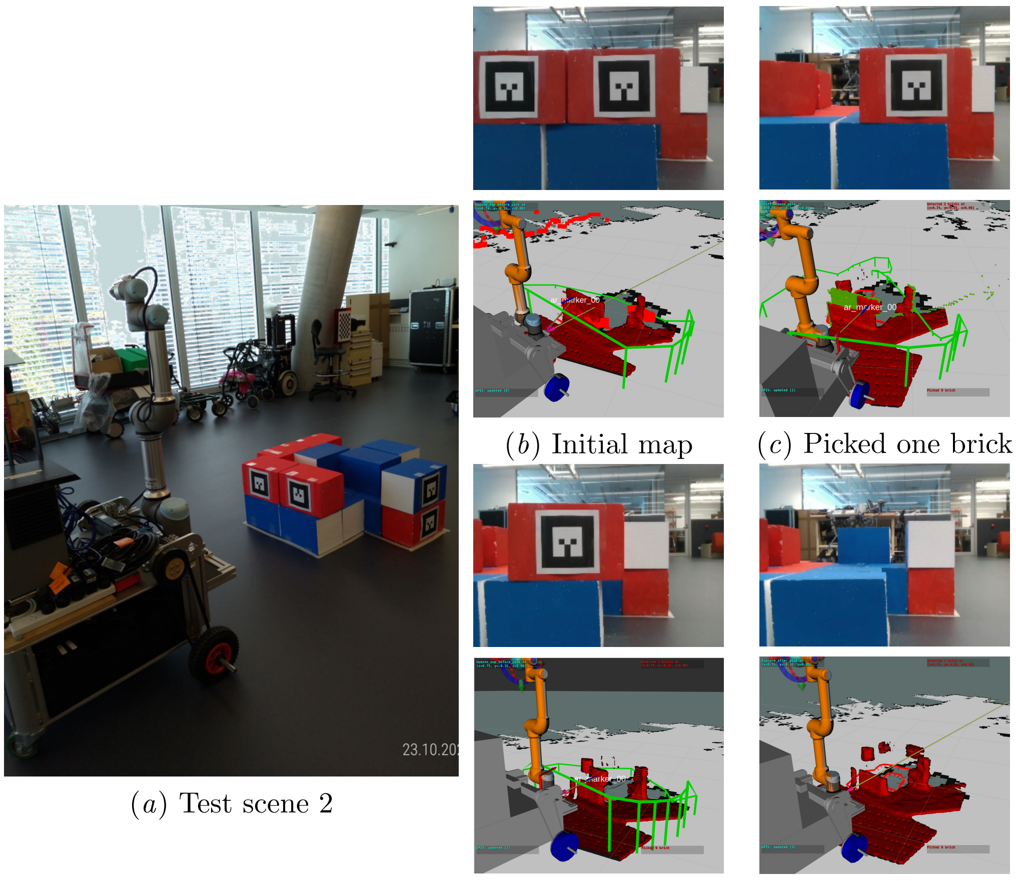

(*a*) Test scene 2 (*b*) Initial map (*c*) Picked one brick (*d*) Moved to NBV (*e*) Picked another brick

FIGURE 4.7: Real-life experiment, active and interactive mapping cycle.

calculated based on the updated map and shifts towards the high brick on the right (*c*). The robot then moved to the NBV position (*d*). Finally, the right brick is picked up from the wall (*e*). This shows the NBV strategy presented in [15] behaved in the desired order.

4.4 Conclusion

This chapter presents the experimental evaluation conducted and the experimental results of implementing an interactive and active mapping framework based on dynamic GPIS for a mobile manipulator platform. Due to the framework's probabilistic nature, the map could update immediately to reflect the dynamically changing environment. The experimental results have shown that the NBV selection scheme of the framework balances the needs of information gain in visited regions, frontier-driven map expansion, and object manipulability. The results also demonstrate the framework's ability to simultaneously navigate and modify the environment with the generated dense map. Both simulations and real-world experiments show that the framework introduced in [15] outperforms other frameworks, and can efficiently explore and interact with different objects in an environment. The framework utilised 3D sensor data to generate and update the map for a mobile manipulator robot that performs pick-and-place tasks. For practical reasons, computer vision can be used to obtain useful information to facilitate human interactions in complex environments. In the following chapter, two systems are presented for object identification using 3D sensor data in both simulated and real-world environments.

Chapter 5

Perception and Estimation

5.1 Introduction

The interactive probabilistic mapping framework for a mobile manipulator conducting a pick-and-place task presented in Chapter 4 did not fully utilise the 3D sensor data available. The sensor data obtained from the RGB-D camera was used mainly to generate the 3D occupancy map. However, various processes can be included in the framework to analyse the scene and provide meaningful information. This chapter introduces two systems that use the point cloud data provided by RGB-D cameras to perform object detection, pose estimation and scene overlay annotation. The first system detects and estimates the poses of rectangular surfaces. The second system extends on the first, detecting all objects above the ground surface. The systems presented in this chapter are closely based on the author's published work [129][130][131].

The first system (System 1) [129] focuses on solving the rectangular-shaped object pose estimation in the robotic bin-picking problem, using data from a single RGB-D camera collecting point cloud data from a fixed position. The key benefit of this system is its ability to accurately and robustly locate an object's position and orientation, which allows for high-precision robotic grasping and placing of such objects. Firstly, the intelligent grasping surface selection is performed, then PCA is used for pose estimation, and finally, rotation averaging is integrated to significantly reduce the data noise. Comparisons between the resulting poses and ones estimated by integrating the traditional ICP algorithm have demonstrated the system's advantages for pose estimation tasks.

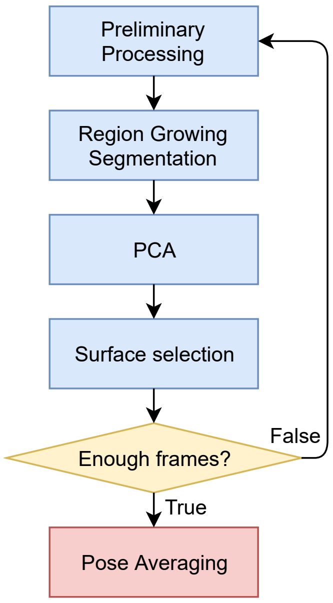
The second system (System 2) aims to detect multiple objects above the ground level and highlight them within the VR environment [130]. Similar to the first system, the noise in the raw data is minimised through preliminary processing. Then RANSAC [132] is implemented to remove all points in the background, including the ground plane. Afterwards, points are grouped based on their Euclidean distance to create clusters. Since each cluster represents an individual object, bounding boxes are generated for all distinct clusters. The boxes are defined by eight corners and are presented to the user in a VR environment. Simulated experiments have been conducted to validate the accuracy of the bounding boxes generated by System 2. Additionally, experiments were conducted to verify the applications of the background removal and clustering processes in the system in simulated and real-world scenarios.

5.1.1 Rectangular-shaped Object Pose Estimation

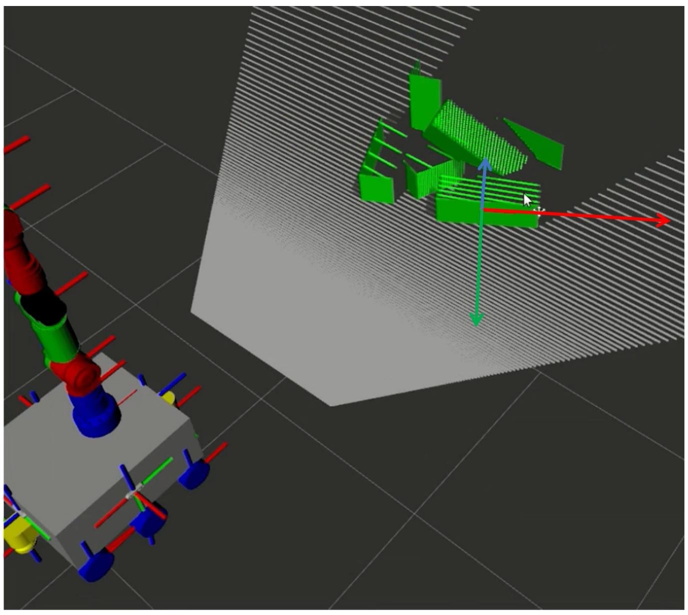
Recent years have seen increased interest in autonomous field operation, aided by high-precision perception capabilities [126] [37] [133]. This system was initially motivated by the 2020 Mohamed Bin Zayed International Robotics Challenge (MBZIRC). An objective in MBZIRC is to utilise an Unmanned Ground Vehicle (UGV) to autonomously build a pre-designed structure in a short time frame, which requires demonstrating robust and precise robotic pick-and-place capabilities.

This point cloud processing system (Figure 5.1) solves a key perception challenge for a UGV to accurately estimate an object's position for grasping and releasing from a single viewing perspective. The system proposed involves identifying the surface of a single rectangular prism-shaped brick within a pile of unorganised bricks of predefined colour and size and performing highly accurate pose estimation and noise reduction by exploiting a novel series of algorithms.

Due to the wall-building objective of the MBZIRC robot, the brick's estimated pose must match the brick's centre and natural orientation for accurate picking. Knowledge of this



(*a*)



(*b*)

FIGURE 5.1: System 1 overview and its demonstration. (*a*) System 1 overview. (*b*) Simulation of an arm-equipped robot detecting the best brick for picking from a pile. Chosen brick is labelled with RGB-axes.

pose can help plan a firm and robust grasping approach. Consequently, precise object placement for building construction becomes more straightforward. The added benefit of accurate pose estimation is that it allows for a less-constrained mechanical gripper design. On the other hand, data obtained from depth sensors typically contain large amounts of noise, which may hinder the system's performance. For outdoor applications, especially in places where sunlight is particularly strong, sensor noise may appear especially pronounced. The reason is that sunlight has a strong infrared (IR) light component that can severely interfere with a camera's IR stereo system. The orientation of the brick's surface generated by PCA will thus exhibit non-negligible variation from frame to frame. To limit the fluctuations of the estimated pose due to noise, multiple depth data sets can be collected over a short period of time, with pose estimation being performed for each time frame. By applying a novel averaging method, it is possible to achieve the best estimate of face orientation. Rotation averaging is thus a key component of the presented high-precision pose estimation system.

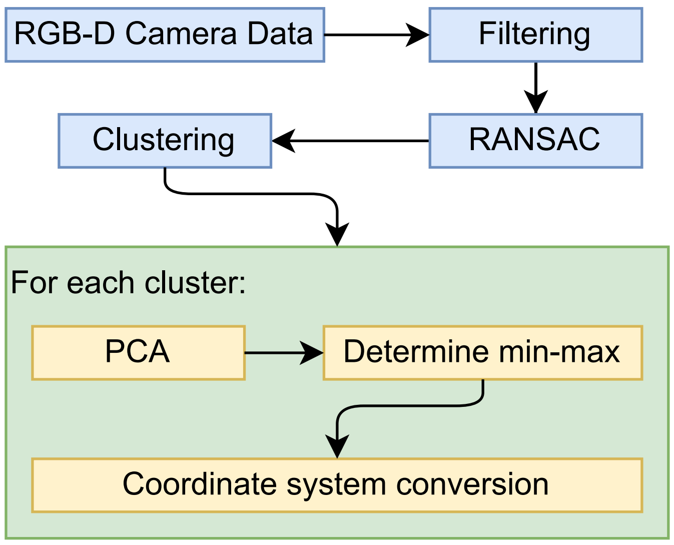


Figure 5.2: System 2 overview.

5.1.2 Overlay Bounding Box Annotation

The point cloud processing system, shown in Figure 5.2, combines multiple algorithms to produce bounding boxes that highlight objects above the scene's ground surface [130]. The system builds upon the author's published work [129], described in detail in 5.2.2, in combination with additional algorithms from PCL [2]. The system utilises the point cloud that is generated by multiple RGB-D cameras and can be received directly without additional processing steps. Concurrently, the point cloud rendered in the VR environment is generated by fusing colour and depth images. This system forms a core part of the second user study that will be presented in Chapter 6. Assumptions that were made about the scene in the FOV from the cameras are listed below:

- The multiple cameras must be extrinsically calibrated, and an overlap between the cameras' FOV must exist to implement the point cloud processing system
- The surface on which the objects to be detected are placed must comprise a large portion of the combined FOV of the cameras.

5.2 Methodology

The first system processes point cloud data received from an RGB-D camera to estimate the pose of a rectangular-shaped object robustly. The collected point cloud is first simplified with a voxel grid filter [134]. Then, it is filtered by distance, colour and number of points. The filtered cloud is segmented into clusters of different surfaces, and PCA is applied to each surface for object recognition and pose estimation. Finally, rotation averaging is applied to the PCA outputs collected over multiple frames for result stabilisation and noise reduction. The second system extends on the first to detect all objects above the ground surface by initially preprocessing the raw point cloud from the RGB-D camera using down-sampling and distance filtering to obtain a filtered point cloud. This filtered point cloud is then used to identify surfaces and generate bounding boxes for individual objects using PCA.

5.2.1 Core Background Fundamentals

5.2.1.1 Region Growing Segmentation

Region growing is a class of algorithms that is best known for its "split and merge technique" [135]. Region growing can be used on point cloud data to extract clusters of surface regions [136].

Given a point cloud, P, the point, $\mathbf{p}_i$ has a normal vector, $\vec{\boldsymbol{n}}_i$ and a curvature value, c_i. All normal vectors are stored in the point normal set, $\mathbb{N}$ and curvature values in the point curvature set, $\mathbb{C}$.

Let $\mathbb{R}$ be the set of all regions extracted from the point cloud and $\mathbb{A}$ be the remaining points. Initially, $\mathbb{R} = \varnothing$ and $\mathbb{A} = \mathbb{P} \setminus \mathbb{R}$.

As long as $\mathbb{A} \neq \varnothing$, region growing will choose the starting point, p_c with the minimum curvature value in $\mathbb{A}$ to start the expanding process to determine the local region of that point.

The chosen point, $\mathbf{p}_c$ will then be added to the local region set, $\mathbb{R}_l$ and removed from $\mathbb{A}$.

$$\mathbb{R}_l = \mathbb{R}_l \cup \{\mathbf{p}_c\} \tag{5.1}$$

$$\mathbb{A} = \mathbb{A} \setminus \{\mathbf{p}_c\} \tag{5.2}$$

Let $\mathbb{N}$ be the set of neighbours around $\mathbf{p}_c$. Each point, $\mathbf{p}_n \in \mathbb{N}$, is checked to see if it satisfies a specific angle threshold, θ_{th} and curvature threshold, c_{th}. A qualifying point set, $\mathbb{Q}$ will be added to $\mathbb{R}_l$ and removed from $\mathbb{A}$.

$$\mathbb{Q} = \{\mathbf{p_n} \in \mathbb{N} \mid \theta_{th} \geq acos(\vec{\boldsymbol{n}}_{\mathbf{n}}.\vec{\boldsymbol{n}}_c),\ c_{th} \geq c_n - c_c\} \tag{5.3}$$

$$\mathbb{R}_l = \mathbb{R}_l \cup \mathbb{Q} \tag{5.4}$$

$$\mathbb{A} = \mathbb{A} \setminus \mathbb{Q} \tag{5.5}$$

where $\vec{\boldsymbol{n}}_n$ and c_n, $\vec{\boldsymbol{n}}_c$ and c_c are the normal vector and curvature values of points $\mathbf{p}_n$ and $\mathbf{p}_c$, respectively.

After all neighbours of $\mathbf{p}_c$ have been considered, $\mathbb{R}_l$ will be added to $\mathbb{R}$. Then the system will choose another point in $\mathbb{A}$ and start the expanding process again until $\mathbb{A} = \varnothing$.

$$\mathbb{R} = \mathbb{R} \cup \mathbb{R}_l \tag{5.6}$$

5.2.1.2 Principal Component Analysis

PCA is a multivariate technique with the goal being to extract and present important data as principal components [137]. Initially, PCA was designed for digital image processing but was later extended to be implemented for point cloud data [138].

In a point cloud, P with N points, each point, $\mathbf{p}_i$ is presented as $[x_i, y_i, z_i]^T$ in the three-dimensional coordinate system. P can be represented as a N columns matrix:

$$\begin{aligned} &\mathrm{P} = [\,\mathbf{p}_1,\, \mathbf{p}_2,\, ...,\, \mathbf{p}_i,\, ...,\, \mathbf{p}_N\,] \in \mathbb{R}^{3\times \mathrm{N}} \\ &\mathbf{p}_i = [\,x_i, y_i, z_i\,]^T \in \mathbb{R}^{3\times 1}, \\ &\quad i \in \mathbb{Z}, N \geq i \geq 0 \end{aligned} \tag{5.7}$$

The empirical mean, $\bar{\mathbf{m}} = [x_m, y_m, z_m]^T$ of P can be calculated as,

$$\bar{\mathbf{m}} = \frac{1}{N}\sum_{i=1}^{N}[x_i, y_i, z_i]^T \tag{5.8}$$

It is then possible to obtain a zero-centred deviation matrix, $\mathrm{D} \in \mathbb{R}^{3\times \mathrm{N}}$ by subtracting $\bar{\mathbf{m}}$ from every point in P,

$$\mathrm{D} = \mathrm{P} - \bar{\mathbf{m}}\,\mathrm{I}_{3\times \mathrm{N}} \tag{5.9}$$

The co-variance matrix, $\mathrm{C} \in \mathbb{R}^{3\times 3}$ can then be calculated as,

$$\mathrm{C} = \frac{1}{N-1}\mathrm{D}\mathrm{D}^T \tag{5.10}$$

C is now symmetrical, and by applying eigendecomposition, the eigenvalues and eigenvectors of C are calculated,

$$\begin{aligned} &\mathrm{C} = \mathrm{V}\Lambda\mathrm{V}^T, \\ &\Lambda = \mathrm{diag}(\,\lambda_1, \lambda_2, \lambda_3\,),\ \lambda_1 > \lambda_2 > \lambda_3, \\ &\mathrm{V}\,\mathrm{V}^T = \mathrm{I}_{3\times 3} \end{aligned} \tag{5.11}$$

where Λ is a diagonal matrix of dimension 3×3, with the eigenvalues along the diagonal sorted in descending order, and V is a unitary matrix whose columns are the eigenvectors corresponding to the eigenvalues.

This operation is similar to performing Singular Value Decomposition (SVD) on D as,

$$\begin{aligned} \mathrm{D} &= \mathrm{U}\Lambda^{\frac{1}{2}}\mathrm{V}^{T}, \\ \Lambda^{\frac{1}{2}} &= \mathrm{diag}(\,\lambda_1^{\frac{1}{2}}, \lambda_2^{\frac{1}{2}}, \lambda_3^{\frac{1}{2}}\,) \end{aligned} \tag{5.12}$$

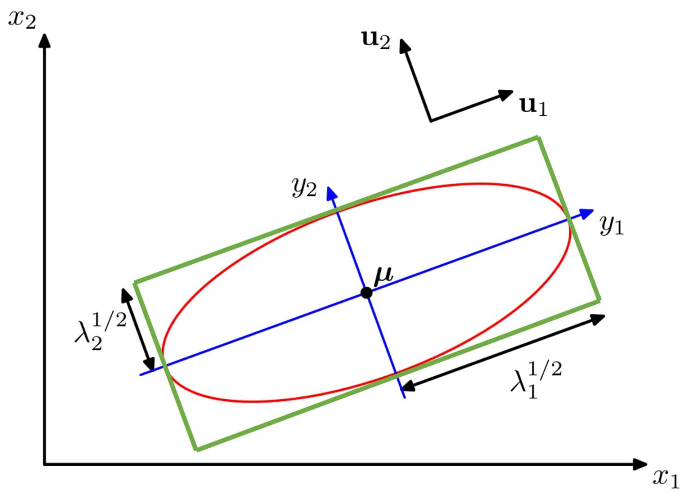


FIGURE 5.3: PCA reveals the natural distribution of normally distributed data [18]. Here eigenvectors are labeled as $\mathbf{u}_1$ and $\mathbf{u}_2$. The data mean is located at the centre of the surface. Standard deviations along each axis are the singular values $\lambda_1^{\frac{1}{2}}$ and $\lambda_2^{\frac{1}{2}}$. The ratio of length-to-height is $\lambda_1^{\frac{1}{2}} : \lambda_2^{\frac{1}{2}}$.

The singular vectors and eigenvectors from both operations are the same. The difference is that the singular values from SVD are square roots of the corresponding eigenvalues. This is because the eigenvectors represent the underlying orthogonal axis of uncorrelated data distribution. For a symmetrical geometrical shape, the eigenvectors give the right choice of shape axis, and the empirical mean coincides with the shape's centroid. In a cloud patch, the PCA axis should be aligned with the rectangle's natural shape axis, and the PCA mean is at the rectangle's centre. This concept is illustrated in Figure 5.3.

Moreover, the singular values are the standard deviation of data distributions along each latent axis. The singular value ratios should indicate the length-to-height ratio of the rectangular brick surface, with each class of brick having a unique length-to-height ratio (see Table 5.4 for details). Therefore, it is possible to classify the brick type by computing this singular value ratio, or eigenvalue ratio. Performing eigendecomposition on a symmetric matrix is less computationally expensive than SVD. Thus, PCA is selected instead of direct SVD.

5.2.1.3 Rotation Averaging

To address the challenge of orientation robustness, earlier works have proposed rotation averaging least square methods [139], rotation averaging for DNA helix [140], and large-scale rotation averaging [141]. [142] provided a thorough review of rotation averaging methods.

The methods can be categorised by the representation domain they operate on. Rotations can be represented in the angle-axis form in the Euclidean domain,

$$\mathbf{r} = \phi\mathbf{u}, \qquad \phi \in \mathbb{R},\ \mathbf{u} \in \mathbb{R}^3,\ \mathbf{u}\cdot\mathbf{u}^T = 1 \tag{5.13}$$

where vector, $\mathbf{r}$ is a rotation of angle, ϕ about an axis in $\mathbf{u}$. The definition is illustrated in Figure 5.4 [19]. Averaging in this context essentially means finding the algebraic mean for $\mathbf{v}$. Rotations can also be represented in the Manifold domain of $\mathbb{SO}(3)$ (the group of all 3D rotations), expressed as 3×3 rotation matrix R or unit quaternion, $\mathbf{q}$. The unit quaternion is defined as,

$$\mathbf{q} = (\,\cos(\theta), \mathbf{u}\sin(\theta)\,), \qquad \theta = \frac{1}{2}\phi \tag{5.14}$$

The set of all unit quaternions forms a unit sphere, S^3 in $\mathbb{R}^4$. Figure 5.5 [19] illustrates rotation in the unit-quaternion form. Averaging can be done as a Chordal L_2-mean in the $\mathbb{SO}(3)$ domain.

