NAVIGATION AND PATH-PLANNING OF MICRO-AUTONOMOUS UNDERWATER VEHICLES IN ENCLOSED AND CLUTTERED UNDERWATER ENVIRONMENTS

A THESIS SUBMITTED TO THE UNIVERSITY OF MANCHESTER

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

IN THE FACULTY OF SCIENCE & ENGINEERING

2019

By

Yibin Peng

Department of Electrical and Electronic Engineering

Contents

List of Figures **8**

List of tables **14**

Abstract **15**

Declaration **16**

Copyright Statement **17**

List of Publications **18**

Acronyms **19**

Acknowledgement **21**

Chapter 1 ..... **22**

Introduction **22**

1.1Background and Motivation 24

1.1.1Nuclear Storage Ponds 25

1.1.2 Previous and Related Work 26

1.1.2.1 AAS4IP Project 26

1.1.2.2 AVEXIS Project 27

1.2 The Main Challenges 28

1.3 Aims and Objectives 29

1.4 Contributions 30

1.5 Thesis Outline 30

Chapter 2 ..... **32**

Review of Navigation Methods **32**

2.1 Introduction 32

2.2 Inertial Navigation 33

2.3 Acoustic Positioning Methods 33

2.3.1 Short Baseline 34
2.3.2 SSBL and LBL 35
2.4 AUV Underwater Mapping Methods 36
2.4.1 Side-Scan Sonar 36
2.4.2 Multi-Beam Sonar 36
2.4.3 Forward-Looking Sonar (FLS) 37
2.4.4 Echo Sounding Sonar 37
2.5 Geophysical Navigation 38
2.5.1 Existing Map 38
2.5.2 Simultaneous Localisation and Mapping (SLAM) 39
2.6 AUV Applications 39
2.7 Summary and Discussion 41

Chapter 3 44
Aerial Mapping and Ray Tracing 44
3.1 Introduction 44
3.2 Acoustic Range Finders Technique 47
3.3 Ultrasound Transducer for Echo Sounding 47
3.4 Important Physical Characteristics of Nuclear Storage Pond 49
3.5 The Approach Taken to Study Aerial Mapping in Pond 51
3.6 Choosing of Ray Tracing Method 51
3.7 Ray-Tracing in Underwater Aerial Survey 52
3.7.1 Ray-Tracing Algorithm 52
3.7.2 Impulse Response in Room Acoustics 54
3.7.3 Applying Ray Tracing to Storage Pond Simulation 56
3.7.3.1 Reflection from a 2D inclined surface 59
3.7.3.2 Position of the Reflection Point 61
3.7.3.3 Estimation of the Inclination Angle 62
3.7.3.4 Reflection from 3D Inclined Surfaces 63
3.8 Depth Measurement Data 65
3.9 Summary 65

Chapter 4 66
Implementation of the Ray-Tracing Algorithm 66

4.1 Introduction 66
4.2 Geometrical Description of the Environment 68
4.3 Decomposition of Ray Projection Angle 69
4.4 Lines-Plane Intersection 70
4.4.1 Ray-Polygon Intersection 70
4.4.2 Visibility Test 72
4.5 Line-Receiver Intersection 73
4.5.1 Ray-Sphere Intersection 73
4.6 Ray Reflection and Energy Impulse Response 74
4.8 Summary 77

Chapter 5 78
Aerial Mapping Simulations and Experiment 78
5.1 Introduction 78
5.2 Simulation Assumptions 78
5.3 Sampling Resolution and Accuracy 79
5.4 Multi-Resolution Survey 84
5.5 Tilted Containers 85
5.6 Special Case Study 88
5.6.1 Case 1: Detect Multiple Obstacles in One Scan 88
5.6.2 Varying the Height of the Survey Plane 90
5.7 Well-Structured Modern Nuclear Storage Ponds 92
5.8 Hollow Containers 93
5.9 Limitations of Aerial Mapping 95
5.10 Aerial Mapping Experiments 95
5.10.1 Configurations of the Aerial Mapping Experiments 96
5.10.2 Experiment 1: Performance of HC-SR04 98
5.10.3 Experiment 2: Study of the Reflection Pulse 99
5.10.4 Experiment 3: Effects of the Beam Angle 103
5.10.5 Experiment 4: Effect of the Height of the Measurement Plane 104
5.10.6 Experiment 5: Gap Distance of Two Neighbour Obstacles 104
5.10.7 Experiment 6: Aerial mapping for a Pond-like Environment with Four Obstacles 106
5.11 Summary and Conclusion 108

Chapter 6 110
Review of Path-planning Algorithms 110
6.1 Introductions 110
6.2 Basic Concept and Terminology 111
6.3 C-Space Modelling Techniques 111
6.4 Path-planning Algorithms 114
6.4.1 Cell Decomposition 114
6.4.2 Voronoi Diagrams 115
6.4.3 Visibility Graph 116
6.4.4 Probabilistic Roadmap 117
6.4.5 Rapidly- Explore Random Trees 118
6.4.6 Artificial Potential Fields 120
6.4.6.1 Improved Artificial Potential Fields Function to Address the Collision Problem 122
6.4.6.2 Improved Artificial Potential Fields Function to Address the Goal Unreachability Problem 122
6.4.6.3 Introduce Simulated Annealing to Address the Local Minimum Problem 123
6.4.7 Dijkstra's Algorithm 124
6.4.8 The A* Algorithm 126
6.4.9 RRT-A* 128
6.4.10 Genetic Algorithms (GAs) 128
6.5 Discussion and Summary 129

Chapter 7 132
Path-planning Simulations for Storage Ponds 132
7.1. Systematic Simulations of the Proposed Environment 132
7.1.1 Performance of APF Simulation 133
7.1.2 Performance of SA-APF Simulations 135
7.1.3 Performance of A* Simulation 136
7.1.4 Performance of RRT and RRT-A* Simulations 137
7.2. Simulations with the Goal inside a Hollow Canister 142
7.2.1 Performance of SA-APF Simulation 143
7.2.2 Performance of A* simulation 144
7.2.3 Performance of RRT-A* simulation 144

7.2.4 Discussion of SA-APF 145
7.3 Discussion and Summary 148

Chapter 8 149
3D Grid Map Construction and A* Path-planning Algorithm 149
8. 1 Introduction 149
8.1.1 General Description 149
8.1.2 Non-inclined objects 152
8.1.3 Single inclined object 152
8.1.4 Multiple inclined objects 152
8.2 Establishment of the 3D Digital Grid Map 152
8.2.1 Boundary Tracing 153
8.2.2 Modified Depth Measurement Data 156
8.2.3 Environment Construction by Point Cloud 157
8.2.4 Grid Method to Establish 3D Occupancy Grid Map 159
8.3 Path-planning on 3D Digital Grid Map Based on A* Algorithm 161
8.4 The Simulation Experiments for Pond-Like Environment 161
8.4.1 The Simulation for Pond-Like Environment Case 1 162
8.4.2 The Simulation for Pond-Like Environment Case 2 164
8.5 Errors in Measurement 167
8.5.1 Bounding Box Expansion in 2D 169
8.5.2 Bounding Box Expansion in 3D 171
8.5.3 Localisation Error 173
8.6 Summary 174

Chapter 9 175
Multiresolution Hierarchical Data Sampling and Path-Planning 175
9.1 Introduction 175
9.2 Hierarchical Data Sampling 176
9.3 Multiresolution A* Path-Planning 178
9.4 Summary 181

Chapter 10 182
Conclusion and Future Work 182
10.1 Thesis Summary and Conclusions 182

10.2 Future Research 184
10.2.1Vertical Survey 185
10.2.2 Further Experiments 185
10.2.3 Further Development of the A* Path-Planning Algorithm 185
10.2.4 True Size μAUV Path-Planning 186

Reference 187

Appendix A: Area Data for Different Sampling Resolution 202

Appendix B: Recorded Aerial Mapping Data 203

Appendix C: MB1340 Distance Measurement Tests 204

Appendix D: Boundary points and vertices 205

Appendix E: A* Path-Planning on Occupancy Grid Map 206

Appendix F: Mapping Result with Gaussian Error 208

Appendix G: Calculation of δ 212

Total Word Count: 47016

List of Figures

Figure 1.1: Modern nuclear storage pond [6] 26
Figure 1.3: µAUV [12] 27
Figure 1.4 The ROV designed in the AVEXIS project [15] 28
Figure 2.1: SBL [25] 34
Figure 2.2: Schematic diagram of SBL 34
Figure 2.7: Side-scan sonar [145] 36
Figure 2.8: Multi-beam sonar [146] 37
Figure 2.8: FLS [147] 37
Figure 2.9: Echo sounding [179] 38
Figure 2.10: AUV CTINEU in water and surface buoy for localising the vehicle [142] 40
Figure 3.1: A land aerial survey [32] 45
Figure 3.2: Nuclear pond aerial mapping 46
Figure 3.3: Mapping route in 2D 46
Figure 3.4: HC-SR04 Working Principle [160] 48
Figure 3.5: Distance measuring [160] 48
Figure 3.6: Outputs of MB1340 [164] 49
Figure 3.7: Modern nuclear storage pond [6] Figure 3.8: Legacy nuclear storage pond [9] 50
Figure 3.9: Sludge at the bottom of a legacy storage pond [166] 50
Figure 3.10: Specular reflection and diffuse reflection [44] 53
Figure 3.11: Scattering coefficient and absorption coefficient [44] 54
Figure 3.12: Ray propagation 55
Figure 3.13: Acoustic impulse response of a room [44] 55
Figure 3.14: Ray reflection for aerial mapping 57
Figure 3.15: A scan of measuring an object 58
Figure 3.16: Impulse response of the scan 58
Figure 3.17: Small beam without normal reflection 59
Figure 3.18: The normal reflection 60
Figure 3.19: Sensor survey to an inclined surface 60

Figure 3.20: 3 sensors array 61
Figure 3.21: Geometry relationship between normal reflection point and scanning point 61
Figure 3.22: Estimation of the inclination angle 62
Figure 3.23: Normal reflection in the 3D scene 63
Figure 3.24: Mapping on to a tilted object 64
Figure 3.25: Front view of scans (along the x-axis) 64
Figure 3.26: Side view of scans (along the y-axis) 64
Figure 4.1: Flow chart ray-tracing algorithm 67
Figure 4.2: 3D pond enclosure 68
Figure 4.3: Rotation 69
Figure 4.4: Example of creating an object 69
Figure 4.5: h_angle and v_angle 70
Figure 4.6: Ray-Polygon intersections at the upper surface 71
Figure 4.7: Line sphere intersection 73
Figure 4.8: line receiver intersection 73
Figure 4.9: Reflection vector 75
Figure 4.10: Diffuse reflection Vector 76
Figure 5.1: 2 different shaped containers 79
Figure 5.2: Simulation result under a sampling resolution of 2m×2m 80
Figure 5.3: Simulation result under a sampling resolution of 1m×1m 80
Figure 5.4: Simulation result under a sampling resolution of 0.5m×0.5m 80
Figure 5.5: Simulation result under a sampling resolution of 0.25m×0.25m 81
Figure 5.6: Simulation result under a sampling resolution of 0.1m×0.1m 81
Figure 5.7: The offset distance with respect to sampling resolution 82
Figure 5.8: (a) Area of measured to plane against sampling resolution; (b) Area discrepancy in percentage against sampling resolution 83
Figure 5.9: (a) Depth measurement points of low sampling resolution; (b) depth measurement points high sampling resolution; (c) depth measurement points that contains both low and high sampling resolution 85
Figure 5.10: (a) MATLAB geometry input for the tilt angle of 10°; (b) depth measurement points; (c) surface plot reconstructed object 86
Figure 5.11: MATLAB geometry input for the tilt angle of 20°; (b) depth measurement points; (c) surface plot reconstructed object 86

Figure 5.12: (a) MATLAB geometry input for the tilt angle of 30°; (b) depth measurement points; (c) surface plot reconstructed object 87
Figure 5.13: Indication of detected and undetected area 87
Figure 5.14: Scene of case 1 88
Figure 5.15: Aerial mapping scene for case 1 89
Figure 5.16: Surface plot of aerial mapping result for case 1 89
Figure 5.17: Scanning height is 9m 90
Figure 5.18: Scanning height is 8m 91
Figure 5.19: Scanning height is 7m 91
Figure 5.20: Scanning height is 6m 91
Figure 5.21: Scanning height is 5m 91
Figure 5.22: MATLAB model 92
Figure 5.23: Simulation result of modern nuclear storage pond aerial survey 93
Figure 5.24: Storage pond in Sellafield [55] Figure 5.25: MATLAB geometry of hollow canister. 94
Figure 5.26: (a) Depth measurement points of the hollow canister; (b) The surface reconstructed object of hollow canister 94
Figure 5.27: Canister lies above another 95
Figure 5.28: Test rig, instrumentations, and block diagram overview 96
Figure 5.29 Used sensors 97
Figure 5.30: Distance measurements along the axis y = 22cm 98
Figure 5.31: Experiment results and simulation results 99
Figure 5.32: Reflections at the side surface 99
Figure 5.33: Test Case 1 configuration 100
Figure 5.34: Analogue envelope of MB1340 for test case 1 101
Figure 5.35: Testing case 2 102
Figure 5.37: Experiment results and simulation results 103
Figure 5.38: MB7060 result for different heights 104
Figure 5.39: Schematic two nearby objects 105
Figure 5.40: Mapping results of different gaps 105
Figure 5.41: Estimation of the gap distance 106
Figure 5.42: The setup of the pond-like environment 107
Figure 5.43: Schematic of the environment 107
Figure 5.44: Aerial mapping results 108

Figure 5.45: Surface plot reconstructed objects 108
Figure 6.1: Workspace *C-space* translation 111
Figure 6.2: 3D space division 113
Figure 6.3: Octree decomposition model [77] 113
Figure 6.4: Vertical cell decomposition diagram [78] 114
Figure 6.5: Approximate Cell Decomposition [79] 115
Figure 6.6: Voronoi diagram [81] 116
Figure 6.7: Visibility graph diagram [81] 117
Figure 6.8: Probabilistic roadmap example [175] 118
Figure 6.9: An example of expanding explore tree [91] 119
Figure 6.10: A gradient map of the attractive potential field, the repulsive potential field, and the composite potential field are shown [95] 120
Figure 6.11: Dijkstra's algorithm 125
Figure 6.12: A* path-planning 127
Figure 7.1: Pond environment 133
Figure 7.2: APF path for starting position of (1, 1, 10) 133
Figure 7.3: APF path for starting position of (1, 1, 8) 134
Figure 7.4: APF path for starting position of (1, 1, 6) 134
Figure 7.5: APF path for starting position of (1, 1, 4) 134
Figure 7.6: APF path for starting position of (1, 1, 2) 134
Figure 7.7: SA-APF path for starting position of (1, 1, 6) 135
Figure 7.8: A* path for starting position (1, 1, 10) 136
Figure 7.9: A* path for starting position (1, 1, 8) 136
Figure 7.10: A* path for starting position (1, 1, 6) 136
Figure 7.11: A* path for starting position (1, 1, 4) 137
Figure 7.12: A* path for starting position (1, 1, 2) 137
Figure 7.13a: RRT-A* path for starting position (1, 1, 10) 137
Figure 7.13b: RRT path for starting position (1, 1, 10) 138
Figure 7.14a: RRT-A* path for starting position (1, 1, 8) 138
Figure 7.14b: RRT path for starting position (1, 1, 8) 138
Figure 7.15a: RRT-A* path for starting position (1, 1, 6) 138
Figure 7.15b: RRT path for starting position (1, 1, 6) 139
Figure 7.16a: RRT-A* path for starting position (1, 1, 4) 139
Figure 7.16b: RRT path for starting position (1, 1, 4) 139

Figure 7.17a: RRT-A* path for starting position (1, 1, 2) 139
Figure 7.17b: RRT path for starting position (1, 1, 2) 140
Figure 7.18: Open top nuclear storage canister 142
Figure 7.19: Hollow canister scene 143
Figure 7.20: SA-APF path in a hollow canister scene (a) 3D view (b) Plan view 143
Figure 7.21: A* path in hollow canister scene 144
Figure 7.22: RRT-A* in hollow canister scene 144
Figure 7.23: Force analysis 145
Figure 7.24: Intermediate checkpoint (a) 3D view (b) Plan view 146
Figure 7.25: Line of sight 146
Figure 7.26: Testing case 147
Figure 7.27: Intermediate goal V 147
Figure 7.28: Intermediate point B 148
Figure 8.1: Survey grid of aerial survey 150
Figure 8.2: The processing flow chart 153
Figure 8.3: 8 Directions respect to the current point 154
Figure 8.4: Boundary Tracing 154
Figure.8.5. Removal of a traced object 155
Figure 8.6: A schematic of the linear perpendicular height interpolation 157
Figure 8.7: A schematic of the linear perpendicular height interpolation 158
Figure 8.8: Determine the grid cell 159
Figure 8.9: The whole environment Figure 8.10: The meshed environment 160
Figure 8.11: 2D occupancy grid map 160
Figure 8.12: A* path-planning on a 2D occupancy grid map 161
Figure 8.13: Pond-like environment Case 1 162
Figure 8.14: Raw depth measurements of the pond environment Case 1 162
Figure 8.15: Height interpolated point cloud for case 1 163
Figure 8.16: 3D grid map for case 1 163
Figure 8.17: A* path: (a) 3D view for case 1. (b) Plan view for case 1. 164
Figure 8.18: Pond-like environment Case 2 165
Figure 8.19: Raw point cloud of pond-like environment Case 2 165
Figure 8.20: Height interpolated point cloud of Case 2 165
Figure 8.21: 3D grid map of Case 2 166
Figure 8.22: A* path: (a) 3D view for case 2. (b) Plan view for case 2 166

Figure 8.23: Object created by measurement errors in height 168
Figure 8.24: Bounding box 170
Figure 8.25: The area of the bounding box, the actual object, and the measured object of each sample 171
Figure 8.26: Plane view of a 3D object 172
Figure 8.27: Cross-section 172
Figure 8.28: Point cloud of the hexagonal prism's bounding box 173
Figure 8.29: Planned path 173
Figure 8.30: Bounding box for the positional error in X and Y-axis 174
Figure 9.1: Geometry model of multiple obstacles 176
Figure 9.2: Depth measurement points obtained by first phase data sampling 177
Figure 9.3: Reconstructed objects obtained by surface plot 177
Figure 9.4: Hierarchical 3D point cloud map 178
Figure 9.5: Multi-resolution search 178
Figure 9.6: A* path on hierarchical 3D point cloud map 179
Figure 9.7: 3D view of A* path on hierarchical 3D point cloud map 179
Figure 9.8: Plan view of A* path on hierarchical 3D point cloud map 180
Figure 9.9: Hierarchical 3D point cloud map 180
Figure 9.10: Multi-resolution A* path-planning for a cluttered environment 180
Figure 9.11: Plan view of multi-resolution A* path-planning for cluttered environment 181

List of tables

Table 2.1: Comparison of mapping transducers 42
Table 5.1: Sensor parameters 97
Table 5.2: Measured Distance and calculated distance 103
Table 6.1: Summary of path-planning algorithms (1) 129
Table 6.2: Summary of path-planning algorithms (2) 130
Table 7.1: Path length of RRT and RRT-A* 140
Table 7.2: Simulation run time of RRT and RRT-A* 141
Table 7.3: Path length of RRT, A*, APF/APF-SA, and RRT-A* 141
Table 7.4: Simulation run time of RRT, A*, APF/APF-SA, and RRT-A* 141
Table 7.5: Path length and simulation run time of RRT-A* 145

Abstract

Deploying a low-cost, micro autonomous underwater vehicle (µAUV) remains a challenge in modern underwater robotics. In the nuclear industry, there is a desire for µAUVs to replace people in the process of gathering information from cluttered layouts and take measurements of pH, radioactivity level, temperature and other relative parameters.

The key challenges in successfully deploying µAUVs in spent fuel storage ponds are underwater navigation and collision-free path-planning. This thesis developed a solution for these challenges.

In this thesis, a navigation approach, that combines storage pond map acquisition and path-planning, is proposed. The proposed approach integrates environment surveys, environment reconstruction, and path-planning as a complete task that involves learning and planning.

An echo-sounding based aerial mapping is developed to gather the topological height map of the storage pond; an efficient map construction method is developed to process the obtained raw depth measurements data to establish a 3D grid map; the A* algorithm is evaluated to be used in the 3D grid map of the cluttered pond environment

Keywords: AUV Navigation; Path-planning; A* algorithm; Occupancy grid map; Height interpolation; Point cloud; Aerial mapping; Echo-sounding; Multi-resolution path-planning; Hierarchical mapping;

Declaration

No portion of the work referred to in the thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institutes of learning.

Copyright Statement

i. The author of this thesis (including any appendices and/or schedules to this thesis) owns certain copyright or related rights in it (the "Copyright") and s/he has given The University of Manchester certain rights to use such Copyright, including for administrative purposes

ii. Copies of this thesis, either in full or in extracts and whether in hard or electronic copy, may be made **only** in accordance with the Copyright, Designs and Patents Act 1988 (as amended) and regulations issued under it or, where appropriate, in accordance with licensing agreements which the University has from time to time. This page must form part of any such copies made.

iii. The ownership of certain Copyright, patents, designs, trademarks and other intellectual property (the "Intellectual Property") and any reproductions of copyright works in the thesis, for example graphs and tables ("Reproductions"), which may be described in this thesis, may not be owned by the author and may be owned by third parties. Such Intellectual Property and Reproductions cannot and must not be made available for use without the prior written permission of the owner(s) of the relevant Intellectual Property and/or Reproductions.

iv. Further information on the conditions under which disclosure, publication and commercialisation of this thesis, the Copyright and any Intellectual Property and/or Reproductions described in it may take place is available in the University IP Policy (see http://documents.manchester.ac.uk/DocuInfo.aspx?DocID=24420), in any relevant Thesis restriction declarations deposited in the University Library, The University Library's regulations (see http://www.library.manchester.ac.uk/about/regulations/) and in the University's policy on Presentation of Theses.

List of Publications

- Y. Peng, P. N. Green, “Acoustic side scan on an enclosed underwater environment”, *UK Robotics and Automation System (UKRAS) conference*, Loughborough, 2019.
- Y. Peng, P. N. Green, “Mapping of an enclosed underwater environment by Acoustic side-scan”, *IEEE International Conference on Mechatronics and Automation conference 2019*.
- Y. Peng, P. N. Green, “3D Digital Grid Map Initialisation and Path-planning for Autonomous Robot Navigation”, *the International Conference on Robotics and Intelligent Systems*, Warsaw 2019. Proceeding - 2019 2nd *International Conference on Service Robotics Technologies*.
- Y. Peng, P. N. Green, “Environment mapping, map constructing, and path-planning for underwater navigation of an AUV in a cluttered closed pond”, *IAES International Journal of Robotics and Automation*, Vol 8, No 4: December 2019.

Acronyms

2D	Two dimensional
3D	Three dimensional
µAUV	Micro Autonomous Underwater Vehicle
UAV	Unmanned Aerial Vehicle
UGV	Unmanned Ground Vehicle
ROV	Remotely Operated Vehicle
SLAM	Simultaneous Localization and Mapping
AAS4IP	Actuated Acoustic Sensor Network for Industrial Process
AVEXIS	Aqua Vehicle Explorer for In-situ Sensor
DOF	Degree of Freedom
TOF	Time Of Flight
INS	Inertial Navigation System
IMU	Inertial Measurement Unit
MEMS	Micro Electronic Mechanical System
DVL	Doppler Velocity Log
SBL	Short Baseline
SSBL	Supper Short Baseline
LBL	Long Baseline
GPS	Global Position System
SONAR	SOund Navigation Ranging
LIDAR	LIght Detection And Ranging
RADAR	RAdio Detection And Ranging
VG	Visibility Graph

PRM	Probabilistic Roadmap
APF	Artificial Potential Field
CD	Cell Decomposition
GD	Grid Decomposition
VD	Voronoi Diagram
RRT	Rapidly-Exploring Random Tree
GA	Genetic Algorithm
FIRAS	Force Inducting an Artificial Repulsion from the Surface

Acknowledgement

I would like to thank Dr Peter N Green for his supervision and support throughout the course of this PhD. This thesis would not have been possible without the enlightening guidance and help from Dr Peter N Green. I particularly appreciate his patience and encouragement to help me with my study throughout the four years. I would like to thank the electronic and mechanical workshop staff, especially Danny Vale, who helps me to build the mapping testing rig and Malcro who lends me a lot of electronics devices for my testing. I would like to thank my colleagues, Hassan Hakim Khalili, for giving feedback and suggestions on my research.

I would like to thank the School of Electrical and Electronic Engineering for providing financial support for attending the UKRAS, ICRIS, IEEE ICMA conferences.

I would also like to thank my family and friends for supporting me when I am stressed and under pressure.

Chapter 1

Introduction

The past twenty years have seen increasingly rapid advances in robotics. The most significant recent developments in this field have been those of autonomous vehicles. Long after manipulator's arms became widely used in industry to optimise production line and replace human workers doing highly repetitive tasks during long working hours, research on autonomous vehicles has become the new trend. Recently, investigations of the autonomous vehicle have been made significant progress. However, there is still a long way to go to achieve complete autonomy of robots. Currently, many partially autonomous or remotely control robots are available. Based on their working environment, they can be categorised into three types. Autonomous vehicles include Unmanned Aerial Vehicles (UAVs), Unmanned Ground Vehicles (UGVs) and Autonomous Underwater Vehicles (AUVs). This research is concerned with the application of AUVs.

UAVs work in the air such as drones, which are used to film, environmental surveillance, geography study, and even parcel delivery [135]. For example, On 15 April 2019, Pairs firefights used the DJI drone to track the progress of the Notre Dame fire and find the best location for the fire hose [156]. UGVs work on the ground such as automated guided vehicles (AGV), which is used in smart factories, smart warehouses. For example, British supermarket Ocado is using robots to make online grocery shopping faster [136]. Although, progress has been made in autonomous robots recently, achieving truly autonomous without any constraints for sophisticated applications in real life still is a big challenge. Aerial vehicles such as drones usually are remotely controlled by a human operator, and ground vehicles such as vacuum cleaners are operated in constrained environments. AUV used in underwater research and industry is an even more sophisticated application. This research is interested in AUV underwater applications.

A diversity of AUVs has already been developed for different purposes and deployed in different underwater situations, which including air crash investigations, marine research, and

industrial processes. Those industrial processes include but are not limited to undersea oil pipeline leakage detection [1], spent nuclear fuel storage pond monitoring [2], and chemical reagent synthesis in large-scale vessels [3]. In the ocean researches, AUVs are deployed to undertake undersea oil exploitation, marine study and information collection, and ocean rescue [4]. As for air crash investigations, AUVs have been used to search wreckages of missing aeroplanes, such as Air France Flight 447 accident [133] and Malaysia Airlines Flight 370 accident [134].

AUVs could be used to replace human to do dangerous jobs in hazardous environments. One example that has received some attention in recent years is the monitoring of nuclear storage ponds to monitor for leaking containers or to find the disposition of material in older ponds. This research is concerned with applying AUVs in a hazardous underwater environment – specifically nuclear storage ponds. The spent nuclear fuel storage pond contains highly radioactive metals such as Uranium and Thorium [41], which are hazardous to human health. Places such as these are nearly impossible for humans to work in even wearing a protective suit. In such circumstances, small AUVs could potentially be used to replace humans to monitor the radiation level and explore the unknown underwater environment. However, so far, very little research has been carried out on small, low-cost sensor-limited[1] AUVs in cluttered storage ponds. One challenging aspect is radiation hardening on electronic devices [181]. The radioactive materials in the storage pond produce gamma radiation and neutron radiation which will affect AUVs' sensors and circuits and cause damage and malfunction of components. It could lead the AUV cannot work properly. The solution to reducing this risk is using lead or some other very dense materials to construct the AUV's hull so that can help protect its internal electronics from radiation. Another challenging aspect is obstacle detection and avoidance. Since the underwater environment is unknown and cluttered, gathering layout of the obstacles in the pond, establishing a 3D grid map, and planning collision-free paths on the established map are the main objectives of this research.

The utilisation of robots to undertake explorations and measurements in extreme environments has been a topic of great interest recently. The University of Manchester took a step ahead in the research of robotics monitoring of nuclear waste storage pools [2]. This

[1] Sensor-limited means sensors characterised by their low accuracy, a cheap sensor with limited accuracy is selected as part of a cost/performance trade-off

thesis aims to develop a navigation strategy that can be used in closed underwater environments.

1.1Background and Motivation

With the development of nuclear technology, nuclear power has become one of the primary power sources in the world. More and more countries are building nuclear power plants. According to the latest International Atomic Energy Agency report, 442 nuclear reactors operate in 31 countries, and 66 are under construction [6]. The increasing number of nuclear power plants leads to an increase in the quantity of nuclear waste produced. Worldwide, there are around 240,000 tonnes of used fuel, about 90% of which locate in storage ponds, and this is increasing at a rate of about 7,000 tonnes per year [6]. Hence, spent nuclear fuel monitoring and disposal has emerged as a massive problem across the world. This research is motivated by the nuclear waste monitoring and disposal problem.

As discussed at the beginning of this chapter, it is too dangerous for humans to work in nuclear storage pools, so robots have sometimes been used. It requires robots to navigate through the underwater environment to monitor the nuclear storage pond in case of leakage accidents or other unexpected accidents happen. However, the development of autonomous technologies is still in an early stage, and many robots that are used in the industrial application need to be operated by humans. Such human-controlled robots are known as ROVs (Remotely Operated Vehicles).

ROVs are widely used in deepwater oil and gas exploration, geotechnical investigations, and mineral exploration [5]. However, the main drawback of ROVs is that tethered with a cable (umbilical) to transmit commands and data between the vehicle and the control centre [5]. The operator determines where to go, so the ROVs are not autonomous. ROVs require operator control during the whole task, costing time and money. Moreover, the cable will limit the working range and sometimes wrap around obstacles causing accidents. Hence, autonomous vehicles with no umbilical AUVs would offer a significant improvement, especially for a cluttered enclosed environment like a nuclear storage pond [7]. Removing human control requires the robot to have the capability of making its own decisions based on the current state and known circumstances of its environment

1.1.1Nuclear Storage Ponds

Nuclear storage ponds are large open-air vessels that filled with water and used to keep spent fuel assemblies, other discarded machinery, and radioactive material until they are sufficiently benign to be subjected to processing for permanently safe storage [180]. In terms of size, Olympic swimming pools (assume 50 m × 25 m) provide a convenient benchmark, although storage ponds are typically much deeper (at least 10 m) [176]. Waste objects are typically stored in canisters and placed on the bottom of the pond with a significant depth of clear water above for shielding purposes [176, 180]. Those materials remain within a pond over fifty years.

Based on the era of construction, existing storage ponds can be classified as legacy ponds or modern ponds. The AUV using cases for each category of ponds are likely to be different. Modern ponds are equipped with regular storage racks designed to hold canisters that contain radioactive material. Hence, the distribution of clutter (the solid objects within the pond) is fairly regular or at least well-understood [176]. See Pile Fuel Storage Pond shown in Figure 1.1. The main uses of the AUV are to patrol around the pond and monitor pond parameters such as radioactivity levels, pH, and temperature.

Old ponds are referred to as legacy ponds (such as First Generation Magnox Storage Pond) and pose different challenges. Originating from times when the hazards of storing nuclear material were not as clearly understood as they are now, and handling technology was less well developed, legacy ponds contain material that was originally stored in a haphazard manner in skips (dumpsters). Some material in legacy ponds has been immersed for many years and has partially disintegrated [177]. As a result of this, the layout of used fuel rod canisters is not regular and not accurately known some are even on their sides. Decomposing fuel cladding and fuel fragments from spent fuel canisters (e.g. heavy metals uranium series) stacked at the bottom of the pool become bottom sludge [177]. Figure 1.2 illustrates an example of a legacy nuclear storage pond. The main objectives of the AUV are to map the pond, identify the clutters and levels of radiation, which would contribute to removal, processing, and disposal of nuclear wastes in the pond.

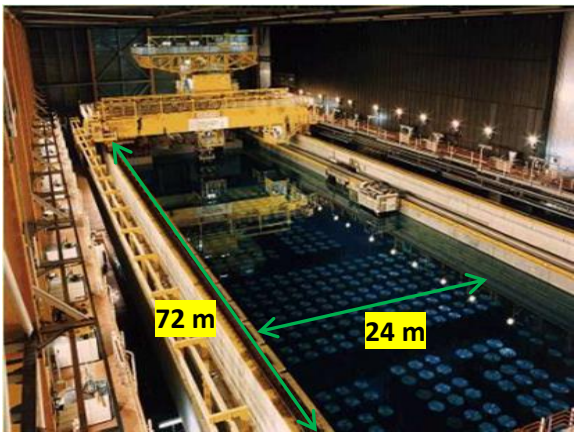


Figure 1.1: Modern nuclear storage pond [6]

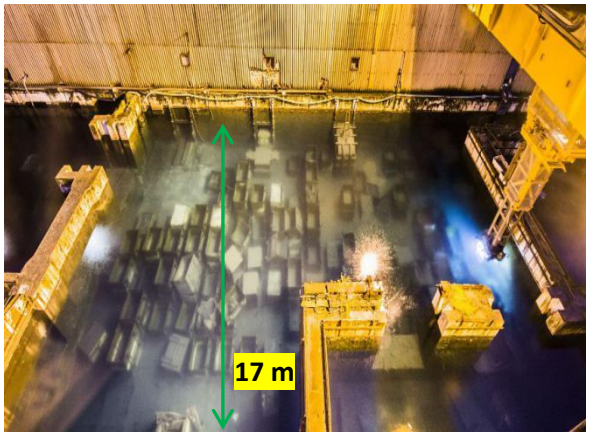


Figure 1.2: Legacy nuclear storage pond [9]

1.1.2 Previous and Related Work

1.1.2.1 AAS4IP Project

This research follows on from earlier work concerned with designing a low-cost sensor network platform AUV for industrial processes. This work was part of the Actuated Acoustic Sensor Network for Industrial Process (AASN4IP) project [12]. The AASN4IP project was undertaken to support the monitoring of water-based industrial processes [12]. The purpose of AASN4IP was to design and develop a low-cost µAUV which can move freely in three dimensions, take measurements of pond parameters and communicate with other vehicles or base stations outside the pond. The relevant technologies involve communication, signal/data processing, localisation, sensing, and motion control [12]. It was intended that the cost of the AUV is within £300, as it is essentially disposable and will stay in the pond becomes one of the nuclear waste afterwards and joins the was removing programs since it will be radioactively contaminated after being deployed in a nuclear storage pond. The low-cost AUV means it will use a cheap sensor with limited accuracy as part of a cost/performance trade-off.

The outcome of the AASN4IP project was a basic functional prototype vehicle shown in Figure 1.3. It has a spherical waterproof hull with a diameter of 15 cm. The propulsion in 3D is provided by six externally mounted thrusters that are based on small-scale DC motors and propellers [12]. Four thrusters are mounted horizontally and two vertically, which provide manoeuvrability with four degrees of freedom (DOF) - heave, surge, sway and yaw [12]. Control electronics were developed in three printed circuit boards, one to provide control, one to support communication, and one to provide power [12].

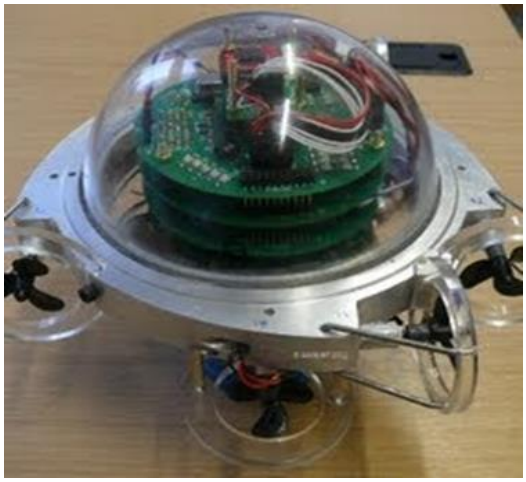


Figure 1.3: µAUV [12]

The control board interfaces with a series of sensors, including a pressure sensor, a digital compass, and a rate gyroscope [12]. These sensors support the motion control of the vehicle.

The communication system is based on acoustic transducers because acoustic signals experience far less attenuation than the radio signal in water [16]. The communication system enables signal transmission between vehicles and a base station outside the pond. A working communication physical layer was built in the LabVIEW environment [12]. However, the development of a system that can be integrated with the vehicle is currently ongoing.

Some work on localisation was carried out during the AASN4IP project since it is crucial that pond measurements are accompanied by information about locations. However, localisation relies on the communication system, and work on the acoustic system to support localisation is still ongoing.

1.1.2.2 AVEXIS Project

The Aqua Vehicle Explorer for In-situ Sensing (AVEXIS) project is the further development and improvement of the previous AASN4IP project. AVEXIS is an ongoing project and is intended to improve the characteristics of the first generation of untethered micro-autonomous underwater vehicles (µAUVs) developed in the AASN4IP project [14].

The second-generation prototype is an ROV, as shown in Figure 1.4. It has a 14 cm diameter 3D printed hull and ten waterjet thrusters providing three degrees of freedom (surge, heave, and yaw) Its horizontal velocity is 0.055 ms^{-1} [14]. The unique characteristic of this prototype is that it can be deployed vertically by detaching an electromagnet to change its

mass distribution. Unlike the first generation, this vehicle is controlled by an operator with a joystick. It has deployed this vehicle in Japan to take radiation measurements [14].

Figure 1.4 The ROV designed in the AVEXIS project [15]

1.2 The Main Challenges

This research is based on the AASN4IP work in that it is concerned with navigation and path-planning for μAUVs (its size within 30 cm) investigating storage ponds. In terms of the system, it requires an approach that enables the robot to work in a pond autonomously. μAUVs used in modern ponds are aimed to patrol about to monitor the radiation level in case of accidental leakage. It is helpful to check for leakage from canisters so that remaining work can be carried out in a timely manner. μAUVs used in the legacy ponds are aimed to determine the dispersal of objects within a pond and take measurements about the distributions of radioactivity materials, temperature, and throughout other parameters. Robots used for the above tasks are expected to gather the map of the environment and able to explore autonomously. Since the designed μAUV must satisfy large-scale commercial manufacture, and it will be disposed of after deployed in the nuclear storage pond as it is a radioactively contaminated item, so the cost should be low, within £300 [12].

The pond exploration relies on the navigation strategy of the μAUV, so a reliable approach that allows a low-cost μAUV to map the underwater environment and to achieve autonomous, collision-free navigation in an underwater environment is required. A map construction algorithm that converts the obtained data to useful information to establish a map is also a necessity. Further, a path-planning algorithm that plans a collision-free path from the start to the desired goal is needed. To clarify the above tasks, the challenging aspects can be defined as environment surveys, map construction, and path-planning.

An Environment Survey aims to gather the geometry information of an environment through a set of measurements. In this work, an environmental survey was performed via an **'aerial survey'**. The aerial survey is an in-air survey technique, which is a method of collecting ground depth information from an overhead position. This term is also used in the water known as the "depth-sounding".

Map construction aims to convert the obtained data into a map that can be used for path-planning. This requires the research about constructing the geometry of the object based on measurement points and the shape of the objects.

Path-planning aims to plan a collision-free path so that an AUV can traverse from the start to the goal without collisions. This requires the research on the path-planning algorithms, and how to implement the path-planning algorithm in the constructed map.

The above three challenges closely relate to each other. The environment survey gathers the pond data, map constructing uses the obtained data to establish a grid map, and path-planning generates feasible paths on the grid map.

The goal of this work is to establish an underwater navigation approach that considers all three challenges mentioned above and which endows an AUV with the capability of navigating in a cluttered enclosed spent fuel nuclear storage pond without collisions.

1.3 Aims and Objectives

The objectives of the research that is described in this thesis are:

- To investigate a mapping approach that allows AUVs to learn the environment and obstacle locations by range sensors
- To construct a map that can be used in robot path-planning based on the data acquired by the active range sensor.
- To investigate path planning approaches for a low-cost small-sized AUV in an enclosed and cluttered/well-organised nuclear storage pond, and to develop a prototype system.

1.4 Contributions

This research contributes to robotic applications in both nuclear storage decommissioning and general liquid-based industrial processes in

- Proposing an echo-sounding based aerial survey method to gather the topological height map of the pond environment and developing a ray-tracing based aerial survey simulator is to study the aerial survey.
- Evaluating which path-planning algorithms are feasible for a spent fuel pond problem.
- Developing an efficient map construction method to process the obtained depth measurements data to establish a 3D grid map.
- Developing a system that integrating the constructed map with an appropriate path-planning algorithm for underwater navigation in the pond environment.

1.5 Thesis Outline

The main body of the thesis consists of nine chapters, which are organised as follows:

Chapter 2 presents a detailed literature review of underwater navigation strategies and underwater mapping techniques, as well as their applications in industrial processes and scientific research. Also compares the methods and justifies the selection of the preferred method for storage pond mapping.

Chapter 3 presents the proposed survey method and discusses the underlying assumptions and issues concerned with its simulation. As ray tracing was the technique chosen for the simulator, it is briefly introduced in this chapter.

Chapter 4 discusses the development of the simulator in MATLAB, and Chapter 5 presents simulation results for a number of pond clutter distributions. Chapter 5 also investigates how sampling resolution impacts survey accuracy and how to mitigate the information loss between two close tilted obstacles. This chapter also presents some practical experiments that were undertaken for validation purposes.

Chapter 6 presents a detailed review of notable path-planning algorithms and discusses their applicability in the 3D underwater storage pond case. This chapter gives a comprehensive overview of the advantages and disadvantages of each path-planning algorithm.

Chapter 7 presents a set of simulations for some path-planning algorithm (the chosen algorithm discussed in the end Chapter 6). Using different pond-like environments to evaluate which path-planning algorithm has the best performance for the cluttered pond-like environment. This chapter compares the simulation outcomes of each algorithm and discusses which has the best performance and explains which are not suitable.

Chapter 8 introduces a novel algorithm that derives a 3D point-cloud model of the pond environment from a 2D aerial survey depth measurement. This method integrates the work reported in the earlier parts of the thesis: the environmental data acquisition, map reconstruction, and path-planning to achieve collision-free underwater navigation. This chapter explains how the 3D point-cloud model is further processed to establish a 3D occupancy grid map for selected path-planning algorithms. This chapter also presents an extensive validation simulation of the proposed framework.

Chapter 9 presents a framework that allows AUVs to conduct Multi-resolution hierarchical aerial mapping and path-planning.

Chapter 10 summarises the work that has been completed on this thesis and provides conclusions based on the work. This chapter also proposes the direction of future research.

Chapter 2

Review of Navigation Methods

2.1 Introduction

There has been and continues to be much research into underwater mapping, navigation and localisation. This continuing interest is partly due to the growth in the application of submersibles and because off-the-shelf technologies like GPS are not applicable in underwater environments [18]. The research is about mapping storage ponds as a precursor to developing plans to explore the pond to sense desired parameters. This requires mapping, localisation, and navigation. Mapping is used to identify the feature of interest in an environment in order to gain knowledge about the environment and to provide data for navigation. Localisation is concerned with estimating a vehicle's position within an environment. Mapping, navigation, and localisation are strongly inter-related. Mapping provides details of the environment for navigation. Navigation finds collision-free paths for exploration and the localisation estimates where a vehicle is located within the space.

Strictly speaking, this chapter mainly concerned with localisation and mapping since if an environment is scanned and a map created for later use [29], then navigation often becomes a path-planning exercise, and path-planning is discussed in Chapter 6. However, some approaches, for example, Simultaneous Localisation and Mapping (SLAM), which maps an environment as a robot navigates through it. SLAM-based methods are briefly considered in this chapter. Many of the methods discussed in this chapter are proposed or used in sub-sea AUV. The application of these techniques to the pond mapping and exploration problem is then discussed. This work is primarily concerned with mapping and navigation. Localisation is the focus of other colleague's work. Most AUVs are used in sub-sea applications, where the most commonly used navigation techniques are Inertial Navigation and Acoustic Positioning.

2.2 Inertial Navigation

Inertial navigation is a classic AUV navigation technique. It uses motion sensors (accelerometers and gyroscopes) to estimate the position, velocity, and attitude of a vehicle based on its previous position, velocity, and attitude. The operation of an inertial navigation system (INS) depends on Newton's laws from classical mechanics.

Inertial navigation only depends on input from sensors directly contained within the AUV, and that does not require any external references [25, 138]. Inertial navigation is achieved through the use of Inertial Measurement Units (IMUs) which consist of at least three elements, including a microcontroller, an accelerometer, and a rate gyroscope [19]. An accelerometer is an electronic device for measuring the translational acceleration of a moving object in one or more directions. [20]. By measuring the acceleration due to gravity, the tilt angle of the device relative to the horizontal plane can also be estimated. A rate gyroscope is a Micro-Electro-Mechanical Systems (MEMS) device used for measuring orientation and angular velocity of a moving object [21]. Ultimately, an embedded microcontroller processes the data gathered from accelerometers and gyroscopes to estimate the state of the robot [19]. Other devices used to support Inertial Navigation include compasses provide a global direction references, pressure sensors to measures the depth, and Doppler velocity log (DVL) to calculate the AUV movement velocity refer to the seabed [25].

An inertial navigation system is **a dead reckoning** system. When applied to an AUV, it is necessary to know the initial state, including the AUV's starting position, velocity, orientation, and heading direction [19]. During AUV motion, sensors measure attitude, velocity, changes in altitude, and gravitational forces acting on an AUV. Hence, the INS can update the new position, velocity, and orientation by processing information received from the sensors. One problem with inertial navigation is that it drifts over time. In order to aid the calculation of past, present, and future prediction positions, the INS usually is often used along with a Kalman filter [20] or particle filter [21]. The accurate measurements of its IMUs largely determine the performance of an INS, so a high-quality device is typically required.

2.3 Acoustic Positioning Methods

Acoustic positioning refers to techniques that use acoustic signals to localise an AUV's position. The idea of acoustic navigation is to build a local positioning system in the

underwater environment [24]. Acoustic Positioning uses the time of flight (TOF) of acoustic signals transmitted between the transceivers located below a surface support ship and the AUV. To measure the distance between the AUV and the transceivers on the surface vehicle, trilateration [178] is then used to estimate the position of the AUV. Three related Acoustic Positioning methods have been used with AUVs [25]: short baseline (SBL), super short baseline (SSBL), and long-baseline (LBL).

2.3.1 Short Baseline

A completed SBL system consists of at least three transceivers mounted under the surface vessel. A minimum of three transceivers is required by the triangulation method. A known distance separates each transceiver. The line that connects a pair of transceivers is called a baseline. The TOF of signals from the AUV to the support vessel's transceivers can be estimated and used in triangulation software to determine the AUV's position [25, 26]. See Figure 2.1 for an SBL system.

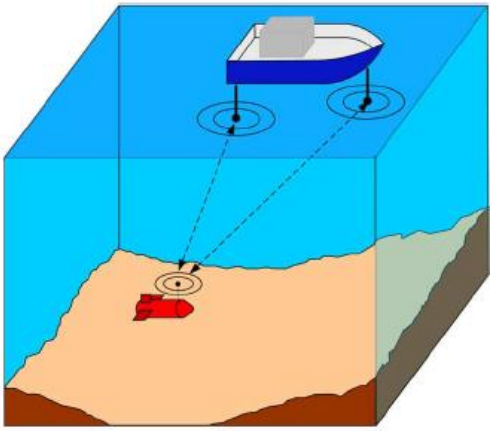


Figure 2.1: SBL [25]

The geometry model of an SBL system is explained by the diagram below:

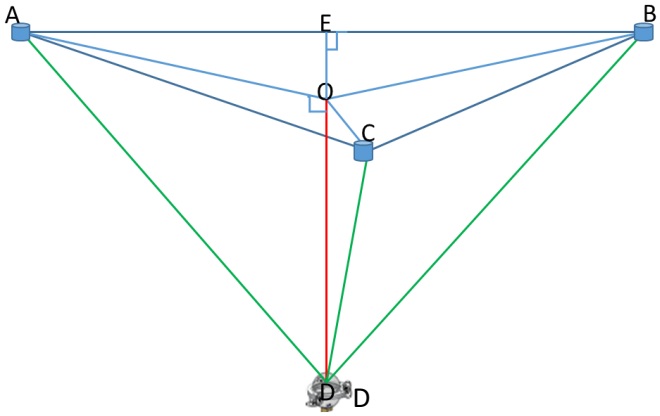


Figure 2.2: Schematic diagram of SBL

A, B and C are three transceivers mounted under the support ship. They are in the same plane (parallel to the sea surface), their positions with respect to ship-based coordinates are known, and A, B in the same horizontal straight line parallel to X-Axis. D represents the AUV. AD, BD and CD are the distances between the AUV and transceivers A, B, and C, respectively. These distances are measured by the TOF of signals transmitted by the AUV's transponders. O is the projection of D in the plane ABC, so DO is perpendicular to AO, CO, and BO. OD is the depth of the AUV, which is measured by a water pressure sensor onboard the AUV. It uses the equations below to calculate the position of O using A as the reference. OE is perpendicular to AB, so the coordinate of O is (AE, OE). Using Pythagoras Theorem [27]:

$$AO = \sqrt{DA^2 - DO^2}$$

$$CO = \sqrt{DC^2 - DO^2}$$

$$BO = \sqrt{BD^2 - DO^2}$$

Apply the cosine rule to calculate ∠BAO:

$$\angle BAO = \cos^{-1}\left(\frac{AO^2 + AB^2 - BO^2}{2AO * AB}\right) \quad (1.1)$$

The Cartesian coordinates (AE, OE) of the AUV are given by:

$$AE = AO * \cos(\angle BAO)$$

$$OE = AO * sin(\angle BAO)$$

2.3.2 SSBL and LBL

SSBL is similar to SBL except that it uses range and phase difference rather than trilateration to calculate the AUV's position. The baseline of SSBL is much shorter than that of SBL, so the advantage of SSBL is that it can be used on a smaller surface vehicle. An LBL system does not rely on the support of the surface vehicle but uses at least three beacon transponders that installed on the seabed at known positions. The limitation of LBL is its lack of mobility, as it is necessary to install beacons on the seafloor, and an AUV can only navigate within the range of the beacons. The LBL estimates the position of the AUV in a similar manner to the SBL, but with an inverse geometric triangulation model.

2.4 AUV Underwater Mapping Methods

An important mission for subsea AUVs is to map the seabed to construct a topological map for navigation or to collect geophysical information of the seabed. Seabed mapping is the measurement of the depth of a given body of water called Bathymetric measurements. They are usually conducted by sonar devices with various methods includes side-scan sonar [145], forward-looking sonar (FLS) [147], multi-beam sonar [146], or echo-sounding [161].

2.4.1 Side-Scan Sonar

The side-scan is shown in Figure 2.7. An acoustic transducer mounted under the bottom of the AUV generates a fan-shaped sound pulse perpendicular to the direction of the track and sweeps the seafloor from underneath to either side [145]. In the side-scan, a topological seabed acoustic intensity image is constructed based on the energy of the returned sound on a time basis.

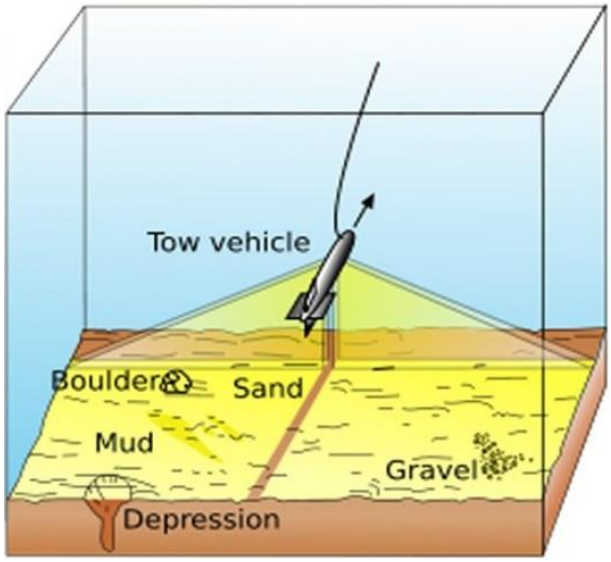


Figure 2.7: Side-scan sonar [145]

2.4.2 Multi-Beam Sonar

Multi-beam sonar is another mapping method. See Figure 2.8. It uses several single beam acoustic transducers, each simultaneously producing a single-beam sound pulse perpendicular to the direction of the track of the AUV to cover a large area beneath the vehicle [146]. The covered area is called a swath, and its width is called the swath area. This technique measures the depth of the water by the time of flight (TOF) of the received reflected signal. The obtained map is a bathymetric map.

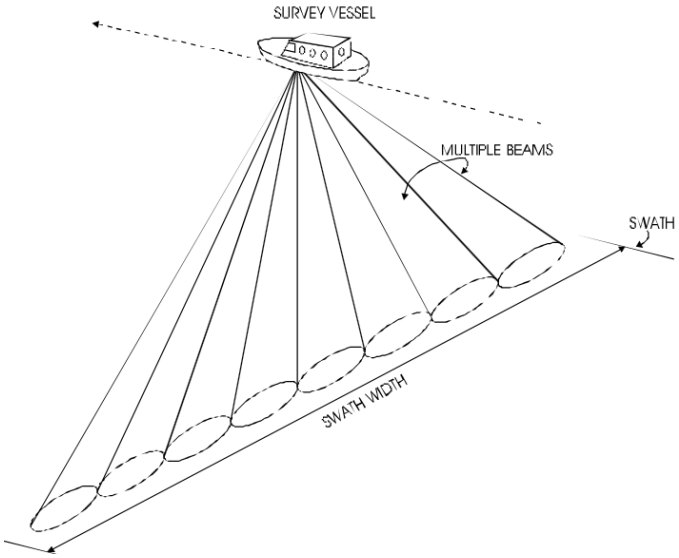


Figure 2.8: Multi-beam sonar [146]

2.4.3 Forward-Looking Sonar (FLS)

FLS systems do not map the seabed directly beneath the AUV but map the seabed ahead of the AUV, see Figure 2.9. The acoustic transducer mounted under the AUV generates a fan-shaped beam that has a narrow spreading angle along the azimuth direction but a wide-spreading angle along the elevation angle direction [147]. A seabed map is built based on the intensity of the sound reflected by the seabed.

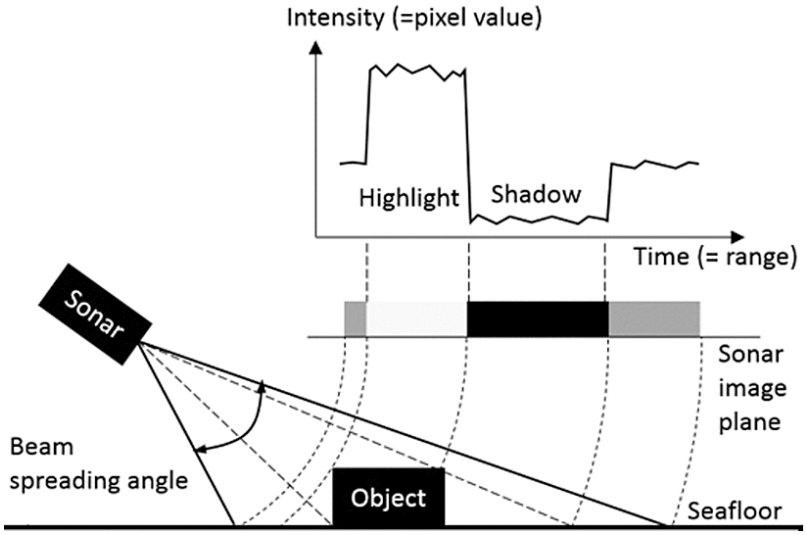


Figure 2.8: FLS [147]

2.4.4 Echo Sounding Sonar

There is a simplified version of the multi-beam sonar mapping technique, the echo sounding sonar. It is used to determine the depth of water by emitting sound wave. It records the

interval between the transmitted pulse and the returned pulse. After that, uses the time interval along with the speed of sound in water to determine the depth of water, the principle of how the echo sounding work is shown in Figure 2.9. High-frequency sound devices are used to measure depths data. Echo-sounding data are used to produce DEMs (digital elevation models) for floors of lakes, seas, and oceans [161].

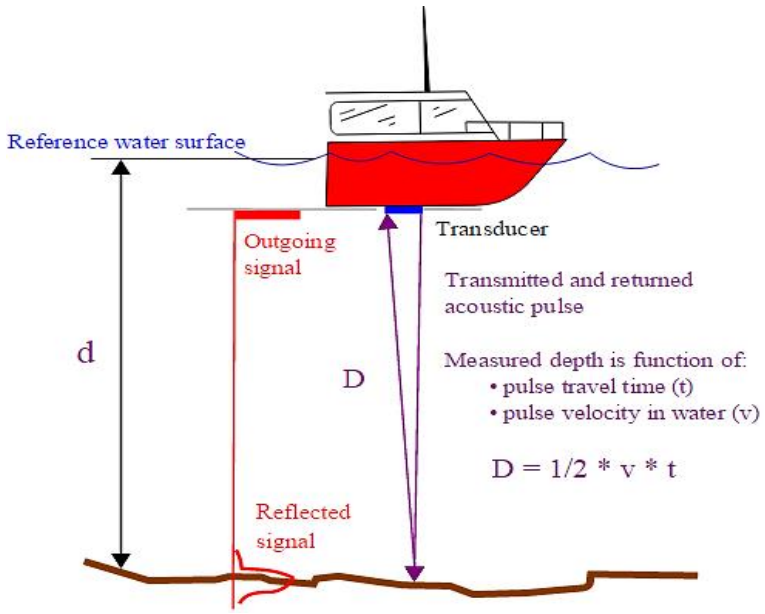


Figure 2.9: Echo sounding [179]

2.5 Geophysical Navigation

Unlike inertial navigation, geophysical navigation (also known as terrain navigation) utilises external environmental features for navigation [29]. This method either requires an existing map or needs to build a map. If an existing map is available, the navigation problem is simplified, because feature locations are already known. If an existing map is not available, then a survey and an off-line map construction are required, or some form of underwater vision-based simultaneous localisation and mapping (SLAM) [11].

2.5.1 Existing Map

On the existing map, it becomes an offline path-planning problem, so planning algorithms can be used directly. The map provides the input to the path-planning algorithm. Based on this, the path-planning algorithm will estimate a collision-free path from the start to the desired goal. Many path-planning algorithms such as A*, randomly-explore random tree, and the probabilistic road map can be applied. See Chapter 6.

2.5.2 Simultaneous Localisation and Mapping (SLAM)

When the map is not available, an AUV either needs to perform mapping to gather the information about the environment's features (see section 2.4) and so build a map, or it can apply a SLAM-based approach. Unlike the existing map method, SLAM will build the map and use a map to deduce AUV's current position at the same time. This approach deals with uncertainty and dynamic environment and allows the robot to react to unforeseen areas of the environment during its movement to the target (in real-time). When an AUV is placed in an unknown environment, the first thing needs to do is to scan the surrounding environments (e.g. via a sonar or a camera [159]) and extract features from the scans. At the same time, it determines where it locates [25]. SLAM is a 'chicken and egg' problem because a map is needed for localisation, and a pose estimate is needed to build a map at the same time.

In feature-based SLAM, sensors detect the landmark information by different sonar sensor swaths include side-scan, multi-beam, forward-looking, mechanical scanning and imaging, and synthetic aperture sonar [25, 30]. The start position will be used as the reference to determine the pose of the AUV, so the localisation is estimated based on the distance to the original position and the landmark information. In vision-based SLAM, the pose of AUV is estimated based on view-based odometry by comparing the current view of the structure from motion (SFM) and its previous view of SFM [25]. SFM is an imaging technique for estimating three-dimensional structure from two-dimensional images. Both feature-based SLAM and vision-based SLAM will work along with Kalman filter/extended Kalman filter for using a recursive predict-update cycle to estimate an AUV's pose and its surrounding map.

2.6 AUV Applications

This section investigates the application of inertial navigation, baseline-based acoustic navigation, and geophysical navigation in subsea environments.

INS is widely used in many marine applications. Chen et al. [22] designed a micro inertial attitude and heading reference system for the "XUANWU" AUV to track the path to a docking station set on nodes of cabled seabed observatories. This system is made up of magnetometer compass, Micro Inertial Measurement Unit (MIMU), Doppler Velocity Log (DVL) and GPS. Their simulation shows high precision when AUV works on the surface of the water.