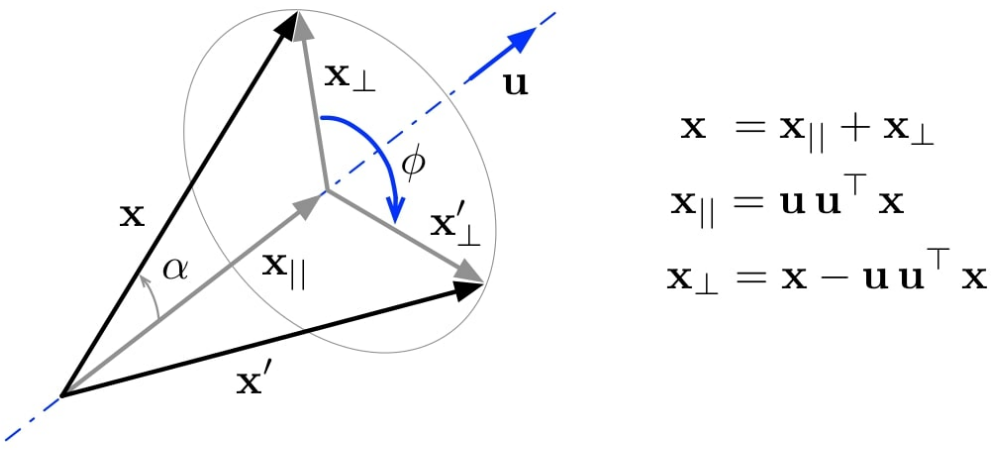

$\mathbf{x}_\perp$
$\mathbf{u}$
ϕ
$\mathbf{x}$
$\mathbf{x}'_\perp$
α
$\mathbf{x}_{||}$
$\mathbf{x}'$
$\mathbf{x} = \mathbf{x}_{||} + \mathbf{x}_\perp$
$\mathbf{x}_{||} = \mathbf{u}\,\mathbf{u}^\top\mathbf{x}$
$\mathbf{x}_\perp = \mathbf{x} - \mathbf{u}\,\mathbf{u}^\top\mathbf{x}$

FIGURE 5.4: Rotation of a vector, $\mathbf{x}$ about the axis, $\mathbf{u}$ by an angle, ϕ [19].

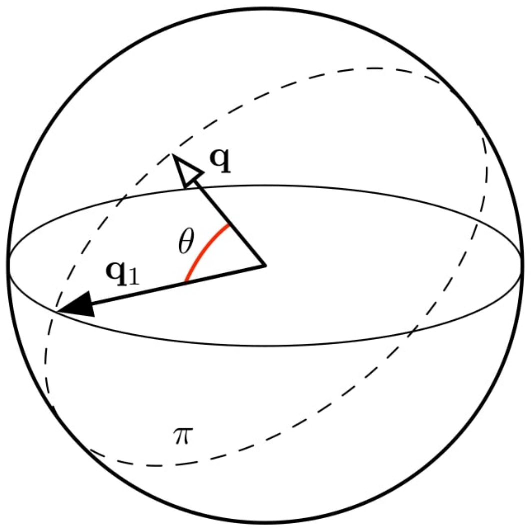


FIGURE 5.5: In unit 3-sphere manifold, quaternion $\mathbf{q}$ defines an angle, $\theta = \frac{1}{2}\phi$ with a unit quaternion, $\mathbf{q}_1$ [19].

The angle-axis rotation representation is inherently error-prone as it allows representation ambiguity. Multiple angle-axis values can all represent the same orientation. For example, the aforementioned angle-axis, $\mathbf{r}$ can also be written as $(2\pi - \phi)(-\mathbf{u})$. As another example, a random angle, ϕ with a zero mean, may show a small magnitude due to noise: $0 < \phi < \epsilon$. Similarly, with a small perturbation, it can swing in the opposite direction and approach the maximum angle, $\phi \to 2\pi$. Averaging over such values will generally lead to a meaningless orientation. For this reason, the $\mathbb{SO}(3)$ form, due to its unique representation, has been chosen. In the implementation, the unit quaternion is used for rotation parameterisation in $\mathbb{SO}(3)$ and performs averaging by minimising the chordal error.

Alternatively, rotation averaging can also be viewed from the perspective of the optimisation method in use. Some $\mathbb{SO}(3)$ based methods may take an iterative form to search where many steps are required to achieve solution convergence. There are also more elegant methods that exploit simple linear algebra techniques if a closed-form solution is theoretically guaranteed. For simplicity and accuracy, this system incorporates the simple "Chordal L_2-Mean method"[142].

5.2.2 System 1: Rectangular-shaped Object Pose Estimation

5.2.2.1 Filtering Strategies to Improve Object Detection in Noisy RGB-D Point Cloud Data

Firstly, consider a fixed RGB-D camera, a robotic arm and the relative location of the camera in the robot's frame. Data collected from the camera is a colour point cloud, and all objects in the field of view are immobile. Let the robotic arm's base be the global frame, the camera's position and orientation be represented by a 3×1 translation vector, $\mathbf{t}$, and a 3×3 rotation matrix, R:

$$\mathbf{t} = \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix}, \mathrm{R} = \mathrm{R}_z(\gamma)\mathrm{R}_y(\beta)\mathrm{R}_x(\alpha) \tag{5.15}$$

where R_x, R_y, R_z are the rotation matrices about axes x, y, z for roll, pitch, yaw angles α, β, γ, respectively.

Note that the point cloud data, P_c coming directly from the RGB-D camera, is noisy and contains much information to be processed. In this system, a few layers of filters are implemented to improve the quality of the received point clouds. Initially, the data from the camera is run through a voxel grid filter to reduce the number of points to a preferred resolution. Voxel grid filtering also guarantees points in the cloud are evenly distributed.

The down-sampled point cloud, P_d is then filtered by the Z-axis value of each point. Since noisy data cannot be processed and points far away from the camera are generally noisier, all $\mathbf{p}_i$ that exceed a distance limit, z_{limit} will be removed. Then,

$$\mathrm{P}_f^d = \{\mathrm{P}_d \setminus \mathbf{p}_i \mid \mathbf{p}_i \in \mathrm{P}_d, z_i \geq z_{limit}\}, \tag{5.16}$$

where P_f^d is the point cloud after filtering by distance. This point cloud is further filtered by the colour of the object of interest. This filter layer's purpose is not to recognise the object but to limit the range of data to be considered for later steps. Therefore, the colour range does not have to be strict. The resulting cloud, P_f^c can be defined as,

$$\mathrm{P}_f^c = \{\mathrm{P}_f^d \setminus \mathbf{p}_i \mid \mathbf{p}_i \in \mathrm{P}_f^d,\ h_{lowlim} \geq h_i \geq h_{hilim}\} \tag{5.17}$$

where the point colour under consideration is described in the Hue Saturation Value (HSV) colour space and h_{hilim} and h_{lowlim} are the high limit and low limit of the “Hue” value of the point.

5.2.2.2 Region Growing Segmentation using Normal Vectors

Region growing segmentation is then implemented on P_f^c. Region growing segmentation requires each point in P_f^c to have a corresponding normal vector. These normal vectors are computed for each point by computing the covariance matrix of the query point and its neighbours. The C of $\mathbf{p}_i$ can be obtained using the following equation,

$$\mathrm{C} = \frac{1}{k}\sum_{i=1}^{k}(\mathbf{p}_i - \bar{\mathbf{p}})(\mathrm{p}_i - \bar{\mathbf{p}})^T,\ \mathrm{C}\vec{\boldsymbol{v}}_{\boldsymbol{j}} = \lambda_j \vec{\boldsymbol{v}}_{\boldsymbol{j}},\ j \in \{0,1,2\} \tag{5.18}$$

where k is the number of neighbor points considered of $\mathbf{p}_i$, $\bar{\mathbf{p}}$ is the centroid of $\mathbf{p}_i$ and its neighbours, and λ_j and $\vec{\boldsymbol{v}}_j$ are the j-th eigenvalue and eigenvector of C.

5.2.2.3 Rectangular Object Recognition and Pose Estimation using PCA

Region growing segmentation results in a set of clusters, $\mathbb{C}_{rg}$. Each element in the cluster is a point considered to be on the same surface. The result from region growing segmentation is further filtered by the number of points, N_{th} in clusters. This step is designed to remove clusters that do not have enough points to produce an accurate final result.

$$\mathbb{C}_{rg}^f = \mathbb{C}_{rg} \setminus \mathbb{C}_{uq}, \tag{5.19}$$

where $\mathbb{C}_{rg}^f$ is the set of qualified clusters and $\mathbb{C}_{uq}$ is a set of clusters with less than N_{th} points. For surface, S_i from $\mathbb{C}_{rg}^f$, PCA is applied to obtain the principal axes X, Y and Z and the centroid.

It is necessary that the unit vector of the Z-axis, $\vec{\boldsymbol{u}}_i^z$ of the obtained cloud on S_i, points away from the camera's viewpoint. This requirement is checked by calculating the dot product, d_i of $\vec{\boldsymbol{u}}_i^z$ and the unit vector of the Z-axis of the camera, $\vec{\boldsymbol{u}}_{cam}^z$. Since all calculations so far are in the camera frame, $\vec{\boldsymbol{u}}_{cam}^z = (0, 0, 1)$. If $d_i \geq 0$, then $\vec{\boldsymbol{u}}_i^z$ is pointing away from the camera. However, if $0 \geq d_i$ then $\vec{\boldsymbol{u}}_i^z$ is pointing towards the camera and the coordinate system obtained from PCA will be turned by π around the X-axis.

The geometric attributes of S_i are considered in the rectangular object recognition task. A comparison between the known model edges length with surface S_i edges is executed to determine if S_i is a surface of the targeted rectangular prism-shaped object. The edge length calculations of S_i are based on the maximum distance, d_x^{max} and d_y^{max} from any point on S_i to the X and Y axes at the centroid of that surface.

The width, w and height, h of the surface will be twice the value of d_x^{max} and d_y^{max}. If the calculated values are similar to the model, S_i is a surface on the targeted rectangular object.

$$w = 2 \times d_x^{max}, \ h = 2 \times d_y^{max} \tag{5.20}$$

5.2.2.4 Best Surface Selection for Rectangular Prism-Shaped Objects using Sigmoid Scores

If S_i is a surface of the targeted rectangular prism-shaped object, it will then be stored in a set, $\mathbb{S}_r$ for the best surface selection process.

$$\mathbb{S}_r = \mathbb{S}_r \cup \{S_i\} \tag{5.21}$$

Each surface S_i of $\mathbb{S}_r$ contains a centroid point, $\mathbf{c}_i$ and the unit vectors of that surface's coordinate system. For best surface selection, the following parameters are considered: the distance of the centroid to the global frame or the robotic arm's base, d_i^g, the angle between S_i and the global frame, θ_i^g, and the number of points, N_i belonging to the surface. Each parameter is calculated and used as a metric in surface candidate selection. To combine all tests fairly with equal voting power, each raw value is mapped to the same score range

[0..1] using the sigmoid function [143] as,

$$\begin{aligned} s_i^d &= \frac{2}{1+\exp\frac{6(0.3-\mathrm{d})}{3}} - 1 \\ s_i^\theta &= \frac{2}{1+\exp\frac{6(0.0-\mathrm{a})}{1.0}} - 1 \\ s_i^N &= \frac{2}{1+\exp\frac{6(1000-\mathrm{ptc})}{20000}} - 1 \end{aligned} \tag{5.22}$$

where s_i^d, s_i^θ, and s_i^N are the scores for distance, angle, and number of points in the region, respectively. And d, a, and ptc are the distance, the angle and the number of points, respectively. The final score, s_i^f is the product of all three scores,

$$s_i^f = s_i^d * s_i^\theta * s_i^N \tag{5.23}$$

The product rule here is to ensure the candidate with a poor score in either criterion will receive the highest penalty. The surface with the highest score will be the best surface with well-balanced attributes. This surface is selected as the final output for subsequent processing.

5.2.2.5 Rotation Averaging for Surface Pose Estimation

The robot is stationary while capturing depth data from the camera. The best surface that is selected will always be the same for multiple runs, $[1..N_T]$ in this time interval. To further improve the result, an averaging calculation is applied over the calculated poses. The results of each run will be collected in $\mathbb{S}_T$. The average calculation will be executed for all elements of $\mathbb{S}_T$.

There are two components needed to be averaged: the centroid position and the orientation. Each surface, $S_t \in \mathbb{S}_T$ has a centroid, $\mathbf{c}_t$ and its orientation presented by the coordinate system with unit vectors: $\vec{\boldsymbol{u}}_t^x$, $\vec{\boldsymbol{u}}_t^y$, and $\vec{\boldsymbol{u}}_t^z$. The averaged centroid position $\mathbf{c}_{avg}$ will be calculated as,

$$\mathbf{c}_{avg} = \frac{1}{N_T}\sum_{t=1}^{N_T}\mathbf{c}_t \tag{5.24}$$

where N_T is the number of elements in the set $\mathbb{S}_T$.

The best orientation, R_{avg} from all results is achieved by solving the single rotation averaging problem as described in [142]. During single rotation averaging, several rotation estimates of a still target are computed over time and then averaged to give the best result [142], thereby reducing the output pose noise. To compute the optimal rotation, a rotation difference metric is defined according to [142]

$$\begin{aligned} \text{Let } & d_{chord}(\mathbf{p},\,\mathbf{q}) = \|\mathbf{p}\cdot\mathbf{q}^{-1} - \mathbf{e}\| \\ \text{where, } & \mathbf{p}\cdot\mathbf{q}^{-1} \in \mathbb{R}^{4\times 1}, \quad \mathbf{e} = (1,0,0,0), \\ \text{define } & \mathbf{p}\cdot\mathbf{q}^{-1} = (\,\cos(\psi), \sin(\psi)\mathbf{u}\,) \\ \text{therefore: } & \|\mathbf{p}\cdot\mathbf{q}^{-1} - \mathbf{e}\| = 2\sin(\psi/2). \end{aligned} \tag{5.25}$$

This derivation suggests the distance between two unit 4-vectors, also known as the chordal distance, is directly related to their separation angle, ψ. [142] then uses the chordal distance as the error metric to define a cost function, $C(\mathrm{R}_{avg})$ under L_2 chordal distance as,

$$\begin{aligned} C(\mathrm{R}_{avg}) &= \sum_{t=1}^{N_T} d_{\text{chord}}(\mathrm{R}_i,\ \mathrm{R}_{avg})^2 \\ &= \sum_{t=1}^{N_T} d_{\text{chord}}(\mathbf{q}_i,\ \mathbf{q}_{avg})^2 \end{aligned} \tag{5.26}$$

where R_t is a rotation of a surface, $S_t \in \mathbb{S}_T$ and $\mathbf{q}_t$ is R_t's $\mathbb{SO}(3)$ equivalent in unit quaternion form.

The rotation averaging problem is then to find R_{avg} such that $C(\mathrm{R})$ is minimised. To this end, the rotation matrix, $\mathrm{R}_t \in \mathbb{R}^{3\times 3}$ is converted to the unit quaternion form: $\boldsymbol{q}_t = [q_w, q_x, q_y, q_z]^T \in \mathbb{R}^{4\times 1}$ and $\|\boldsymbol{q}_t\| = 1$.

$$\begin{aligned} q_w &= \sqrt{1 + \mathrm{R}_{00} + \mathrm{R}_{11} + \mathrm{R}_{22}}\,/\,2 \\ q_x &= (\mathrm{R}_{21} - \mathrm{R}_{12})\,/\,(4\,q_w) \\ q_y &= (\mathrm{R}_{02} - \mathrm{R}_{20})\,/\,(4\,q_w) \\ q_z &= (\mathrm{R}_{10} - \mathrm{R}_{01})\,/\,(4\,q_w) \end{aligned} \tag{5.27}$$

In the next step, the matrix, A $\in \mathbb{R}^{4\times4}$ is constructed as the sum of the dyadic product of $\boldsymbol{q}_t\boldsymbol{q}_t^T$ over time.

$$\mathrm{A} = \sum_{t=1}^{N_T} \boldsymbol{q_t}\boldsymbol{q_t}^T \tag{5.28}$$

Hartley et al., [142] provided proof that the eigenvector, $\boldsymbol{s}_{max}$ of the matrix, A corresponding to its largest eigenvalue minimises the cost function in (5.26). Thus, $\boldsymbol{s}_{max}$ is the averaged rotation in the optimal sense.

5.2.3 System 2: Overlay Bounding Box Annotation

The raw point cloud, P_r received directly from the RGB-D camera contains millions of points with noise that proportionally increases with the distance between objects and the camera lens. Due to the overwhelming amount of information and noise in P_r, preliminary processes need to be applied. Consequently, P_r is down-sampled with the voxel grid filter from PCL and additionally, a distance threshold is applied to filter points that are outside of the required bounds. The distance filter aims to remove noisy data recorded from distanced objects [129].

After filtering, the point cloud, P_f remains from P_r. Due to the assumptions listed above, the surface where objects are located will be the surface determined by RANSAC [132]. The list of points that belong to this surface will be denoted as P_R.

Before the clustering step, all the points that reside on and below S_R relative to the camera position, c must be removed. The algorithm implemented to determine the point's relative position to a surface is shown in Algorithm 1. Let P_{temp} be the point cloud, P_f after removing P_R. The projection point of c onto S_R is denoted as p_j. The vector from the projected point to c is $\vec{v_{pc}}$. For every point, p_i in the P_{temp} vector, $\vec{v_{pp}}$ is the mapping from p_i to p_j. The relative position of point, p_i to a surface, S_R and c is determined based on the dot product dp of $\vec{v_{pc}}$ and $\vec{v_{pp}}$. If $dp < 0$, p_i and c are on the same side relative to S_R, or are on opposite sides if $dp > 0$. All points p_i that satisfy $dp < 0$ are stored in the point cloud, P_{ss}.

The Euclidean distance algorithm [144] is applied to P_{ss} to obtain a set that contains clusters of points. Each cluster, C_i represents an individual object on the surface, S_R.

Algorithm 1: Two points relative position to a surface

```
$P_{temp} = P_f \setminus P_R$ for $p_i \in P_{temp}$ do
    $dp = \vec{v_{pc}} \bullet \vec{v_{pp}}$ if $dp < 0$ then
        $p_i$ and $c$ are on the same side relative to $S_R$ $p_i \to P_{ss}$
    if $dp > 0$ then
        $p_i$ and $c$ are not on the same side relative to $S_R$
```

To generate a bounding box that encompasses each of the objects, a three-step process is implemented. The three steps involve the application of PCA, extracting the minimum and maximum values of the axes generated by PCA and converting the corners of the bounding box from the PCA coordinate frame to the camera's coordinate frame.

PCA is a multivariate statistical technique that extracts important information from an existing data set and represents the set accordingly. This results in a set of new orthogonal variables labelled principal components [145]. For each cluster, C_i, a three-dimensional coordinate system, O_i, can be derived from the eigenvectors obtained from the PCA technique. Each Cartesian point, $p_j^W \in C_i$ is converted from the camera coordinates, O_c, to point, p_j^O of coordinate O_i. The minimum and maximum values on each of the axes, $[x_{min}, x_{max}, y_{min}, y_{max}, z_{min}, z_{max}]$ among all points, p_j^O are utilized to define the set of corners of the bounding box, B_i^O, wrapping the cluster, C_i. The corner points are then converted from PCA coordinates, O_i, to the camera frame, O_c, for visualisation. The set of corner points in O_c coordinate system is denoted as B_i^W.

5.3 Experimental Result

5.3.1 System 1: Rectangular-shaped Object Pose Estimation

Two experiments were conducted to display the results of System 1. The first compared the pose estimation algorithm from System 1 with the well-known ICP algorithm [146]. The results are evaluated based on the ground truth data. The second experiment tested the ratio between eigenvalues and the size of multiple rectangular objects with different dimensions.

All experiments were conducted in the Gazebo simulation environment on a machine with an Intel Core i7-7700HQ CPU, 16 GB RAM, and Geforce GTX 1050 GPU.

5.3.1.1 Comparative Study: System 1 and ICP-integrated System

The simulation consists of a mobile base with a simulated model of the Realsense D435 attached. The mobile base will be placed in front of a pile of green bricks, which are a class of bricks with the same dimensions: $0.6 \times 0.2 \times 0.2$ meters. Point cloud data is obtained directly from the camera's point cloud topic. Figure 5.6 shows five different scenarios of the experiment. For each scenario, System 1 will be implemented to obtain the best surface for the picking task. ICP will then try to fit this surface into a generated model. This model is a cloud of a surface with the same colour and size as the brick. The model contains points evenly distributed, similar to the result after the voxel grid filter process. The results from the ICP-integrated system and System 1 are compared to the ground truth value. Figure 5.7 shows the segmented point cloud in each scenario. The bricks on which the selected surfaces belong can be observed. Then, the ground truth value of the surface can be determined from the Gazebo world model.

The result of each system is evaluated by the distance error and angular error. The distance error, e_d is calculated by the distance between the result's centroid and the ground truth, as,

$$e_d = ||\mathbf{c}_t - \mathbf{c}_g|| \tag{5.29}$$

where $\mathbf{c}_t$ is the centroid point position from either of the two systems, and $\mathbf{c}_g$ is the ground truth value.

The angular error, e_{th} is calculated by the difference between the Z-axis obtained from the result and the ground truth Z-axis.

$$e_{th} = cos^{-1}(\vec{\boldsymbol{u}}_r^z \cdot \vec{\boldsymbol{u}}_g^z) \tag{5.30}$$

where $\vec{\boldsymbol{u}}_r^z$ and $\vec{\boldsymbol{u}}_g^z$ are the unit vectors of the Z axes obtained from the roll, pitch and yaw values from the systems and the ground truth data. The unit vector of the Z-axis can be

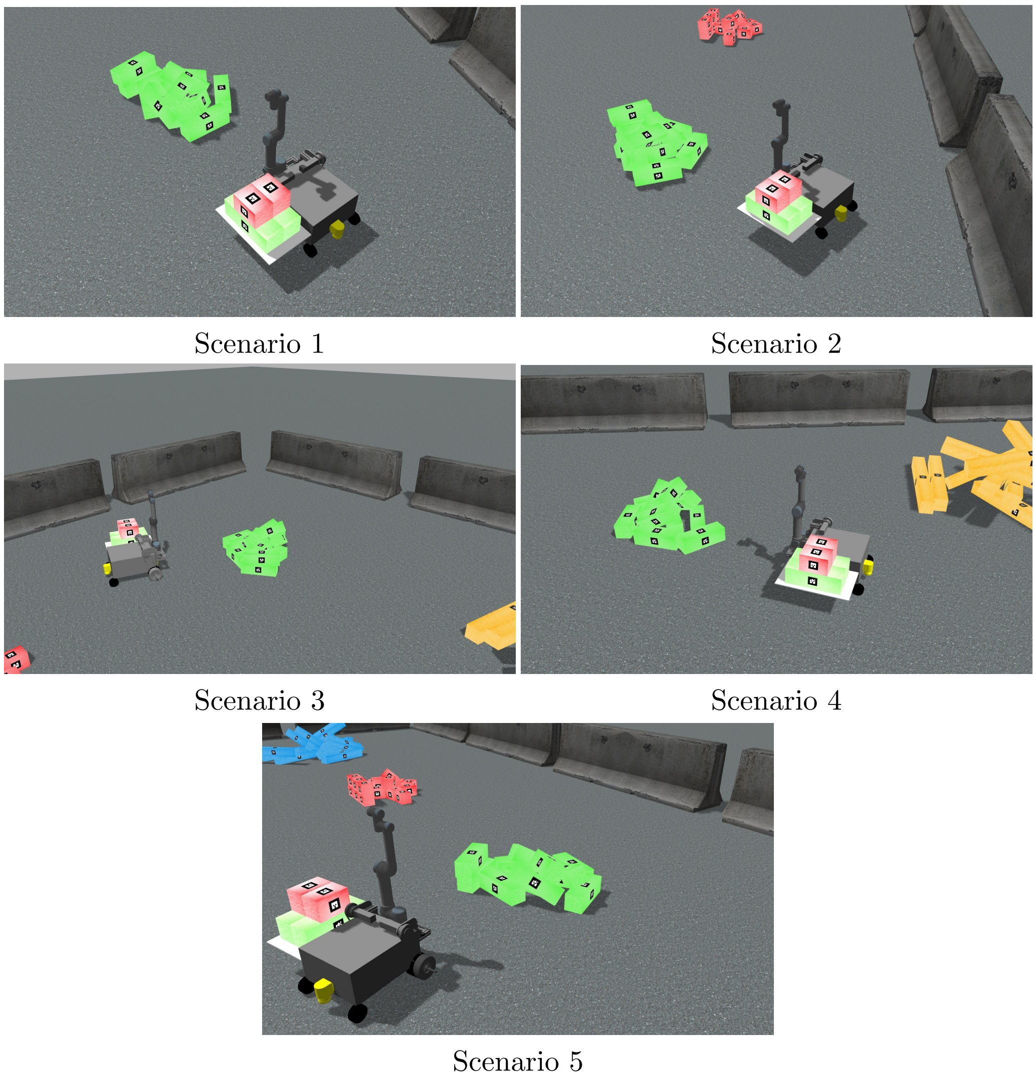


FIGURE 5.6: The five Experiment 1 scenarios.

calculated as,

$$\vec{\boldsymbol{u}}^z = \begin{pmatrix} \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma \\ \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma \\ \cos\beta\cos\gamma \end{pmatrix} \tag{5.31}$$

The comparative centroid estimation results are recorded in Table 5.1. Table 5.2 contains the estimated orientation and the calculated orientation error. Table 5.3 displays the execution time of each system in milliseconds and the ratio between the two systems' execution times.

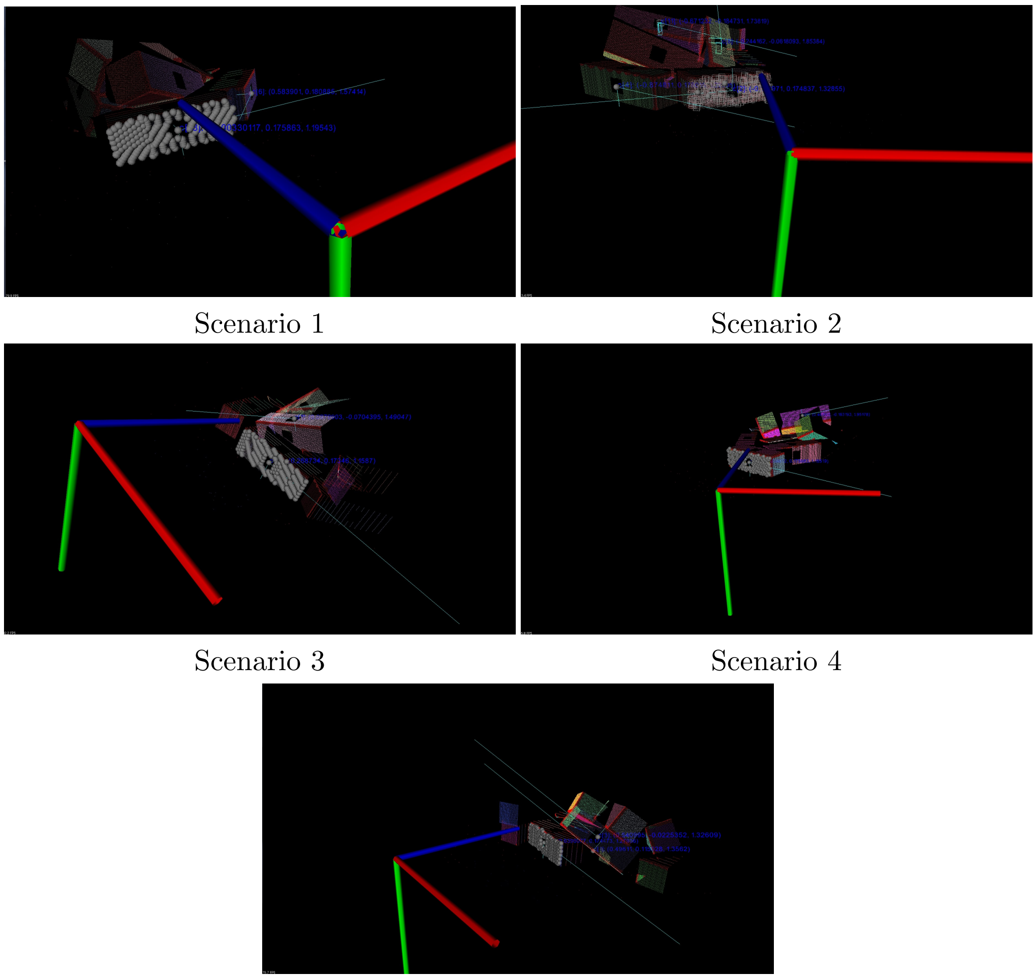


FIGURE 5.7: Segmented point cloud results from Experiment 1.

5.3.1.2 Ratio Between Eigenvalue and Object Dimensions

This experiment evaluates the relationship between the eigenvalue ratios and the dimension ratio for each type of brick. The simulation environment for this experiment is similar to the previous experiment, with two extra piles of blue and orange bricks added. The newly added blue bricks and orange bricks, where the dimensions are $1.2 \times 0.2 \times 0.2$ meters and $1.8 \times 0.2 \times 0.2$ meters, respectively. The mobile base is placed at multiple points of view for each pile. PCA is then executed on the point cloud that is recorded from each perspective. The two larger eigenvalues, corresponding to the length and height of a surface, are considered. Each brick colour has a different length-to-height ratio. Therefore,

Table 5.1: A comparison of the centroid estimation accuracy between the ICP-integrated system and System 1.

Ground Truth Centroid (m)			Output Centroid (m)						Error (m)	
			ICP			Proposed			ICP	Proposed
x	y	z	x	y	z	x	y	z		
-5.3	5.5	0.1	-5.4	5.5	0.1	-5.4	5.5	0.1	0.05	0.05
-5.7	4.9	0.1	-5.9	4.6	0.1	-5.7	4.9	0.1	0.31	0.01
-5.2	4.3	0.1	-5.7	4.5	0.1	-5.2	4.3	0.1	0.31	0.04
-4.6	4.2	0.1	-4.5	4.3	0.1	-4.5	4.2	0.1	0.13	0.06
-4.6	4.8	0.1	-4.5	4.7	0.1	-4.6	4.8	0.1	0.08	0.06
Average			-5.2	4.72	0.1	-5.08	4.74	0.1	0.176	0.044
Standard Error			0.297	0.206	0	0.231	0.233	0	0.056	0.009
Root Mean Square Error			0.253	0.173	0	0.077	0	0	N/A	N/A

Table 5.2: A comparison of the orientation estimation accuracy between the ICP-integrated system and System 1.

Ground Truth Orientation (radian)			Output Centroid (radian)						Error (radian)	
			ICP			Proposed			ICP	Proposed
x	y	z	x	y	z	x	y	z		
1.6	0	0.5	-1.6	0	-2.7	1.6	0	0.5	0.06	0
0	0	1.7	-1.6	0	-1.7	1.6	0	1.5	0.16	0.02
-1.6	0	2.8	-1.6	0	-0.3	-1.6	0	-0.3	0.06	0.06
0	0	0.6	-1.6	0	-2.6	-1.6	0	0.6	0.06	0.01
3.1	0	-1.5	-1.6	0	1.7	-1.6	0	1.7	0.06	0.06
Average			-1.6	0	-1.12	-0.32	0	0.8	0.08	0.03
Standard Error			0	0	0.826	0.784	0	0.363	0.02	0.013
Root Mean Square Error			2.737	0	3.221	2.333	0	1.994	N/A	N/A

Table 5.3: A comparison of execution time (in seconds) between the ICP-integrated system and System 1.

ICP	Proposed	Ratio
33	496	15.03
48	485	10.10
41	434	10.59
53	338	6.38
33	365	11.07

the ratio of the two larger eigenvalues obtained from PCA is expected to change between different bricks exponentially.

The experiment data are recorded in Table 5.4.

TABLE 5.4: A comparison of the brick's PCA ratio with size ratio. $(\frac{L}{H})^2$ refers to the length-to-height ratio squared, and λ_i refers to the i'th eigenvalue from PCA.

Brick colour	Expected dimension	Expected $(\frac{L}{H})^2$	Test$^{(1)}$: λ_1/λ_2 Test$^{(2)}$: λ_1/λ_2 Test$^{(3)}$: λ_1/λ_2	$= \lambda_{ratio}^{(1)}$ $= \lambda_{ratio}^{(2)}$ $= \lambda_{ratio}^{(3)}$
green -brick	length: 0.6		155.8 / 16.4	= **9.5**
	width: 0.2	9 : 1	84.3 / 8.86	= **9.5**
	height: 0.2		103.5 / 12.2	= **8.5**
blue -brick	length: 1.2		1223.7 / 31.8	= **38.5**
	width: 0.2	36 : 1	1042.6 /30.7	= **34.0**
	height: 0.2		1215.6/ 32.1	= **37.8**
orange -brick	length: 1.8		4033.6 / 47.9	= **84.2**
	width: 0.2	81 : 1	4016.8 / 47.8	= **84.0**
	height: 0.2		1637.6 / 20.2	= **80.9**

5.3.1.3 Discussion

The first experiment demonstrated the ability of System 1 to precisely estimate the pose of a single rectangular surface within a pile of objects with similar colours and dimensions. Compared with using ICP for pose estimation, the results from System 1 are superior for both locating the centroid point and estimating the surface's orientation.

Table 5.1 shows the distance error between the ground truth value and the centroid points calculated by both the ICP-integrated system and System 1. The results showed that System 1 can be up to 30 times better regarding distance error in certain scenarios. Furthermore, the distance errors produced from System 1 are less than 0.1 meters, while those of the ICP-integrated system have a higher chance (60%) to exceed 0.1 meters. The suggested system surpasses the ICP-integrated system because it considers the effect of noise on point cloud data, while the ICP-integrated system does not.

Table 5.2 shows the rotation error between the normal vectors of the ground truth plane and the Z-axis produced by both the ICP-integrated system and System 1. The rotations estimated by both systems have high precision. However, implementing ICP on a single surface results in multiple solutions. ICP can find more than one possible solution since there are no conditions set for the rotation obtained by ICP. Therefore, the ICP-integrated

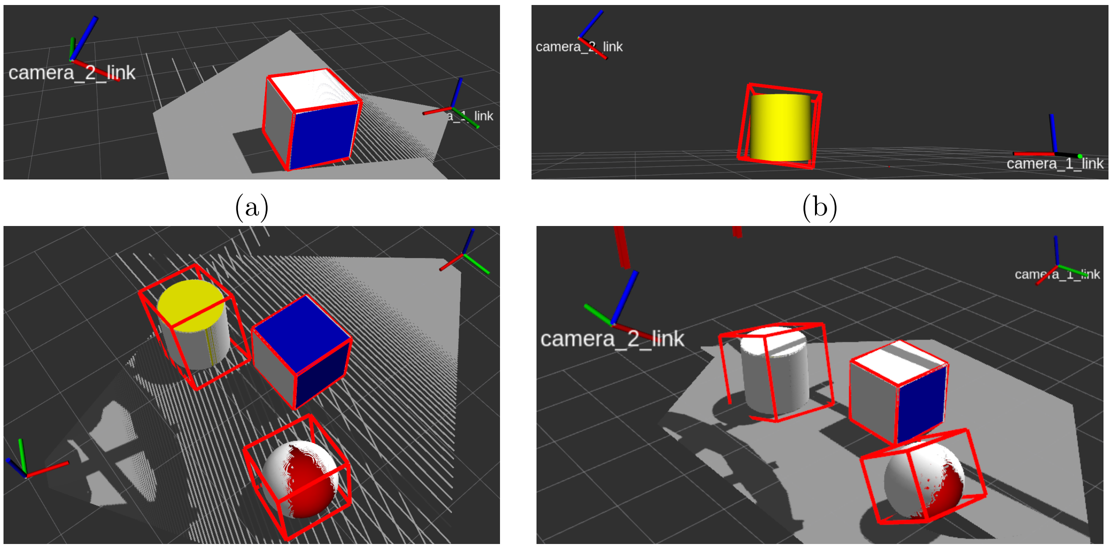


Figure 5.8: Bounding boxes from the simulated experiments: (a), (b) individual objects experiment; (c), (d) multiple objects experiment with different cameras viewpoints.

system's results are inconsistent. Due to this inconsistency, rotation averaging cannot be applied. During experiments, there is one scenario where the result from the ICP-integrated system is offset by π, which violates the requirement of the Chordal L_2-Mean, i.e., the difference between two input rotations must not exceed $\pi/4$.

The second experiment displays the relationship between the calculated eigenvalues ratio of PCA and the length-to-height ratio of the surface. Table 5.4 shows the ratio of the calculated eigenvalues within 10% of the expected value over nine tests on three different models.

Table 5.3 demonstrates the trade-off between accuracy and processing time. Despite the high accuracy and robustness of System 1, as discussed above, it requires significantly more time to execute compared to the ICP-integrated system.

5.3.2 System 2: Overlay Bounding Box Annotation

5.3.2.1 Simulated Experiments

A simulated Gazebo environment was created containing multiple simple-shaped objects and a system of two calibrated cameras. The experiment includes seven unique trials, and for each trial, the cameras are placed facing the objects from multiple viewpoints. The environment used for the first three trials contains one object, as shown in the first row of Figure 5.8. The object's shape and location vary between trials. For the latter trials, multiple objects were placed within the camera's view and remained stationary between trials, as seen in the second row of Figure 5.8.

Figure 5.8 shows the bounding box generated by the system in RViz. The system is evaluated by comparing the centre of the generated bounding box and the centre of the object in the simulated environment. The objects' centres are ground truth values obtained directly from the simulation. The error is calculated as the Euclidean distance between the bounding box centre and the objects' centre. The bounding boxes' centre and error values are shown in Table 5.5.

Table 5.5: Bounding boxes centre and error values

Exp	Bounding Box Centre (m)		Error (m)
1	Box	0.000,-0.402,2.836	0.002
2	Sphere	0.263,-0.395,4.494	0.0086
3	Cylinder	-0.124,-0.403,4.198	0.0059
4	Box	0.003,-0.401,3.836	0.0032
	Sphere	-1.499,-0.394,5.120	0.041
	Cylinder	1.400,-0.401,4.797	0.0221
5	Box	0.001,-0.237,3.956	0.0033
	Sphere	-1.761,-0.685,5.159	0.07
	Cylinder	1.770,-0.540,4.864	0.0319
6	Box	0.000,-0.825,3.689	0
	Sphere	-1.756,-1.902,4.393	0.0778
	Cylinder	1.761,-1.611,4.236	0.0437
7	Box	0.000,-0.827,3.684	0.0054
	Sphere	-1.748,-1.903,4.358	0.1094
	Cylinder	1.753,-1.606,4.230	0.0526

5.3.3 System 2: Background Removal and Clustering Process

To evaluate the performance of the ground removal and clustering processes implemented in the second system presented in this chapter, experiments in both simulated and real-world scenarios were conducted.

5.3.3.1 Simulated Experiment

A simulated Gazebo environment was created to test the background removal and clustering processes of the second system introduced. The environment consists of a 3D model of a rough surface and three simple-shaped objects which are distinctively above the ground level (Figure 5.9). The RGB-D camera is also simulated as part of the environment and was placed at coordinate $(0, 0, 0.3)$ in the global frame, and rotated by 30 degrees on the Y-axis. Figure 5.10 shows the point cloud obtained from the simulated environment. The results stage-by-stage when implementing System 2 are demonstrated in Figure 5.11

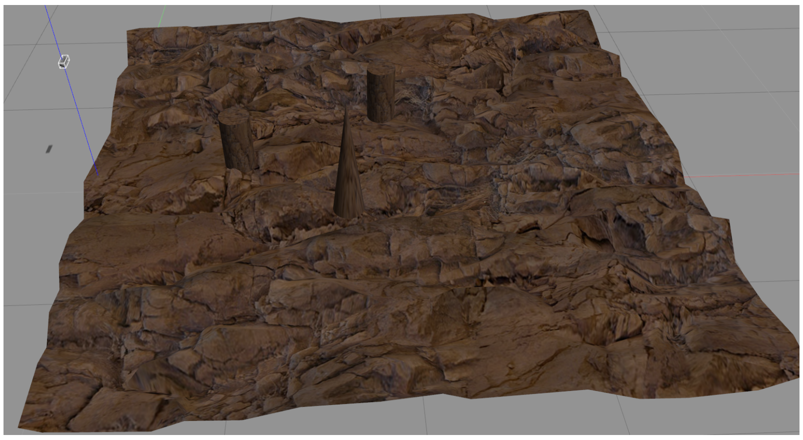


FIGURE 5.9: System 2: Background removal and clustering process Gazebo simulated environment

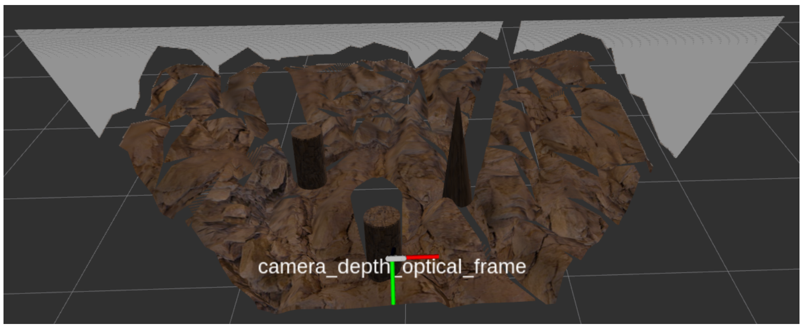
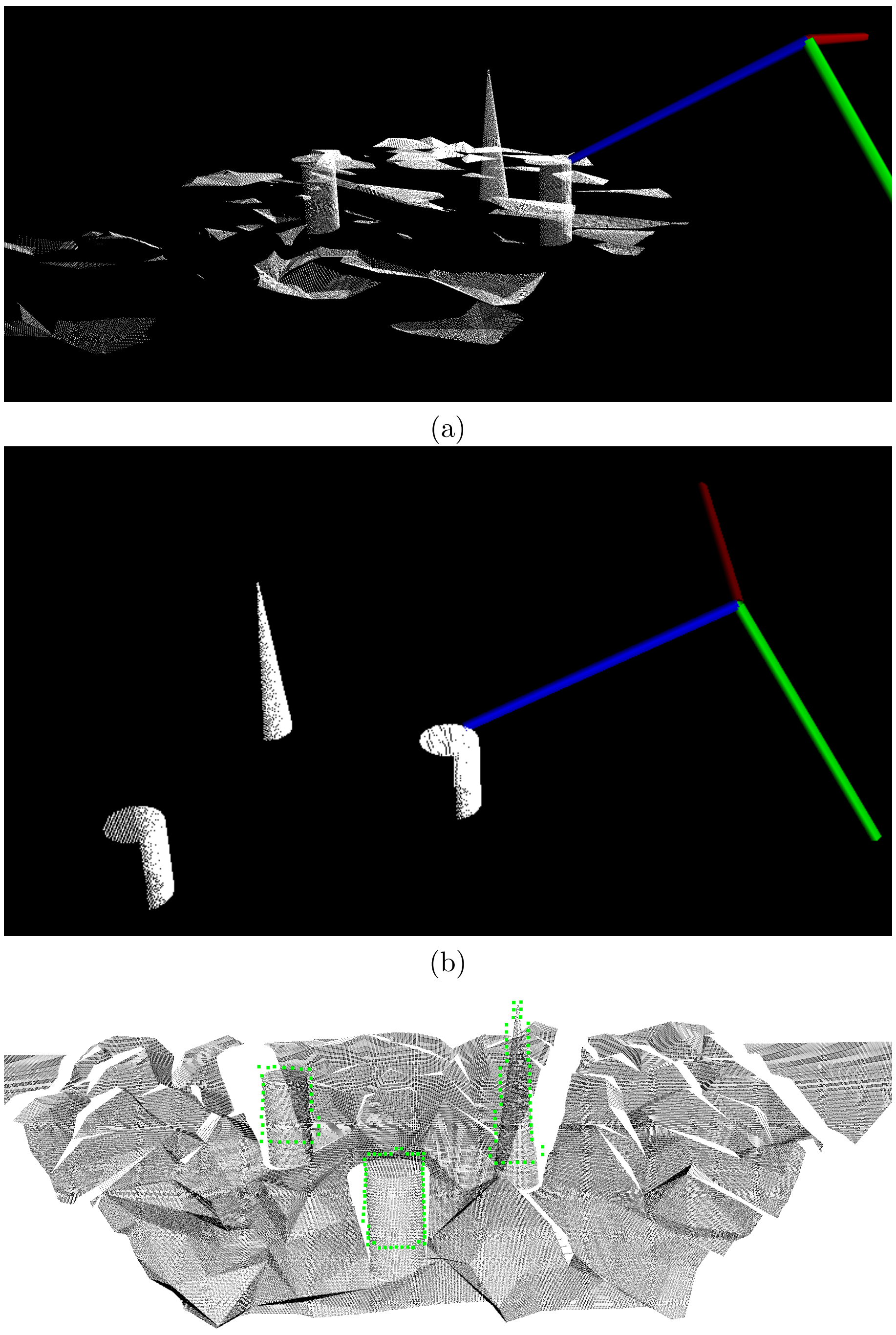


FIGURE 5.10: System 2: Background removal and clustering process simulated environment view in RViz.