The Monterey Bay Aquarium Research Institute [23] operated an AUV equipped with an INS and a DVL and GPS to perform tests and to take scientific measurements of nitrate concentration, salinity, temperature, and track the relative movement of the ice drift in the area between Svalbard and Greenland. Their research was successfully completed, and the INS provided accurate and reliable navigation for their trials. The INS provides very accurate short-term changes in velocity and attitude, and it shows the position and velocity error is less than one nautical mile per hour (NMH) and 1 m/s long term [23].

Acoustic navigation methods can work along with INS and Kalman filters to improve the accuracy of AUV navigation. [139] introduced SSBL to enhance dead-reckoning of the CTINEU AUV when it was used to map and inspect a hydroelectric dam accurately. The SSBL system set up with transponders mounted on the vehicle, and surface buoy with transceivers, see Figure 2.10. The SSBL navigation system provides a precision of a few centimetres in positioning, which enabled the AUV autonomously follow a trajectory in front of the wall of the dam to gain a set of images, navigation data, and other information.

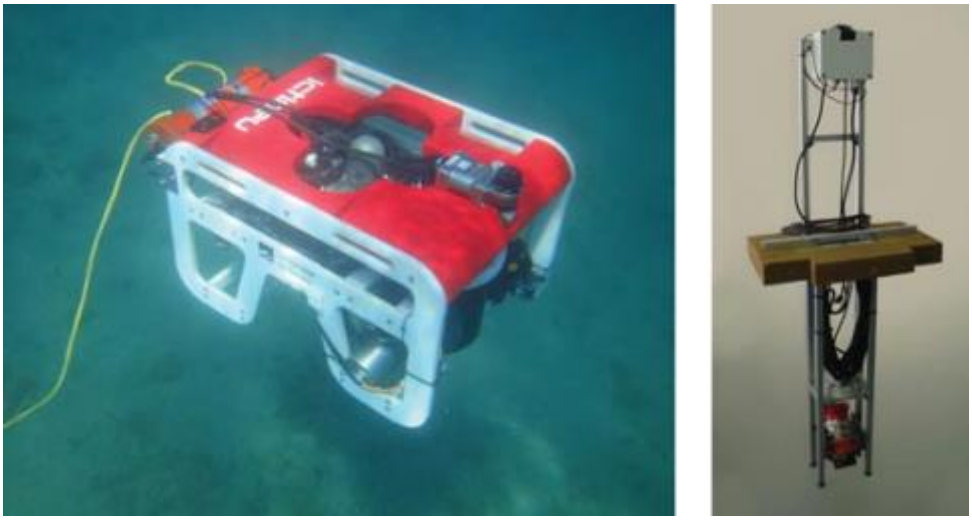


Figure 2.10: AUV CTINEU in water and surface buoy for localising the vehicle [142]

[142] introduced an implementation of LBL navigation on an AUV used for a deep-sea near-bottom survey. Before the AUV survey, an array of two or more acoustic transponders installed near the seafloor and separated by large distances (long baselines). GPS gives the georeferencing location of each transponder. During the survey, the AUV repeatedly interrogates the transponders and measures the time between the transmitted signal and the received signal. Hence, the position of the AUV can be estimated.

An example of Geophysical Navigation is reported in [143]. It introduces an implementation of SLAM used by AUV during a ship hull inspection mission (the ship is 36.2 meters in length and 13.4 meters in width). It describes a SLAM implementation using forward-looking sonar (FLS) data. The Exactly Sparse Extended Information Filter (ESEIF) algorithm is used to extract features from FLS images [143]. The results demonstrate that the AUV can effectively map a ship hull by SLAM. Paper [144] presents an application of underwater vision-based SLAM navigation by using a stereo camera.

2.7 Summary and Discussion

This chapter has briefly reviewed inertial navigation, acoustic positioning, seabed mapping methods, and geographical navigation, all of which are used in AUV operation in a sea or river environment. This section will discuss the applicability of these methods in storage pool mapping and exploration. Methods which are only concerned with localisation are not discussed in this section, as localisation is outside the scope of this work.

Majority of AUVs are designed for undersea applications, so they must be robust and flexible, which makes them expensive; an average cost of $70,000 per vehicle is quoted in [148]. They are also relatively large, most being torpedo-shaped and several metres in length. However, pond environments differ from most sub-sea environments. This factor alone justifies the AASN4IP [12] and AVEXIS [13] need to develop a new small-scale AUVs since sea-going AUVs are typically too large be used to map and explore storage ponds.

Although seagoing AUVs are not suitable for the storage pond application that is the focus of this thesis, it is possible that the techniques used in large AUVs for localisation, mapping, and navigation can be applied to the pond problem. Consider the characteristics of the pond environment. As discussed in Section 1.1, storage ponds are tens of metres in length and width, and around ten metres deep. They have well-defined hard boundaries and the distribution of solid objects on the bottom. The arrangement and orientation of these objects might be partly disordered or wholly random. However, it assumes that the position of objects is unknown and that most objects are cuboid or prismatic (discussed in Section 3.4). Ponds are typically a static environment, where the layout of solid objects on the bottom does not change over relatively long periods. Finally, storage ponds are radioactive. Any object, such as AUVs and beacons that placed in a pond will become contaminated, increasing the amount of dangerous material. For these reasons above, it is important to minimise the

amount of hardware that must be placed in the pond to support mapping and exploration. This implies that the AUVs used in ponds should be treated as disposable and therefore should be of low cost.

As stated in Section1.1.1, it assumed that the position of the clutter in a pond is either completely unknown or inaccurately known. Hence, the initial task is to construct a map. The advised solutions discussed earlier are off-shore mapping and SLAM. The mapping approaches discussed in Section 2.5 appear to be adaptable to the pond environment. However, the multi-beam sonar devices and FLS are typical too large to fit into a small AUV build for AAS4IP or AVEXIS, and the price for such a system is too high. A table of comparison among

Table 2.1: Comparison of mapping transducers

Sensor type	Size	Price
Multi-beam sonar SONIC-2020 [150]	155 mm*145 mm*150 mm	£500
Echopilot FLS Standard Transducer	60 mm *60 mm* 146 mm	£320
Garmin GT20-TM Transom Mount Transducer [153]	150 mm * 52 mm * 22 mm	£89.99

The Multi-beam sonar SONIC-2020 is too big (the diameter of the designed µAUV in AAS4IP is only 15 cm [12]) and too expensive for a µAUV. Echopilot FLS Standard Transducer is also too big and too expensive for a µAUV. Side-scan sonar has the lowest price among them, but still very high compared to the budget. What is more, the map obtained by side-scan sonar is an acoustic intensity image, can not be directly used for path-planning. The basic principle of multi-beam sonar could be implemented via low-cost ultrasound ranging devices, which is known as echo-sounding sonar explained in Section 2.4.4, and some work (communication and localisation) on acoustic devices already accomplished in AASN4IP in a pond although not for distance measurement. Therefore, echo-sounding is an appropriate approach to map the pond.

As for the SLAM, since the storage pond is a static environment, and its environment will not change during the navigation of an AUV, so there is unnecessary to consider real-time path-

planning via SLAM in the beginning. Moreover, sensors that used for underwater SLAM are FLS or vision sensor [159], as discussed earlier, FLS is not feasible in this application, and the pond is a turbid environment, so it is hard for the vision sensor to identify environmental features. Hence, SLAM is not considered.

Therefore, an echo-sounding technique using an inexpensive ultrasonic sensor is considered (within £20). A two-stage approach of measurement with cheap ultrasonic sensor, off-line map construction and path-planning appears to offer a relatively simple, low-cost solution to the pond mapping and exploration. INS and baseline positioning techniques can be used to support the localisation of μAUV when navigating on a pond on the planned path, but it is not considered at the current stage. The next chapter will explain the concept of the echo sounding based aerial survey to map the pond.

Chapter 3

Aerial Mapping and Ray Tracing

3.1 Introduction

The previous chapter described different underwater scanning methods, and it is found that the echo-sounding technique provides a low-cost solution for mapping the nuclear pond. Sound devices mounted under the bottom of a µAUV project a conical-shaped beam onto the bottom of the pond to conduct the echo-sounding scanning. This vertical downward mapping is named as ‘aerial mapping’ or ‘aerial survey’. ‘Aerial mapping’ or ‘aerial survey’ originally came from an airborne land survey, which is a method of mapping the geographic features of large areas. It is widely used to build geometric maps of agricultural or forestry sites, urban areas, industrial plants. It requires a UAV (unmanned aerial vehicle) or a small aeroplane flying at a specified height above sea level, with an electronic system that projects a scanning pulse of high frequency vertically down to a ground area. The electronic system receives a reflection of the pulse from the ground object that it intersects. This enables the distance from the reflection point and coordinates of the reflection point to be determined [31].

The working principle of the land aerial survey is shown in Figure 3.1. This shows that the scan is based on ray-tracing, which treats the scanning beam as a finite number of rays, with each ray obeying the law of reflection. Once a ray strikes a ground object, the surface reflection will take place, and its energy will be dissipated significantly. The time of flight (TOF) of the echo signal (the time it takes for the pulse to return) will be used to calculate the distance between the scanner’s position and detected surface.

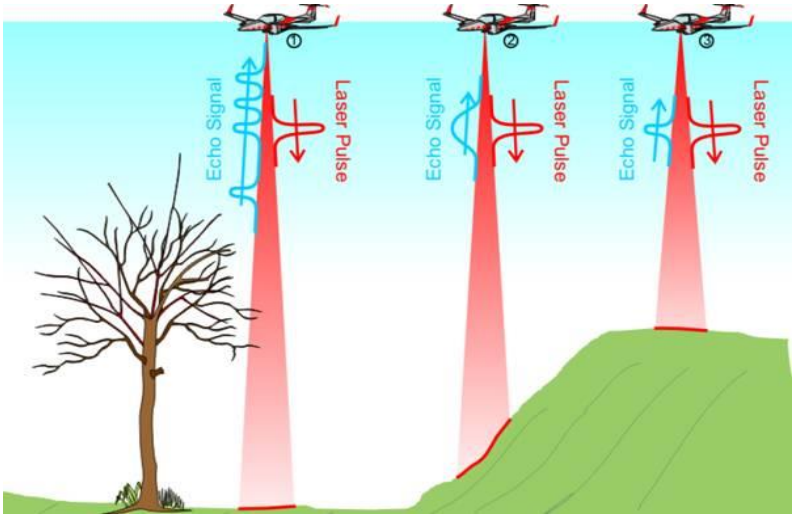


Figure 3.1: A land aerial survey [32]

Land aerial surveys are usually conducted with LiDAR (**Li**ght **D**etection **A**nd **R**anging) sensor which implement vertical optomechanical scanning of an area by laser emission with high-frequency pulses. The LiDAR technique provides very accurate results, so it is commonly used to make high-resolution maps [33].

It is clear that the underwater mapping technique presented in Section 2.4 is closely related to the aerial survey concept, and this will form the basis of the proposed pond-mapping scheme. Similar to land aerial surveys, an underwater aerial survey is simply a set of depth measurements taken on a regular measurement grid at a fixed height plane within the clear water region of the ponds, see Figure 3.2. SONAR is used for sensing (Section 2.4) rather than RADAR or LIDAR because:

- First of all, the light gets scattered by water, and its penetration is depended on the clarity of the water or the turbidity [34], so LIDAR is not feasible in a turbid underwater environment. As for RADAR, it also not suitable in water, because water is a denser medium than the air so the wave with a short wavelength is unable to pass through water molecules or easily to be absorbed by the water. Sounds/ultrasounds wave have a longer wavelength than radio waves, so they are able to pass through water molecules undisturbed and have better performance in the water [35].
- Secondly, the SONAR range finder system is cheaper than the LIDAR/RADAR range finder system in most cases [36]. Since the µAUVs are intended to be low-cost, a SONAR range finder is good for a low-cost µAUV.
- Thirdly, the size of the LIDAR instrument might not fit a µAUV, because of its size usually too big for a small (15cm diameter) AUV.

- Finally, SONAR transducer can provide a large beam angle than LiDAR, which is useful to map tilted objects and irregular orientated objects.

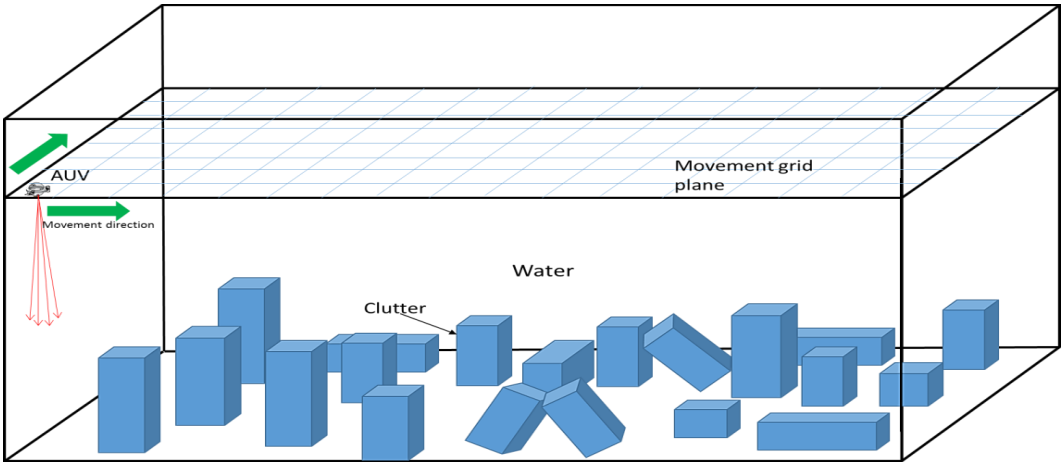


Figure 3.2: Nuclear pond aerial mapping

The proposed approach to mapping a pond is to perform echo-sounding using acoustic transducers mounted on the bottom of µAUVs to each sampling grid. The aerial mapping yields a depth map that can be turned into a point cloud based on assumptions of obstacles' shape is prismatic (this assumption is discussed in Section 3.4). An example of the aerial mapping route is shown in Figure 3.4. The movement plane is above the clutter and sampled in a uniformly spaced two-dimensional grid which is the mapping grid (see the blue grid on the movement plane Figure 3.2). The µAUV will move to each grid point and measure the corresponding depth in sequence on the mapping route given in Figure 3.3. A localisation system is required for the µAUV to follow the given route and measure the depth at the correct position, which allows the µAUV to know its data sampling position. The issue of sampling accuracy is bound up with issues of localisation accuracy. However, as mentioned in Section 2.1, localisation is the focus of other colleague's work, so will not be studied in this thesis.

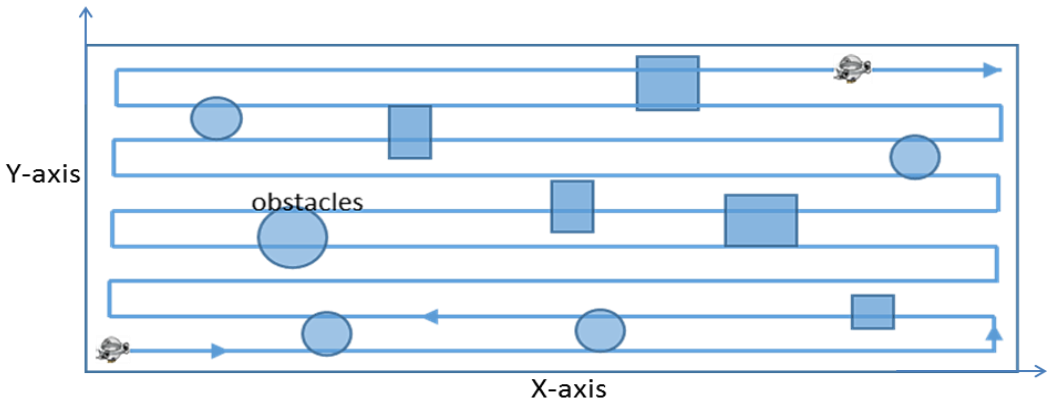


Figure 3.3: Mapping route in 2D

The result of an aerial survey is an array of tuples, whose components are measurement positions and its corresponding height. These elevation point data is called ‘depth measurements data’. This depth measurement data provides geometric information about the bottom surface.

3.2 Acoustic Range Finders Technique

Acoustic range-finding techniques include the measurement of Time-of-Flight (TOF), phase shift, or intensity of reflection. The Time-Of-Flight (TOF) technique refers to the time it takes for a signal (pulse) to travel from its transmitter to an observed object and then back to the receiver [39]. The range can be estimated by knowing the speed of the acoustic signal (assumes 1450m/s in water) and the measured TOF. This gives the round-trip distance, and the desired distance is half of this. The phase shift measurement technique continuously transmits a signal towards the target and determines the phase shift between the transmitted and reflected signals and uses it to estimate the distance [40]. The intensity of reflections technique is based on the energy dissipated during the propagation of an acoustic signal [40]. The initial energy is emitted towards an object and will be reflected and subsequently sensed by the receiver [40]. Acoustical ranging can be implemented by using the above techniques or a combination of these techniques. The acoustical ranging technique considered in this thesis is a combination of the TOF technique and the intensity of reflections techniques.

3.3 Ultrasound Transducer for Echo Sounding

The working principle of acoustic transducers to conduct echo sounding relies on modulating acoustic waves to sense a physical object. These devices transduce electrical signals to acoustic waves, receive reflected acoustic waves, and convert received acoustic waves back to electrical signals [46]. Take the HC-SR04 ultrasound range finder as an example. This device is a classic low-cost PWM based distance measuring device. It works at a frequency of 40 kHz. When a 10uS input trigger TTL electrical pulse is applied to its trigger pin, this module will generate eight 40 kHz sound wave pulses [160]. At the same time, the echo pin will pull up to a higher electrical potential, and the timer starts to count. When the pulse returns, the echo pin will pull down a low electrical potential, and the timer stops counting. As Figure 3.4 shown, the width of the pulse that outputs from the echo pin is the TOF of the sound signal that reflected off its nearest object.

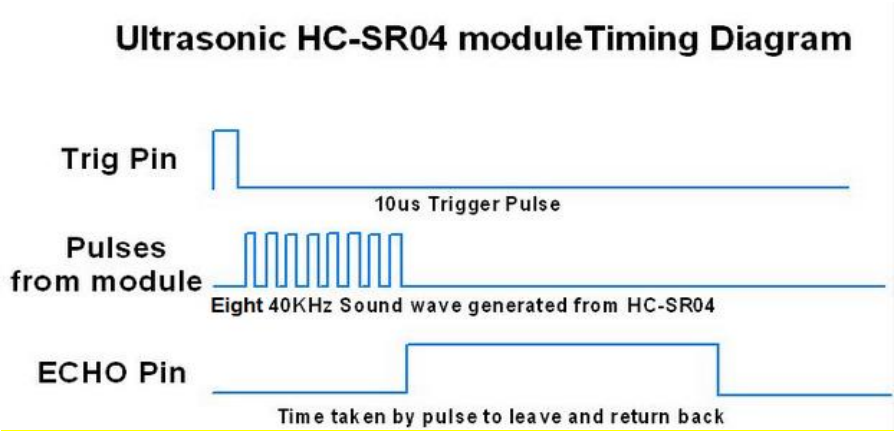


Figure 3.4: HC-SR04 Working Principle [160]

Equation 3.1 below can calculate the distance to the nearest object:

$$Distance = v * \frac{t}{2} \tag{3.1}$$

Where v is the speed of sound in water, which is known; t is the time of flight (TOF) of the received reflection signal. The mechanism of how the HC-SR04 measures distance is shown in Figure 3.5.

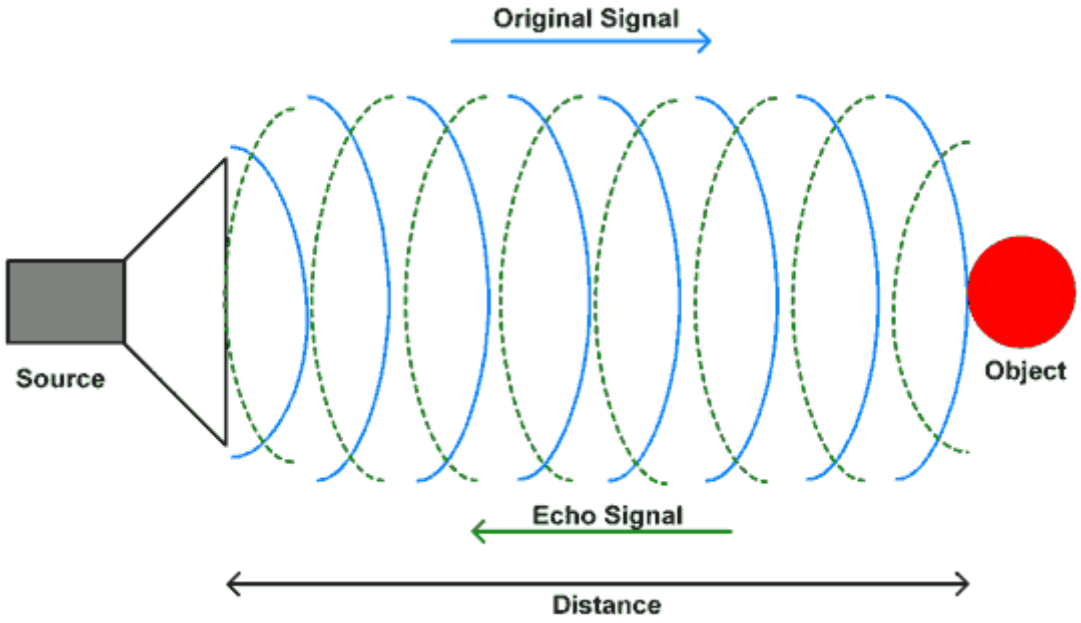


Figure 3.5: Distance measuring [160]

Some sensors have more sophisticated outputs than the HC-SR04 – in particular, some sensors that have an analogue envelope output of the received signal, such as the MB1340. The ranger finder's analogue envelope output allows a more detailed analysis of the structure of the reflections, which may be helpful in cluttered environments. Each pulse in the analogue envelope output represents a detected object. Typically, the highest peak in the

signal will represent the target object. To measure the distance in this way, it needs to find the highest impulse of the response signal, and then find the corresponding TOF to calculate the distance, see the outputs of the analogue envelope based ultrasound sensor in Figure 3.6.

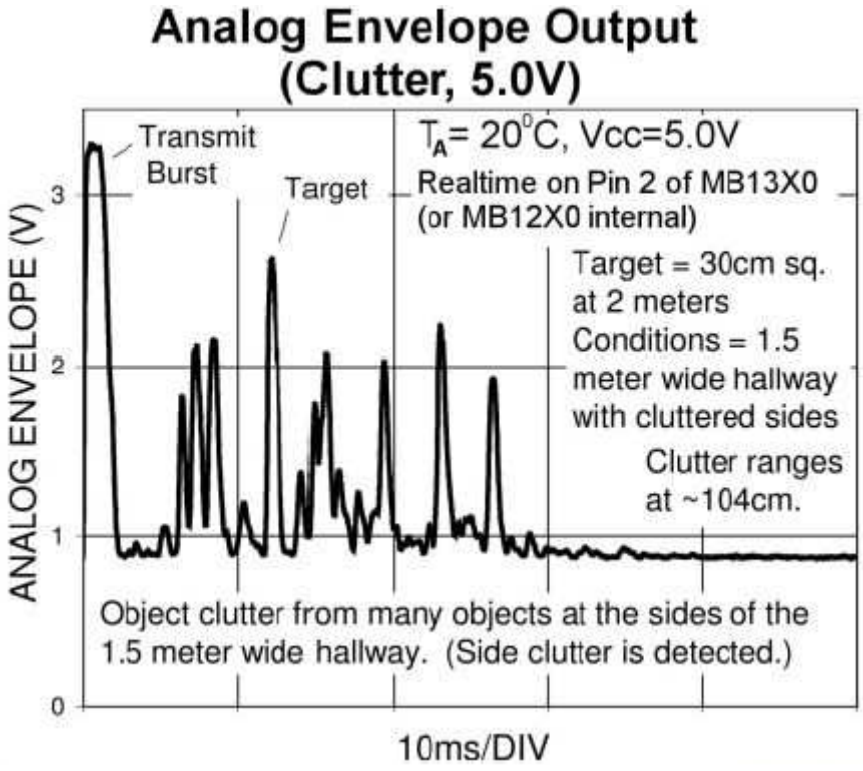


Figure 3.6: Outputs of MB1340 [164]

3.4 Important Physical Characteristics of Nuclear Storage Pond

Since the echo-sounding technique is used, the physical characteristics of the pond are a big concern. This section will explain aspects of storage ponds that will influence the propagation of sound. Figure 3.7 and 3.8 are images taken from modern storage pond and legacy storage pond, respectively. It assumes that the shape of canisters to be prismatic (objects have a polygonal base and a cross-section that does not vary with height, this includes both the cuboid and cylindrical containers that are typically used in most ponds), which is a reasonable assumption for those canisters because canisters shown in Figure 3.7 and 3.8 are cuboid and cylinder. In general, most canisters are cuboid or cylinder. This is a crucial assumption because objects construction in the latter map construction process is based on this assumption.

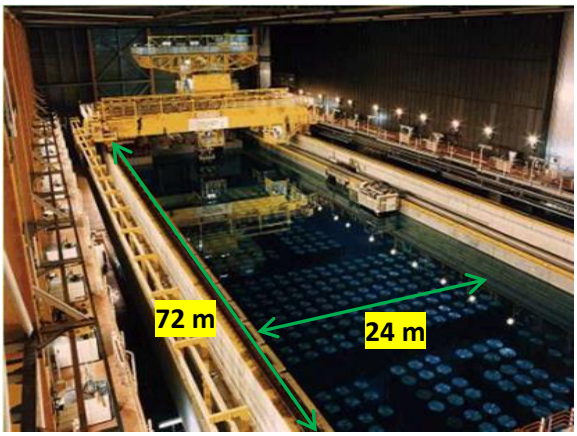


Figure 3.7: Modern nuclear storage pond [6]

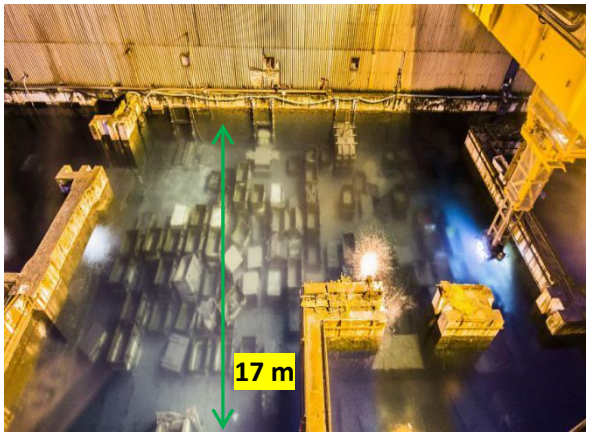


Figure 3.8: Legacy nuclear storage pond [9]

In modern storage pond, all surfaces are likely to be acoustically hard – boundaries are concrete, spent fuel canisters are metallic, such as zirconium alloys, stainless steel alloy [41]. This means that much of the incident acoustic wave will be reflected via the law of reflection. As for legacy pond, their boundaries are similar to the modern pond. Canisters are also likely to be metallic although after long immersion may not be as smooth as those in modern. The key difference between modern and legacy is the presence of sludge and the disorder of the canister layout, as mentioned in Section 1.1. Bottom sludge which formed by corroded materials of from fuel rods, which may contain such as pockets of hydrogen and other solid objects, see Figure 3.9. It is likely that the sludge is not highly reflective. Finally, private communications from storage pond organisations indicate that there are temperature gradients within ponds, as might be expected due to some waste material being hot. The temperature gradient will lead to the speed of sound changes in gradient [42]. However, there is no information about the size of the temperature gradients.

Figure 3.9: Sludge at the bottom of a legacy storage pond [166]

3.5 The Approach Taken to Study Aerial Mapping in Pond

The ideal approach to study the pond aerial mapping is to set up underwater mapping experiments. However, there are several reasons that the work should be carried on computer simulation rather than underwater experiments. First of all, there is no such swimming pool-sized water pond available in the university. Perhaps it can find a swimming pool outside the university (e.g. aquatics swimming centre). Even the aquatics swimming pool is available to use, there are too many constraints to design such an experiment. Because set up equipment, conduct experiments, and collect data will take a long time (at least a couple of hours), the time slot allows to use the swimming pool is very limited as it is a commercialised facility and cannot be borrowed for too long, it is hard to complete the experiment within a short time. Regular accessing to the pond to repeat the experiment is also an issue. Moreover, design such an experiment need a lot of human resources to carry equipment, set up equipment, and record data. Secondly, the high measurement error in the small-scale model pond, Griffith [49] tried to take experiments in a necessarily small-scale model pond. However, the results are not optimistic. There are too many multipath effects on a small-scale model pond, which results in significant noise in the data, so there is unnecessary to repeat this experiment again. Finally, issues of the pond aerial mapping were not well understood at the beginning, so computer simulation study can be the start of this research. Moreover, the advantage of computer simulation is that there are no external constraints on facilities or settings and much safer, most importantly, they are capable of delivering good quality results, see Chapter 5. Hence, the approach taken is to study pond mapping and path planning via computer simulation with some supporting validation open-air experiments. The rest of this chapter concerns how the simulations are conducted.

3.6 Choosing of Ray Tracing Method

Modelling acoustic propagation in ponds has similarities to room acoustics used in the design of concert halls because they both concern acoustic propagation in an enclosed space. This section will discuss some indoor acoustic modelling methods. In general, there are three acoustic propagation modelling methods for an enclosed environment, which are empirical methods, wave-based method, and the geometrical method. The empirical method provides an estimation of the parameter, the Sabine and Eyring models is a typical empirical model to predict the volume of the enclosure by knowing absorption coefficients and reverberation

time [45]. The wave-based methods try to proximate the numerical solution. Several methods include finite element method, the boundary element method, and the finite-difference time-domain method give a numerical solution of the wave equation by introducing a discretisation of space, surface or time, respectively [162]. The ray-tracing based geometrical method regards the acoustic wave propagated as rays and ignored wave nature but emphasised geometric reflections, enabling the complexity of wave interactions to be simplified [44]. It traces the trajectory of acoustic waves in the course of their propagation.