In the experiment, ideally, RANSAC would obtain three clusters of points. However, due to the roughness of the 3D model, there are multiple smaller clusters of points besides the three that represent the objects expected (Figure 5.11a). Two approaches that can eliminate these unwanted small clusters are proposed as follows. The first approach is



(a)

(b)

(c)

FIGURE 5.11: Results of the implemented ground removal and clustering processes. (a) The point cloud after ground plane removal through RANSAC. (b) The point cloud after the removal of clusters below ground level and small clusters. (c) The position of clusters in (b) in the original point cloud.

to increase the “inlier” threshold of RANSAC. This was implemented in the simulated experiments conducted. Figure 5.11b shows that only the three expected clusters remain after increasing the threshold value and removing clusters below the ground plane. However, due to the increased threshold, approximately a third of the objects’ point cloud representation is missing after RANSAC, as shown in Figure 5.11c. The second approach utilises a threshold value in the clustering process to remove the small clusters created by the rough surface. The threshold value defines the minimum number of points required in a group to be considered a cluster; any group of points smaller than the threshold will be removed during the clustering process. The second approach is implemented in real-world experiments.

5.3.3.2 Real-world Experiments

Multiple data sets with different lighting conditions and surface roughness were collected for real-world experiments. The data sets are stored as rosbags. Each rosbag contains colour, depth images and point cloud data obtained from a Lidar Camera Realsense L515 and the camera’s intrinsic and extrinsic settings. Figure 5.12 shows the data sets that were utilised during the experiments. The first two data sets recorded a wall with minor roughness at nighttime (Dataset 1) and daytime on a cloudy day (Dataset 2). The latter two data sets recorded a beach with rock piles located randomly. These data sets were recorded in the daytime when the camera was exposed to direct sunlight.

The experiment results for the data sets are displayed in Figure 5.13. Figure 5.13(a) shows the point cloud from Dataset 2 after the removal of the RANSAC inliners (ground plane) and all points below ground level. Figure 5.13(b) shows the point cloud after the clustering stage, all small groups of points were removed, and points that belong to the same cluster are represented with a similar colour. Figure 5.13(c) and (d) show results of a similar process in 5.13(a) and (b) respectively, but with Dataset 4 as the input point cloud.

The experimental result displayed in Figure 5.13(a) and (b) was obtained by applying a threshold value during the clustering stage. By limiting the minimum number of points in each cluster, System 2 was able to remove small fragments of the point cloud after the removal of the ground plane. Compared to the RANSAC threshold expand approach

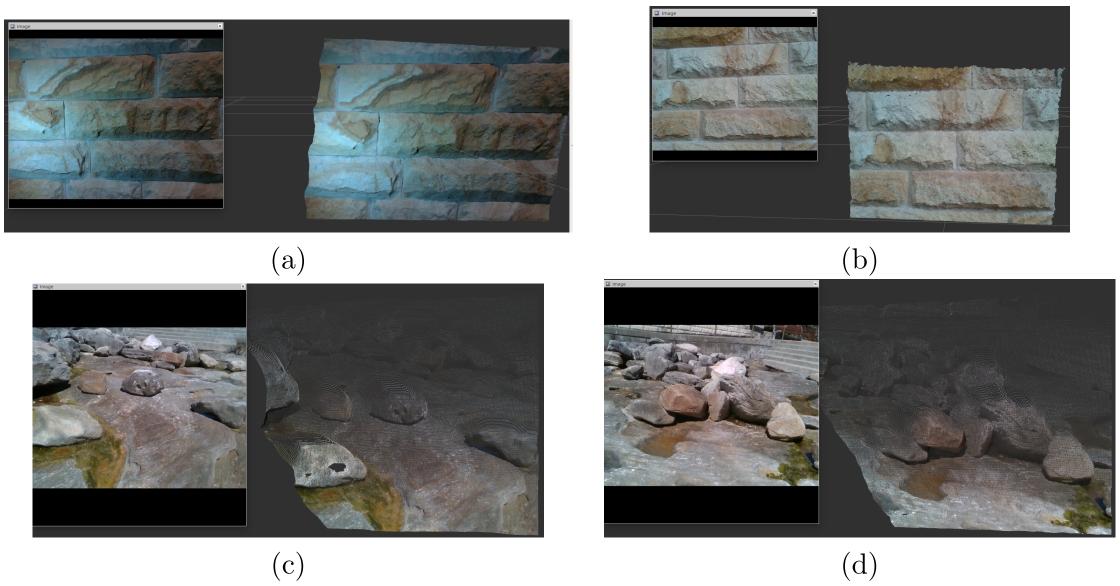


FIGURE 5.12: System 2: Background removal and clustering process real-world experiments data set. (a) Dataset 1; (b) Dataset 2; (c) Dataset 3; (d) Dataset 4.

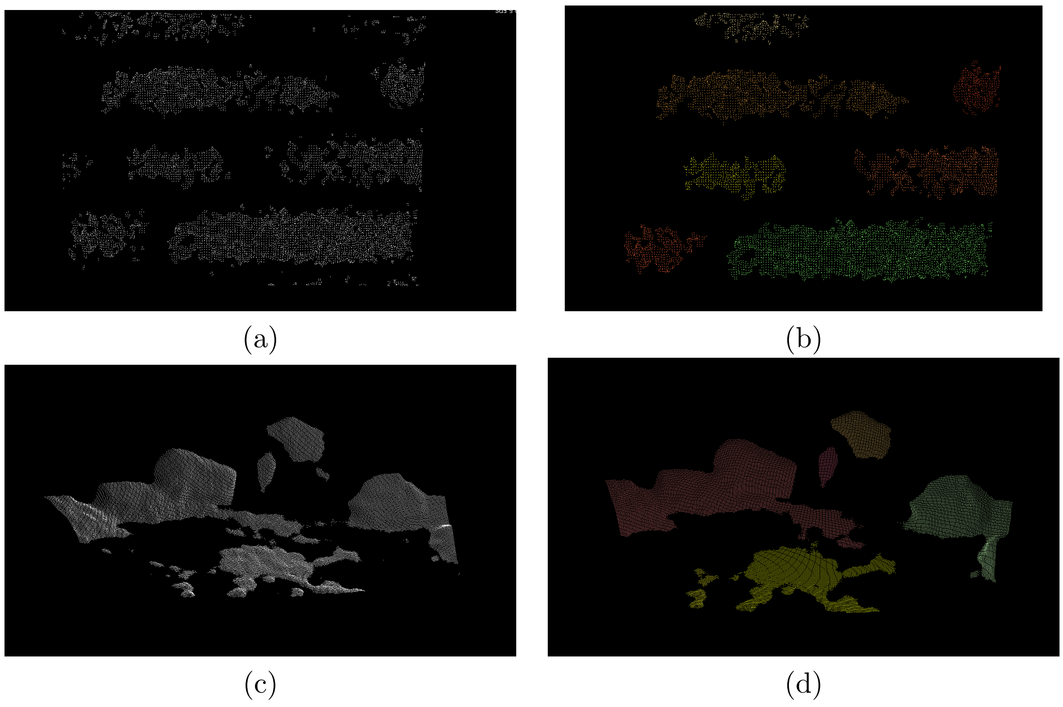


FIGURE 5.13: System 2: Background removal and clustering process real-world experiments result. (a)(b) Wall with minor roughness. (c)(d) Beach rock under sunlight.

utilised in the simulated environment experiments, the minimum limitation set during the clustering process does not cause losses in the data of the interesting objects. Additionally, since the small fragments of the point cloud are removed during the clustering stage, RANSAC's settings can have a strict error threshold, providing a precise approximation of the ground plane.

However, Figure 5.13(c) and (d) demonstrated a significant difference in System 2 performance while operating under sunlight exposure conditions. Due to the decrease in data reliability and precision, the radius, R has to be changed to ensure a connection between points in a cluster. This results in scenarios where small fragments sometimes cannot be easily removed. Although they should not be considered clusters due to the minimum number of points threshold, they have a "bridge" connection to other clusters and thus satisfy the threshold requirements.

5.4 Conclusion

This chapter has detailed two systems that use the point cloud data from RGB-D cameras to perform object detection, pose estimation and scene overlay annotation. System 1 solves the problem of rectangular prism-shaped object recognition and precise pose estimation. Experiments conducted have shown that this system is successful in obtaining the pose of a single object within a set of multiple similar objects. Experiments also proved that System 1 is more accurate and consistent than the ICP-integrated system. System 2 highlights objects on a surface by creating bounding boxes and displaying them as an overlay over the raw data. The system initially estimates a "ground" surface and removes the "ground" and all components below it. Afterwards, point cloud clustering algorithms are implemented to group the points, and bounding boxes are generated for each cluster of points. A possible metric to evaluate the accuracy of the bounding box is Intersection Over Union (IoU) of boxes [147]. Evaluating the system accuracy using IoU is outside the scope of this thesis but is considered to be future work. This system forms a core part of the VR-based collaboration framework [130][131] presented in the next chapter, which enables human supervisors to visualise the sensor data in real-time while interacting with the robotic system.

Chapter 6

Using Virtual Reality for Collaborative Autonomy

6.1 Introduction

In recent years, hardware and software development for Virtual Reality (VR) has enabled researchers to leverage representations of the real world in a virtual environment. With the help of depth cameras, the geometry and state of objects in an environment can be captured in real-time, and VR can strategically render this data. This presents several research challenges: what data to show users; how to effectively represent and transfer information-rich and dense data sets to a VR environment in real-time; and what influence data limitations have on a user's ability to perform manipulation tasks.

This chapter introduces a VR-ROS framework for collaborative autonomy and experimental results from implementing this framework for a remote robotic control system. The original framework has an RGB-D camera streaming colour and depth images in real-time. In addition, VR is utilised to visualise this sensor data for the user. A user study was performed on this framework to determine the optimal settings to visualise the data for the human operator. Based on the finding of the first study, the original framework was improved to prepare for the second study that was conducted to validate the practicality of the implementation of the point cloud processing algorithms to annotate the scene, highlighting important information to assist the user during a task.

6.1.1 User Study 1: The Influence of Sensor Data Visualisation Configuration on Users' Performance

The first user study investigates the effect of various data visualisation configurations on users in a VR environment manipulating objects using streamed sensor data, such as point clouds and images [148]. Other visualisation modalities that exist, such as meshes, are not considered within the scope of this study since a fair equivalent implementation remains unrealised. An RGB-D camera was used to capture colour and depth images, and visualisation of this real-time sensor data is done in VR. The study was conducted, and data from 12 participants performing the task 95 times in total were collected. Relevant measurements of participants' performance under various settings were recorded, including their completion time and subjective cognitive workload when performing the task.

6.1.2 User Study 2: An Investigation of User Performance in Virtual Reality-based Annotation-assisted Remote Robot Control

The second user study investigates the use of point cloud processing algorithms to provide annotations for robotic manipulation tasks completed remotely via VR [130][131]. A VR-based system has been developed that receives and visualises the processed data from real-time RGB-D camera feeds. A real-world robot model has also been developed to visualise the robot's joint state and user control feedback. The targets and the robot model are reconstructed in a VR environment and presented to users in different modalities. The modalities and available information are varied between experimental settings, and the associated task performance is recorded and analysed. The results accumulated from 288 experiments completed by 12 participants showed that point cloud data is sufficient for completing the task. Additional information, neither image stream nor preliminary processes presented as annotations, was found to have a significant impact on the completion time. However, the combination of image stream and coloured point cloud data was found to greatly enhance a user's performance accuracy, with the number of target centres missed being reduced by 25%.

6.2 Methodology

6.2.1 The VR-ROS Framework For Collaboration Autonomy

6.2.1.1 Hardware and High-level Overview

A multi-computer setup was devised to acquire, process, transmit and render real-time RGB-D sensor data in a VR environment. Chapter 5 described how the point cloud data is processed in this implementation.

Two machines are connected via Gigabit Ethernet to minimise latency. One machine (Linux PC: Intel Core i7-7700HQ CPU, 4GB RAM) obtains data from an RGB-D (RealSense D435) camera via ROS. The raw depth and colour images from the camera are compressed before being sent through the network. The second machine (Windows PC: AMD Ryzen 5 1600 CPU, 16GB RAM, GTX 1070 8GB GPU) receives and constructs a point cloud from the image data, then renders it in VR.

6.2.1.2 Real-time Point Cloud Transmission for VR Rendering using Compressed Depth Images

To achieve real-time point cloud transmission between multiple machines, the data is transferred using compressed images. However, depth images are encoded in a 16-bit format, which is unconventional for general image compression. Consequently, two images are constructed; an upper and a lower 8-bit image. The single colour and the two depth images are then compressed using JPEG, with the highest compression quality, to minimise data loss. The image manipulation is achieved using the image_transport package [149]. The VR machine's software was created using the Unity development platform. The software communicates with the ROS machine using a native .NET package [150], which enables TCP/IP connections via a network socket interface. After successful transmission,

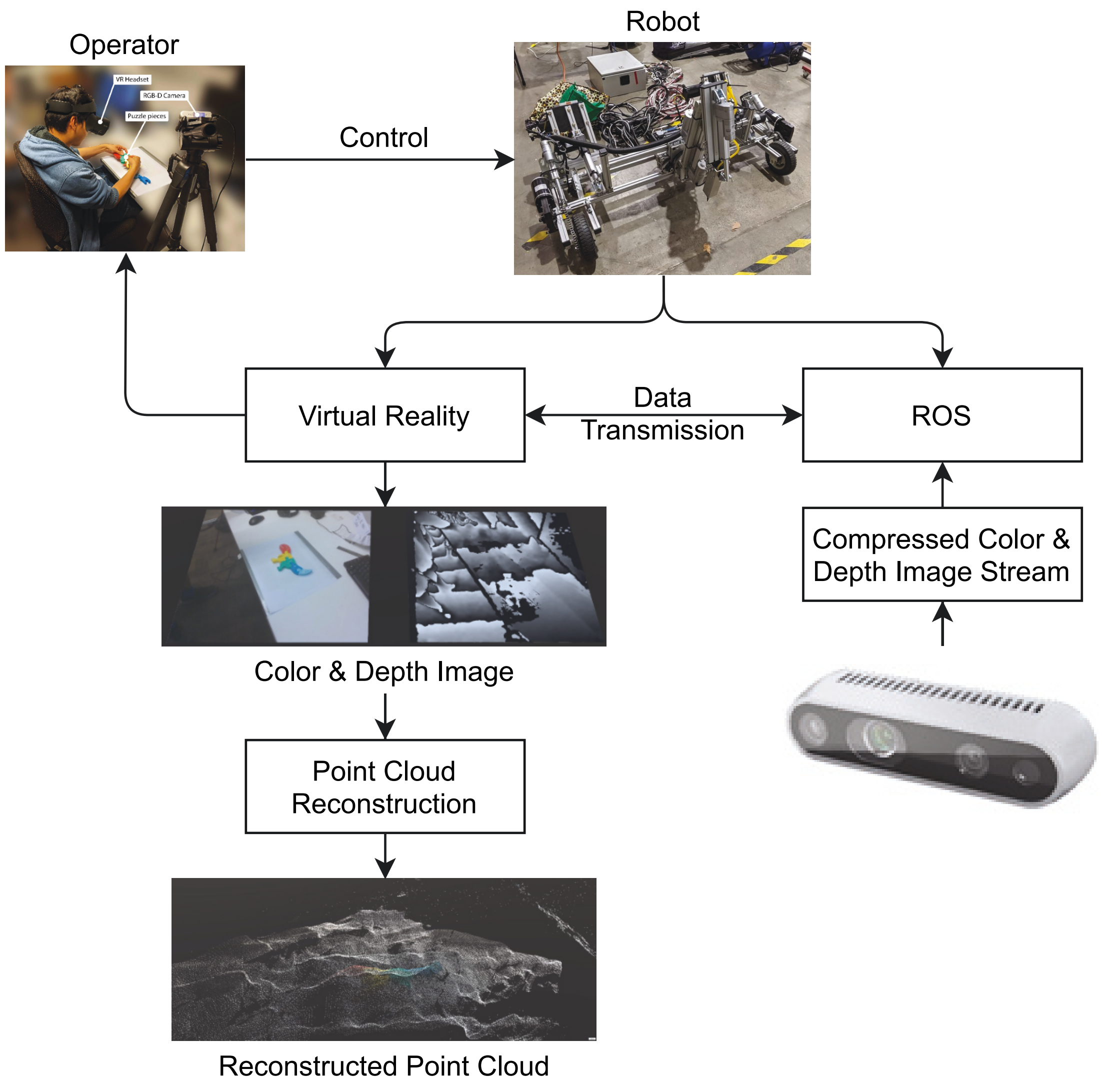


FIGURE 6.1: Operator interacting with a remote robot in VR with the visual aid of processed point cloud data that is transmitted from RGB-D cameras.

the software decompresses the images, and, when required, reconstructs and renders the point cloud in the VR environment.

6.2.1.3 Point Cloud Reconstruction

Point cloud reconstruction is achieved by implementing the pinhole camera model to construct 3D points from individual camera pixels. A mono-coloured point cloud, PC_m is obtained by processing the raw depth image. Pixel, P_i^d at row, u_i and column, v_i of the

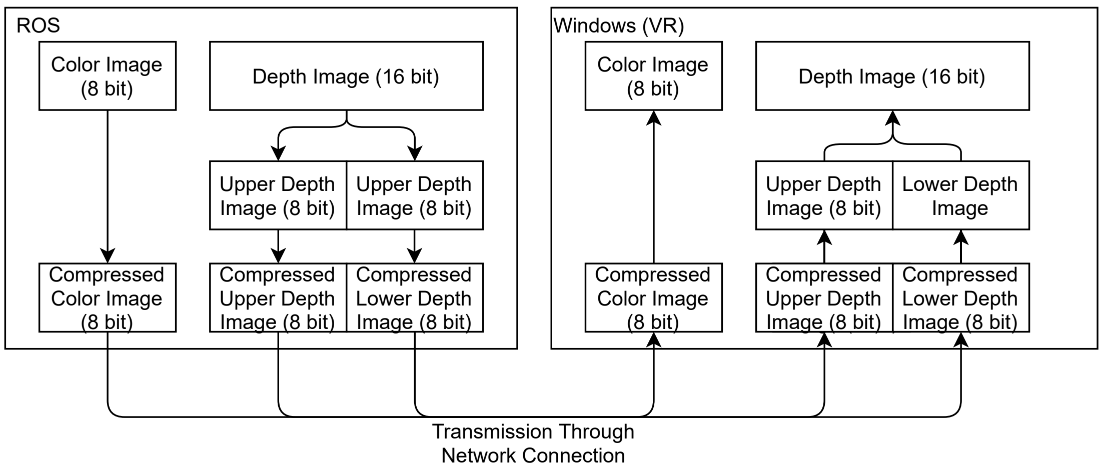

ROS
Color Image (8 bit)
Depth Image (16 bit)
Upper Depth Image (8 bit)
Upper Depth Image (8 bit)
Compressed Color Image (8 bit)
Compressed Upper Depth Image (8 bit)
Compressed Lower Depth Image (8 bit)
Windows (VR)
Color Image (8 bit)
Depth Image (16 bit)
Upper Depth Image (8 bit)
Lower Depth Image
Compressed Color Image (8 bit)
Compressed Upper Depth Image (8 bit)
Compressed Lower Depth Image (8 bit)
Transmission Through Network Connection

FIGURE 6.2: A flowchart of the lossless colour and depth image data compression and transmission process.

depth image has its 3D position in the camera coordinate system calculated by:

$$\begin{aligned} X_i &= \frac{v_i - c_x^d}{f_x^d} \times Z_i \\ Y_i &= \frac{u_i - c_y^d}{f_y^d} \times Z_i \\ Z_i &= z_i \end{aligned} \tag{6.1}$$

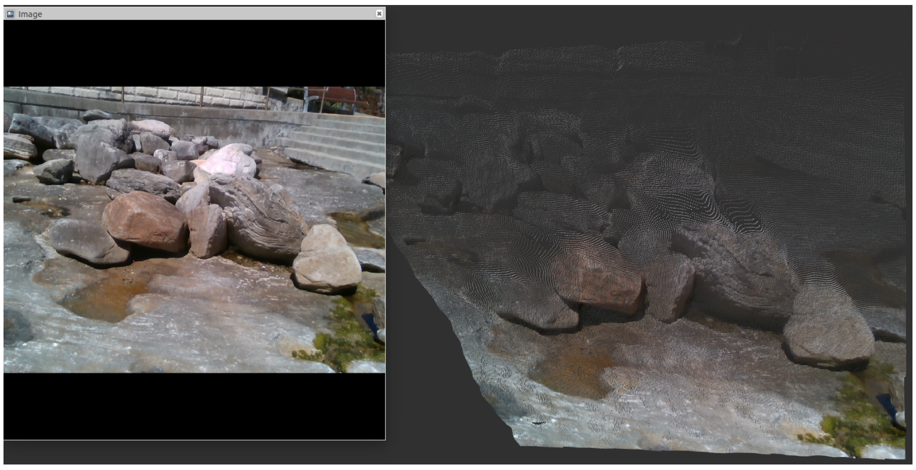
where (c_x^d, c_y^d) is the principal point of the depth image, f_x^d and f_y^d are the focal lengths of the depth camera, z_i is the depth value of pixel, P_i^d.