Among the above room acoustics modelling methods, empirical methods and wave-based methods are mathematical modelling methods, and they are not very suitable for modelling the aerial mapping. Because the main purpose of the underwater aerial survey is to measure the distance between the detected objects and the scanner and acquire geometry information of the pond. Instead, the ray-tracing method is suitable for modelling the aerial mapping of a storage pond. The ray-tracing method traces the ray propagation between the scanner and the intersected objects, which will provide the distance relationship between the scanner and the intersected objects, so the ray-tracing method is an appropriate technique for the aerial mapping simulation. The ray tracing is applicable for the storage pond case because the dimension of the canisters is larger than the wavelength of the ultrasound wave, so the propagation of the sound can be approximated as a ray.

3.7 Ray-Tracing in Underwater Aerial Survey

3.7.1 Ray-Tracing Algorithm

The ray-tracing method can be used to model wave propagation problems in the case that wave is travelling through environments where the solid objects have dimensions that are very much larger than the wavelength of the wave. Under these conditions, wave behaviour tends toward simple ray behaviour [44]. The technique has many applications in electromagnetic and acoustic [50, 167], and in computer graphics [168]. The storage pond problem satisfies the condition of ray tracing as the solid objects (spent fuel canisters) has dimensions that are very much larger than the wavelength of the ultrasound wave. Moreover, as mentioned in section 3.4, those solid objects are acoustically hard, so reflections are the main concern.

The concept of acoustic ray tracing is: the sound waves emitted by the acoustic sensor propagates along with rays that are normal to wavefronts, so they can be described as a beam

of rays travelling at the speed of sound without considering the wave interference and diffraction [44]. It assumes that each ray carries the same amount of initial energy. Over time, the carried energy will be dissipated due to the reflections of objects. In distance measurement, reflection is the key factor. There are two types of reflection: specular reflection and diffuse reflection. See Figure 3.10.

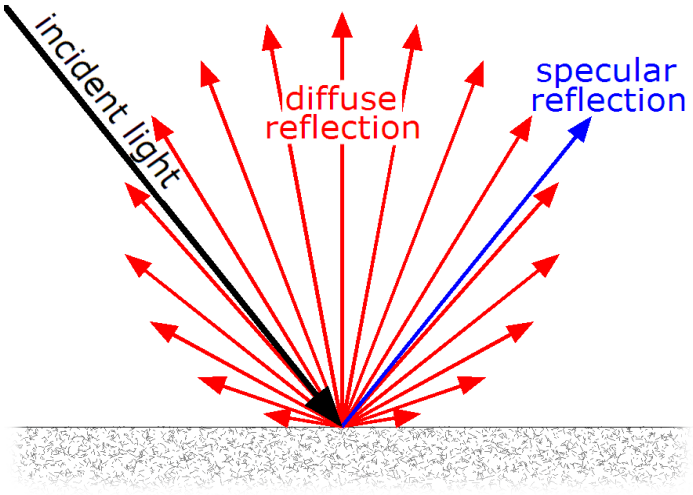


Figure 3.10: Specular reflection and diffuse reflection [44]

Specular reflection occurs when rays reflect from hard, smooth surfaces and obey Law of reflection, namely that the angle of reflection is equal to the angle of incidence. The special case that incident ray is perpendicular (normal reflection) to a surface relates to depth measurements is particularly important for depth measurement.

Diffuse reflection is the reflection from a rough surface where a ray incident on the surface scatters in many directions. Lambert's emission law [47] states that incident rays on a surface scatter in all directions toward the half-space adjacent to the surface rather than just one direction. This means that, regardless of the incident angle of the ray, there always some rays reflected back in the direction of the incident ray. The use of a scattering coefficient is the most popular method to weight the relationship between specular reflection and diffuse reflection. The scattering coefficient δ is given by:

$$\delta = \frac{W_{non-specular}}{W_{non-specular} + W_{specular}} \tag{3.2}$$

Where $W_{non-specular}$ is the energy of non-specular reflections (scattering) and $W_{specular}$ is the energy of specular reflections. δ is the fraction of scattering energy of the overall energy. The scattering coefficient is depended on the roughness and material of the surface, so the

energy intensity of a scattered ray also depends on the roughness and material of the surface [48].

Another factor that affects specular and diffuse reflection is the absorption coefficient α, and it depends on the hardness and material of the surface [163]. The relationship between δ and α is illustrated in Figure 3.11.

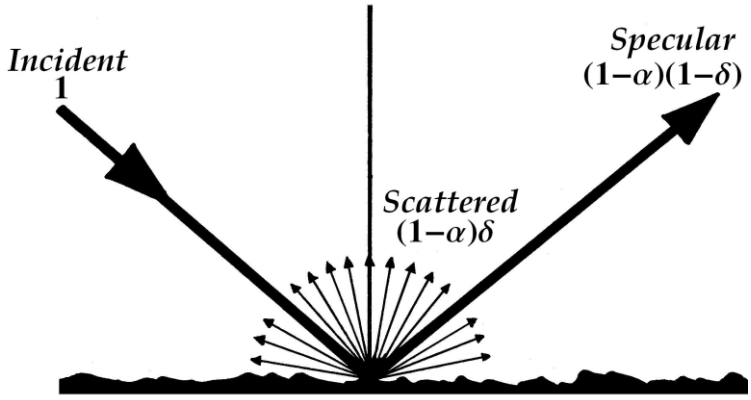


Figure 3.11: Scattering coefficient and absorption coefficient [44]

The transceiver will receive a normal reflection ray and a diffuse reflection ray. However, the energy intensity for the normal reflection signal and diffuse reflection signal is different, which depends on the surface material and its hardness. In general, for acoustically hard material, the energy intensity of the specular reflection signal is stronger than that of the diffuse reflection signal due to its low scattering coefficient [44]. Based on this fact, the specular reflection signal and diffuse reflection signals can be distinguished. The ray will not stop bouncing in the enclosure unless set threshold energy. Once the remaining energy of the ray is lower than the threshold energy, the ray disappears, otherwise, keep bouncing.

3.7.2 Impulse Response in Room Acoustics

The impulse response in the time domain plays a key role in measuring distance. The computing of the impulse response of room acoustics using the ray-tracing method is explained in [165]. In the room acoustics, the transmitter and receiver are placed in two separate locations, and the transmitter produces omni-direction rays. Figure 3.12 shows the diagram of the propagation of acoustic rays in a room.

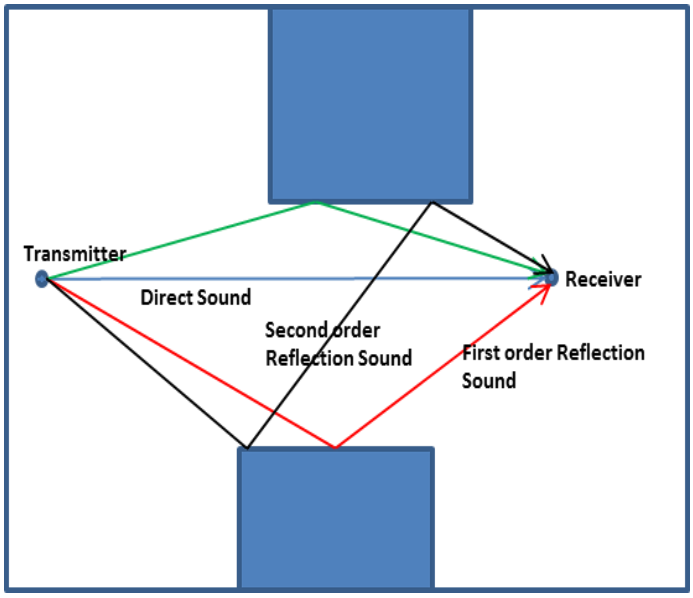


Figure 3.12: Ray propagation

The sound waves created by the source (transmitter) bounce around the room. The direct sound (blue line in Figure 3.13) is a straight path from the transmitter to the receiver. It arrives first and has the highest energy impulse response, as the first pulse in Figure 3.13. The first order reflection sound (red line and green line) is reflected off from objects to the receiver. It arrives late and has a lower energy impulse response, as the next couple of pulses in Figure 3.14. The multiple reflection sound (black line) is the sound signal that is reflected off from objects multiple times before arriving at the receiver. It has a much lower energy impulse response, corresponds to the rest of the pulses in Figure 3.13 (the remaining pulses also includes diffuse reflections that are not shown in Figure 3.12). Rays disappear when carried energy lower than the threshold energy.

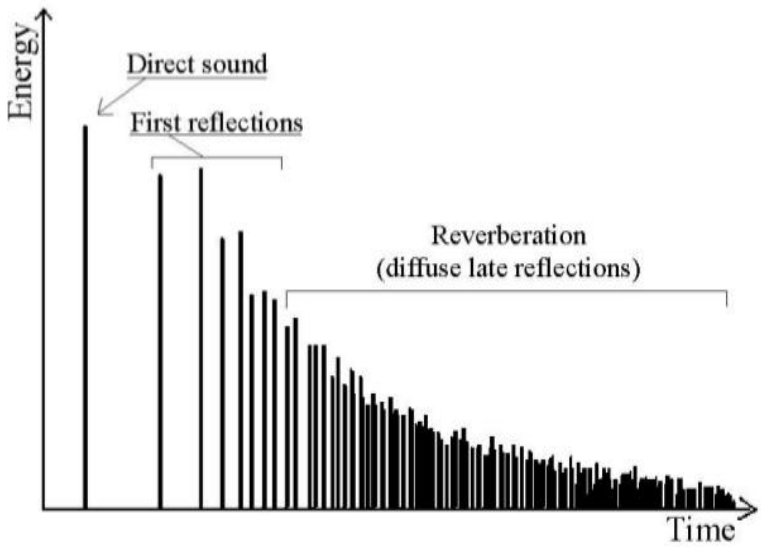


Figure 3.13: Acoustic impulse response of a room [44]

The multipath effect is a problem in room acoustics, since the transmitted signals are omnidirectional, so there a possibility that two different reflections arrive at the same time, Gu [50] and his colleague investigated the impact of multipath reflections on indoor visible light communication positioning.

3.7.3 Applying Ray Tracing to Storage Pond Simulation

Basic ideas of ray tracing are discussed in the previous section. However, its application in the storage pond aerial mapping simulation is slightly different. The differences are rather than producing omni-direction rays, the scanner will produce a cone-shaped beam downward, and it combines the transmitter and the receiver in a single housing (transceiver). An envisaged scenario of the measurement can be found in Figure 3.2 in section 3.1. The application of ray tracing is discussed here to simulate the behaviour of ultrasonic propagation in a pond environment. To do this, it assumes:

- Speed of sound is constant, assume it is 1450 m/s
- All clutter in the pond is prismatic.
- All surfaces are acoustically hard, and so specular reflections overwhelm diffuse reflection.
- Assume that propagation from the sensor is conical.
- The sensor emits a beam with a central vertical ray will reflect back from bottom other rays in the beam will reflect at different angles.
- The only diffuse reflection ray that will be considered is the one that returns to the incident direction.
- In the storage pond, reflection from adjacent clutter will also be received. However, the normal reflection is the key to depth measurement, so assuming the TOF of the signal that has the highest impulse response is used for depth measurement.
- Assumes the threshold energy is 1/11 of the initial energy.

The impulse response in storage pond aerial mapping adopts the same manner that in room acoustics, so it can be used to find the incident ray that is perpendicular (normal reflection) to a surface. Walter discussed the impulse response of an ocean waveguide in his PhD thesis [17].

Figure 3.14 shows the schematic diagram of the propagation trajectory of projected rays and its echo-sounding reflections to a horizontally placed object. As Figure 3.14 shown, the

scanner produces a wide beam towards a horizontally placed object, which creates different types of reflection that includes normal reflection, diffuse reflections, lateral reflections, and not received reflections. Among all types of reflections, the only one that is useful for calculating the distance is the first order normal reflection (the red arrow) because it is perpendicular to the intersection plane. Hence, the X, Y position of the reflection point is the same as the scanner's position which is known. The distance between the scanner and the reflection point can be calculated by Equation 3.1. As for the other reflections (blue arrows), the X, Y position of the reflection point and the propagation trajectory are unknown so it cannot extract useful geographical information.

The ray is the carrier of energy, and its energy will dissipate during propagation and collisions. Based on this fact, the ray that reflected from the nearest obstacles should have the highest remaining energy [17]. Therefore, TOF and impulse response energy levels can be used to find the normal reflection from the received rays in the time domain. Specifically, after filtering the noise signal, the pulse that has the highest energy response in the time domain is the normal reflection signal, its corresponding TOF can be used to calculate the depth.

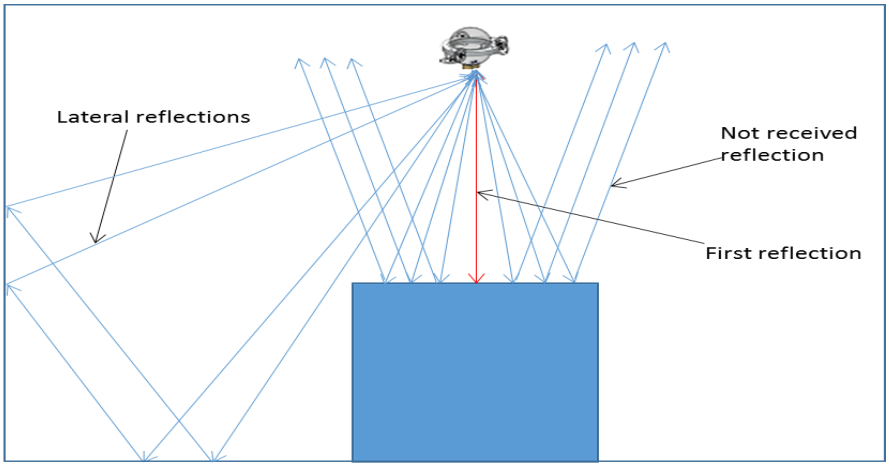


Figure 3.14: Ray reflection for aerial mapping

Figure 3.15 is the testing case shown in Figure 3.14 constructed in MATLAB simulator. The testing conditions are:

- Measurement position: (25, 20, 0)
- Pond depth: 10m
- Clutter height: 5m
- Beam angle: 20 degree
- Number of rays: 21

- The angular separation of rays: 1 °
- The energy carried per ray: 150 units
- The threshold energy: 13.63 units (1/11 of the original energy)
- Energy remaining upon reflection: 50% for specular reflection 10% for diffuse reflection

Figure 3.16 shows the corresponding energy impulse response of each received ray in the time domain. It illustrates how the normal reflection signal (pulse) is found from the received impulse response. The key reflection for depth measurement is the received pulse that has the highest impulse response. This corresponds to the normal reflection. The TOF of this signal is 0.007 s; the distance between the scanner and the reflection point is 0.007*1450/2=5.075m. Hence the coordinate of the reflection point is (25, 10, -5.075). The height of the object is 10-5.075=4.925m.

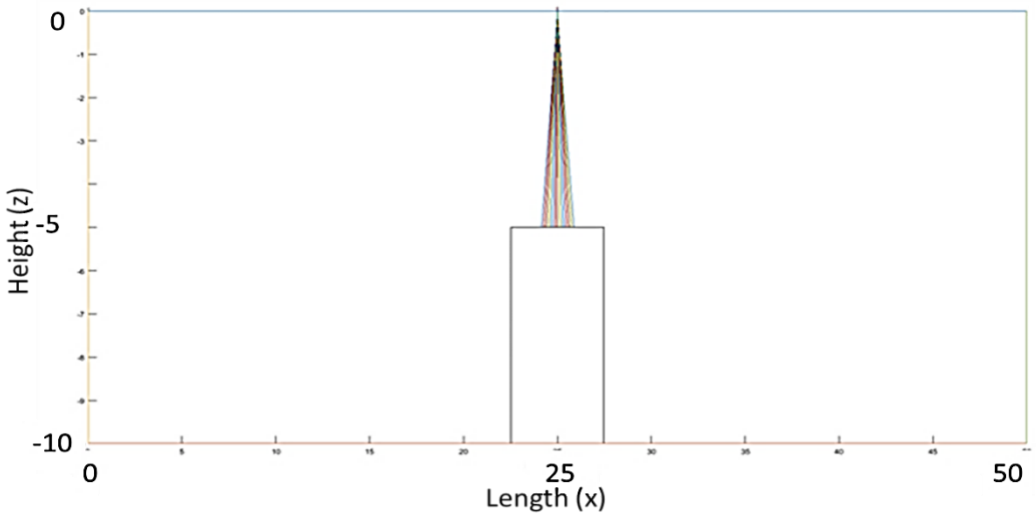


Figure 3.15: A scan of measuring an object

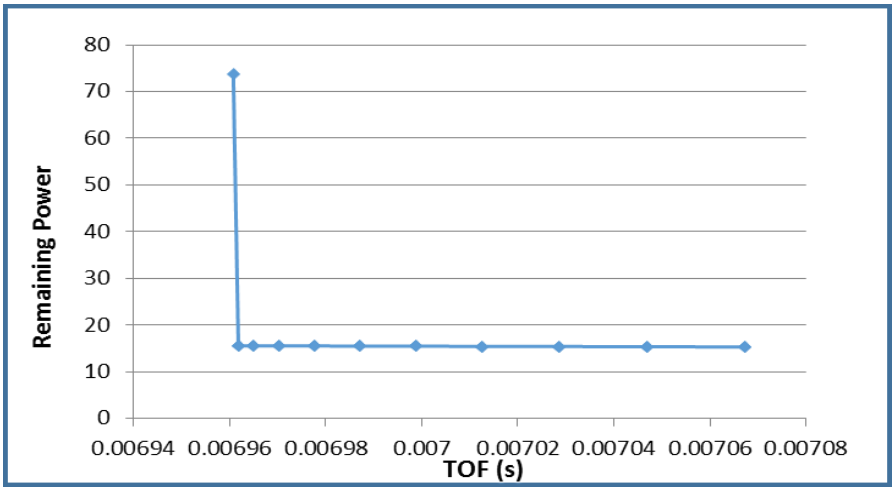


Figure 3.16: Impulse response of the scan

There is one concern about this approach. If there are two scans (measurements) take place in a short time, how to avoid the second scan receives a lateral reverberation of the first scan and treats it as its own first-order reflection (Because the speed of sound is slow compare to the light, the reverberation time is long). To address this problem, it needs to know the absorption index and the scattering coefficient, the absorption index of concrete is about 0.01, and the scattering coefficient of concrete is about 0.1 [163]. And then, it needs to either set up an energy threshold, only the impulse response signals that larger than this energy threshold are considered, or set up the modulating frequency carefully, e.g. reflections created by the previous scan dissipated within the time interval.

3.7.3.1 Reflection from a 2D inclined surface

To conduct an aerial survey, the entire enclosure is defined in one coordinate system, and the scanner (transceiver) will move along X-axis and takes depth measurement in sequence, refer to Figure 3.3 in section 3.1. This setup applies to all diagrams below.

For a horizontal surface, the transceiver will always receive normal reflection signals. However, for the case of an inclined surface, it is possible that normal reflection signals do not exist. Because there is no ray that perpendicular to the inclined surface, see Figure 3.17.

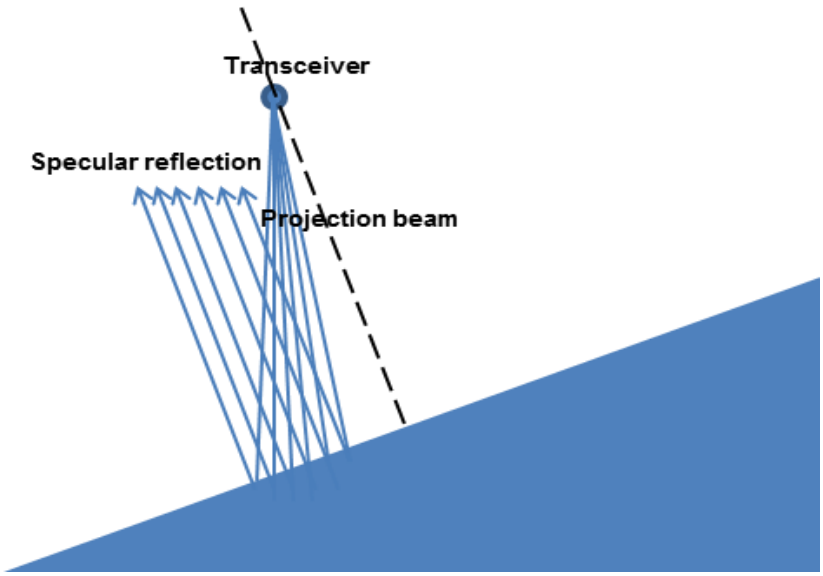


Figure 3.17: Small beam without normal reflection

This problem could be addressed in several ways, which all involve producing a projection ray perpendicular to the inclined surface, as shown in Figure 3.18. This can be achieved by using a wide beam-angle transducer, designing a sweep module, control the µAUV to perform roll and pitch motions or design a sensors array with n-degree intervals. However,

the µAUV developed in AVEXIS do not support roll and pitch, and how to achieve this in µAUV is beyond this thesis.

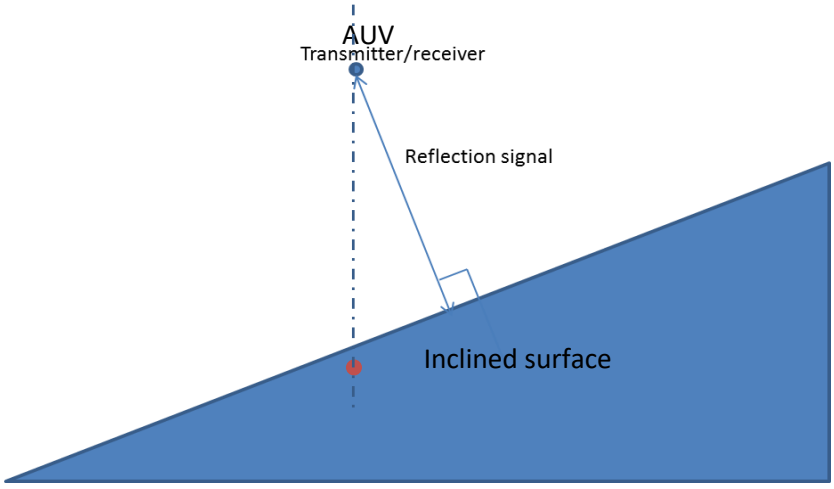


Figure 3.18: The normal reflection

As Figure 3.19 shows, transducer C projects a wide conical beam to detect an inclined surface. A is the aimed detection point because A is the projection of C at the inclined surface. B is the actual detected point because B is the normal reflection point. The sensor aims to detect point A, but its reflected signal is missing (as Figure 3.19 shown) because of specular reflections. The point detected by the sensor is B. Hence, the true measured distance is CB. It is the 'echo-sounding' assumption that caused the problem, i.e. that the first reflection comes from vertically below the sensor.

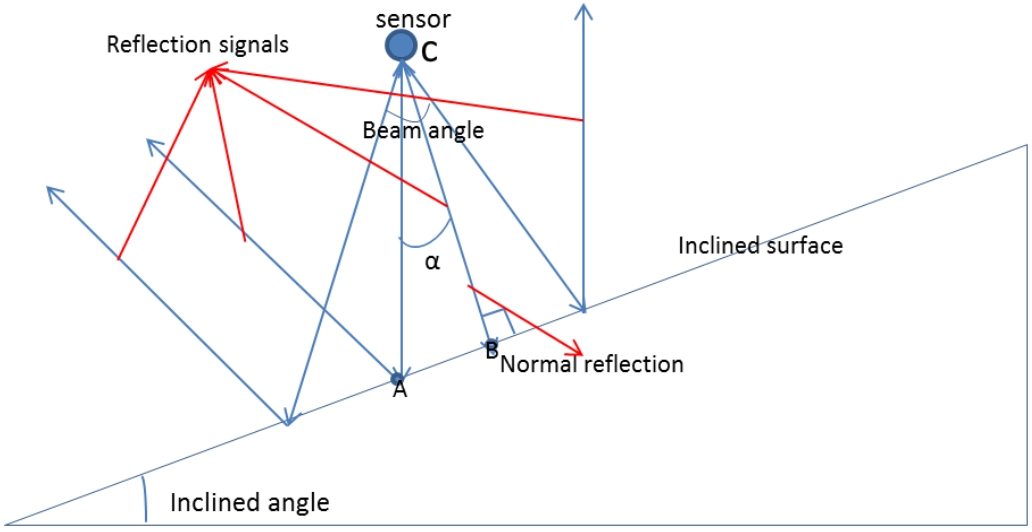


Figure 3.19: Sensor survey to an inclined surface

As long as the beam angle is no smaller than the inclination angle, the sensor will always receive a normal reflection signal. If an AUV tries to take a depth measurement but depth = INF, then clearly it can be inferred that there is a surface with a slope angle greater than the

beam angle. To avoid this, it needs an array of sensors or controls the µAUV to perform the rolling. As mentioned earlier, the µAUV developed in AVEXIS does not support rolling and pitching, so design an array of sensors is the better option. It can use three narrow beam sensors to create a sensor array and ensure that the beam of each sensor does not overlap with each other; see the design in Figure 3.20. To avoid the cross-talk between adjacent sensors, each sensor has its own frequency and executes in orders with a small interval, so that each sensor can distinct its own produced signal and rejects interferences from nearby sensors.

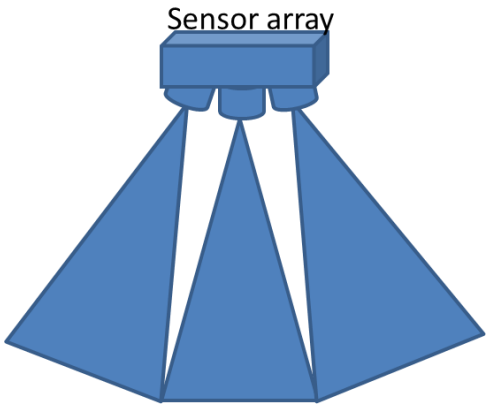


Figure 3.20: 3 sensors array

3.7.3.2 Position of the Reflection Point

As discussed earlier, the position of the reflection point on a horizontal surface is the same as the position of the sensor, but the position of the reflection point on an inclined surface is different from the position of the scanner. As shown in Figure 3.21, B is the true detected point, but its position is unknown, and so it is required to calculate the position of B, with CB being the measured 'depth'. The scanner's position $C(x_c,\ z_c)$ is known.

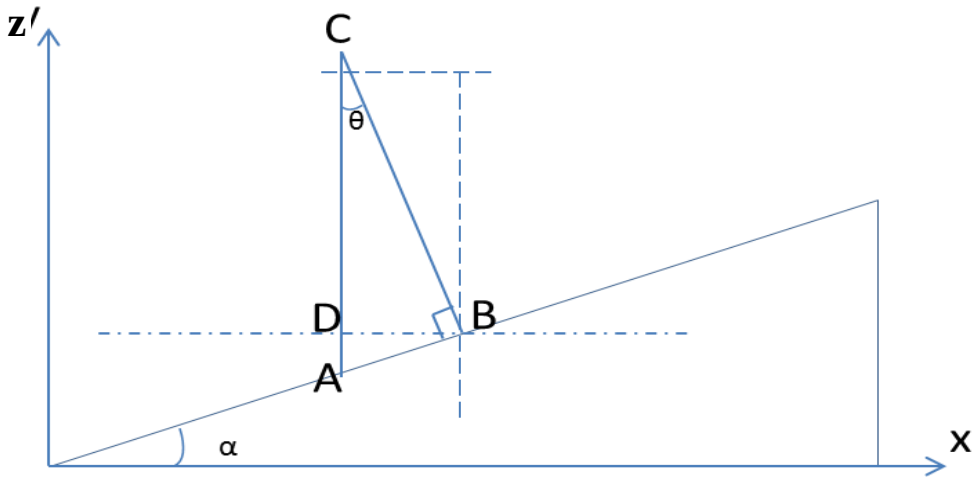


Figure 3.21: Geometry relationship between normal reflection point and scanning point

Coordinate of B can be calculated by trigonometry below:

$$x_b = x_c + CB * \sin(\alpha)$$

$$z_b = z_c - CB * \cos(\alpha)$$

3.7.3.3 Estimation of the Inclination Angle

The key element to calculate the position of the reflection point is the inclination angle of the tilted surface. However, the inclination angle is unknown, so the first step is to estimate it. Figure 3.22 shows the 2D situation where the scanner moves along the measurement plane (X-axis) and takes a sequence of measurements.

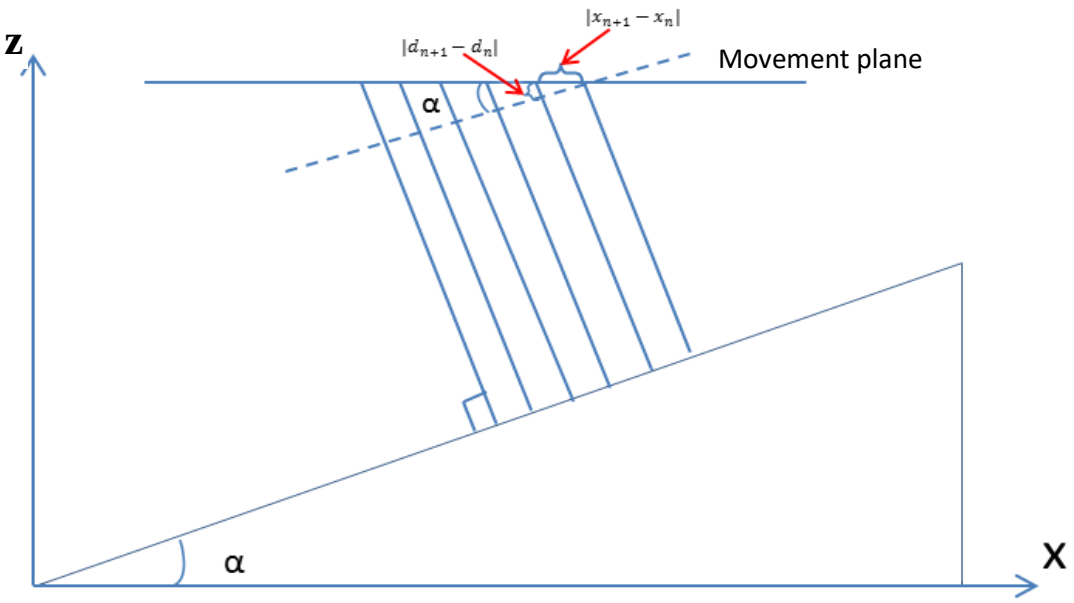


Figure 3.22: Estimation of the inclination angle

Based on the average rate of change of measured distances along X-axis, the inclination angle can be estimated as:

$$\alpha = \frac{1}{m}\sum_{i=1}^{m} sin^{-1}\left(\frac{d_{n+1}-d_n}{x_{n+1}-x_n}\right) \quad (3.3)$$

Where $(x_{n+1} - x_n)$ is the gap between two consecutive scanning positions, $(d_{n+1} - d_n)$ is the difference of two consecutive measured distances. If a set of scans is performed on the same continuous inclined plane, the rate of change of measured distances for each measurement position should be the same, and the average of the changing rate should equal to the inclined angle of this plane, vice versa. If it is a horizontal plane, the changing rate (α) is zero or nearly zero. Based on this, the ground and the inclined plane can be determined. As for the practical AUV scans, if α≈zero, the detected surface is either the pond ground or the

upper surface of a non-tilted spent fuel canister (whether it is the ground or the upper surface of a non-tilted spent fuel canister is determined by the depth). If α≠zero, the detected surface is the upper surface of a tilted spent fuel canister. However, this is only true for a linear slope.

3.7.3.4 Reflection from 3D Inclined Surfaces

Clutter that is sloping may not be parallel to coordinate axes. Its upper surface may slope in two directions. Hence, it needs a more sophisticated approach to calculating 3D inclination. Now refer to Figure 3.23.

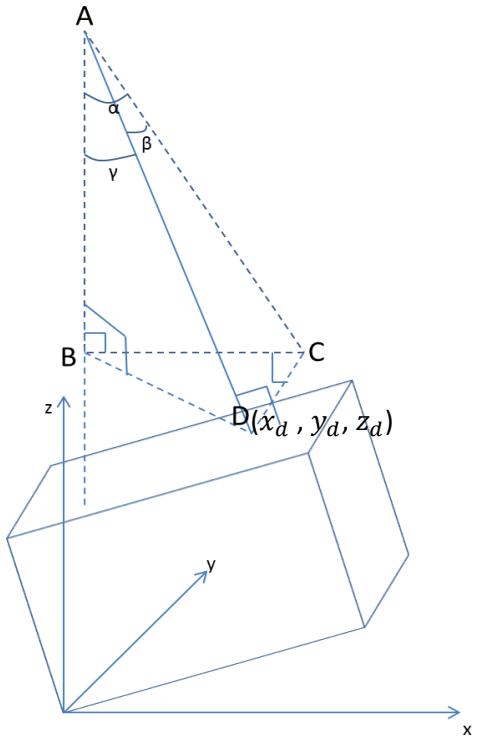


Figure 3.23: Normal reflection in the 3D scene

AD is perpendicular to the upper surface of the inclined cuboid, A is the scanner's position, and D is the normal reflection point. γ is the angle between the normal reflection and the Z-axis (also known as the inclination angle of the cuboid), α and β are the two projecting angle of γ with respect to the X and Y-axes, respectively. Figure 3.24 shows the measurements obtained by the aerial scan over the inclined object measurement plane in the X and Y directions. α can be calculated by the sequence of measurements along X-axis (see Figure 3.25), and β can be calculated by the sequence of measurements along Y-axis (see Figure 3.26).

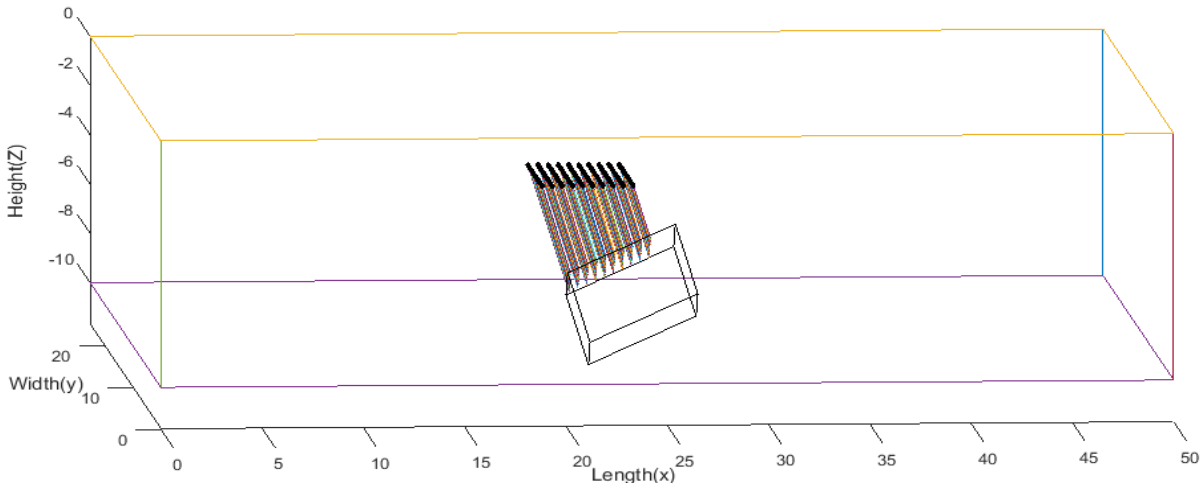


Figure 3.24: Mapping on to a tilted object

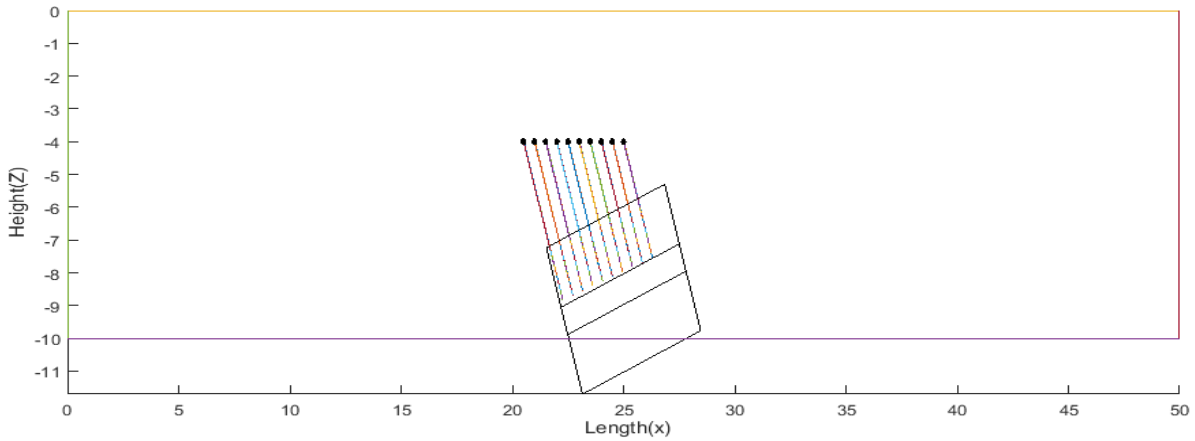


Figure 3.25: Front view of scans (along the x-axis)

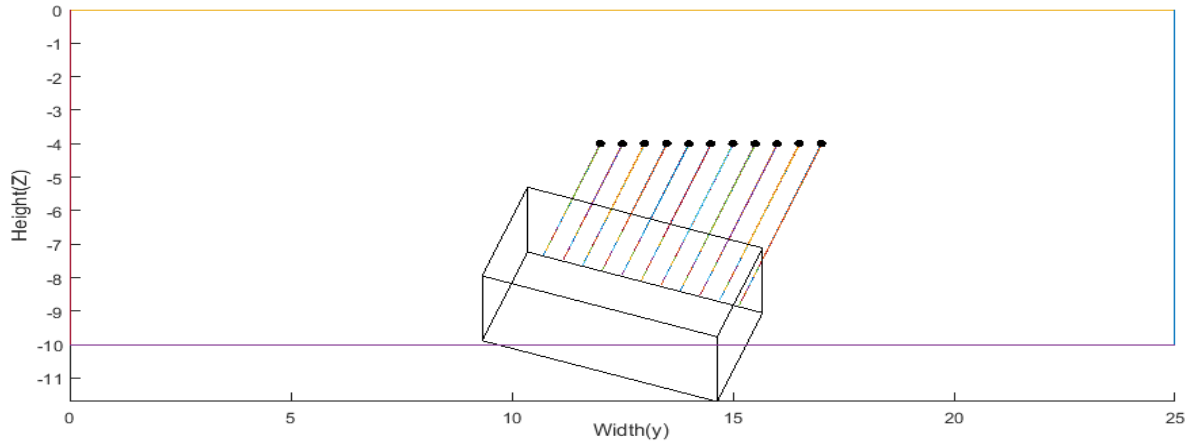


Figure 3.26: Side view of scans (along the y-axis)

Since α and β can be calculated by the same method as discussed for the 2D case, apply equation 3.3 to calculate α and β:

$$\alpha = \frac{1}{m}\sum_{n=1}^{m} \sin^{-1}\left(\frac{d_{n+1}\text{-}d_n}{x_{n+1}\text{-}x_n}\right)$$

$$\beta = \frac{1}{m}\sum_{n=1}^{m} \sin^{-1}\left(\frac{d_{n+1}-d_n}{y_{n+1}-y_n}\right)$$

Hence, point D's coordinate given by:

$$x_d = x_a + BC = x_a + AD * \cos(\beta) * \sin(\alpha)$$

$$y_d = y_a - DC = y_a - AD * \sin(\beta)$$

$$z_d = z_a - AB = z_a - AD * \cos(\beta) * \cos(\alpha)$$

3.8 Depth Measurement Data

The data set acquired by a storage pond aerial survey consists of many point depth measurements taken on a horizontal (X-Y) grid. This information can be used to produce a 'height map', for example, contours of the bottom of the pond. The map only represents a projection of the clutter on to the horizontal measurement plane. However, since objects may be stacked or overlie one another, this scan information does not completely represent the three-dimensional distribution of objects on the pond floor.

3.9 Summary

This chapter has introduced the approach of performing aerial surveys (echo-sounding) in a nuclear storage pond. As argued in section 3.1, the ultrasound sensor is appropriate for conducting echo sounding for reasons of low-cost and good performance underwater. However, difficulties in developing a realistic test environment forced research to focus on simulation. As discussed in section 3.7, the propagation of the ultrasound wave can be approximated to a ray, so the ray-tracing method is used to simulate the echo sounding. Two key concepts of the ray-tracing method: time of flight (TOF) and energy impulse response of the normal reflection are used to estimate the distance between the scanner and the reflection point. As for the issue of the slope, a wide projection beam is proposed, and calculation in section 3.7.3.2 and 3.7.3.3 explains how to estimate the true detection point. The consequence of aerial mapping is an array of topological data of the pond. The next chapter will explain how to build a ray-tracing simulator in MATLAB.

Chapter 4

Implementation of the Ray-Tracing Algorithm

4.1 Introduction

There are three ways of getting a ray tracer, either devolving one in MATLAB, using one from the Internet or buying one. There are many commercial ray-tracing packages available, such as ODEON and EASE. However, they are too expensive. A basic ODEON costs €4363 [169]. A standard EASE costs €2110 [170]. They are very good for industrial applications, such as modelling a concert hall, but its price is not affordable for this project. Getting a free ray-tracing program is a good idea, however, not very much available, and they only have the basic function like trace the ray, do not have the function of modelling energy dissipation. Therefore, developing a ray tracer is a good choice. This chapter discusses the ray tracing aerial mapping simulator developed in MATLAB. The flowchart in Figure 4.1 shows how the simulator works.

The first step is to configure all the geometry of the pond by the program, including the geometry of the enclosure (assumed to be a cuboid), and the shape, orientation, height, position and numbers of clutters. Next, set the position of the scanner that is the source of rays. After that, initialise the ray, including the number of the rays, the beam angle, the initial energy carried by each ray, the energy losses index via propagation through the water and reflection upon clutters, and the threshold energy discussed in section 3.7.2. Then, project a ray downward and trace its propagation. After that, identify the true intersection surface via the visibility test (see section 4.2), if the remaining energy after the intersection is larger than the threshold energy, trace the reflection via s Law of reflection, otherwise, return to create another ray (block 4) and repeat above again. The next step is to check if the reflected ray intersects with the scanner (block 12), if the reflected ray intersects the scanner, measures its remaining energy and TOF, otherwise keep tracing the ray for the further reflection until its remaining energy lower than the threshold energy. Although both specular reflections and diffuse reflections are modelled in the program, the specular reflections are the main

consideration, and the only diffuse reflection that will be considered is the one that returns to the incident direction. After that, return to project another (block 4) ray and repeat the above procedures until all rays are shot. Once all rays are shot, find out the TOF of the ray that has the highest remaining energy, and then calculate its travelled distance and save it. After that, move to a new position of the scanner (block 2) and restart all over again until finishing the aerial mapping. In the end, the ray tracer program will output an array of mapping data.

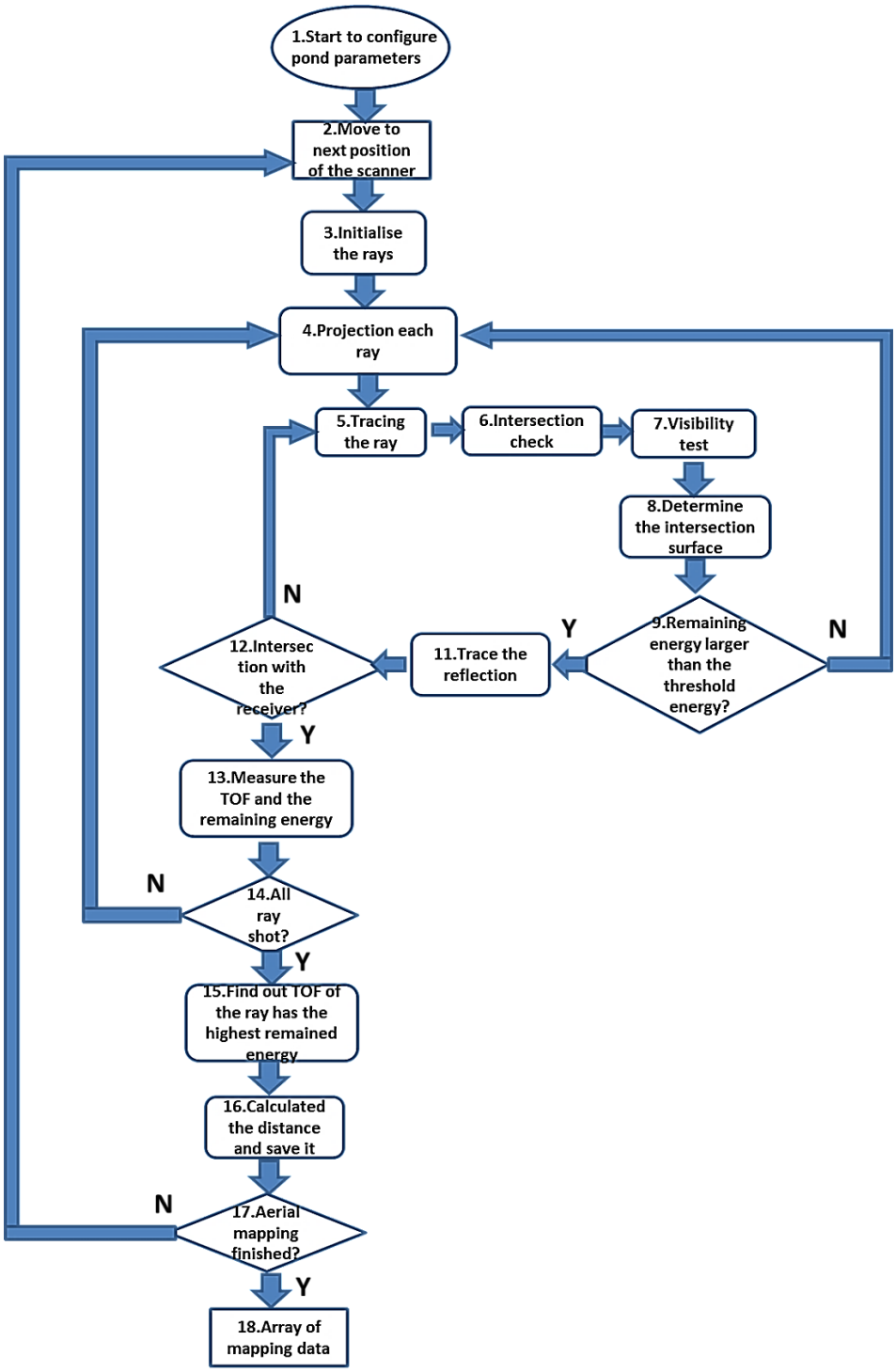


Figure 4.1: Flow chart ray-tracing algorithm

4.2 Geometrical Description of the Environment

The geometry of nuclear storage pond is represented by a cuboid enclosure with a length of A, a width of B, and a depth of C. The definition of the enclosure in MATLAB is given by:

$$pool\_dim = [0\ 0\ 0\ A\ B\ C];$$

This syntax creates an object with a length of A, a width of B, and a height of C, see Figure 4.2.

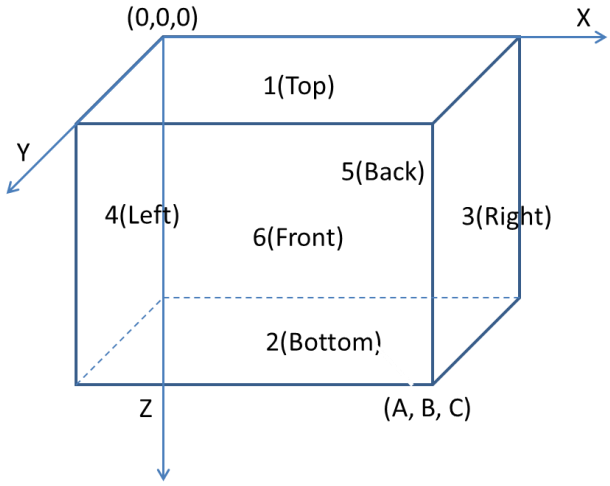


Figure 4.2: 3D pond enclosure

The geometric model of the used fuel canisters is more complicated than that of the pond enclosure because the shape might vary and may be inclined. The definition of a used fuel canister is given by:

$$clut\_dim = [x\ y\ z\ r\ h\ n\ \alpha\ \beta\ \theta];$$

Where x, y, z is the centre point of the top surface, r is the distance between the centre and the top surface's vertex, h is the height of the object, n is the number of side surfaces, α, β, and θ are the clockwise rotation angles about the X, Y and Z axes, respectively, see Figure 4.3. This syntax creates an object that can vary in size, shape, and orientation direction. For example, clut_dim =[50 25 -5 2.5 5 8 20 10 30] create an object that has eight side surfaces, a height of 5 m, centre point of top plane at (50, 25, -5), rotated about the X, Y and Z axes by 20 °, 10 ° and 30 °, respectively, see Figure 4.4.

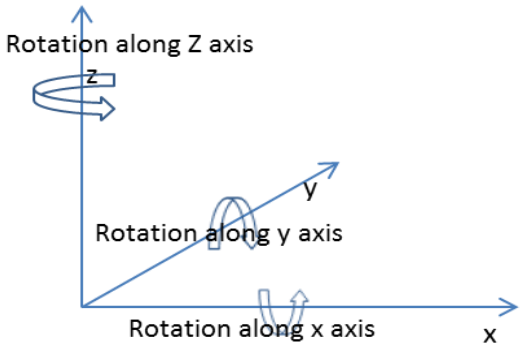


Figure 4.3: Rotation

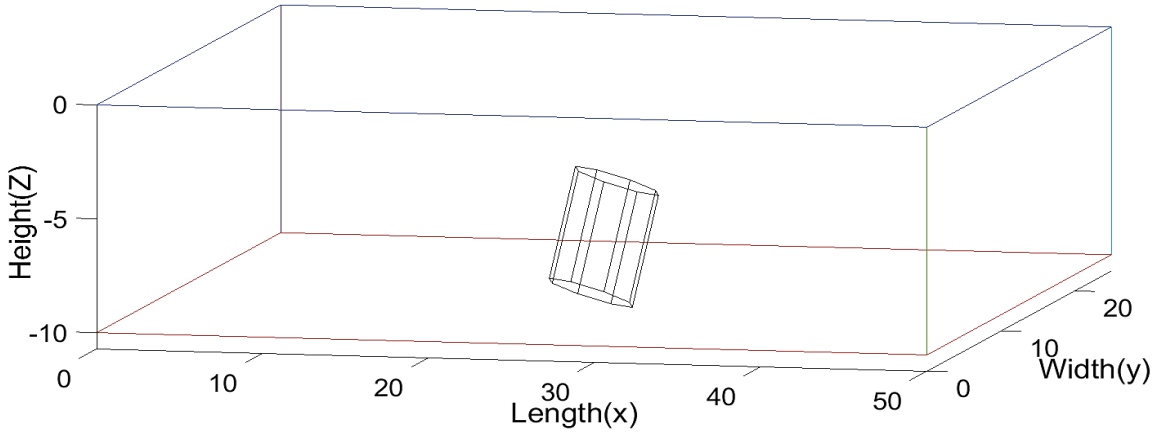


Figure 4.4: Example of creating an object

The scanner is a transceiver whose receiver and transmitter are combined in a single housing, represented by a spherical object with a radius of *r* and above the object. The reason that uses a spherical model to represent the scanner is that it is a simple geometry model, and the sphere can represent an omni-directional transceiver that produces and registers rays from all directions with equal weight.

4.3 Decomposition of Ray Projection Angle

The rays projected by the scanner reflect within the pond enclosure. To create a beam of rays in MATLAB, the projection angle of the ray is decomposed into two angles (v_angle) and (h_angle). h_angle is the angle between the projection of the source ray in the X-Y plane (as h_angle on Figure 4.5 shown), the h_angle's range is 0 ° to 360 °, which creates a cone beam. v_angle is the angle between the source ray and Z-axis (as Figure 4.5 shown), and it is also regarded as the angle of the beam.

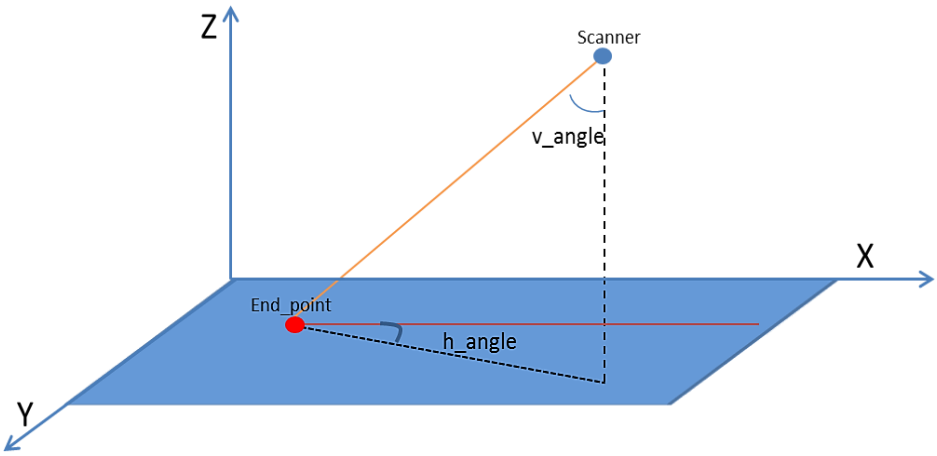


Figure 4.5: h_angle and v_angle

4.4 Lines-Plane Intersection

Since the position of the scanner (tx) and the projection angle is known, the ray equation is given by following MATLAB equations and statements.

The Start Point is tx=[tx(1), tx(2), tx(3)];

*d = D*cos (v_angle); // D is the 1 unit magnitude of the ray,*

*c = D*sin (v_angle);*

*a = c*cos (h_angle);*

*b = c*sin (h_angle);*

line_vector=[tx(1), tx(2), tx(3); tx(1)+a, tx(2)+b, tx(3)+d];

Since the *line_vector* is given, clutters and pond boundaries are a set of surfaces that it could intersect; the problem is to find the true point of reflection, which is defined in the next section.

4.4.1 Ray-Polygon Intersection

As stated in Section 3.3, it assumes that the clutter is prismatic, so each constituent surface is a rectangle as shown in Figure 4.3, so the ray-polygon intersection method [130] can be used to find out the intersection point. A polygon has at least three vertices. Take the scene in Figure 4.6 as an example. The polygon has four vertices D, E, F, G, a normal vector $\vec{n}$ and the incident ray is $\vec{R}$.

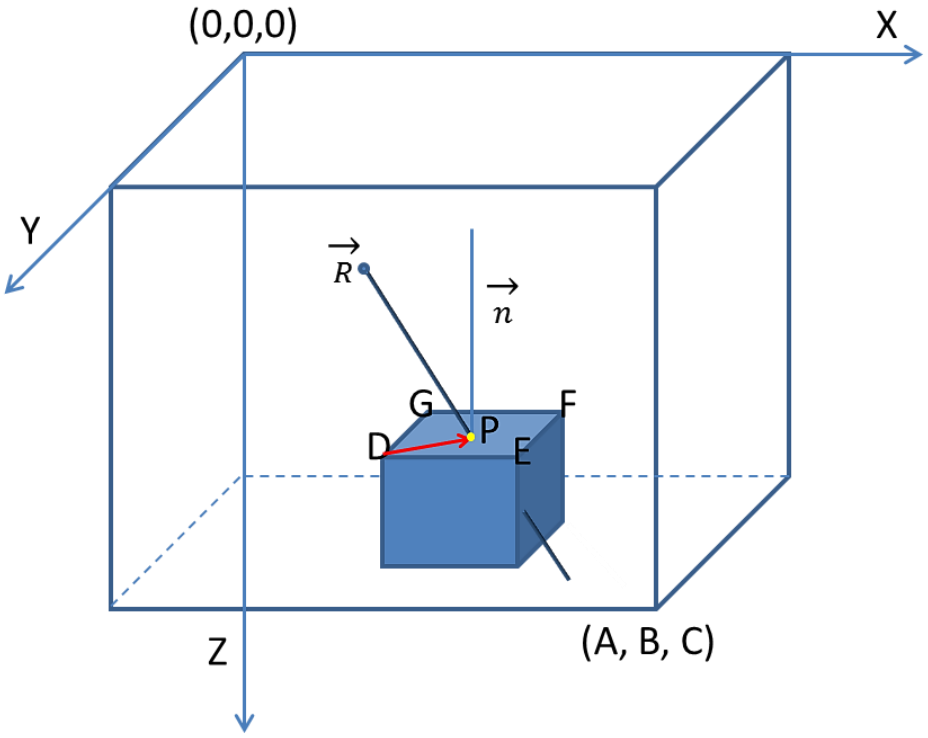


Figure 4.6: Ray-Polygon intersections at the upper surface

The normal unit vector $\vec{n}$ can be calculated by the cross product of side vectors $\overrightarrow{DE}$ and $\overrightarrow{DG}$:

$$\vec{n} = \frac{\overrightarrow{DE} \times \overrightarrow{DG}}{\sqrt{|DE|^2 + |DG|^2 - 2|DE \cdot DG|}} \tag{4.1}$$

If $\vec{R}$ intersects with the plane surface DEFG, and the intersection point is P, hence:

$$\overrightarrow{DP} \cdot \vec{n} = 0 \tag{4.2}$$

Note this is also true for the vectors connecting the other vertices to P. The coordinates of P is written as the equation $\vec{P} = \vec{e} + t \cdot \vec{d}$ (e is the start point of ray, d is the unit vector in the direction between the start and the endpoint, and t is the magnitude). Equation 4.2 can be rewritten as:

$$(\vec{e} + t \cdot \vec{d} - \vec{D}) \cdot \vec{n} = 0$$

$$t = \frac{(\vec{D} - \vec{e}) \cdot \vec{n}}{\vec{d} \cdot \vec{n}} \tag{4.3}$$

The t can be calculated from the above equations. If the ray crosses the surface DEFG, the intersection point P can be determined from:

$$\vec{P} = \vec{e} + t \cdot \vec{d} \tag{4.4}$$

There is a problem in ray tracing. Mathematically, any line will intersect any plane as long as the line is not parallel to the plane. This causes the ray to intersect almost every plane in the pond environment. However, the surface that testing for intersection is a finite portion of a plane, so the calculated intersection point might locate on the plane but outside the surface. The next step is to check if the intersection point locates on the surface. The Crossing Number method [172] is used to determine if a point is inside a surface. The concept of this method is to count the number of times that a ray starting from the point crosses the polygon boundary edges. If "crossing number" is even, the point is outside; otherwise, the point is inside.

Another important aspect of ray tracing is the need for checking intersections between a ray and every surface in the environment. In most of the cases, the ray will intersect multiple surfaces, e.g. the incident ray $\vec{R}$ in Figure 4.6 intersects three surfaces, top and right-hand surfaces of the object and the bottom of the pond. It is essential to have a visibility check to find out the true intersection surface and the true inspection point.

4.4.2 Visibility Test

There are three intersection points for the incident ray $\vec{R}$ in the scene shown in Figure 4.6. It is essential to determine which intersection point is the true intersection point. The visibility test here is to determine which surface is the true intersection surface. If there are multiple intersections, then the closest one is the true intersection surface since rays cannot penetrate objects, so the true reflection surface can be found by comparing the travel distance between the scanner and the intersection point. The travel distance is calculated by:

$$Distance = \sqrt{(start_x - intersection_x)^2 + \left(start_y - intersection_y\right)^2 + (start_z - intersection_z)^2} \quad (4.5)$$

Where $start_{x,y,z}$ and $intesection_{x,y,z}$ are coordinate of the start point and intersection point, respectively. The surface that has the shortest distance to the scanner is the true reflection surface. In the case shown in Figure 4.6, the true intersection surface is the top surface of the object.

4.5 Line-Receiver Intersection

In practice, the ray propagation will end up either hitting the receiver or disappearing after too many reflections. However, in aerial survey, the distance between the scanner and the collision point is the key, the TOF of the received signal will be used to calculate the distance, and so the implementation only considers the first-order reflection. As stated in Section 4.2, the scanner is represented by a sphere, so the line-receiver intersection problem becomes a line-sphere intersection problem. In the case shown in Figure 4.7, the scanner is represented by a spherical object with a radius of r (r=0.025 m in the simulation, approximate the cubic size of MB1340 ultrasound range finder).