Each pixel, P_i^D, forms a 3D point, $p_i^m(X_i, Y_i, Z_i)$ which is added into PC_m. The final mono-coloured point cloud PC_m consists of all the 3D points generated by the pixels in the raw depth image with a fixed colour value.

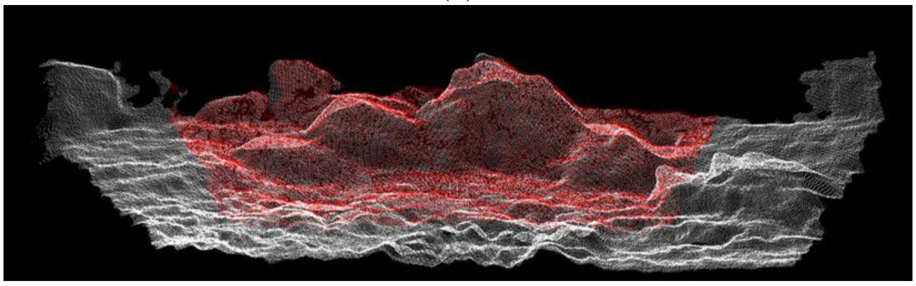
The RGB image is then used to obtain the colour point cloud PC_c, where every point, p_i^m of PC_m, is associated with a corresponding pixel in the colour image, P_i^c using:

$$\begin{aligned} v_c &= f_x^c \times \frac{X_p}{Z_p} + c_x^c \\ u_c &= f_y^c \times \frac{Y_p}{Z_p} + c_y^c \end{aligned} \tag{6.2}$$

where (u_c, v_c) is the row and column index of P_C, (c_x^C, c_y^C) is the principal point of the



(a)



(b)

Figure 6.3: Point cloud reconstruction experimental results. (a) Raw point cloud data view in RViz. (b) Comparison of the reconstructed point cloud (white) and original point cloud (red).

colour image, and f_x^c and f_y^c are the focal lengths of the colour camera. The colour point cloud, PC_c will consist of all the points, p_i^m of PC_m, where the colour value of p_i^m is set to the colour values of the corresponding pixel, $P_i^c(u_c, v_c)$.

Experiments were conducted on previously acquired data sets to perform experiments in Chapter 5 (Figure 5.12). Figure 6.3(b) presents a point cloud reconstructed from the depth image obtained from data set 4, shown in Figure 5.12(d).

It can be observed from Figure 6.3(b) that there's a difference in the FOV between the reconstructed and the original point cloud. The difference was caused by the camera's depth and colour image coverage disparity. The colour image's FOV ($69° \times 42°$) is smaller in comparison to the depth image's FOV ($85° \times 58°$) as shown in Figure 6.4. Since the original point cloud from the camera's ROS topic message is a coloured point cloud, the colour image's FOV limits the directly obtained point cloud's FOV. On the other hand,

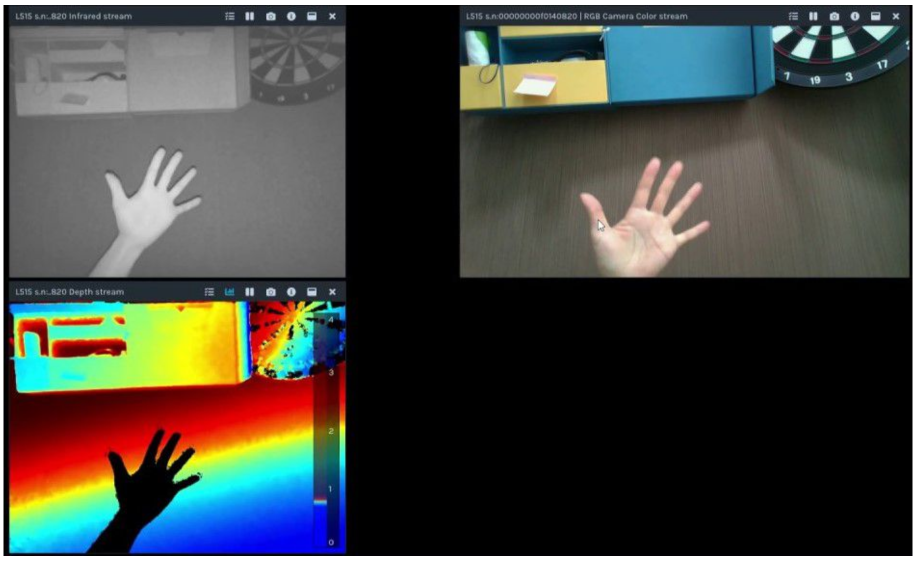


FIGURE 6.4: Field of view comparison between depth and colour images.

	Participants	Scene Setups	VR Configurations	Repetitions per Participants	Total Repetitions
User Study 1	12	9	9	9	95
User Study 2	12	6	4	24	288

TABLE 6.1: Summary of experimental design information for the two user studies.

since the reconstructed point cloud in Figure 6.3(b) is a non-colour point cloud, the depth image limits the FOV of the reconstructed point cloud. Therefore, the white point cloud representing the reconstructed point cloud is larger than the original point cloud.

6.2.2 User Study 1: Experiment Setup

An experiment was designed to investigate the effect of visualisation parameters on the participants' performance and VR experience. Summarised information concerning this experiment design can be found in Table 6.1. The user study conducted involved the assembly of a puzzle consisting of five parts whilst immersed in a VR environment. The variations of interest for the assembly task include the visualisation modality, sensor resolution, and frame rate. The criteria for assessing participant performance is based on the completion time and subjective cognitive workload.

Twelve healthy participants with no neuromuscular or debilitating visual impairments volunteered to partake in the study. Prior to completing the study, participants provided

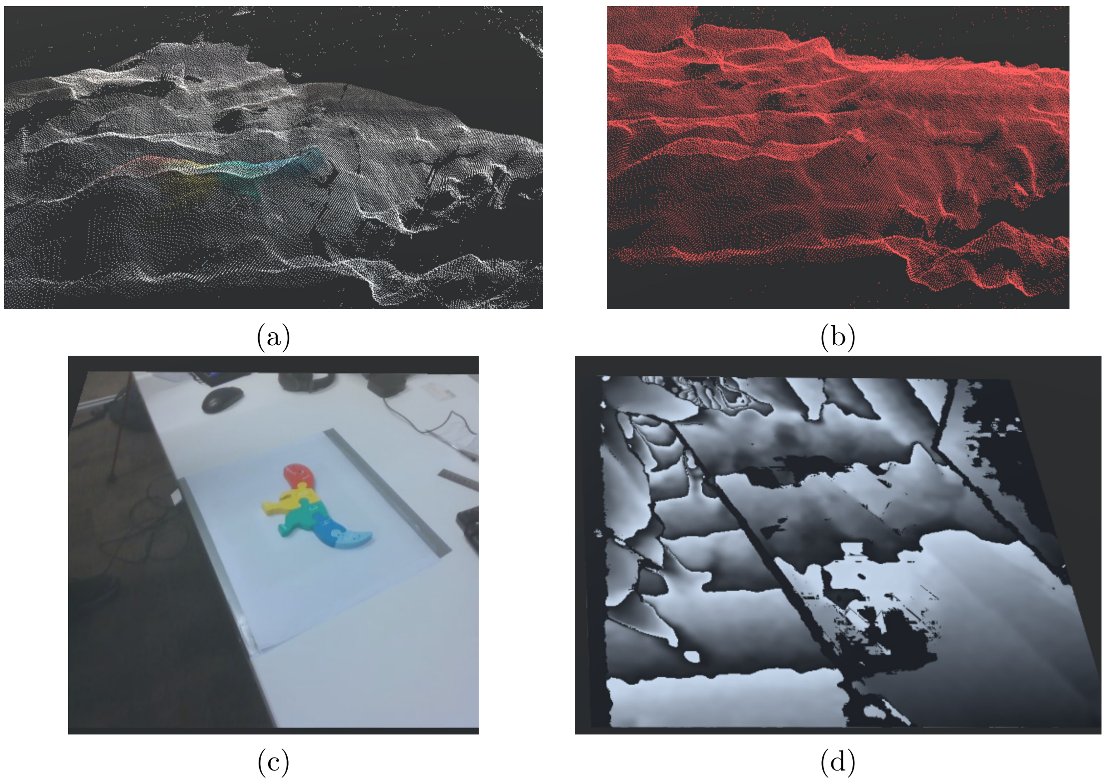


Figure 6.5: Participant view in VR: (a) Participant view: colour point cloud (C-PC); (b) Participant view: mono-coloured point cloud (MC-PC); (c-d) Participant view: colour and depth image stream (C-D).

informed consent approved by an ethics committee (UTS, Australia, approval number ETH21-5929, as detailed in Appendix A). The participants were also asked to provide feedback regarding their experience after each trial.

Table 6.2: Experimental trial visualisation settings.

Settings	Description	FPS	Quality	Resolution
1	C-D	30	80	640x480
2	C-D	5	80	640x480
3	C-D	30	5	640x480
4	MC-PC	5	100	100%
5	MC-PC	15	100	50%
6	MC-PC	25	100	33%
7	C-PC	5	100	100%
8	C-PC	15	100	50%
9	C-PC	25	100	33%

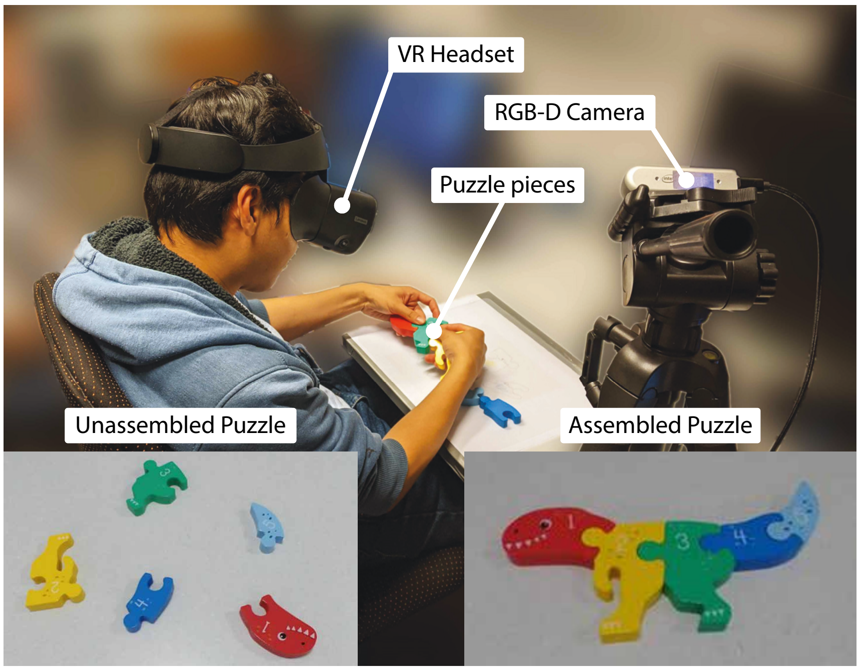


FIGURE 6.6: A participant assembling a puzzle while visualising it via RGB-D camera stream in a VR headset. Inset: The puzzle used in the experiment (left): Unassembled start configuration, (right) Successfully completed assembly.

6.2.2.1 Experiment Procedure

Participants were given a set of written instructions and shown how to fit and adjust the VR headset. Participants were asked to remain seated with the VR headset on during the experiment for their health and safety. The task involved assembling the pieces of a puzzle. It was chosen to be straightforward to minimise the advantage from personal experience. Figure 6.6 (inset left) is an example of the initial setting of the unassembled puzzle pieces, and Figure 6.6 (inset right) is the puzzle assembled. The pieces of the puzzle are stationary on a static surface. Both the camera and the VR headset position are localised based on the puzzle's position. Participants had 30 seconds before wearing the VR headset to familiarise themselves with the puzzle.

Each participant performed the assembly task 9 times using a random sequence of unassembled puzzle positions with an independently randomised trial visualisation setting. Table 6.2 lists the settings in the 9 experiments, which are selected such that visual differences between settings are obvious when presented to a user. "Quality" is the compression level of depth and RGB images, where 0 is the highest compression rate with the lowest quality and 100 indicates 100% quality compression with negligible loss. "Description" is the type of data displayed to the user in VR: "C-PC" indicates a coloured point cloud (Figure 6.5a) that fuses the RGB camera's colour data with the depth data; "MC-PC" is a mono-coloured point cloud (Figure 6.5b), "C-D" represents colour image (Figure 6.5c) and depth image (Figure 6.5d) streams, respectively. "FPS" (Frames Per Second) refers to the update frequency of the sensor data. "Resolution" indicates the quality of images and point clouds in VR. For images, "Resolution" is the width × height in pixels, and for point clouds, it is the percentage of points shown, which is equivalent to the point cloud density.

Participants were not informed of the initial unassembled puzzle arrangements, and the VR data visualisation was disabled while the visualisation parameters were prepared. During this time, the puzzle pieces were arranged in one of the predefined layouts. Then, the VR data visualisation was made available and the participant was instructed to begin. The participant verbally indicated when they were satisfied that they had completed the assembly task. Participants had a 2-minute time limit to complete each task.

6.2.2.2 Experimental Measurement 1: Objective Task Completion Time

The task completion time is the time between when the VR data visualisation is shown and when the participant verbally indicates that the task is completed. The participant completion rate for each setting is measured by the number of pieces they successfully connected out of the puzzle's total number of pieces.

6.2.2.3 Experimental Measurement 2: Subjective Cognitive Workload

The participant's subjective cognitive workload is evaluated with a 3-item questionnaire that utilises the Likert scale [151]. The questionnaire is derived from the NASA-TLX, only selecting relevant questions and rephrasing them to complement the study. Statement (1), "I relied more on perceptual than tactile feedback during the task", aimed to determine what type of feedback was relied upon during the task. An answer of 1 indicated that the participant was strongly dependent on tactile feedback, and 5 indicated that the participant was strongly dependent on visual feedback. Statement (2), “I felt stressed and annoyed during the task”, assessed their stress and annoyance levels during the task. A response of 1 refers to no stress and annoyance experienced, and 5 refers to extreme stress and annoyance . Statement (3), “I felt like I needed to work hard to complete the task”, examined the participant’s perceived cognitive load for completing the task. A response of 1 indicated a reduced cognitive workload and 5 indicated that a higher perceived cognitive workload was required to complete the task.

6.2.3 User Study 2: Experimental Setup

In order to evaluate if the annotations computed from point clouds, or additional information such as image streams would have a significant impact on user performance, a VR-based user study was conducted. Table 6.1 presents summarised information about the conducted user study design. Twelve healthy participants with no neuromuscular or debilitating visual impairments volunteered to partake in the study. Prior to completing the study, participants provided informed consent approved by an ethics committee (UTS, Australia, approval number ETH21-5929). The participants were asked to perform the experiments pertaining to 4 sets of six repetitions.

6.2.3.1 Hardware and High-level Overview

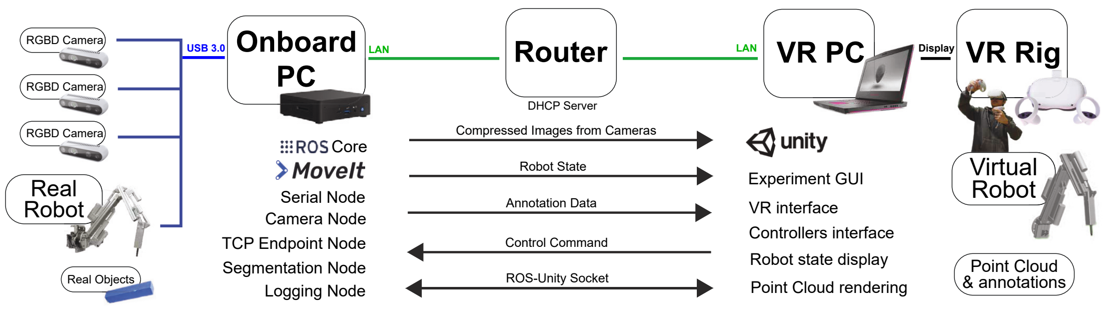


FIGURE 6.7: Hardware block diagram showing data passed from a robotic arm with cameras to a remote operator controlling the system via a VR rig that displays the virtual robot and the transmitted perception data.

The system (Figure 6.7) consists of a custom-built robotic arm with remote joystick control and a calibrated camera system. The camera system includes two static Intel Realsense D435 RGB-D cameras with different viewpoints and partially overlapping fields of view. The cameras were mounted such that they achieved a high visual coverage and limited the data loss from occlusions caused by the arm during motion. Both colour and depth images are captured and utilised to generate point clouds in real-time within the VR environment. Object detection and bounding box creation are performed using the combined data from the cameras. The bounding boxes are then transmitted to the VR environment for visualisation.

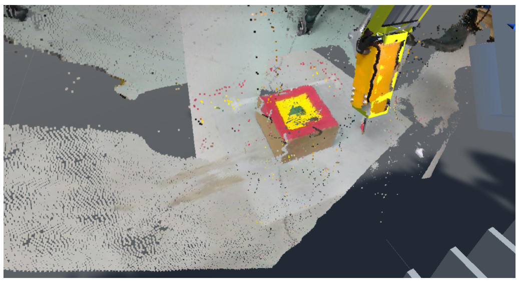
6.2.3.2 Custom Robotic Manipulator

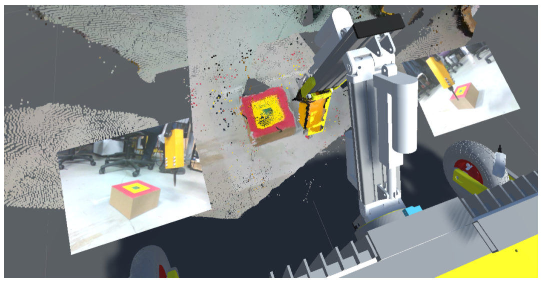
The manipulator is a 5-DOF robotic arm with a prismatic rail as the first joint and four revolute joints. The last three revolute joints are actuated by linear motors; the linear-to-angular control and feedback conversions are done automatically by the onboard controller. The end-effector is chisel-shaped with a small contact area and can be moved to various locations in a one-meter-squared work volume in front of the robot.

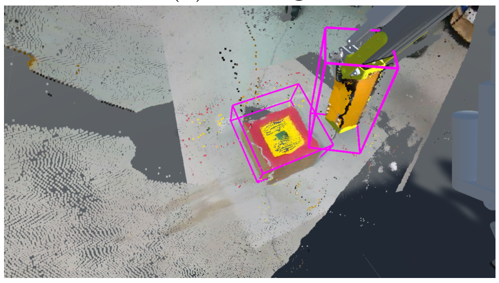
A kinematic and dynamic model was constructed from the CAD model and fed into a kinematic solver that utilises the MoveIt Motion Planning Framework [152]. The robot is equipped with joystick control allowing the user to send control commands via ROS, in either the joint space or the end-effector Cartesian space. All target locations in the experiment are within the robot's reachable volume, and no joint states exhibited by the robot are near singularities or joint limits. The robot's joint state is displayed in real-time as the model reflects the robot's real state in the virtual environment. To ensure that the point cloud of the robot arm is accurately superimposed on the model of the arm in VR, the cameras are extrinsically calibrated with respect to the robot arm.

6.2.3.3 Data Transmission and Point Cloud Rendering

The pipeline used to transmit point cloud data from the sensors via a ROS network and render it within the VR environment is based on Section 6.2.1. In our original work, [148],


(a) Setting 1


(b) Setting 2


(c) Setting 3

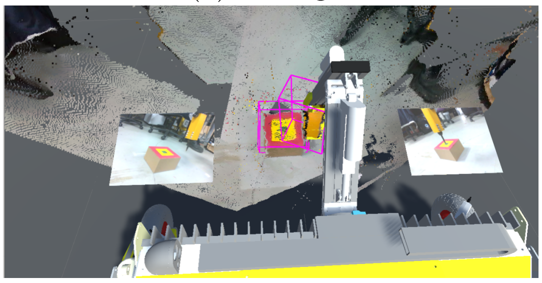

(d) Setting 4

FIGURE 6.8: Participant's view in VR with differing settings: (a) Setting 1 - Point cloud only; (b) Setting 2 - Point cloud and two image streams; (c) Setting 3 - Point cloud and annotations; (d) Setting 4 - Point cloud, two image streams, and annotations.

the point cloud frame rate and resolution were variables that were altered to obtain their effect on user performance as a trade-off exists between the frame rate and resolution. The pipeline implemented for this user study overcomes the original limitation and can transmit and render point clouds at the maximum resolution and frame rate to the VR environment.

6.2.3.4 Experimental Procedure

TABLE 6.3: Experimental visualisation settings (i.e., images, coloured point cloud (PC) and bounding box annotations) indicating which information is available "A" or not available "N/A" to participants.

Setting	Available Information		
	coloured PC	Images	Bounding Box
1	A	N/A	N/A
2	A	A	N/A
3	A	N/A	A
4	A	A	A

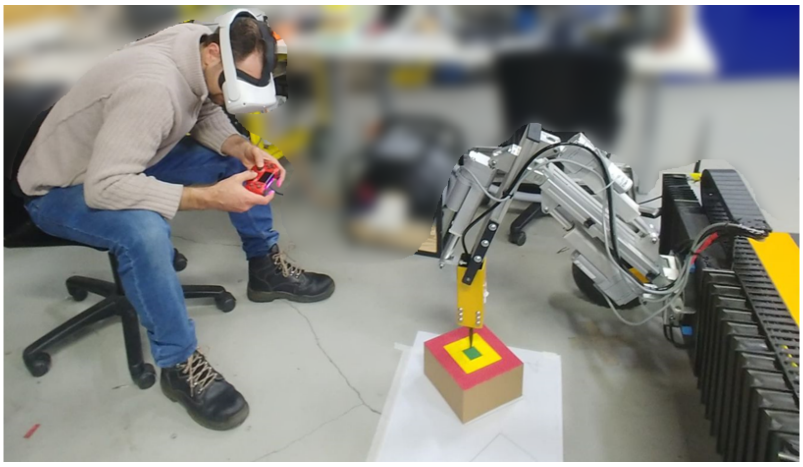


FIGURE 6.9: A participant manipulating the end-effector to the green area of the target while in VR.

Participants were given a set of written instructions and shown how to fit and adjust the VR headset. Participants were asked to remain seated with the VR headset on during the experiment for their health and safety.