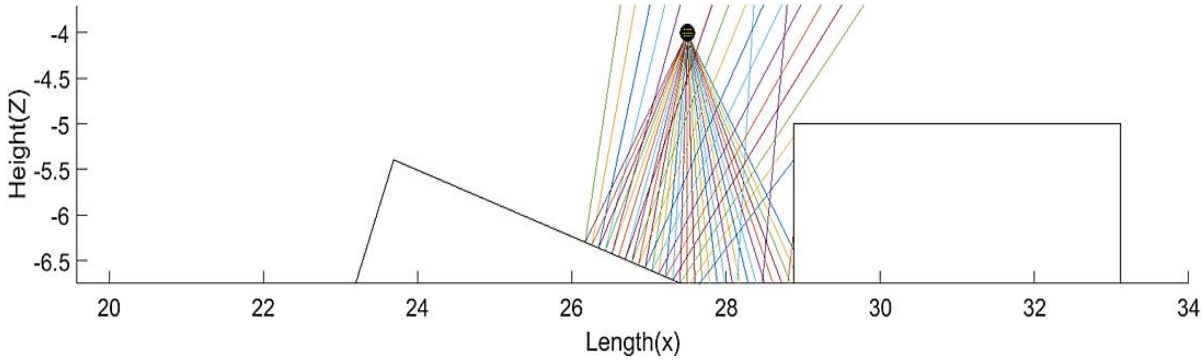


Figure 4.7: Line sphere intersection

4.5.1 Ray-Sphere Intersection

The ray-sphere intersection method [131] is used to check if the ray hit the receiver. Figure 4.8 shows a simple line receiver collision case.

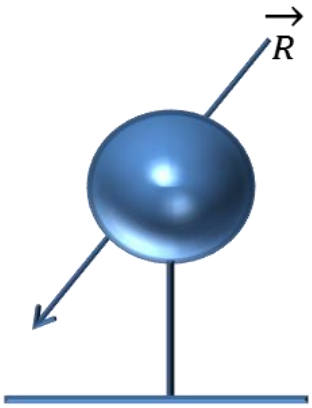


Figure 4.8: line receiver intersection

The equation of the ray is the same as Equation 4.4:

$$\vec{P} = \vec{e} + t \cdot \vec{d}$$

$\vec{e}$ is the coordinate of the start point (x0, y0, z0), d is the direction unit vector from the start point to the surface intersection point, the coordinate of the surface intersection point is (x1, y1, z1), t is the magnitude. Equation 4.4 can be changed to:

$$\vec{P}(t) = (x0 \;\; y0 \;\; z0) + t\{(x1 - x0) \;\; - \;\; (y1 - y0) \;\; (z1 - z0)\} \tag{4.6}$$

Assume the receiver C is a sphere and centred at point C (x_c, y_c, z_c,) with radius R. The equation of the receiver C is given by:

$$(x - x_c)^2 + (y - y_c)^2 + (z - z_c)^2 - R^2 = 0. \tag{4.7}$$

Any point that satisfied equation 4.7 locates on the sphere. So equation 4.7 can also be changed into vector form:

$$\{\vec{P}(t) - \vec{C})\} \cdot \{\vec{P}(t) - \vec{C})\} - R^2 = 0 \tag{4.8}$$

Replace$\vec{P}(t)$ with $\vec{e} + t \cdot \vec{d}$, equation 4.8 becomes:

$$(\vec{e} + t\vec{\mathrm{d}} - \vec{C}) \cdot (\vec{e} + t\vec{\mathrm{d}} - \vec{C}) - R^2 = 0 \tag{4.9}$$

Rearrange and expand equation 4.9 and obtained a new equation:

$$(\vec{d} \cdot \vec{d})t^2 + 2\vec{\mathrm{d}} \cdot (\vec{e} - \vec{C})t + (\vec{e} - \vec{C}) \cdot (\vec{e} - \vec{C}) - R^2 = 0 \tag{4.10}$$

Equation 4.10 is a quadratic function, whose solutions are given by:

$$t = \frac{-2d*(e-c) \pm \sqrt{\left(2d*(e-c)\right)^2 - 4d^2*((e-c)(e-c) - R^2)}}{2d^2} \tag{4.11}$$

In order to have a real number solution, it must satisfy:

$$\rho = \left(2d * (e - c)\right)^2 - 4d^2 * ((e - c)(e - c) - R^2) \geq 0$$

ρ is called determinant, if $\rho < 0$, Equation 4.11 does not have real solutions, in which case the ray does not intersect the source. $\rho = 0$ has one real solution, which represents the situation where the ray is tangential to the sphere. $\rho > 0$ has two real solutions, which is the case where the ray fully intersects the sphere. As a result, if ρ satisfies ($\rho \geq 0$), the ray will intersect with the receiver.

4.6 Ray Reflection and Energy Impulse Response

Reflection occurs when the ray intersects a surface. The reflected ray is the key to measuring the distance between the scanner and the object. There are two types of reflection, the

specular reflection and diffuse reflection. For specular reflection, there is only one specular reflection ray, and its reflection angle is the same as its incident angle see Figure 4.9. The specular reflection equation is derived in [171]:

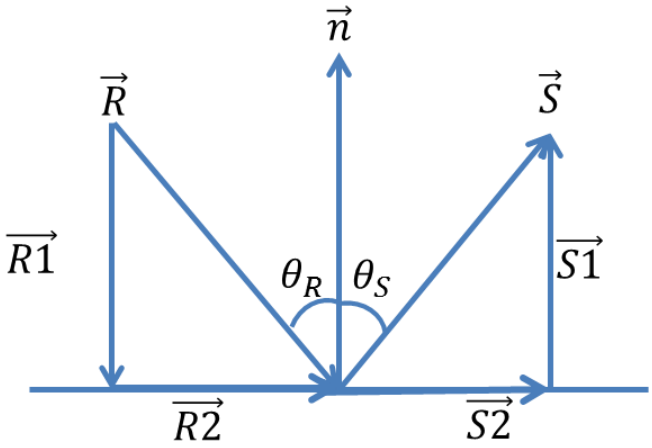


Figure 4.9: Reflection vector

$$\overrightarrow{R1} = (\vec{R} \cdot \vec{n}) \cdot \vec{n}$$

$$\theta_R = \theta_S$$

$$\overrightarrow{R1} = -\overrightarrow{S1}$$

$$\overrightarrow{R2} = \overrightarrow{S2} = \vec{R} - \overrightarrow{R1}$$

$$\vec{S} = \overrightarrow{S1} + \overrightarrow{S2} = -\overrightarrow{R1} + \overrightarrow{R2} = \vec{R} - 2\overrightarrow{R1}$$

$$\vec{S} = \vec{R} - 2(\vec{R} \cdot \vec{n}) \cdot \vec{n} \tag{4.12}$$

Where $\vec{R}$ is the incident vector, $\vec{n}$ is the normal vector of the intersection plane, $\vec{S}$ is the specular reflection vector.

As for the diffuse reflection, there are many diffuse reflections scattered at many angles. However, as stated at the beginning of this chapter, this program only considers the diffuse reflection ray that returns to the scanner, it has the opposite direction of the incident ray, as Figure 4.10 shown.

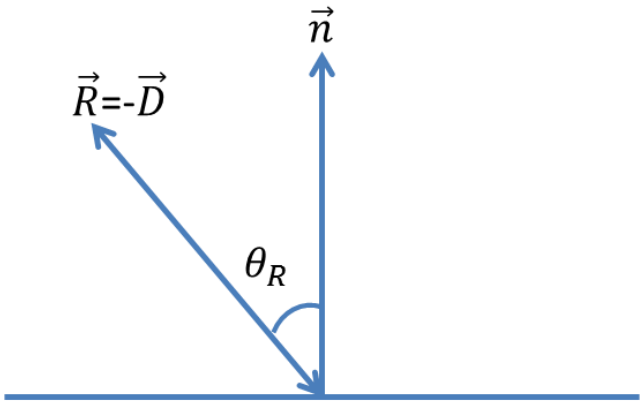


Figure 4.10: Diffuse reflection Vector

$$\vec{R} = -\vec{D} \tag{4.13}$$

The initial energy carried by the beam is constant so that the energy carried by each ray is the same. The energy losing happened in two cases. The first case is during the propagation of the water, and the remaining energy is given by the equation below [132]:

$$Energy_{remained} = Energy_{start} * \exp(-distance_{receiver} * \gamma) \tag{4.14}$$

Where γ is the energy dissipation index in the water during propagation and $distance_{receiver}$ is the travelled distance.

The second energy losing case is the intersection. Energy is significantly reduced due to impinging. The remaining energy is given by the equation below [132]:

$$Energy_{remained} = Energy_{start} * (1 - \sigma)^n \tag{4.15}$$

σ is the energy dissipation index due to the intersection. Assume the energy dissipation index is 50% for specular reflection and 90% for diffuse reflection. n is the order of reflections.

The remaining energy at the receiver calculated by the combination of equation 4.14 and 4.15 is given below [132]:

$$Energy_{receiver} = Energy_{start} * \exp(-distance_{receiver} * \gamma) * (1 - \sigma)^n \tag{4.16}$$

Assume the energy threshold for stopping tracing the ray is 1/11 of the initial energy; this is made to reduce the complexity of calculation and ray propagation. Based on this assumption, in mathematics, for specular reflection ray, it disappears after four specular reflections, as for diffuse reflection ray, it disappears after one diffuse reflection. Since it assumes that pond

made up of acoustically hard surfaces, specular reflections will overwhelm diffuse reflections. A simulation was already shown in section 3.7.2.

4.8 Summary

This chapter explained how to implement the ray-tracing aerial survey in MATLAB. The key elements are the reflection of the ray, the reception of the ray, and the energy dissipation of the ray. A ray-tracing aerial mapping simulator was successfully built. The next chapter will use the aerial mapping simulator to study different pond simulations.

Chapter 5

Aerial Mapping Simulations and Experiment

5.1 Introduction

The previous two chapters explained “aerial mapping” and how a ray tracing aerial mapping simulator was developed in MATLAB. This chapter uses aerial mapping simulator to study its mapping performance in different pond cases. Initially, the effect of scanning resolution on scan quality is investigated. Then, the impact of scanning height is also investigated. After that, several test cases representative of real storage ponds are then presented.

5.2 Simulation Assumptions

The following assumptions and configurations apply to the simulations in this chapter:

- The pond is a 50 m×25 m swimming pool-sized enclosure with a depth of 10m.
- The scanner is a 0.025 m radius sphere transceiver.
- The scanning beam angle is 20 °.
- The energy threshold for stopping tracing the ray is 1/11 of the initial energy.
- The energy dissipation caused by specular reflection is 50%, diffuse reflection is 90%.
- The sampling resolution is defined as the interval between two consecutive sample points, so a smaller value means a higher resolution because there will be more sample points in the same area. It is written as (**a**×**b**), where **a** is the separation distance of the X-axis and **b** is the separation distance of the Y-axis, assuming **a** = **b** in the simulation.
- The surface plot function is used. The surface plot function of the MATLAB using Delaunay triangulation function [137] to present the mapping result visually, which creates a continuous surface based on the data obtained at discrete sampling points
- All pond clutters are assumed prismatic.

5.3 Sampling Resolution and Accuracy

A simple case with two different shaped canisters is used to investigate the effect of sampling resolution on scan results. It is a simple representative of two adjacent canisters with upright orientation. This consists of a hexagonal prismatic object and a cuboid object of the same height of 5m placed closely in the middle of the pond, as shown in Figure 5.1. The hexagonal prismatic object centred at (25, 13.5) and the cuboid object centred at (29, 13.5), the distance between them is 0.75m. Aerial mapping samples (black dots in Figure 5.1) cover the area of two containers from (20, 10) to (33, 18).

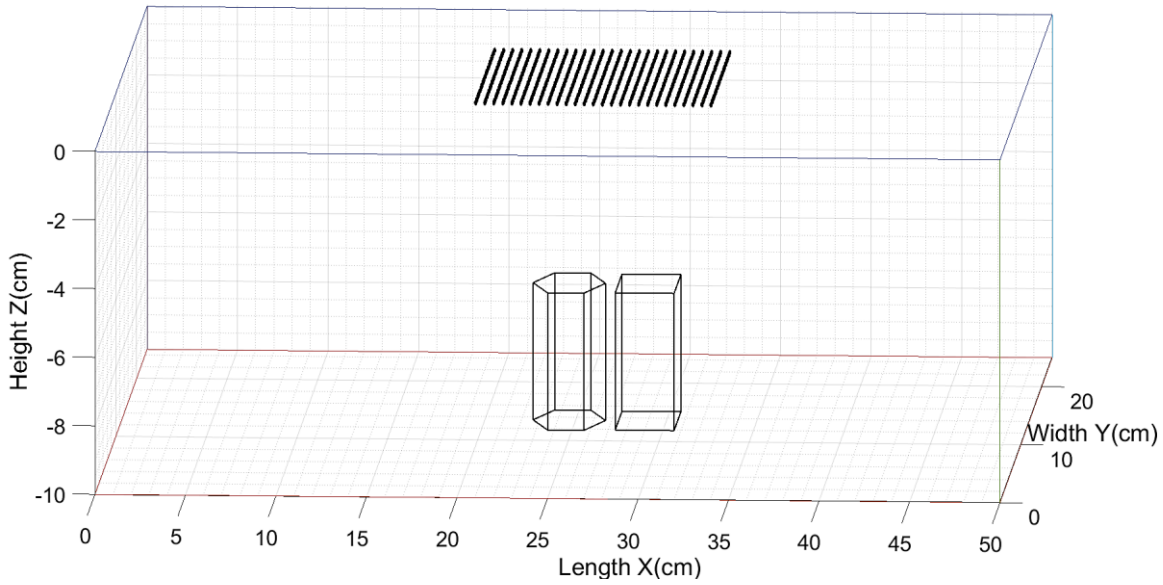
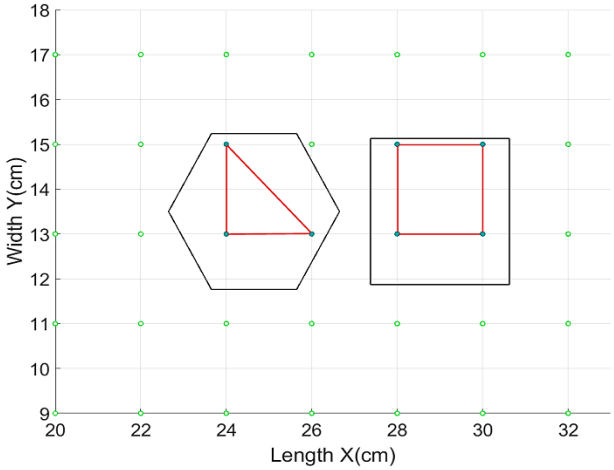
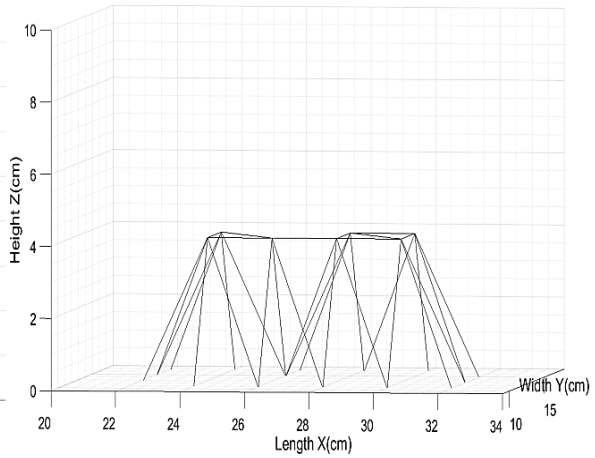


Figure 5.1: 2 different shaped containers

Surveys were carried out with different sampling resolutions. Figure 5.2 to Figure 5.6 show survey results for 2 m×2 m, 1 m×1 m, 0.5 m×0.5 m, 0.25 m×0.25 m, and 0.1 m×0.1 m resolutions, respectively.

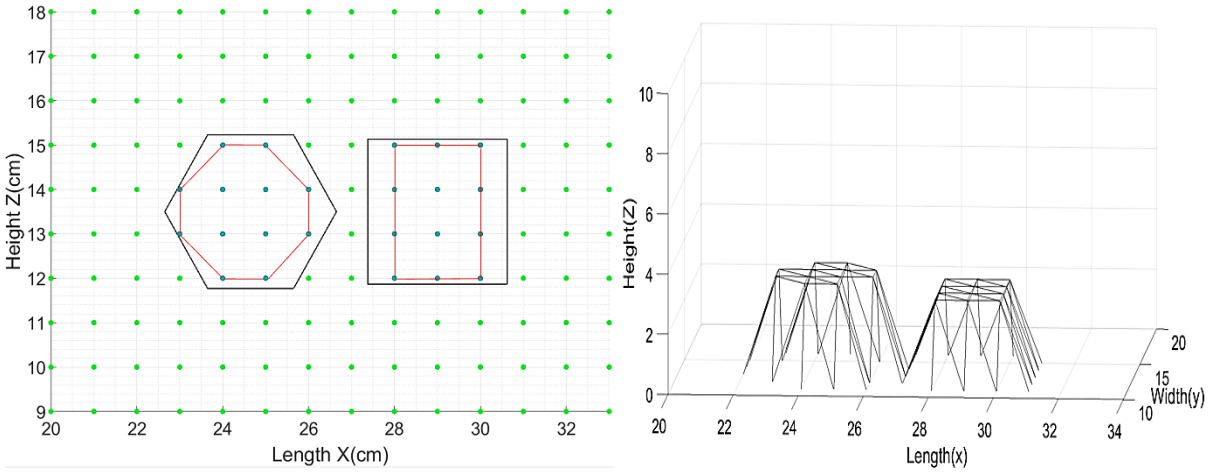


(a) Sampling pixel



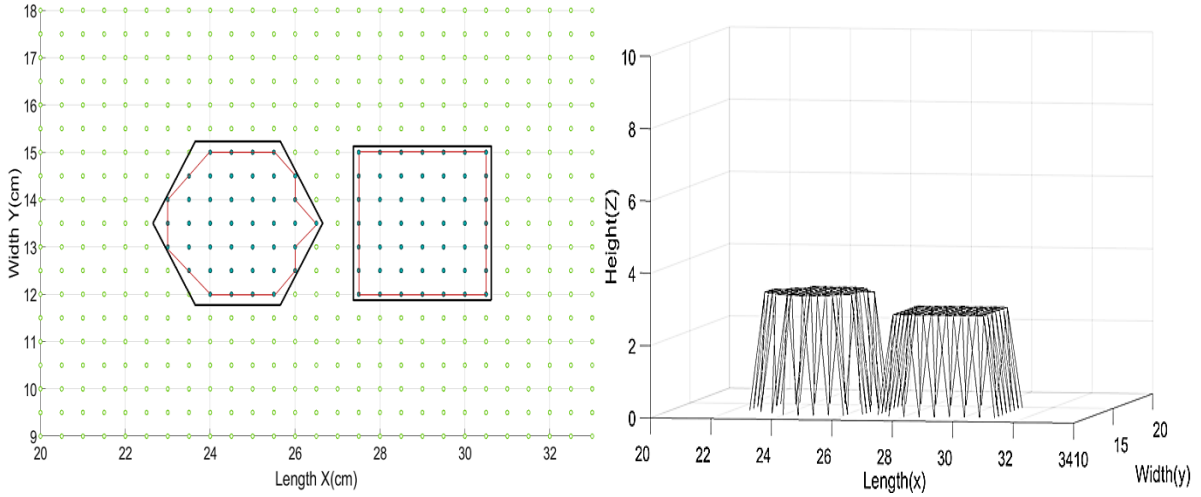
(b) Surface plot reconstructed object

Figure 5.2: Simulation result under a sampling resolution of 2 m×2 m



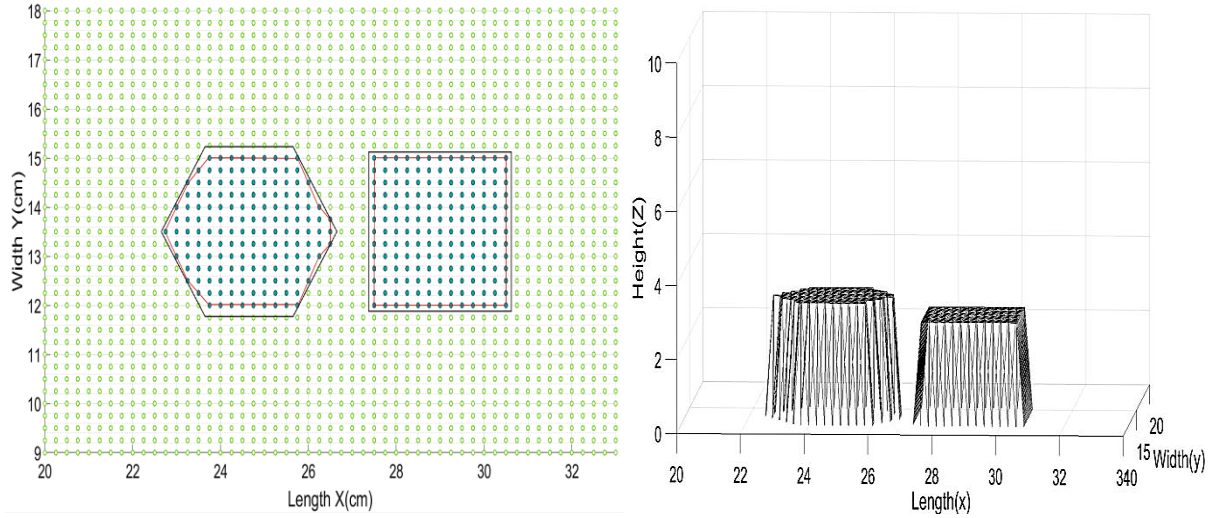
(a) Sampling pixel (b) Surface plot reconstructed object

Figure 5.3: Simulation result under a sampling resolution of 1 m×1 m



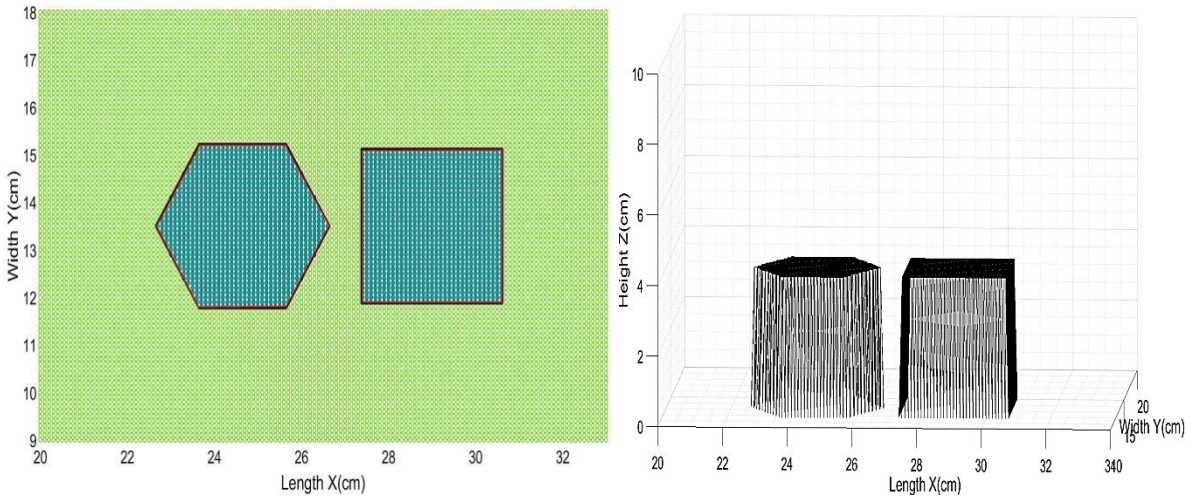
(a) Sampling pixel (b) Surface plot reconstructed object

Figure 5.4: Simulation result under a sampling resolution of 0.5 m×0.5 m



(a) Sampling pixel (b) Surface plot reconstructed object

Figure 5.5: Simulation result under a sampling resolution of 0.25 m×0.25 m



(a) Sampling pixel (b) Surface plot reconstructed object

Figure 5.6: Simulation result under a sampling resolution of 0.1 m×0.1 m

The darker green dots are depth measurement points of the upper surface of objects, and the lighter green dots are depth measurement points of the pond ground. As shown in Figure 5.2 to Figure 5.6, with the increment of the sampling resolution, the boundary of the top surface's measurement point (red line) is getting coincide with the boundary of the object (black line), and the corresponding surface plot is getting closer to the object (the sketched shape changed from a triangular prism to a hexagonal prism). Because the true edge of the object must lie between two adjacent sampling pixels, if the spacing between the two consecutive sampling pixels is reduced, the measured edge is closer to the real edge. Increasing the sampling resolution will also add more sampling points to the object area and get more detailed information. The only problem is that the time consuming increasing exponentially with an increment of the sampling resolution. However, it is not a big problem, because the mapping is underwater offline, so the time consuming is not a concern.

It is important to note that the sloping sides are a function of the surface plot drawing algorithm not part of the mapping algorithm (this applies to all surface plot results), and there is no data on the sloping sides. Hence, it requires further processes to the aerial mapping data for the sloping sides. However, the work about how to deal with the sloping side is presented in Section 8.2.3.

In addition, accuracy can be quantified by comparing the position of the measured and true centres of gravity of the object, and the area of the measured top plane and that of the true top

plane. The function that used to calculate the centre of gravity of the upper surface is expressed in Equation (5.1):

$$O(x,y) = \frac{\sum_{i=1}^{n}(x_i,y_i)}{n} \quad (5.1)$$

- O is the centre of gravity point
- n is the number of sampling points, and
- (x_i, y_i), i=1, 2, … , n, is coordinates of each sample.

The centre of gravity with different sampling resolutions for the hexagonal prismatic object (left-hand side object in Figure 5.1). The true centre of gravity location is (25, 13.5). The offset distance is the shifted distance between the centre of gravity of the measured object (calculated by Equation 5.1) and the true object. The variation of offset of the central centre of gravity with respect to the sampling resolution for the hexagonal prismatic object is shown in Figure 5.7.

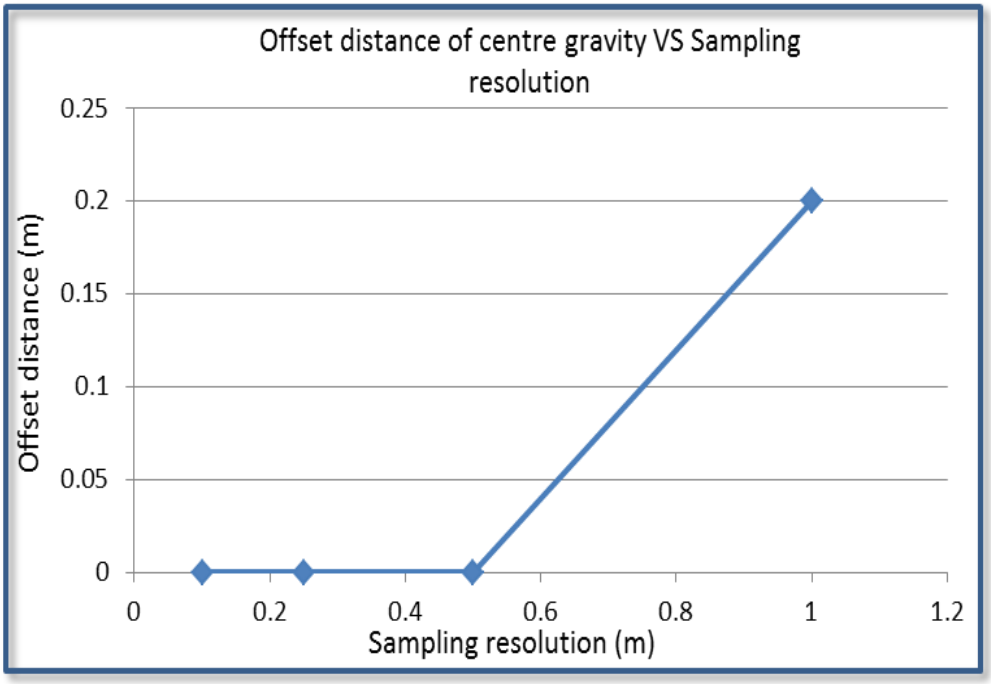


Figure 5.7: The offset distance with respect to sampling resolution

The offset distance remains unchanged until the sampling resolution is (0.5 m×0.5 m), so the measurement object will not shift at a sampling resolution of 0.1 m×0.1 m, 0.25 m×0.25 m and 0.5 m×0.5 m. If the sampling resolution is too small, for example, 1m×1m, the entire measured object drifts, causing an inaccurate mapping result.

The area of the measured top plane is given by the Surveyor's Area Formula [53]:

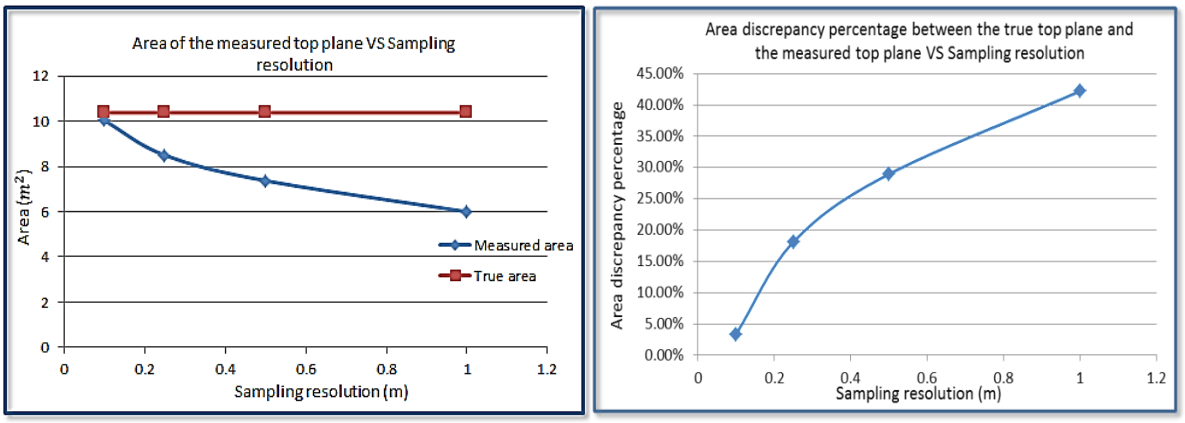
$$A = \frac{1}{2}\left|\sum_{i=1}^{n-1} x_i y_{i+1} + x_n y_1 - \sum_{i=1}^{n-1} x_{i+1} y_i + x_1 y_n\right| \quad (5.2)$$

- A is the area of the polygon (i.e. the surface),
- n is the number of sides of the polygon, and
- (x_i, y_i), i = 1, 2,..., n, are the coordinate of vertices (or "corners") of the polygon.