The experiment required participants to perform a series of remote manipulation tasks. The task designed for the study involves using a joystick controller to manipulate a robotic arm whilst wearing a VR headset. Within the VR environment, camera data ascertained from the 2 RGB-D cameras is rendered as a point cloud. Depending on the task configuration, the bounding boxes and colour image streams may be presented.

Each participant is required to perform the manipulation task 24 times in four sets of six. The scene layout was varied after each repetition. Six different scene layouts were designed and presented to the participants in a randomised sequence. The four different settings shown in Figure 6.8, which visualise the environment with a combination of data modalities, are detailed in Table 6.3. For each repetition of the experiment, one target is in the sensors' fields of view. The participant's aim is to move the robot's end-effector to the centre of the targets, indicated by the green area, as quickly as possible (Figure 6.9). The additional colours are shown as supplemental information available to the participant.

Before putting on the VR headset, participants were given three minutes to perform the manipulation task while looking at the real-world experiment setup. This is to aid participants in familiarising themselves with the task, thus reducing the effect of the learning component. The layout used for this practice is a unique scene layout and is not one of the six scene layouts used for the study.

The participants are placed in a VR waiting room after the VR headset is put on. They are asked to refrain from moving the robotic arm while waiting. When the scene layout is prepared, the administrator begins the VR simulation, and the scene is revealed to the participants in VR. During the experiment, the participants must verbally indicate when they believe the arm has touched the target based on their own judgment. After the participants have indicated that the task is complete or two minutes have elapsed, the VR simulation is terminated, and the participants are returned to the VR waiting room. The administrator then configures the scene for the next task and returns the arm to its predefined home position. After a set of tasks, the participants remove the VR headset and rest for 3 minutes before continuing the next set of repetitions. This minimises the effect of fatigue on the participants' performance.

6.3 Experimental Result

6.3.1 User Study 1 Results

6.3.1.1 Objective Task Completion Time

A one-way ANOVA test was conducted with the null hypothesis being that there is no significant difference between the mean completion time of the various settings implemented during the study. An F-statistic value of 1.9 and a p-value of 0.0702 were obtained from the analysis, showing no statistical significance.

The completion time for Setting 1 in Table 6.2 is lower than that of the remaining settings (median = 26 s). Variations of MC-PC (Settings 4, 5, and 6) were tested, and in multiple trials, participants failed to assemble the puzzle, resulting in a "completion time" equal to the time limit being recorded. Figure 6.10a shows that the completion time of the MC-PC experiments are higher than in other settings.

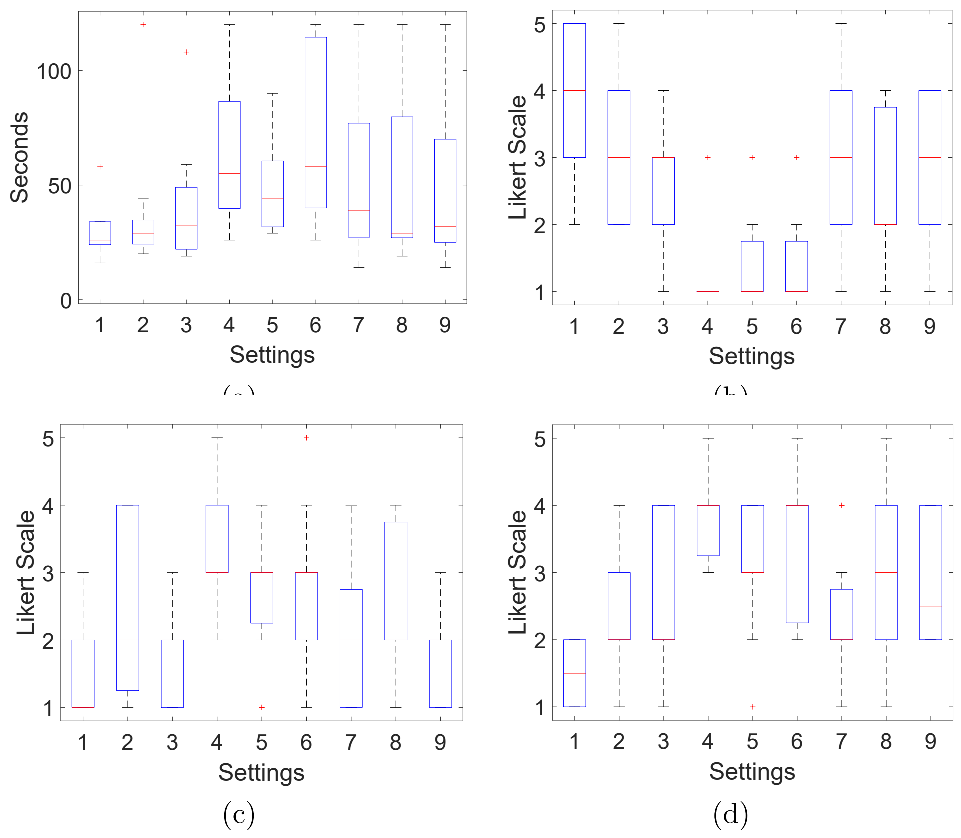


FIGURE 6.10: Experimental results: (a) Completion time; (b)-(d) for statements 1 to 3 the participants' responses on the Likert scale.

The C-PC trials (Settings 7, 8, and 9) also contained instances where participants failed to complete the task within the time limit. However, the median completion time is lower than the MC-PC trials. The participants' completion rate indicates that Setting 1 from Table 6.2 is the only one where no participants failed to complete the task. Settings 2, 7, and 9 each have 1 case where a participant was unable to complete the puzzle. For Settings 3, 4, 5 and 8, the average completion rate is below 90%.

6.3.1.2 Subjective User Experience Results

Figure 6.10b shows participants are more likely to agree that they were dependent on visual feedback when C-D image streams or the C-PC is presented, with 48.13% and 31.23% of responses agreeing with statement (1), respectively. For Settings 4, 5, and 6, when the visualisation was limited to the MC-PC, 84.85% of responses disagreed with the statement, implying that participants relied on tactile feedback.

Figure 6.10c demonstrates a general disagreement (58.33% of responses) to a high level of stress and annoyance experienced in the experiments. Participants indicated they were least stressed and annoyed when working with the image streams, with 74.19% of responses disagreeing with statement (2), followed by C-PC at 65.63% and MC-PC at 36.67%.

When participants were shown C-D image streams, most reported a relatively low cognitive workload, with 77.42% of responses disagreeing with statement (3). On the contrary, participants expressed higher cognitive workload with the MC-PC and C-PC settings, with 54.55% and 28.13% of responses agreeing to the statement, respectively.

6.3.1.3 Discussion

The experimental results indicate that RGB and depth image streams are the preferred data visualisation modalities. However, several factors may confound the findings of the experiment. Since only one camera was used during the experiment, the reconstructed point cloud could easily be occluded by the participant's hand and surrounding objects. Additionally, the straightforward assembly task could have meant that the supplementary information provided by a point cloud was redundant. Furthermore, the image stream can be obtained at a higher frame rate and resolution than the point cloud data. In this particular user study, the point cloud reconstruction rate has been limited to 5Hz. Thus, a dependence existed between the point cloud data frame rate and resolution. The hypothesis for this work stated that the inclusion of point cloud data would enhance the user's perception of the environment and thus improve the participants' performance. However, objective completion time results did not reflect this.

The results indicate that for settings where participants were shown the colour image stream, their performance and VR experience improved. This indicates that they were assisted by their visual perception when performing the task. In contrast, when the point cloud data was shown, tactile feedback was relied on. The participants' responses about their reliance on visual and tactile feedback suggest that coloured point cloud data enabled them to incorporate visual perception to help complete the task. Generally, participants reported the highest dependence on visual feedback when image streams were available. An aspect that has not been explored is the participant's inclination to favour tactile feedback if the visual feedback is poor. This learning component means a participant could eventually complete the task using only tactile feedback.

The majority of participants reported disagreement or a neutral response, to both the use of tactile feedback and a "heavy cognitive workload", when working with an RGB and depth image stream or a coloured point cloud. This suggests that a framework incorporating sensor data collection and VR for data visualisation is viable for real-time applications. Coloured point clouds representing an environment in VR seem to enhance a user's performance if the point cloud is presented at a high frame rate and density. Furthermore, the user study reported negative feedback from participants when working with mono-coloured point clouds, even at high frame rates and density, meaning the colour information is essential.

This work showed depth and colour images being compressed and transmitted with minimal latency and a visually imperceptible loss of quality. Other visualisation modalities exist, such as triangular meshes, visual hulls and annotations. In our preliminary investigations, the added step to triangulate the extracted point clouds caused unacceptable latency, and slight VR sickness since the refresh rate we achieved did not keep up with movements. However, it is anticipated that research into improved filters, compression and GPU processing techniques, will enable new modalities, such as meshes, to be compared alongside the current ones.

6.3.2 User Study 2 Results

Experimental results collected from 12 participants performing a total of 288 repetitions of the task are shown in Figure 6.11.

6.3.2.1 Objective Task Completion Time

The task completion time is determined as the time from when the participant starts manipulating the robotic arm to when they verbally indicate that the task is completed. A one-way ANOVA test was conducted with the null hypothesis being that there is no significant difference between the mean completion time of the various settings implemented

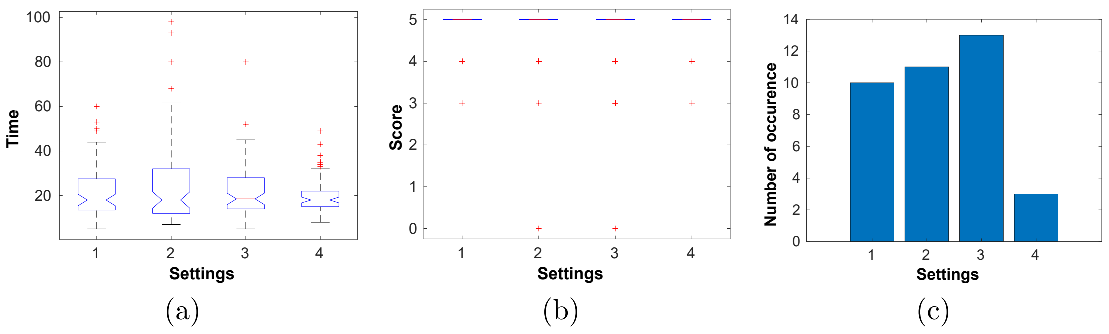


FIGURE 6.11: Experimental results for each setting: (a) Completion time; (b) Precision quantified score; (c) Histogram of repetitions where participants missed the green target area.

during the study. An F-statistic value of 1.71 and a p-value of 0.1644 were obtained from the analysis, showing no statistical significance.

6.3.2.2 Objective Task Precision

To quantify the participant's accuracy during the experiment, the participants receive a point score based on whether the end-effector touches the target. The participants are given 5 points if the end-effector touches the green area on the target surface when they indicate they have completed the task. Similarly, if the end-effector stops when touching the yellow area, the participants receive 3 points, and 1 point is allocated for touching the red area. If the end-effector is not touching the target surface, the participant is not given any points.

Similar to Section 6.3.2.1, a one-way ANOVA test was also conducted, with the null hypothesis being that there is no significant difference between the mean accuracy of the above-mentioned settings. An F-statistic value of 2.37 and a p-value of 0.0705 were reported, supporting the previous statement.

6.3.2.3 Discussion

The experimental results indicate that the availability of the information presented through various mediums, namely 2D images, and bounding box annotations, did not significantly impact the objective task completion time, as seen in Figure 6.11a. The median completion time of the participants collectively for all settings is similar, despite the varied fluctuation range. Figure 6.11b shows that among the four settings investigated, participants are able to manipulate the end-effector to the target's centre in most trials. However, Figure 6.11c demonstrates a difference in the resulting score when images are available. In settings where images are available (Setting 2 and 4), participants failed to hit the target's centre 14 times in total. Within the 14 misses, there were 12 occurrences (85.7%) where the end-effector touched the edge of the green target area, resulting in a score of 4. On the other hand, in other settings, participants missed the centre 23 times. The participants only scored 4 a total of 15 times (65%). Amongst all the investigated settings, participant performance improved with Setting 4, as indicated by the low number of times the participants missed the green area.

The experimental results suggest that the non-annotated visualisation, the presented coloured point cloud alone, was sufficient for this remote-controlled manipulation task. However, this may be due to the controlled and ideal indoor setup where the depth sensors are not exposed to excessive infrared noise, such as from the sun. Furthermore, bounding box annotations did not impact the overall performance as the target could be easily identified in the scene. In other scenarios, where the target cannot be distinguished through colour or if the colour information is unavailable, the target becomes less recognisable, and annotations may better assist a user.

6.4 Conclusion

This chapter presents a framework to transmit and visualise sensor data collected by a mobile manipulator system in VR. The first user study was conducted to investigate the effect of variances in real-time VR-based sensor data visualisation on user performance when an assembly task was conducted. The sensor data variations are visualisation modality, sensor resolution, data compression, and frame rate. In addition to RGB and depth image streams, the VR-based system was shown to transfer colour point clouds in real-time. Experiments were conducted where the visualisation parameters were varied between trials of the assembly task. In the first user study of the framework, the point cloud data cannot achieve the highest resolution and frame rate simultaneously. This was addressed in the second study, where multiple cameras were combined, and data could be visualised at the highest resolution while maintaining a high frame rate. An improved framework was utilised for the second study to examine the practicality and effectiveness of implementing computer vision algorithms to enhance user performance via annotating and highlighting important information.

Chapter 7

Conclusions

This thesis has presented the implementation of multiple frontier detection algorithms for 2D exploration while also comparing their performance with each other. Experiments were conducted in both simulations and the real world to provide insightful conclusions about the performance and possible applications of the algorithms. Additionally, a 3D interactive and active mapping framework for a mobile manipulator platform based on dynamic GPIS was implemented to validate its efficiency while mapping, exploring and interacting with the environment simultaneously. The results showed that the framework can perform immediate map updates for a dynamically changing environment due to its probabilistic nature. Two perception systems that employ the point cloud data in the framework are presented to perform object detection, pose estimation and scene overlay annotation. The first system is a novel solution for rectangular-shaped object pose estimation in the robotic bin-picking problem. The second system aims to detect objects above ground level and generate a bounding box that wraps the objects. The second system is implemented as part of a user study utilising a VR-ROS framework. Lastly, a VR-ROS framework for sensor data visualisation in VR environments is presented. User studies were conducted to determine the optimal configuration for user performance and the effects of annotation information created by the object detection algorithm on human operators.

7.1 Summary Of Contributions

7.1.1 Efficiency Evaluation of Frontier Detection Algorithms for Map Exploration

Six frontier detection algorithms were implemented in simulated and real-world scenarios to evaluate their efficiency. Discussions and conclusions are drawn from the experimental results. Three different experiments were implemented to gather relevant data, including the overall computational time, and the number of cells processed and evaluated. The conclusions drawn from the results can be summed up as follows. The frontier detection algorithm should first be chosen depending on the characteristics of the map and whether the speed is an important consideration for the application. Table 3.1 was generated to summarise the recommended algorithms for specific scenarios, taking into account the unique characteristics and behaviours of each algorithm. For all algorithms, the results did not clearly demonstrate a connection between computation time and map exploration. In other words, the calculation time for the Naïve, NaïveAA algorithms does not proportionally grow with increasing open space. Due to its speed, robustness, and efficiency over Naïve frontier detection, NaïveAA should be taken into consideration as a new benchmark for evaluating future frontier detection algorithms. On the other hand, FTFD is best suited for applications in which the implementation of frontier detection is required after each observation.

7.1.2 Validation of an Active and Interactive 3D Mapping Framework for Mobile Manipulator Platforms

A 3D interactive and active mapping framework for a mobile manipulator platform based on dynamic GPIS was implemented in simulations and real-life environments to validate its efficiency. The validation process included comparisons of the investigated framework with other off-the-shelf available frameworks in the number of bricks picked up and the map completion. The results demonstrated that the active and interactive framework could perform immediate mapping updates for a dynamically changing environment. The results also show that the framework's NBV scheme can balance the needs of information gain in visited regions, frontier-driven map expansion and object manipulability.

7.1.3 Development of Perception Systems for Object Detection and Pose Estimation using Point Cloud Data

Two perception systems that employ point cloud data to perform object detection, pose estimation and scene overlay annotation are presented. The first system was developed to perform object detection and pose estimation with one static RGB-D camera. The system minimises the effect of sensor noise by pre-processing the raw point cloud data. The region growing algorithm is then implemented to determine individual surfaces before PCA is applied to estimate the pose of detected surfaces. At the final stage, a rotation averaging method is performed on the estimated pose, enhancing its precision. The second system aims to detect multiple objects above ground level and highlight them for visualisation in a VR environment. Similar to the first system, the noise in the raw data is minimised through preliminary processing. Then RANSAC is implemented to remove all points in the background, including the ground plane. Afterwards, points are grouped based on their Euclidean distance to create clusters. Since each cluster represents an individual object, bounding boxes are generated for all distinct clusters. The boxes are transmitted and presented in VR as an overlay over the raw data.

7.1.4 Enhancing Perception for Collaborative Autonomy: A Framework and User Studies for Sensor Data Visualisation in VR Environments

A framework and two user studies for sensor data visualisation in VR environments have been presented. The framework is a multi-computer setup devised to acquire, transmit and render real-time 3D sensor data in a VR environment. Experiments were conducted on this framework to determine the optimal configuration for user performance. The data was presented to participants in various ways to investigate the effect of data modalities, frame rate, colour information, and resolution on the participants' completion time and subjective cognitive workload. For the second user study, the original framework is improved and optimised to remove the limitations of the previous version regarding point cloud resolution and frame rate. The improved framework also involved annotation by implementing the perception algorithms presented in this thesis. The second user study was conducted to determine the practicality and effectiveness of implementing point cloud processing algorithms to annotate the VR environment.

7.2 Discussion of Limitations

Implementing active perception and estimation frameworks in semi-autonomous mobile manipulators is an ongoing research challenge. On the other hand, integrating VR for collaborative autonomy is a relatively new research topic with increasing popularity due to improved VR technologies. This thesis has implemented previous works in active perception and estimation on mobile manipulators to automate low-level processes while utilising VR approaches to visualise sensor data and allow human supervisors to collaborate with a robot and make high-level decisions.

Though the frontier detection algorithms implemented in Chapter 3 will work in 3D, it is not obvious how much advantage these algorithms would provide when applied to 3D applications compared to other algorithms developed for 3D exploration. Additionally, the algorithms were not implemented or experimented with in any 3D environment, whether simulations or real-world scenarios. Therefore, these algorithms are strictly limited to 2D applications until further experiments using 3D data are performed and analysed.

The implemented framework in Chapter 4 utilised 3D sensor data to generate and update the map for a mobile manipulator robot that aims to perform pick-and-place tasks. However, the framework did not fully utilise the 3D sensor data available. The sensor data obtained from the RGB-D camera was used solely to generate the 3D occupancy map. However, many perception and estimation methods utilising 3D sensor data can be implemented to obtain helpful information to optimise automated processes and improve human interaction and data visualisation. Therefore, Chapter 5 addresses this limitation by introducing the two separate point cloud processing systems.

The systems presented in Chapter 5 have demonstrated some limitations. The first system relies on the region-growing algorithm to detect individual surfaces. Therefore, it suffers from a similar drawback to the region-growing algorithm: the possibility of multiple surfaces connected through curved edges. Furthermore, the desirable surfaces are defined based on the size and ratio generated by PCA. This introduces special scenarios where non-rectangular-shaped surfaces can be considered desirable results as long as they can be wrapped inside a rectangle with a specified size. The second system heavily depends on the completeness of the object available in the point cloud. In scenarios where the object is partially represented in the point cloud, the bounding box will be generated based on the partially completed object, resulting in an undesirable outcome. Another possible scenario where the system fails is when an object is presented with two distinct clusters of points instead of one due to the disruption in its point cloud representation. In addition, both systems were developed based on assumptions, limiting their applications.

The framework utilising VR for collaborative autonomy is presented in Chapter 6 and is limited to the quality of the raw point cloud received from the sensors. Since the point cloud in VR is reconstructed from RGB and depth images, no preliminary processes were implemented to filter or minimise noise data before presenting it to the user. In addition, the point cloud from multiple sensors was displayed on top of each other without performing a point cloud fusion algorithm. This leads to the quality of the overlapped point cloud solely relying on the precision of the sensors' calibration processes. A limitation also exists in both user studies presented in Chapter 6. In both studies, the experimental results are affected by the learning factor. The learning factor is the ability of a participant to familiarise themselves with the task, which significantly affects their performance during the experiments. Unfortunately, this factor can only be minimised by including a long practice duration before the experiments to practice the task, which was impossible in this research.

7.3 Future Work

This thesis has shown the implementation and experimental results of exploration approaches on mobile manipulators. In addition, systems and frameworks are introduced to utilise VR for collaborative autonomy with mobile manipulators. However, limitations of the current work have been outlined, showing that further research is still required. These works are beyond the scope of this thesis, but it is hoped that these topics will inspire future work related to the topics presented.

The second system in Chapter 5 contributes a core component to the second framework presented in Chapter 6, and the major limitation of the system can be improved by increasing the completeness of the point cloud's presentation of the targets. Since the framework involves multiple sensors, future work should optimise the sensors' viewpoints selection while reducing the number of required sensors and still achieving a high level of coverage. By improving the coverage, leading to a more thorough and complete point cloud, it is hypothesised that both the user's performance and the precision of the object detection and estimation systems will be improved.

Future work should also investigate the implementation of point cloud processing algorithms and the utilisation of other data modalities, such as meshes and voxels, in the VR environment. Raw point cloud data contains a large amount of inaccurate information. This is frequently addressed by including layers of filters and noise-reducing algorithms in the raw data. However, since the VR simulation uses a completely different programming language than the existing libraries that process point clouds, the raw point cloud data cannot be processed before it is presented to the user. In addition, other modalities could have certain advantages over a point cloud in terms of immersiveness and usefulness. Therefore, user studies that investigate the application of other data modalities in VR are also required.