To calculate the measured area of the top surface, it needs to extract the measured boundary points of the top surface and regard them as vertices (refer to boundary tracing in Section 8.2.1). Substituting each boundary point into Equation (5.2), n is the number of boundary edges, and (x_i, y_i) is the x, y coordinate of each boundary point. Therefore, the area of the measured top surface can be calculated. In the current example, the true area of the top surface is 10.38m^2. The difference in the area between the true top plane and the measured top plane is given by:

$$area\ discrepancy = \frac{|measured\ area - true\ area|}{true\ area}\% \quad (5.3)$$

The plot of area discrepancy between the true top plane and the measured top plane against the sampling resolution is shown in Figure 5.8.



(a) (b)

Figure 5.8: (a) Area of measured to plane against sampling resolution; (b) Area discrepancy in percentage against sampling resolution

The discrepancy between the true and measured top surface areas with low sampling resolution is large in numerical terms. The plot of the area discrepancy against the sampling resolution also illustrates the fact that the higher the resolution, the smaller errors. This error graph can be used to find which accuracy range corresponds to which sampling. If a specific

error margin is required, the error graph can be used to determine the desired sampling resolution is required. Therefore, select the appropriate sampling resolution.

Above study find out that a higher sampling resolution would cause a better result. Simulations conducted in this thesis are undertaken by the (0.5 m×0.5 m) resolution.

5.4 Multi-Resolution Survey

Higher sampling resolution will get a more detailed and accurate result, but some areas of the pond are devoid of clutter, while, other areas accumulate of clutter, so applying one sampling resolution is not adequate. One way to solve this problem is to implement hierarchical multi-resolution sampling [54]. For instance, perform high-resolution sampling in the obstacle regions and perform low-resolution sampling in non-obstacle areas.

As the name suggests, the multi-resolution survey has at least two survey phases. In the first phase, a whole environmental scan is conducted at a low sampling resolution (**a**×**a**), and the matrix of the obtained result is A. After the first phase survey, the obstacle regions and non-obstacle regions can be identified from the result. In the second phase, the identified obstacle regions will be scanned by a high sampling resolution (**b**×**b**), and the resultant matrix is B. Matrix A and B contribute to a completed hierarchical depth measurement data of this environment. An example of multi-resolution sampling is shown in Figure 5.9.

Figure 5.9(a) shows the depth measurement points for the entire environment with a low sampling resolution (1m×1m in this case). Figure 5.9(b) shows the depth measurement points for the obstacle region with a higher sampling resolution (0.5m×0.5m). Figure 5.10(c) shows depth measurement points for the entire environment with both high and low sampling resolutions, by combining the point clouds in Figure 5.9 (a) and Figure 5.9 (b). A detailed description of how to implement the multi-resolution survey will be found in Chapter 9.

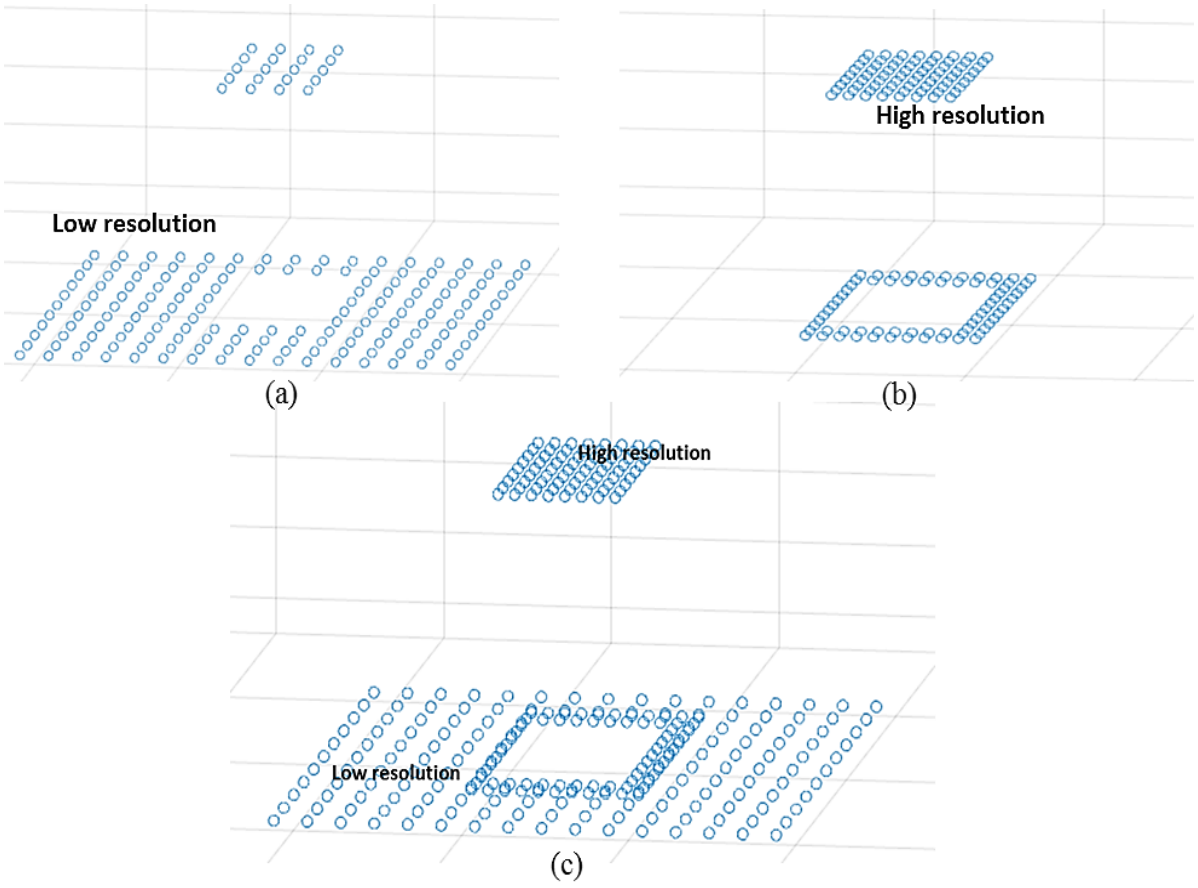
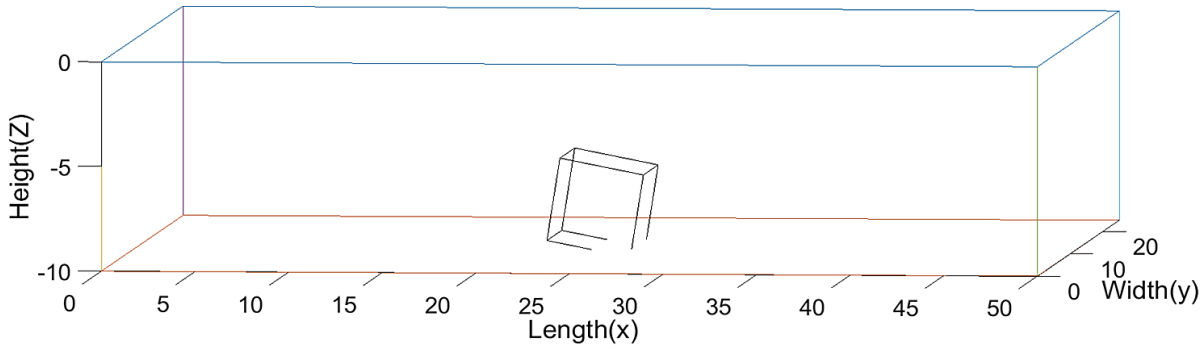


Figure 5.9: (a) Depth measurement points of low sampling resolution; (b) depth measurement points high sampling resolution; (c) depth measurement points that contains both low and high sampling resolution

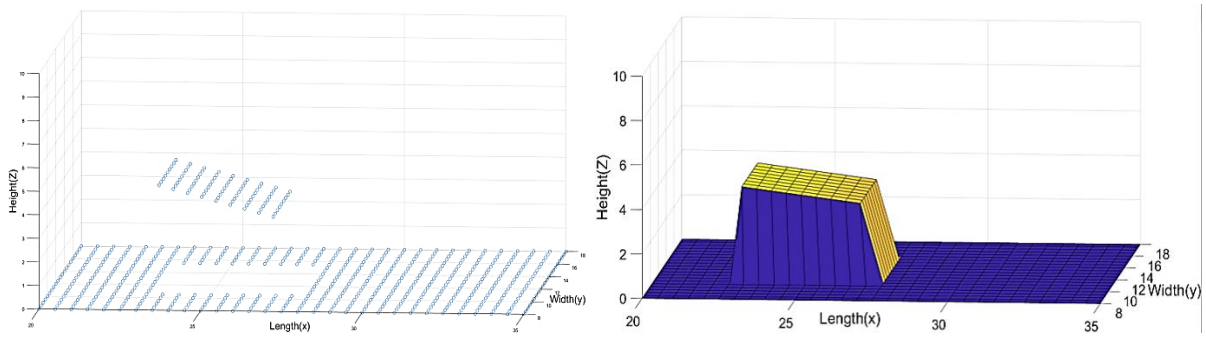
5.5 Tilted Containers

Aerial mapping results of non-tilted used fuel canisters are shown above. However, many used fuel canisters are not well placed in legacy ponds, and some may be randomly tilted at different angles in practice. This section aims to study and discuss the results of an aerial survey of tilted containers.

The simulation below shows the aerial survey results of objects with tilt angles varying from 10° to 30°. The point cloud data and surface map of the aerial survey for each tilt angle is shown in Figures 5.10 to 5.12, respectively.

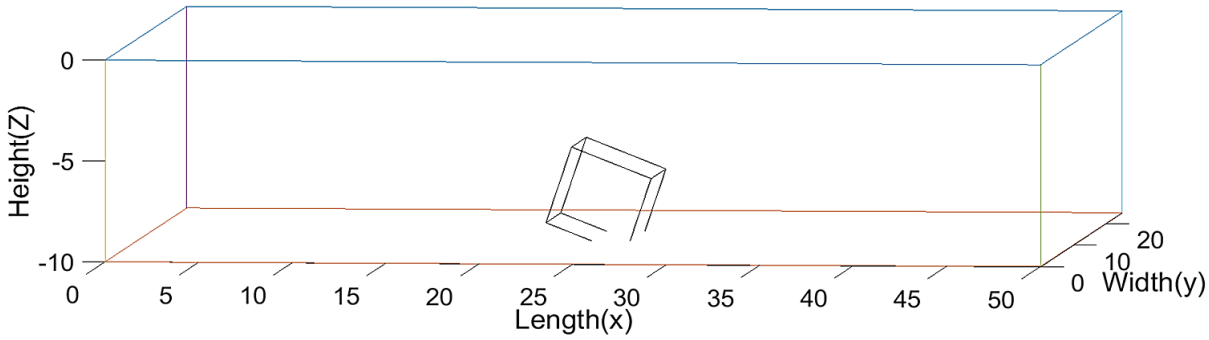


(a)

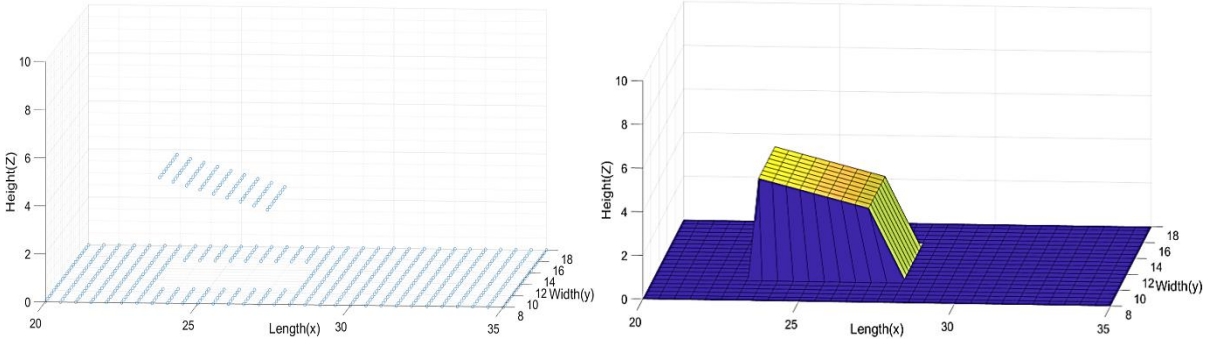


(b) (c)

Figure 5.10: (a) MATLAB geometry input for the tilt angle of 10°; (b) depth measurement points; (c) surface plot reconstructed object

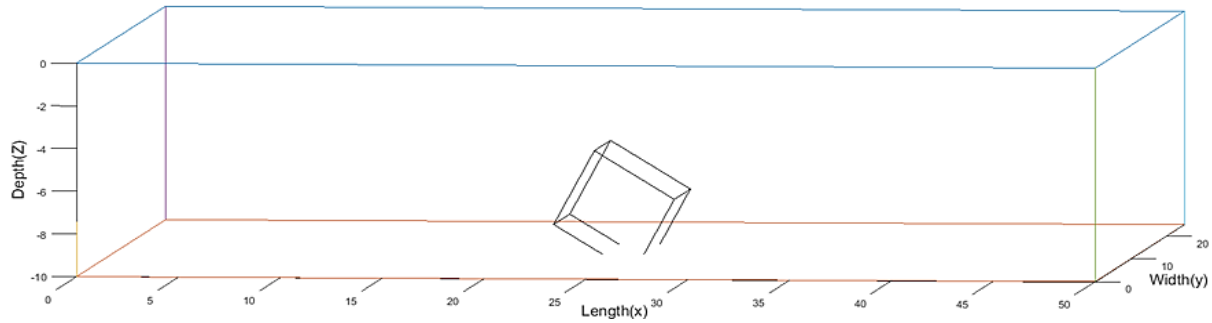


(a)

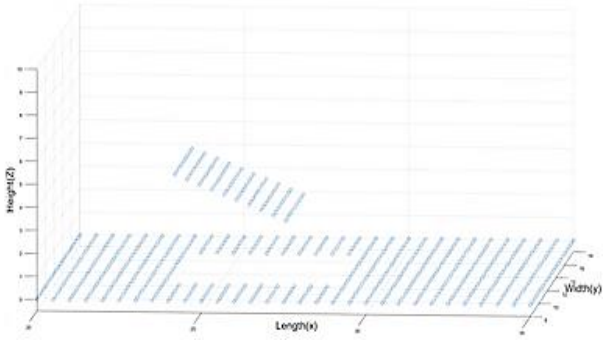


(b) (c)

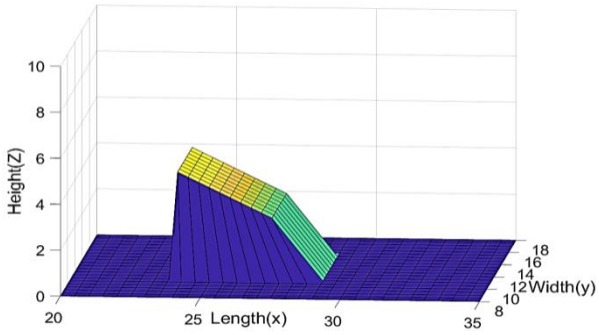
Figure 5.11: MATLAB geometry input for the tilt angle of 20°; (b) depth measurement points; (c) surface plot reconstructed object



(a)



(b)



(c)

Figure 5.12: (a) MATLAB geometry input for the tilt angle of 30°; (b) depth measurement points; (c) surface plot reconstructed object

The surface plot shows that as the tilt angle increases, the ramp that connects the detected plane (top surface of the obstacle) and the ground (bottom of the pond) becomes less steep. The ramp is an artefact of the rendering method, which offers how much information is unknown. Ideally, the side surface is perpendicular to the measured plane (as it assumes clutter is prismatic refer to Section 3.4), so the information between the measured plane and the ground cannot be obtained, see the diagram explanation in Figure 5.13.

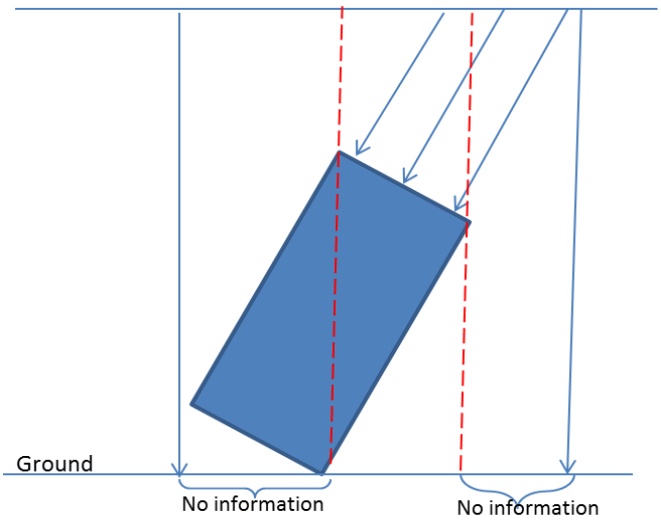


Figure 5.13: Indication of detected and undetected area

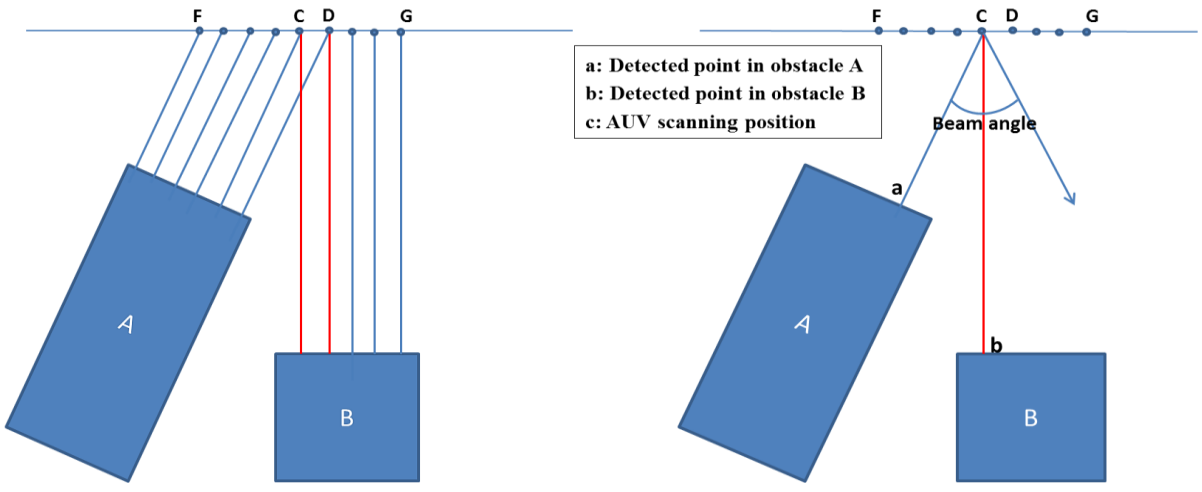
Based on the normal reflection assumption and the vertical reflection concept, the right-hand side “No information” area cannot be measured because the scanner in the position above the “No information” area will measure the tilted top surface of the obstacle. The left-hand side “No information” area cannot be measured because the scanner cannot receive a normal reflection from the object’s side surface. This problem also appears in section 5.6. However, in the later data processing of the depth measurement data, the “No information” area will be filled with new point cloud data by the perpendicular point interpolation algorithm. See section 8.2.3 for details.

5.6 Special Case Study

It uses the depth measurement data obtained by aerial mapping to determine the shape and layout of obstacles. However, some layouts and shapes are hard to determine. Such cases are discussed below.

5.6.1 Case 1: Detect Multiple Obstacles in One Scan

The proposed aerial mapping process will use sensors with a finite beam angle, as discussed in section 3.7.3. Since the detection principle relies on the assumption of the vertical reflection and beam angle (refer to section 3.7.3), if the beam angle is wide, it can detect multiple obstacles in one measurement. This case is shown in Figure 5.14 below.



(a) Scan from F to G (b) Scan at C

Figure 5.14: Scene of case 1

Assume the scanner has a beam angle of 20°. A and B are two objects placed at the bottom of the pond, and object A leans towards object B about 20°. Scans were taken from point F to

point G. The beam projected by the scanner at **C** can detect both point **a** of obstacle A and point **b** of obstacle B because ray ***Ca*** is perpendicular to object A, and ray ***Cb*** is also perpendicular to object B. In this example, the length of ***Ca*** is shorter than the length of ***Cb***, so based on the early reflection assumption, obstacle A will be regarded as the detected object rather than B. This is obviously a problem, since it appears that the depth of the obstacle vertically below ***C*** is ***Cb*** rather than ***Ca***. A simulation scene of this case is shown in Figure 5.15.

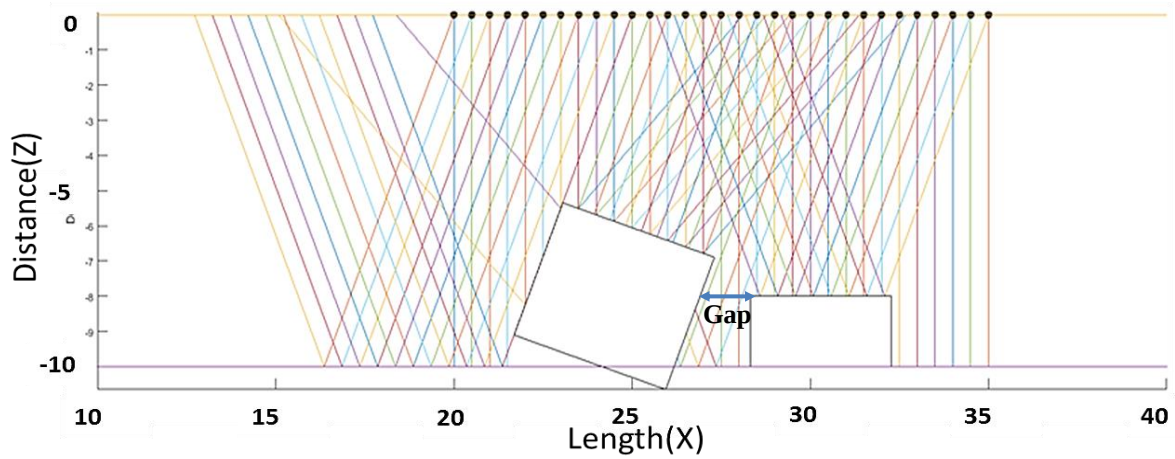
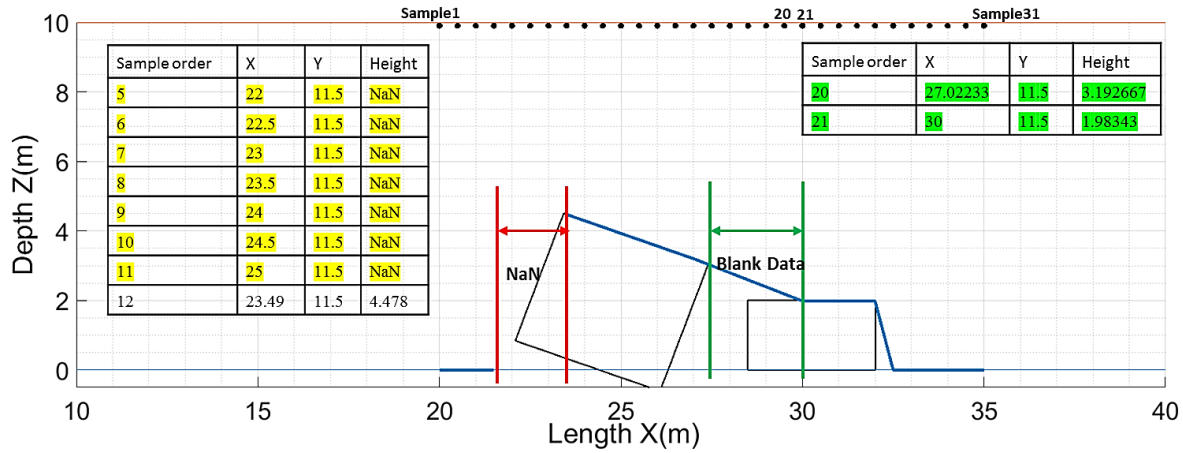


Figure 5.15: Aerial mapping scene for case 1

In this simulation, the scanner (20 ° beam angle) scans along the x-axis from left (x=20 m) to the right (x=35 m) with a sampling interval of 0.5m. There are 31 sampling points in total. The left-hand side object tilts 20 ° to the right-hand object. The gap between them is 1.07 m. The obtained depths and coordinates of each sample from left to right in orders appear in a table in Appendix B, some key data is indicated in Figure 5.16.



Sample order	X	Y	Height
5	22	11.5	NaN
6	22.5	11.5	NaN
7	23	11.5	NaN
8	23.5	11.5	NaN
9	24	11.5	NaN
10	24.5	11.5	NaN
11	25	11.5	NaN
12	23.49	11.5	4.478

Sample order	X	Y	Height
20	27.02233	11.5	8.192667
21	30	11.5	1.98343

Figure 5.16: Surface plot of aerial mapping result for case 1

As shown in Figure 5.16, the blue line is the depth measurement results of each sample; the black rectangles represent actual objects. The obtained survey results differ from the true

object distribution significantly. This shows the multiple obstacles detection problem stated earlier, half of the right-hand non-tilted object and the gap between the two objects (Blank data) cannot be detected, and the side surface (NaN discontinuous part) cannot be observed (there is no height information from sample 5 to 11 because the reflection at the side surface cannot be directly received by the scanner). The other interesting sample points are sample 20 and 21, at sample 20, the scanner detects the point (27.022, 11.5, 3.192) on the left-hand side object, while at sample 21, the scanner detects the point (30, 11.5, 1.983), this shows the information between x=27.022 m to x=30 m is unable to be obtained. This simulation illustrates the shortcoming of aerial mapping: the aerial survey cannot distinguish two adjacent obstacles if they are close together and one of them is leaning against the other. Varying the height of the aerial survey movement plane can be used to solve or mitigate this shortcoming. The next section will discuss the effects of the height of the survey plane on the results. As for the side surface data, it will not be discussed here, because it relates to the map reconstruction in the path-planning, those details will be discussed in Section 8.2.3.

5.6.2 Varying the Height of the Survey Plane

The simulations below aim to study how the height of the survey plane affects the mapping results. Take the same case that one obstacle leans against the other at an angle of 20 ^{o} shown in Figure 5.14 as the example. The height of the measurement plane varies from 10m to 5m with a descending step of 1m. Figure 5.17 to 5.21 show the aerial mapping simulation results for each height.