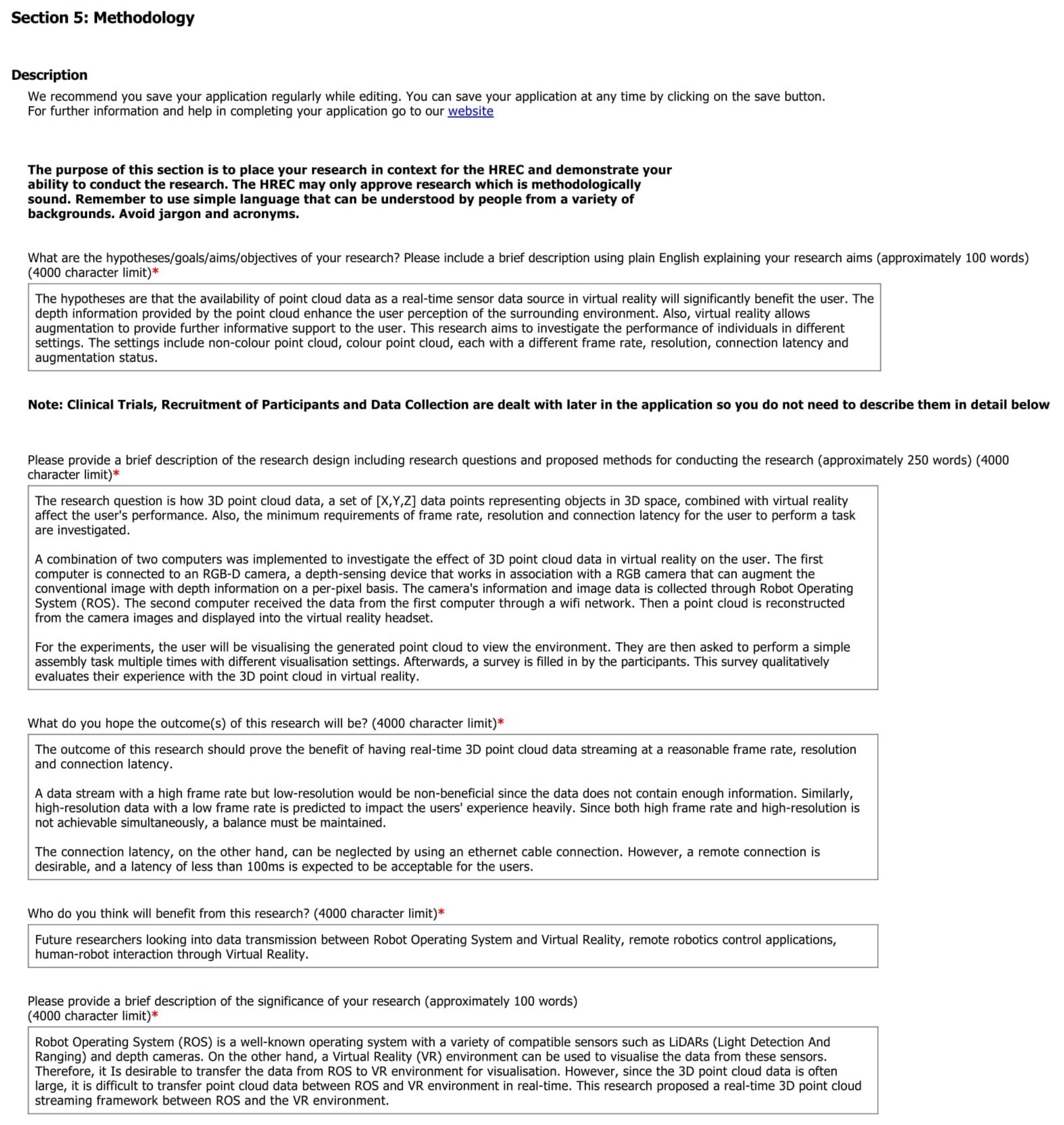
Appendix A

Human Ethics Application

ResearchMaster

Human Ethics Application

Application ID :	ETH21-5929
Application Title :	Investigating the effect of sensor data visualisation variances in Virtual Reality
Date of Submission :	30/03/2021
Primary Investigator :	Dr Gavin Paul (Chief Investigator)
Other Personnel :	Mr Long Long Vu (5Research Student)
	Miss Sheila Sutjipto (5Research Student)
	Mr Pham Dinh Thong Vu (8Research Associate)
	Mr Yujun Lai (5Research Student)
	Mr Khoa Dac Dang Nguyen (3Assoc. Investigator)
	Mr Tony Le (Chief Investigator)
	Mr Nuwan Imiyage Nuwan Prabhath Munasinghe (5Research Student)



Section 5: Methodology

Description

We recommend you save your application regularly while editing. You can save your application at any time by clicking on the save button.
For further information and help in completing your application go to our website

The purpose of this section is to place your research in context for the HREC and demonstrate your ability to conduct the research. The HREC may only approve research which is methodologically sound. Remember to use simple language that can be understood by people from a variety of backgrounds. Avoid jargon and acronyms.

What are the hypotheses/goals/aims/objectives of your research? Please include a brief description using plain English explaining your research aims (approximately 100 words) (4000 character limit)*

The hypotheses are that the availability of point cloud data as a real-time sensor data source in virtual reality will significantly benefit the user. The depth information provided by the point cloud enhance the user perception of the surrounding environment. Also, virtual reality allows augmentation to provide further informative support to the user. This research aims to investigate the performance of individuals in different settings. The settings include non-colour point cloud, colour point cloud, each with a different frame rate, resolution, connection latency and augmentation status.

Note: Clinical Trials, Recruitment of Participants and Data Collection are dealt with later in the application so you do not need to describe them in detail below

Please provide a brief description of the research design including research questions and proposed methods for conducting the research (approximately 250 words) (4000 character limit)*

The research question is how 3D point cloud data, a set of [X,Y,Z] data points representing objects in 3D space, combined with virtual reality affect the user's performance. Also, the minimum requirements of frame rate, resolution and connection latency for the user to perform a task are investigated.

A combination of two computers was implemented to investigate the effect of 3D point cloud data in virtual reality on the user. The first computer is connected to an RGB-D camera, a depth-sensing device that works in association with a RGB camera that can augment the conventional image with depth information on a per-pixel basis. The camera's information and image data is collected through Robot Operating System (ROS). The second computer received the data from the first computer through a wifi network. Then a point cloud is reconstructed from the camera images and displayed into the virtual reality headset.

For the experiments, the user will be visualising the generated point cloud to view the environment. They are then asked to perform a simple assembly task multiple times with different visualisation settings. Afterwards, a survey is filled in by the participants. This survey qualitatively evaluates their experience with the 3D point cloud in virtual reality.

What do you hope the outcome(s) of this research will be? (4000 character limit)*

The outcome of this research should prove the benefit of having real-time 3D point cloud data streaming at a reasonable frame rate, resolution and connection latency.

A data stream with a high frame rate but low-resolution would be non-beneficial since the data does not contain enough information. Similarly, high-resolution data with a low frame rate is predicted to impact the users' experience heavily. Since both high frame rate and high-resolution is not achievable simultaneously, a balance must be maintained.

The connection latency, on the other hand, can be neglected by using an ethernet cable connection. However, a remote connection is desirable, and a latency of less than 100ms is expected to be acceptable for the users.

Who do you think will benefit from this research? (4000 character limit)*

Future researchers looking into data transmission between Robot Operating System and Virtual Reality, remote robotics control applications, human-robot interaction through Virtual Reality.

Please provide a brief description of the significance of your research (approximately 100 words)
(4000 character limit)*

Robot Operating System (ROS) is a well-known operating system with a variety of compatible sensors such as LiDARs (Light Detection And Ranging) and depth cameras. On the other hand, a Virtual Reality (VR) environment can be used to visualise the data from these sensors. Therefore, it Is desirable to transfer the data from ROS to VR environment for visualisation. However, since the 3D point cloud data is often large, it is difficult to transfer point cloud data between ROS and VR environment in real-time. This research proposed a real-time 3D point cloud streaming framework between ROS and the VR environment.

Bibliography

[1] Phillip Quin, Dac Dang Khoa Nguyen, Thanh Long Vu, Alen Alempijevic, and Gavin Paul. Approaches for efficiently detecting frontier cells in robotics exploration. *Frontiers in Robotics and AI*, 8:1, 2021. doi: 10.3389/frobt.2021.616470.

[2] Radu Bogdan Rusu and Steve Cousins. 3d is here: Point cloud library (pcl). In *2011 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–4. IEEE, 2011. doi: 10.1109/ICRA.2011.5980567.

[3] Carlos Moreno and Ming Li. A comparative study of filtering methods for point clouds in real-time video streaming. In *World Congress on Engineering and Computer Science*, volume 1, pages 388–393, 2016.

[4] Sergio Orts-Escolano, Vicente Morell, José García-Rodríguez, and Miguel Cazorla. Point cloud data filtering and downsampling using growing neural gas. In *The 2013 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2013. doi: 10.1109/IJCNN.2013.6706719.

[5] Samuel Dambreville, Romeil Sandhu, Anthony Yezzi, and Allen Tannenbaum. Robust 3d pose estimation and efficient 2d region-based segmentation from a 3d shape prior. In *European Conference on Computer Vision*, pages 169–182. Springer, 2008. doi: 10.1007/978-3-540-88688-4_13.

[6] Rolf Adams and Leanne Bischof. Seeded region growing. *IEEE Transactions on pattern analysis and machine intelligence*, 16(6):641–647, 1994. doi: 10.1109/34.295913.

[7] Zheng Lin, Jesse Jin, and Hugues Talbot. Unseeded region growing for 3d image segmentation. In *ACM International Conference Proceeding Series*, volume 9, pages 31–37, 2000.

[8] Anil K Jain, M Narasimha Murty, and Patrick J Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999. doi: 10.1145/331499.331504.

[9] Levent Ertoz, Michael Steinbach, and Vipin Kumar. A new shared nearest neighbor clustering algorithm and its applications. In *Workshop on clustering high dimensional data and its applications at 2nd SIAM international conference on data mining*, pages 105–115, 2002.

[10] Tianhao Zhang, Zoe McCarthy, Owen Jow, Dennis Lee, Xi Chen, Ken Goldberg, and Pieter Abbeel. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5628–5635. IEEE, 2018. doi: 10.1109/ICRA.2018.8461249.

[11] Yujun Lai, Sheila Sutjipto, Matthew D Clout, Marc G Carmichael, and Gavin Paul. Gavre 2: towards data-driven upper-limb rehabilitation with adaptive-feedback gamification. In *2018 IEEE international conference on robotics and biomimetics (ROBIO)*, pages 164–169. IEEE, 2018. doi: 10.1109/ROBIO.2018.8665105.

[12] Dinh Tung Le, Sheila Sutjipto, Yujun Lai, and Gavin Paul. Intuitive virtual reality based control of a real-world mobile manipulator. In *2020 16th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pages 767–772. IEEE, 2020. doi: 10.1109/ICARCV50220.2020.9305517.

[13] Vikash Kumar and Emanuel Todorov. Mujoco haptix: A virtual reality system for hand manipulation. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pages 657–663. IEEE, 2015. doi: 10.1109/HUMANOIDS.2015.7363441.

[14] Neobotix GmbH. Mobile robot mp-700, 2020. URL `https://www.neobotix-robots.com/products/mobile-robots/mobile-robot-mp-700`.

[15] Liyang Liu, Simon Fryc, Lan Wu, Thanh Long Vu, Gavin Paul, and Teresa Vidal-Calleja. Active and interactive mapping with dynamic gaussian process implicit

surfaces for mobile manipulators. *IEEE Robotics and Automation Letters*, 6(2): 3679–3686, 2021. doi: 10.1109/LRA.2021.3061324.

[16] Bhoram Lee, Clark Zhang, Zonghao Huang, and Daniel D Lee. Online continuous mapping using gaussian process implicit surfaces. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6884–6890. IEEE, 2019. doi: 10.1109/ICRA.2019.8794324.

[17] Simon Fryc, Liyang Liu, and Teresa Vidal Calleja. Efficient pipeline for mobile brick picking. In *Australian Conference on Robotics and Automation (ACRA)*. ARAA, 2019.

[18] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[19] Joan Solà. Quaternion kinematics for the error-state kalman filter, 2017.

[20] Armin Hornung, Kai M Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. Octomap: An efficient probabilistic 3d mapping framework based on octrees. *Autonomous robots*, 34(3):189–206, 2013. doi: 10.1007/s10514-012-9321-0.

[21] Parliamentary of Australia. The mineral sector, 2010. URL `https://www.aph.gov.au/About_Parliament/Parliamentary_Departments/Parliamentary_Library/pubs/BriefingBook43p/mineralssector`.

[22] Queensland government. 2019-20 data for the queensland mines and quarries safety performance and health report, 2020. URL `https://www.publications.qld.gov.au/dataset/`.

[23] Joshua A Marshall, Adrian Bonchis, Eduardo Nebot, and Steven Scheding. Robotics in mining. In *Springer handbook of robotics*, pages 1549–1576. Springer, 2016.

[24] Dusan Paredes and David Fleming-Muñoz. Automation and robotics in mining: Jobs, income and inequality implications. *The Extractive Industries and Society*, 8 (1):189–193, 2021.

[25] Tomas Berglund, Andrej Brodnik, Håkan Jonsson, Mats Staffanson, and Inge Soderkvist. Planning smooth and obstacle-avoiding b-spline paths for autonomous

mining vehicles. *IEEE Transactions on Automation Science and Engineering*, 7(1): 167–172, 2009. doi: 10.1109/TASE.2009.2015886.

[26] Dae Hwan Kim, Giang Hoang, Min-Ji Bae, Jin Wook Kim, Suk Min Yoon, Tae-Kyeong Yeo, Hong Sup, and Sang-Bong Kim. Path tracking control coverage of a mining robot based on exhaustive path planning with exact cell decomposition. In *2014 14th International Conference on Control, Automation and Systems (ICCAS 2014)*, pages 730–735. IEEE, 2014. doi: 10.1109/ICCAS.2014.6987875.

[27] Saikat Roy, Soumalya Sarkar, and Avranil Tah. A bluetooth-based autonomous mining system. In *Intelligent Computing, Networking, and Informatics*, pages 57–65. Springer, 2014. doi: 10.1007/978-81-322-1665-0_6.

[28] Fan Zeng, Adam Jacobson, David Smith, Nigel Boswell, Thierry Peynot, and Michael Milford. Lookup: Vision-only real-time precise underground localisation for autonomous mining vehicles. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 1444–1450. IEEE, 2019. doi: 10.1109/ICRA.2019.8794453.

[29] Jeffrey N Twigg, Jonathan R Fink, Paul L Yu, and Brian M Sadler. Efficient base station connectivity area discovery. *The International Journal of Robotics Research*, 32(12):1398–1410, 2013. doi: 10.1177/0278364913488634.

[30] Brian Yamauchi. A frontier-based approach for autonomous exploration. In *1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97.'Towards New Computational Principles for Robotics and Automation'*, pages 146–151. IEEE, 1997. doi: 10.1109/CIRA.1997.613851.

[31] Elena Digor, Andreas Birk, and Andreas Nüchter. Exploration strategies for a robot with a continously rotating 3d scanner. In *International Conference on Simulation, Modeling, and Programming for Autonomous Robots*, pages 374–386. Springer, 2010. doi: 10.1007/978-3-642-17319-6_35.

[32] Phillip Quin, Gavin Paul, Alen Alempijevic, Dikai Liu, and Gamini Dissanayake. Efficient neighbourhood-based information gain approach for exploration of complex 3d environments. In *2013 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1343–1348. IEEE, 2013. doi: 10.1109/ICRA.2013.6630745.

[33] Robbie Shade and Paul Newman. Choosing where to go: Complete 3d exploration with stereo. In *2011 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2806–2811. IEEE, 2011. doi: 10.1109/ICRA.2011.5980121.

[34] Gavin Paul, Phillip Quin, Andrew Wing Keung To, and Dikai Liu. A sliding window approach to exploration for 3d map building using a biologically inspired bridge inspection robot. In *2015 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER)*, pages 1097–1102. IEEE, 2015. doi: 10.1109/CYBER.2015.7288098.

[35] Gavin Paul. *Autonomous exploration and mapping of complex 3d environments by means of a 6dof manipulator*. PhD thesis, University of Technology Sydney, 2010.

[36] Christian Dornhege and Alexander Kleiner. A frontier-void-based approach for autonomous exploration in 3d. *Advanced Robotics*, 27(6):459–468, 2013. doi: 10.1080/01691864.2013.763720.

[37] Gavin Paul, LiYang Liu, and Dikai Liu. A novel approach to steel rivet detection in poorly illuminated steel structural environments. In *2016 14th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pages 1–7. IEEE, 2016. doi: 10.1109/ICARCV.2016.7838630.

[38] Jan Faigl and Miroslav Kulich. On determination of goal candidates in frontier-based multi-robot exploration. In *2013 European Conference on Mobile Robots*, pages 210–215. IEEE, 2013. doi: 10.1109/ECMR.2013.6698844.

[39] Robert Reid, Andrew Cann, Calum Meiklejohn, Liam Poli, Adrian Boeing, and Thomas Braunl. Cooperative multi-robot navigation, exploration, mapping and object detection with ros. In *2013 IEEE Intelligent Vehicles Symposium (IV)*, pages 1083–1088. IEEE, 2013. doi: 10.1109/IVS.2013.6629610.

[40] Mahdi Hassan, Dikai Liu, and Gavin Paul. Collaboration of multiple autonomous industrial robots through optimal base placements. *Journal of Intelligent & Robotic Systems*, 90(1):113–132, 2018. doi: 10.1007/s10846-017-0647-x.

[41] Kai M Wurm, Armin Hornung, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. Octomap: A probabilistic, flexible, and compact 3d map representation

for robotic systems. In *ICRA 2010 workshop on best practice in 3D perception and modeling for mobile manipulation*, volume 2, 2010.

[42] Matan Keidar and Gal A Kaminka. Robot exploration with fast frontier detection: Theory and experiments. In *11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, pages 113–120, 2012.

[43] Matan Keidar and Gal A Kaminka. Efficient frontier detection for robot exploration. *The International Journal of Robotics Research*, 33(2):215–236, 2014. doi: 10.1177/0278364913494911.

[44] PGCN Senarathne, Danwei Wang, Zhuping Wang, and Qijun Chen. Efficient frontier detection and management for robot exploration. In *2013 IEEE International Conference on Cyber Technology in Automation, Control and Intelligent Systems*, pages 114–119. IEEE, 2013. doi: 10.1109/CYBER.2013.6705430.

[45] PGCN Senarathne and Danwei Wang. Incremental algorithms for safe and reachable frontier detection for robot exploration. *Robotics and Autonomous Systems*, 72:189–206, 2015. doi: 10.1016/j.robot.2015.05.009.

[46] Wenchuan Qiao, Zheng Fang, and Bailu Si. Sample-based frontier detection for autonomous robot exploration. In *2018 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1165–1170. IEEE, 2018. doi: 10.1109/ROBIO.2018.8665066.

[47] Juraj Oršulić, Damjan Miklić, and Zdenko Kovačić. Efficient dense frontier detection for 2-d graph slam based on occupancy grid submaps. *IEEE Robotics and Automation Letters*, 4(4):3569–3576, 2019. doi: 10.1109/LRA.2019.2928203.

[48] Phillip Quin, Alen Alempijevic, Gavin Paul, and Dikai Liu. Expanding wavefront frontier detection: An approach for efficiently detecting frontier cells. In *Australian Conference on Robotics and Automation (ACRA)*, 2014.

[49] Micha Sharir. A strong-connectivity algorithm and its applications in data flow analysis. *Computers & Mathematics with Applications*, 7(1):67–72, 1981. doi: 10.1016/0898-1221(81)90008-0.

[50] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping: part i. *IEEE robotics & automation magazine*, 13(2):99–110, 2006. doi: 10.1109/MRA.2006.1638022.

[51] David Filliat and Jean-Arcady Meyer. Map-based navigation in mobile robots:: I. a review of localization strategies. *Cognitive Systems Research*, 4(4):243–282, 2003. doi: 10.1016/S1389-0417(03)00008-1.

[52] Wolfgang Hess, Damon Kohler, Holger Rapp, and Daniel Andor. Real-time loop closure in 2d lidar slam. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1271–1278. IEEE, 2016. doi: 10.1109/ICRA.2016.7487258.

[53] Jun Jiang, Renato Miyagusuku, Atsushi Yamashita, and Hajime Asama. Glass confidence maps building based on neural networks using laser range-finders for mobile robots. In *2017 IEEE/SICE International Symposium on System Integration (SII)*, pages 405–410. IEEE, 2017. doi: 10.1109/SII.2017.8279246.

[54] Maurice F Fallon, Hordur Johannsson, and John J Leonard. Efficient scene simulation for robust monte carlo localization using an rgb-d camera. In *2012 IEEE international conference on robotics and automation (ICRA)*, pages 1663–1670. IEEE, 2012.

[55] Frank Dellaert, Dieter Fox, Wolfram Burgard, and Sebastian Thrun. Monte carlo localization for mobile robots. In *1999 IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 1322–1328. IEEE, 1999. doi: 10.1109/ROBOT.1999.772544.

[56] Dieter Fox. Adapting the sample size in particle filters through kld-sampling. *The international Journal of robotics research*, 22(12):985–1003, 2003. doi: 10.1177/0278364903022012001.

[57] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *2011 10th IEEE international symposium on mixed and augmented reality*, pages 127–136. IEEE, 2011. doi: 10.1109/ISMAR.2011.6092378.

[58] Stanimir Dragiev, Marc Toussaint, and Michael Gienger. Gaussian process implicit surfaces for shape estimation and grasping. In *2011 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2845–2850. IEEE, 2011. doi: 10.1109/ICRA.2011.5980395.

[59] Fabio Ramos and Lionel Ott. Hilbert maps: Scalable continuous occupancy mapping with stochastic gradient descent. *The International Journal of Robotics Research*, 35(14):1717–1730, 2016. doi: 10.1177/0278364916684382.

[60] Ransalu Senanayake and Fabio Ramos. Bayesian hilbert maps for dynamic continuous occupancy mapping. In *Conference on Robot Learning*, pages 458–471. PMLR, 2017.

[61] Oliver Williams and Andrew Fitzgibbon. Gaussian process implicit surfaces. In *Gaussian Processes in Practice*, 2006.

[62] Lan Wu, Raphael Falque, Victor Perez-Puchalt, Liyang Liu, Nico Pietroni, and Teresa Vidal-Calleja. Skeleton-based conditionally independent gaussian process implicit surfaces for fusion in sparse to dense 3d reconstruction. *IEEE Robotics and Automation Letters*, 5(2):1532–1539, 2020. doi: 10.1109/LRA.2020.2969175.

[63] Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning.* MIT press Cambridge, MA, 2006.

[64] Cansen Jiang, Danda Pani Paudel, Yohan Fougerolle, David Fofi, and Cedric Demonceaux. Static-map and dynamic object reconstruction in outdoor scenes using 3-d motion segmentation. *IEEE Robotics and Automation Letters*, 1(1):324–331, 2016. doi: 10.1109/LRA.2016.2517207.

[65] Berta Bescos, José M Fácil, Javier Civera, and José Neira. Dynaslam: Tracking, mapping, and inpainting in dynamic scenes. *IEEE Robotics and Automation Letters*, 3(4):4076–4083, 2018. doi: 10.1109/LRA.2018.2860039.

[66] Dirk Hahnel, Rudolph Triebel, Wolfram Burgard, and Sebastian Thrun. Map building with mobile robots in dynamic environments. In *2003 IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 1557–1563. IEEE, 2003. doi: 10.1109/ROBOT.2003.1241816.

[67] Frederic Bourgault, Alexei A Makarenko, Stefan B Williams, Ben Grocholsky, and Hugh F Durrant-Whyte. Information based adaptive robotic exploration. In *IEEE/RSJ international conference on intelligent robots and systems*, volume 1, pages 540–545. IEEE, 2002. doi: 10.1109/IRDS.2002.1041446.

[68] Zhengdong Zhang, Theia Henderson, Sertac Karaman, and Vivienne Sze. Fsmi: Fast computation of shannon mutual information for information-theoretic mapping. *The International Journal of Robotics Research*, 39(9):1155–1177, 2020. doi: 10.1177/0278364920921941.

[69] Maani Ghaffari Jadidi, Jaime Valls Miro, and Gamini Dissanayake. Gaussian processes autonomous mapping and exploration for range-sensing mobile robots. *Autonomous Robots*, 42(2):273–290, 2018. doi: 10.1007/s10514-017-9668-3.

[70] Kanrun Huang and Tucker Hermans. Building 3d object models during manipulation by reconstruction-aware trajectory optimization. *arXiv preprint arXiv:1905.03907*, 2019.

[71] Marija Popović, Teresa Vidal-Calleja, Gregory Hitz, Inkyu Sa, Roland Siegwart, and Juan Nieto. Multiresolution mapping and informative path planning for uav-based terrain monitoring. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1382–1388. IEEE, 2017. doi: 10.1109/IROS.2017.8202317.

[72] Gabriela Zarzar Gandler, Carl Henrik Ek, Mårten Björkman, Rustam Stolkin, and Yasemin Bekiroglu. Object shape estimation and modeling, based on sparse gaussian process implicit surfaces, combining visual data and tactile exploration. *Robotics and Autonomous Systems*, 126:103433, 2020. doi: 10.1016/j.robot.2020.103433.

[73] Rui Rocha, Jorge Dias, and Adriano Carvalho. Cooperative multi-robot systems:: A study of vision-based 3-d mapping using information theory. *Robotics and Autonomous Systems*, 53(3-4):282–311, 2005. doi: 10.1016/j.robot.2005.09.008.

[74] Brian J Julian, Sertac Karaman, and Daniela Rus. On mutual information-based control of range sensing robots for mapping applications. *The International Journal of Robotics Research*, 33(10):1375–1392, 2014. doi: 10.1177/0278364914526288.

[75] Sergio Caccamo, Yasemin Bekiroglu, Carl Henrik Ek, and Danica Kragic. Active exploration using gaussian random fields and gaussian process implicit surfaces. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 582–589. IEEE, 2016. doi: 10.1109/IROS.2016.7759112.

[76] Ji Zhang and Sanjiv Singh. Loam: Lidar odometry and mapping in real-time. In *Robotics: Science and Systems*, 2014.

[77] Thomas Whelan, Michael Kaess, Maurice F. Fallon, Hordur Johannsson, John J. Leonard, and John B. McDonald. Kintinuous: Spatially extended kinectfusion. In *AAAI 2012*, 2012.

[78] Thomas Whelan, Hordur Johannsson, Michael Kaess, John J Leonard, and John McDonald. Robust real-time visual odometry for dense rgb-d mapping. In *2013 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5724–5731. IEEE, 2013. doi: 10.1109/ICRA.2013.6631400.

[79] Tomasz Dziubich, Julian Szymański, Adam Brzeski, Jan Cychnerski, and Waldemar Korłub. Depth images filtering in distributed streaming. *Polish Maritime Research*, 23(2):91–98, 2016.

[80] Marius Miknis, Ross Davies, Peter Plassmann, and Andrew Ware. Near real-time point cloud processing using the pcl. In *2015 International Conference on Systems, Signals and Image Processing (IWSSIP)*, pages 153–156. IEEE, 2015. doi: 10.1109/IWSSIP.2015.7314200.

[81] Thanh Long Vu, Liyang Liu, Gavin Paul, and Teresa Vidal Calleja. Rectangular-shaped object recognition and pose estimation. In *Australian Conference on Robotics and Automation (ACRA)*, 2019.

[82] Xian-Feng Han, Jesse S Jin, Ming-Jie Wang, Wei Jiang, Lei Gao, and Liping Xiao. A review of algorithms for filtering the 3d point cloud. *Signal Processing: Image Communication*, 57:103–112, 2017. doi: 10.1016/j.image.2017.05.009.

[83] Xian-Feng Han, Jesse S Jin, Ming-Jie Wang, and Wei Jiang. Guided 3d point cloud filtering. *Multimedia Tools and Applications*, 77(13):17397–17411, 2018. doi: 10.1007/s11042-017-5310-9.

[84] Anh Nguyen and Bac Le. 3d point cloud segmentation: A survey. In *2013 6th IEEE conference on robotics, automation and mechatronics (RAM)*, pages 225–230. IEEE, 2013. doi: 10.1109/RAM.2013.6758588.

[85] Bastian Oehler, Joerg Stueckler, Jochen Welle, Dirk Schulz, and Sven Behnke. Efficient multi-resolution plane segmentation of 3d point clouds. In *International Conference on Intelligent Robotics and Applications*, pages 145–156. Springer, 2011. doi: 10.1007/978-3-642-25489-5_15.

[86] Alexander JB Trevor, Suat Gedikli, Radu B Rusu, and Henrik I Christensen. Efficient organized point cloud segmentation with connected components. *Semantic Perception Mapping and Exploration (SPME)*, 2013.

[87] Giancarlo Iannizzotto and Lorenzo Vita. Fast and accurate edge-based segmentation with no contour smoothing in 2-d real images. *IEEE Transactions on Image Processing*, 9(7):1232–1237, 2000. doi: 10.1109/83.847835.

[88] Vrushali Patil, Indraneel Patil, V Kalaichelvi, and R Karthikeyan. Extraction of weld seam in 3d point clouds for real time welding using 5 dof robotic arm. In *2019 5th International Conference on Control, Automation and Robotics (ICCAR)*, pages 727–733. IEEE, 2019. doi: 10.1109/ICCAR.2019.8813703.

[89] Yusheng Xu, Wei Yao, Ludwig Hoegner, and Uwe Stilla. Segmentation of building roofs from airborne lidar point clouds using robust voxel-based region growing. *Remote Sensing Letters*, 8(11):1062–1071, 2017. doi: 10.1080/2150704X.2017.1349961.

[90] Arun Das, James Servos, and Steven L Waslander. 3d scan registration using the normal distributions transform with ground segmentation and point cloud clustering. In *2013 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2207–2212. IEEE, 2013. doi: 10.1109/ICRA.2013.6630874.

[91] M Lamine Tazir, Tawsif Gokhool, Paul Checchin, Laurent Malaterre, and Laurent Trassoudaine. Cicp: Cluster iterative closest point for sparse–dense point cloud registration. *Robotics and Autonomous Systems*, 108:66–86, 2018. doi: 10.1016/j.robot.2018.07.003.

[92] F James Rohlf. 12 single-link clustering algorithms. *Handbook of statistics*, 2:267–284, 1982. doi: 10.1016/S0169-7161(82)02015-X.

[93] Pierre Hansen and Michel Delattre. Complete-link cluster analysis by graph coloring. *Journal of the American Statistical Association*, 73(362):397–403, 1978. doi: 10.1080/01621459.1978.10481589.

[94] Ying Zhao and George Karypis. Comparison of agglomerative and partitional document clustering algorithms. Technical report, Minnesota Univ Minneapolis Dept Of Computer Science, 2002.

[95] Ross Tredinnick, Markus Broecker, and Kevin Ponto. Progressive feedback point cloud rendering for virtual reality display. In *2016 IEEE Virtual Reality (VR)*, pages 301–302. IEEE, 2016. doi: 10.1109/VR.2016.7504773.

[96] Daniele Bonatto, Ségolène Rogge, Arnaud Schenkel, Rudy Ercek, and Gauthier Lafruit. Explorations for real-time point cloud rendering of natural scenes in virtual reality. In *2016 International Conference on 3D Imaging (IC3D)*, pages 1–7. IEEE, 2016. doi: 10.1109/IC3D.2016.7823453.

[97] Liang Gong, Jonatan Berglund, Dennis Saluäär, and Björn Johansson. A novel vr tool for collaborative planning of manufacturing process change using point cloud data. *Procedia CIRP*, 63:336–341, 2017. doi: 10.1016/j.procir.2017.03.089.

[98] Florian Wirth, Jannik Quehl, Jeffrey Ota, and Christoph Stiller. Pointatme: efficient 3d point cloud labeling in virtual reality. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 1693–1698. IEEE, 2019. doi: 10.1109/IVS.2019.8814115.

[99] Manuel Veit and Antonio Capobianco. Go'then'tag: A 3-d point cloud annotation technique. In *2014 IEEE Symposium on 3D User Interfaces (3DUI)*, pages 193–194. IEEE, 2014. doi: 10.1109/3DUI.2014.6798886.

[100] Andrew C Boud, David J Haniff, Chris Baber, and SJ Steiner. Virtual reality and augmented reality as a training tool for assembly tasks. In *1999 IEEE International Conference on Information Visualization (Cat. No. PR00210)*, pages 32–36. IEEE, 1999. doi: 10.1109/IV.1999.781532.

[101] Justin D Bric, Derek C Lumbard, Matthew J Frelich, and Jon C Gould. Current state of virtual reality simulation in robotic surgery training: a review. *Surgical endoscopy*, 30(6):2169–2178, 2016. doi: 10.1007/s00464-015-4517-y.

[102] Ilkay Yavrucuk, Eser Kubali, and Onur Tarimci. A low cost flight simulator using virtual reality tools. *IEEE Aerospace and Electronic Systems Magazine*, 26(4):10–14, 2011. doi: 10.1109/MAES.2011.5763338.

[103] Hiroaki Nishino, Kouta Murayama, Tsuneo Kagawa, and Kouichi Utsumiya. A japanese calligraphy trainer based on skill acquisition through haptization. In *2010 24th IEEE International Conference on Advanced Information Networking and Applications*, pages 1225–1232, 2010. doi: 10.1109/AINA.2010.112.

[104] Elias Matsas and George-Christopher Vosniakos. Design of a virtual reality training system for human–robot collaboration in manufacturing tasks. *International Journal on Interactive Design and Manufacturing (IJIDeM)*, 11(2):139–153, 2017. doi: 10.1007/s12008-015-0259-2.

[105] Nirit Gavish, Teresa Gutiérrez, Sabine Webel, Jorge Rodríguez, Matteo Peveri, Uli Bockholt, and Franco Tecchia. Evaluating virtual reality and augmented reality training for industrial maintenance and assembly tasks. *Interactive Learning Environments*, 23(6):778–798, 2015. doi: 10.1080/10494820.2013.815221.

[106] Francesca De Crescenzio, Massimiliano Fantini, Franco Persiani, Luigi Di Stefano, Pietro Azzari, and Samuele Salti. Augmented reality for aircraft maintenance training and operations support. *IEEE Computer Graphics and Applications*, 31(1): 96–101, 2010. doi: 10.1109/MCG.2011.4.

[107] Jacopo Aleotti and Stefano Caselli. Grasp recognition in virtual reality for robot pregrasp planning by demonstration. In *2006 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2801–2806, 2006. doi: 10.1109/ROBOT.2006.1642125.

[108] Zaid Gharaybeh, Howard Chizeck, and Andrew Stewart. *Telerobotic control in virtual reality*. IEEE, 2019. doi: 10.23919/OCEANS40490.2019.8962616.

[109] David Whitney, Eric Rosen, Daniel Ullman, Elizabeth Phillips, and Stefanie Tellex. Ros reality: A virtual reality framework using consumer-grade hardware for ros-enabled robots. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–9. IEEE, 2018. doi: 10.1109/IROS.2018.8593513.

[110] David Whitney, Eric Rosen, Elizabeth Phillips, George Konidaris, and Stefanie Tellex. Comparing robot grasping teleoperation across desktop and virtual reality with ros reality. In *Robotics Research*, pages 335–350. Springer, 2020. doi: 10.1007/978-3-030-28619-4_28.

[111] Luis Pérez, Eduardo Diez, Rubén Usamentiaga, and Daniel F García. Industrial robot control and operator training using virtual reality interfaces. *Computers in Industry*, 109:114–120, 2019. doi: 10.1016/j.compind.2019.05.001.

[112] Ali Ahmad Malik, Tariq Masood, and Arne Bilberg. Virtual reality in manufacturing: immersive and collaborative artificial-reality in design of human-robot workspace. *International Journal of Computer Integrated Manufacturing*, 33(1):22–37, 2020. doi: 10.1080/0951192X.2019.1690685.

[113] Patrick Rückert, Laura Wohlfromm, and Kirsten Tracht. Implementation of virtual reality systems for simulation of human-robot collaboration. *Procedia Manufacturing*, 19:164–170, 2018. doi: 10.1016/j.promfg.2018.01.023.

[114] Max Krichenbauer, Goshiro Yamamoto, Takafumi Taketom, Christian Sandor, and Hirokazu Kato. Augmented reality versus virtual reality for 3d object manipulation. *IEEE transactions on visualization and computer graphics*, 24(2):1038–1048, 2017. doi: 10.1109/TVCG.2017.2658570.

[115] LL Gong, SK Ong, and AYC Nee. Projection-based augmented reality interface for robot grasping tasks. In *2019 4th International Conference on Robotics, Control and Automation*, pages 100–104, 2019. doi: 10.1145/3351180.3351204.

[116] AWW Yew, SK Ong, and AYC Nee. Immersive augmented reality environment for the teleoperation of maintenance robots. *Procedia Cirp*, 61:305–310, 2017. doi: 10.1016/j.procir.2016.11.183.

[117] Michael Gradmann, Eric M Orendt, Edgar Schmidt, Stephan Schweizer, and Dominik Henrich. Augmented reality robot operation interface with google tango. In *ISR 2018; 50th International Symposium on Robotics*, pages 1–8. VDE, 2018.

[118] Yuan Lin, Shuang Song, and Max Q-H Meng. The implementation of augmented reality in a robotic teleoperation system. In *2016 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, pages 134–139. IEEE, 2016. doi: 10.1109/RCAR.2016.7784014.

[119] D Ni, AWW Yew, SK Ong, and AYC Nee. Haptic and visual augmented reality interface for programming welding robots. *Advances in Manufacturing*, 5(3):191–198, 2017. doi: 10.1007/s40436-017-0184-7.

[120] Hamidreza Houshiar and Andreas Nüchter. 3d point cloud compression using conventional image compression for efficient data transmission. In *2015 XXV International Conference on Information, Communication and Automation Technologies (ICAT)*, pages 1–8. IEEE, 2015. doi: 10.1109/ICAT.2015.7340499.

[121] Chenguang Yang, Zunran Wang, Wei He, and Zhijun Li. Development of a fast transmission method for 3d point cloud. *Multimedia Tools and Applications*, 77(19): 25369–25387, 2018. doi: 10.1007/s11042-018-5789-8.

[122] Bhaskar Anand, Vivek Barsaiyan, Mrinal Senapati, and Pachamuthu Rajalakshmi. Real time lidar point cloud compression and transmission for intelligent transportation system. In *2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring)*, pages 1–5. IEEE, 2019. doi: 10.1109/VTCSpring.2019.8746417.

[123] Georg Bartels and Michael Beetz. Perception-guided mobile manipulation robots for automation of warehouse logistics. *KI-Künstliche Intelligenz*, 33(2):189–192, 2019. doi: 10.1007/s13218-019-00585-2.

[124] Mikko Lauri, Nikolay Atanasov, George Pappas, and Risto Ritala. Active object recognition via monte carlo tree search. In *ICRA 2015 Workshop: Beyond Geometric Constraints*, 2015.

[125] Fouad Sukkar, Graeme Best, Chanyeol Yoo, and Robert Fitch. Multi-robot region-of-interest reconstruction with dec-mcts. In *2019 International conference on*

robotics and automation (ICRA), pages 9101–9107. IEEE, 2019. doi: 10.1109/ICRA.2019.8793560.

[126] Gavin Paul, Shuyuan Mao, Liyang Liu, and Rong Xiong. Mapping repetitive structural tunnel environments for a biologically-inspired climbing robot. In *ASSISTIVE ROBOTICS: Proceedings of the 18th International Conference on CLAWAR 2015*, pages 325–333. World Scientific, 2016. doi: 10.1142/9789814725248_0041.

[127] Soohwan Kim and Jonghyuk Kim. Continuous occupancy maps using overlapping local gaussian processes. In *2013 IEEE/RSJ international conference on intelligent robots and systems*, pages 4709–4714. IEEE, 2013. doi: 10.1109/IROS.2013.6697034.

[128] UNIVERSAL ROBOT UR5e, a flexible and lightweight robotic arm, 2020. URL `https://www.universal-robots.com/products/ur5-robot`.

[129] Thanh Long Vu, Liyang Liu, Gavin Paul, and Teresa Vidal Calleja. Rectangular-shaped object recognition and pose estimation. In *Australian Conference on Robotics and Automation (ACRA)*, 2019.

[130] Thanh Long Vu, Dac Dang Khoa Nguyen, Dinh Tung Le, Sheila Sutjipto, and Gavin Paul. Investigation of annotation-assisted user performance in virtual reality-based remote robot control. In *Australian Conference on Robotics and Automation (ACRA)*, 2022.

[131] Thanh Long Vu, Dac Dang Khoa Nguyen, Sheila Sutjipto, Dinh Tung Le, and Gavin Paul. Investigation of user performance in virtual reality-based annotation-assisted remote robot control. In *Proceedings of the 28th ACM Symposium on Virtual Reality Software and Technology*, pages 1–2, 2022.

[132] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[133] Gavin Paul, Ngaiming Kwok, and Dikai Liu. A novel surface segmentation approach for robotic manipulator-based maintenance operation planning. *Automation in Construction*, 29:136–147, 2013. doi: 10.1016/j.autcon.2012.08.007.

[134] Taosong He, Lichan Hong, Arie Kaufman, Amitabh Varshney, and Sidney Wang. Voxel based object simplification. In *6th conference on Visualization'95*, page 296. IEEE Computer Society, 1995. doi: 10.1109/VISUAL.1995.485142.

[135] Rolf Adams and Leanne Bischof. Seeded region growing. *IEEE Transactions on pattern analysis and machine intelligence*, 16(6):641–647, 1994. doi: 10.1109/34.295913.

[136] Anh-Vu Vo, Linh Truong-Hong, Debra F Laefer, and Michela Bertolotto. Octree-based region growing for point cloud segmentation. *ISPRS Journal of Photogrammetry and Remote Sensing*, 104:88–100, 2015. doi: 10.1016/j.isprsjprs.2015.01.011.

[137] Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459, 2010. doi: 10.1002/wics.101.

[138] Rocco Furferi, Lapo Governi, Matteo Palai, and Yary Volpe. From unordered point cloud to weighted b-spline-a novel pca-based method. In *Applications of Mathematics and Computer Engineering-American Conference on Applied Mathematics, AMERICAN-MATH*, volume 11, pages 146–151, 2011.

[139] Sahan S Hiniduma Udugama Gamage and Joan Lasenby. New least squares solutions for estimating the average centre of rotation and the axis of rotation. *Journal of biomechanics*, 35(1):87–93, 2002. doi: 10.1016/S0021-9290(01)00160-9.

[140] James C Wang. Variation of the average rotation angle of the dna helix and the superhelical turns of covalently closed cyclic λ dna. *Journal of molecular biology*, 43 (1):25–39, 1969. doi: 10.1016/0022-2836(69)90076-X.

[141] Avishek Chatterjee and Venu Madhav Govindu. Efficient and robust large-scale rotation averaging. In *IEEE International Conference on Computer Vision*, pages 521–528, 2013.

[142] Richard Hartley, Jochen Trumpf, Yuchao Dai, and Hongdong Li. Rotation averaging. *International journal of computer vision*, 103(3):267–305, 2013. doi: 10.1007/s11263-012-0601-0.

[143] John F Kros, Mike Lin, and Marvin L Brown. Effects of the neural network s-sigmoid function on kdd in the presence of imprecise data. *Computers & operations research*, 33(11):3136–3149, 2006. doi: 10.1016/j.cor.2005.01.024.

[144] Alexander JB Trevor, Suat Gedikli, Radu B Rusu, and Henrik I Christensen. Efficient organized point cloud segmentation with connected components. *Semantic Perception Mapping and Exploration (SPME)*, 2013.

[145] Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459, 2010. doi: 10.1002/wics.101.

[146] David A Simon, Martial Hebert, and Takeo Kanade. Real-time 3-d pose estimation using a high-speed range sensor. In *1994 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2235–2241. IEEE, 1994. doi: 10.1109/ROBOT.1994.350953.

[147] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv:2007.08501*, 2020.

[148] Thanh Long Vu, Dinh Tung Le, Dac Dang Khoa Nguyen, Sheila Sutjipto, and Gavin Paul. Investigating the effect of sensor data visualization variances in virtual reality. In *Proceedings of the 27th ACM Symposium on Virtual Reality Software and Technology*, pages 1–5, 2021.

[149] ROS.org. image_transport, 2020. URL `http://wiki.ros.org/image_transport`.

[150] Unity-Technologies. Ros-tcp-connector, 2021. URL `https://github.com/Unity-Technologies/ROS-TCP-Connector`.

[151] Tomoko Nemoto and David Beglar. Likert-scale questionnaires. In *JALT 2013 conference proceedings*, pages 1–8, 2014.

[152] David Coleman, Ioan Alexandru Sucan, Sachin Chitta, and Nikolaus Correll. Reducing the barrier to entry of complex robotic software: a moveit! case study. *ArXiv*, abs/1404.3785, 2014.