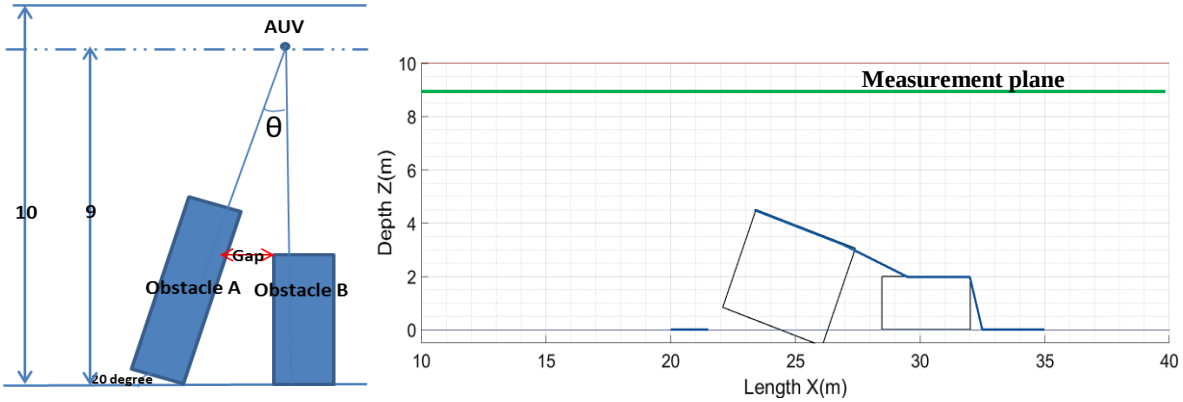


Figure 5.17: Scanning height is 9 m

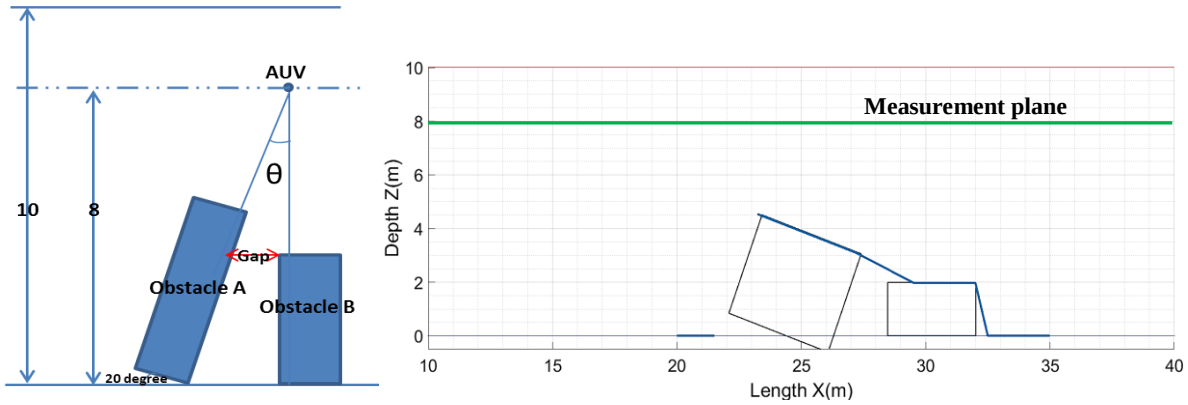


Figure 5.18: Scanning height is 8 m

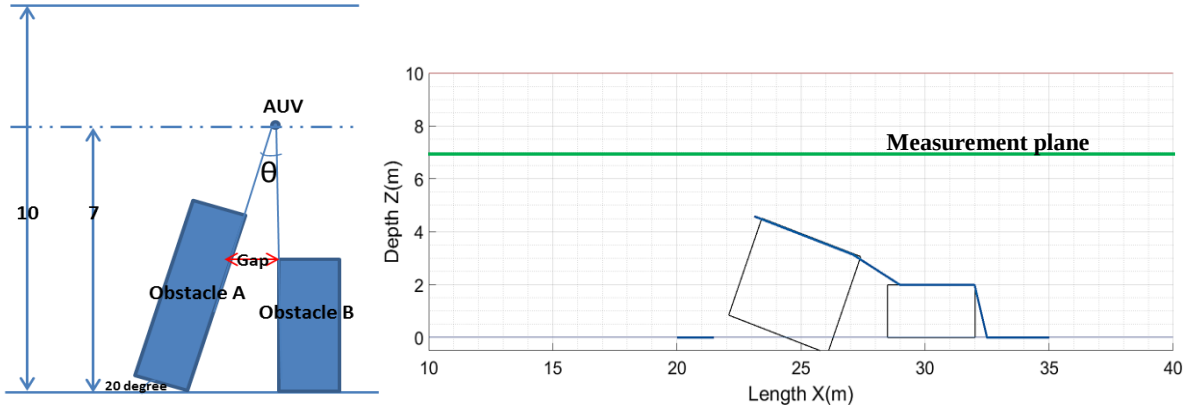


Figure 5.19: Scanning height is 7 m

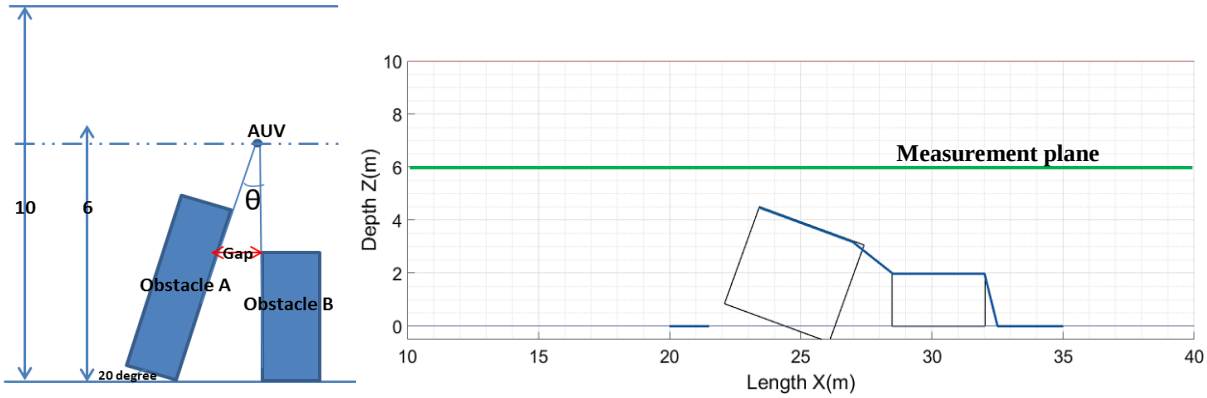


Figure 5.20: Scanning height is 6m

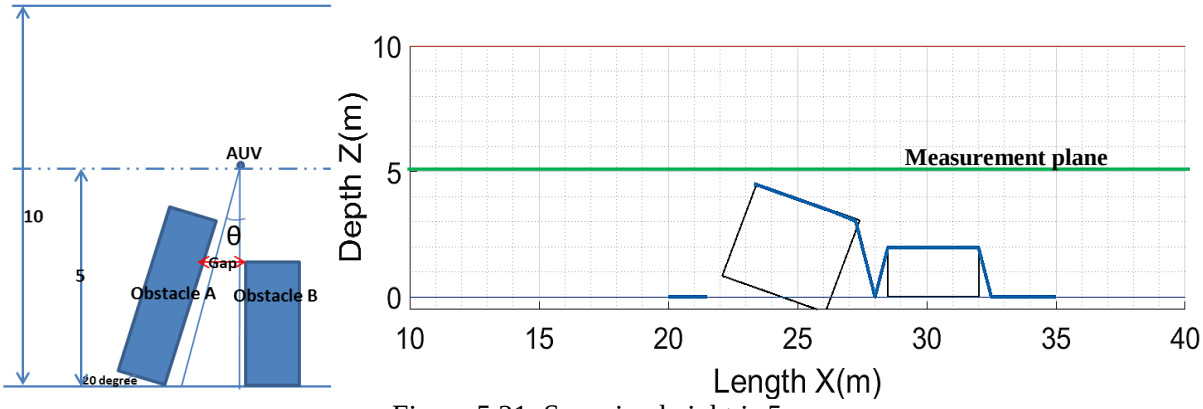


Figure 5.21: Scanning height is 5 m

Left-hand diagram of each figure is the scanning scene. The blue line in the right-hand figures represents the depth measurements of each sample. The green line is the measurement plane. The black block represents the true object. As the measurement plane decreases, the blank data range getting smaller and smaller. The case of scanning height is 5 m can even detect the small gap between the two close obstacles. The above results show that the measurement plane can obtain a more accurate result if it is close to the detected object. This suggests that scanning should fairly close to the objects, all other things being equal. The practical experiments in Section 5.10 prove this.

5.7 Well-Structured Modern Nuclear Storage Ponds

This section aims to study the mapping result of a modern storage pond. As discussed in Section 1.1, spent fuel containers in modern nuclear storage pond are well-organised and placed in a regular structure, so assumes the testing pond case has 32 spent fuel containers, and the dimension of each container is 4 m×4 m×5 m. They are arranged in four rows, eight per row, with a gap of 2m between each. See Figure 5.22 for the MATLAB geometry model of the assumed storage pond.

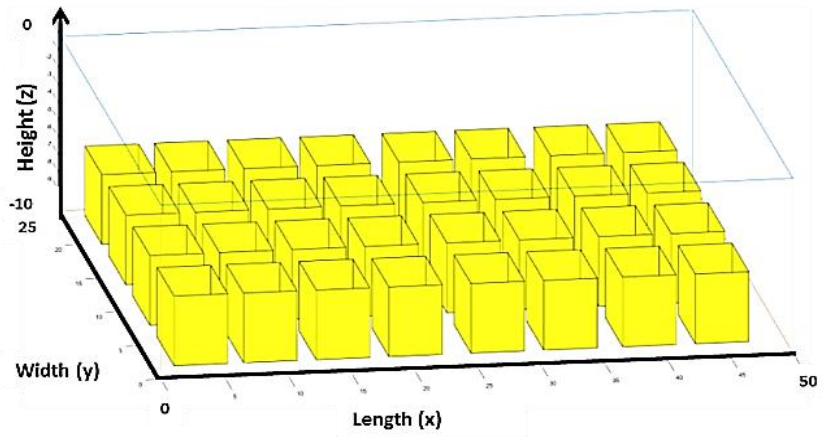
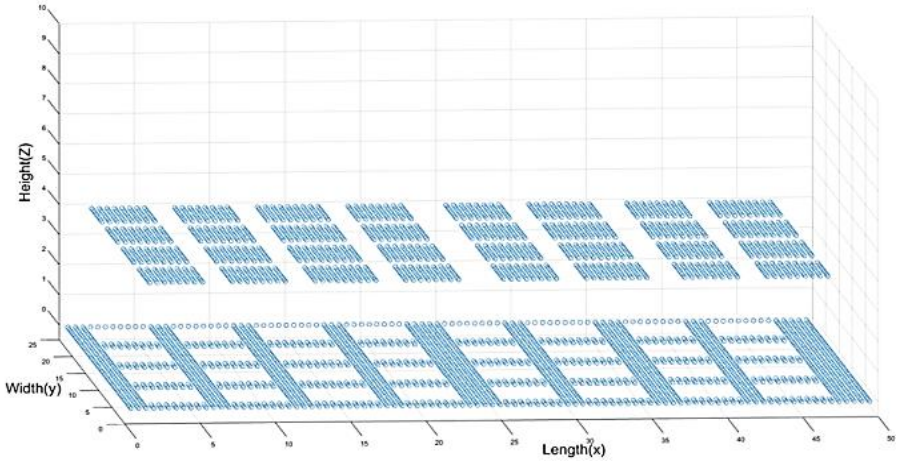
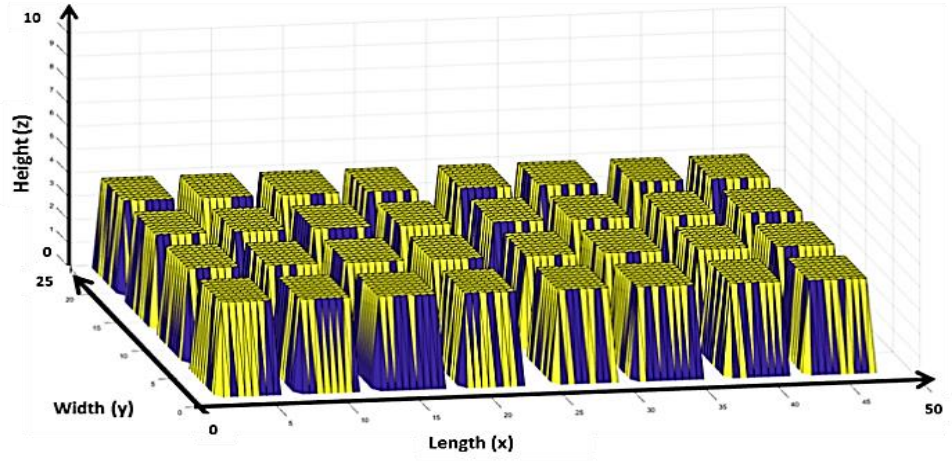


Figure 5.22: MATLAB model

Assume the aerial survey sampling resolution is 0.5 m×0.5 m (a discussion of sampling resolution is given in section 5.3). The obtained topological point cloud data and its corresponding surface plot are shown in Figure 5.23(a) and 5.23(b), respectively.



(a) Depth measurement points



(b) Environment reconstruction by surface plot

Figure 5.23: Simulation result of modern nuclear storage pond aerial survey

The upper points in Figure 5.23(a) are the information obtained on the top surface of the canisters. The lower points are the information obtained on the ground of the pond. Figure 5.23(b) is the surface plot constructed pond based on the contained information shown in Figure 5.23(a). This simulation shows an image of what information is obtained by an aerial survey. The results obtained by aerial surveys need further processing to be used for path planning to plan a collision-free path in the pond; details can be found in Chapter 8.

5.8 Hollow Containers

In practice, many spent fuel containers are open-cap and their internal materials are partially dissolved [55]. Figure 5.24 is a photograph taken from the Sellafield nuclear site, shows an old storage pond. A MATLAB geometric model (use a simple hollow geometry model to

represent a single hollow canister) of the single hollow vessel was generated to study the aerial survey of such scenarios, see Figure 5.25.

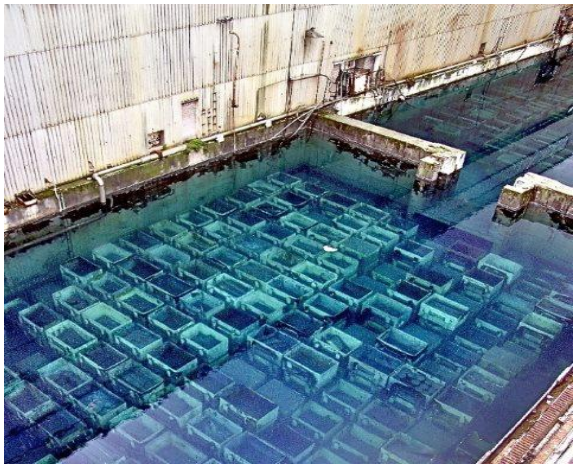


Figure 5.24: Storage pond in Sellafield [55]

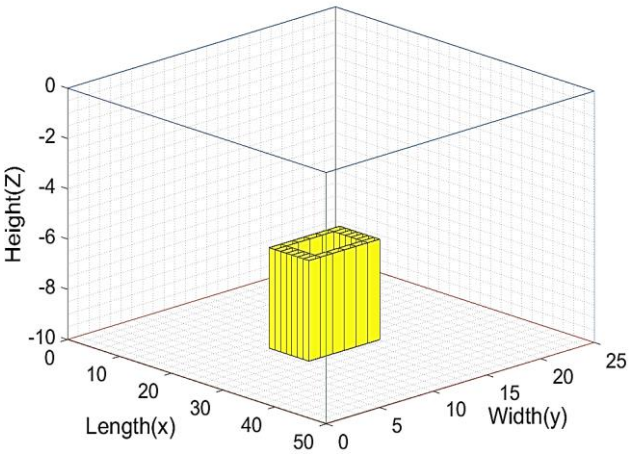
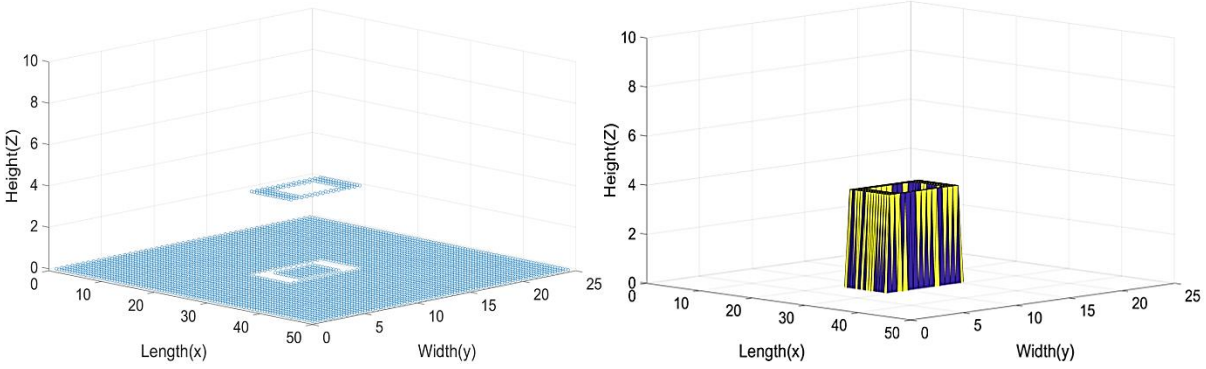


Figure 5.25: MATLAB geometry of hollow canister.

In the MATLAB geometric model, assuming the thickness of the container wall is 1.1 m, so 0.5m×0.5m sampling resolution can detect the container wall, also assuming that the dimension of this hollow container model is 8 m×8 m×4 m. The purpose of this simulation is to study mapping results for a hollow canister. The aerial survey simulation result is shown in Figure 5.26(a) and Figure 5.26(b), respectively.



(a) (b)

Figure 5.26: (a) Depth measurement points of the hollow canister; (b) The surface reconstructed object of hollow canister

In practice, the sampling resolution for mapping a hollow container needs to be considered carefully. Otherwise, it might miss the hollow canister if the sampling resolution is too small to detect the canister's wall.

5.9 Limitations of Aerial Mapping

In some special cases, aerial mapping cannot capture enough information, for example, in the case where one canister lies above another, as shown in Figure 5.27. In such cases, aerial mapping cannot collect information about the 'shadow area'. The aerial survey will regard them as one solid obstacle. See the simulation results below.

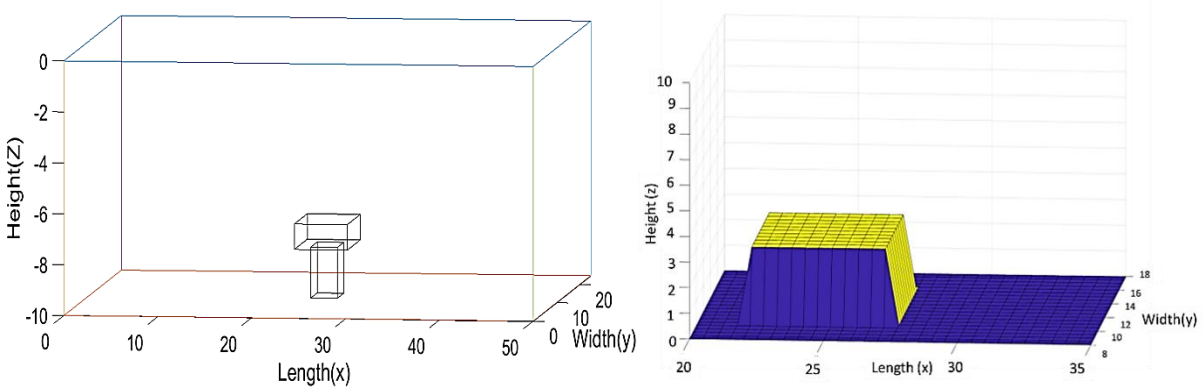


Figure 5.27: Canister lies above another

In this case, the contribution of the aerial survey is very limited, and more investigations are needed to support the establishment of a complete map, such as an online survey[2] or a vertical survey[3], which are not considered further in this thesis. However, this situation is probably unlikely to occur in most nuclear storage ponds.

5.10 Aerial Mapping Experiments

The above work is based on simulations, so some experiments must be done to study aerial mapping. As discussed in Section 3.5, since no big water tank was available, the following experiments were carried out in an open-air environment. Sound waves propagation in the air in the same way as in water (the sound is always longitudinal unless intersect with solid objects), so only the speed of sound is different, but the mode of propagation remains the same. Therefore, the open-air experiments can be approximated to the underwater experiments. The experiments aim to study the performance of aerial mapping in practice and to validate some simulations above. The experimental environment and testing rig are set as follows:

[2] Real-time survey, update map information in real-time
[3] Produce forward scans at different scanning heights

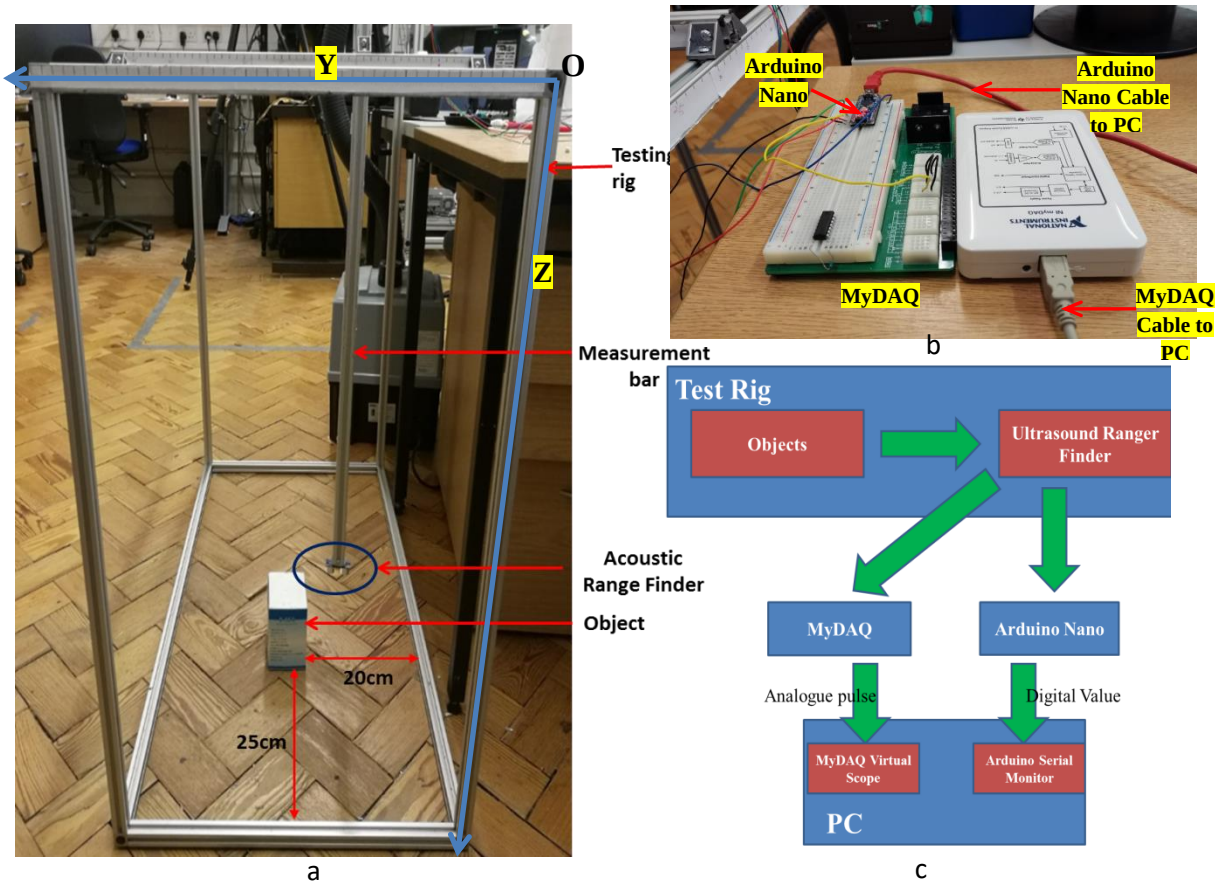


Figure 5.28: (a) Test rig; (b) instrumentations; (c) block diagram overview

5.10.1 Configurations of the Aerial Mapping Experiments

The settings of the experiment tool and testing environment are:

- The testing rig is 3d cuboid frame with open sides to avoid multi-path. Its dimension is 95×45×80 cm. The reason for choosing this size was to represent a swimming pool-sized pond on a smaller ratio (about 1/50 in size and 1/10 in depth). In Figure 5.28, the right-upper test rig corner is regarded as the original coordinate, its left joint is the Y-axis, and its forward joint is the X-axis. The measurement bar is adjustable in the X, Y, Z directions. This mechanism enables the sensor to be placed at the selected (X, Y) position of the measurement plane and also makes the height of the measurement plane adjustable.

- A cuboid object with a length of 9 cm, a width of 7 cm, and a height of 10.5 cm placed in the middle of the testing rig. The reason to choose this size is to represent a spent fuel canister on a smaller ratio (about 1/50 in size and depth). The cuboid has hard surfaces and can be regarded as an acoustic hard object.

- National Instrument MyDAQ is used. It is used as a breadboard. Through it, wires connect the output from the sensor to input pins of an Arduino Nano. Its oscilloscope channel is also used. The MyDAQ virtual oscilloscope in the desktop (PC) is used to observe the analogue output from the acoustic range finder (see block diagram in Figure 5.28).

- Arduino Nano is a micro-controller, which is used to read the output from the acoustic range finder and output a digital range value. It uses serial communication through a USB cable to the PC, so the distance can be displayed in the PC's screen (see block diagram in Figure 5.28).

- An ultrasound range finder (MB7060, HC-SR04, MB1340 are used in the following experiments) is attached to the end of the measurement bar. Characteristics of each sensor are shown in Table 5.1.

Table 5.1: Sensor parameters

Sensor Name	Beam angle (°)	Analogue/PWM output	Measure range	Error (cm)	Price(£)
MB7060	11	PWM	25cm to 765cm	1	85.12
MB1340	38	Analogue envelope	20cm to 765cm	1	32.33
HC-SR04	30	PWM	0cm to 400cm	0.3	3.76

Figure 5.29 Testing sensors

- The height of the measurement plane is 35 cm (the distance from the ultrasound range finder to the ground). Experiments about adjusting the height are described in Section 5.10.5.

- The sampling resolution is 1 cm×1 cm for following experiments because the error of MB7060 and MB1340 is 1 cm, the minimum scale of the test rig is 1 cm, and the 1cm interval is easy to set manually.

5.10.2 Experiment 1: Performance of HC-SR04

This experiment is aimed to study the mapping result of the **HC-SR04** ultrasound sensor**.** How the HC-SR04 measures the distance was already explained in Section 3.3. In this experiment, a series of distance measurements performed along the axis y = 22 cm (cross the middle of the object) from x=18 cm to x=40 cm, see Figure 5.30. This scan was intended to investigate how well the edges of objects could be resolved.

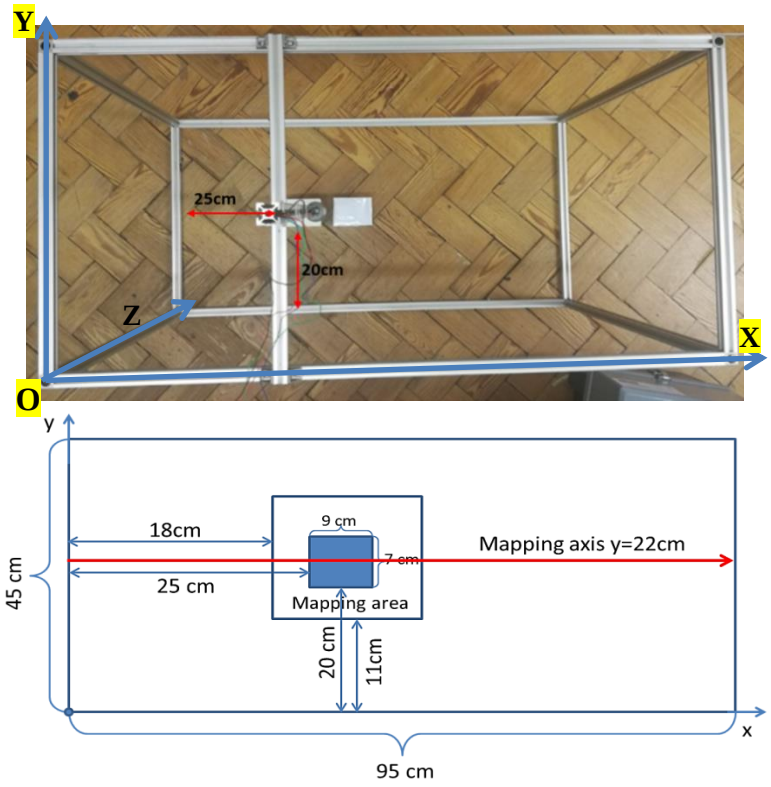


Figure 5.30: Distance measurements along the axis y = 22cm

Both simulation (the same environment is built in the aerial mapping simulator under with the same sampling resolution) and experiment result are shown in Figure 5.31.

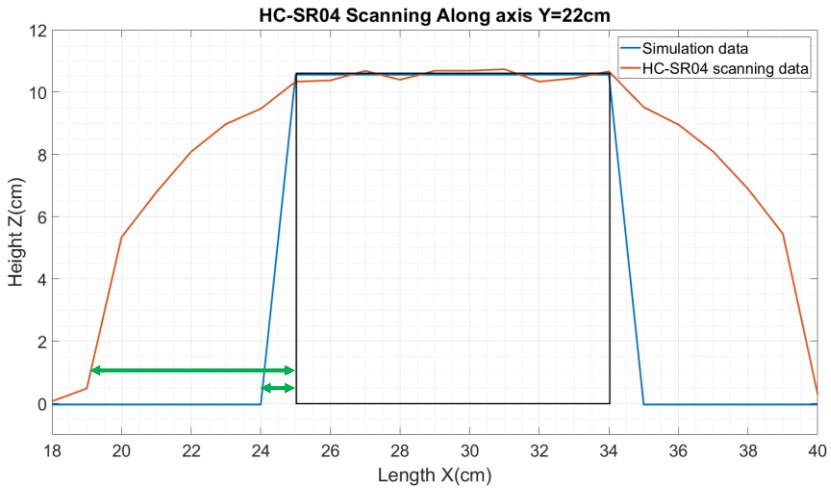


Figure 5.31: Experiment results and simulation results

The blue line is the simulation result. The red line is the measurement result. The black box is a sketch of the cross-section of the true object (or front view). The simulation result is consistent with the resolution (1 cm sampling resolution) used but the experimental result does not resolve edges well, resulting in a measured object that is much bigger than the true object.

5.10.3 Experiment 2: Study of the Reflection Pulse

The experiment conducted by the HC-SR04 ultrasound range finder above shows that the measured object is larger than the actual object. Because the HC-SR04 has a beam angle of 30 °, the reflections (back-scattering) on the edge of the object cause this problem. See the diagram below:

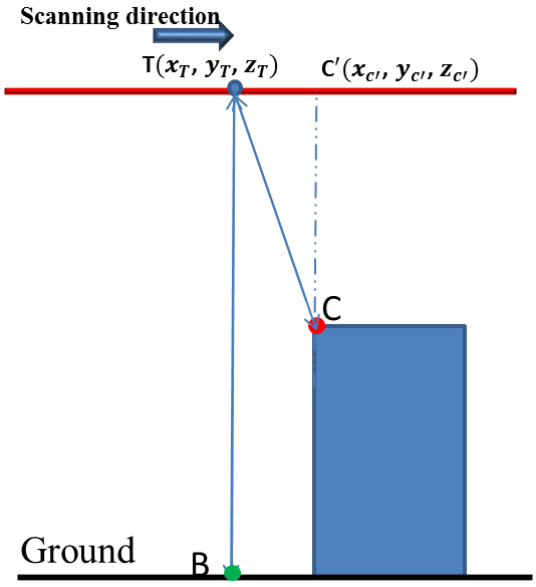


Figure 5.32: Reflections at the side surface

When the range finder scans on the measurement plane along the X-axis direction (scans from the left-hand side to the right-hand side at intervals of 1cm), there is a point where the distance from it to the ground longer than that of the distance to the object (TB<TC). Based on the early reflection (refer to section 3.7.2), before point T, the range finder will measure its distance to the ground (TB), but after point T, the range finder will measure its distance to the object (TC). This seems to depend on the beam angle of the sensor. A new experiment was designed to validate the above statement.

To study why the above case happens and why the beam angle will affect the result, the ultrasonic range finder **MB1340** whose output is the analogue envelope impulse response is used in this experiment. The reason for using **MB1340** is that it outputs the received signal rather than simply providing an edge that represents the arrival of a single reflection, so it can be used to observe each received pulse. The mechanism of how **MB1340** measures the distance was already explained in Section 3.3. In Testing Case 1 shown in Figure 5.33, the ultrasonic range finder MB1340 was placed at point A to measure the distance to the ground, and AC=35 cm. The distance to edge corner B was AB=25.58 cm. The sensor's output is shown in Figure 5.34.

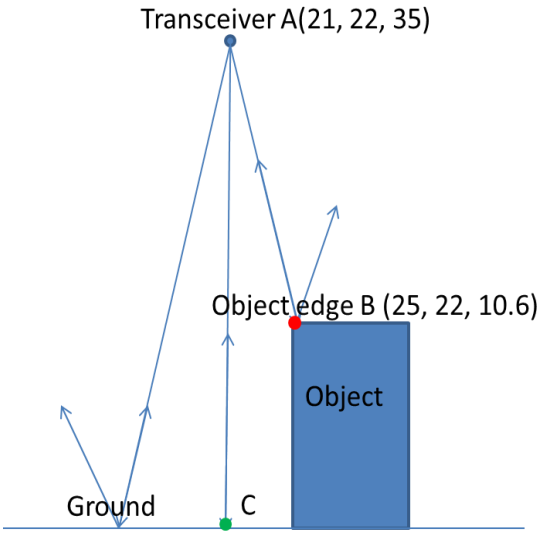
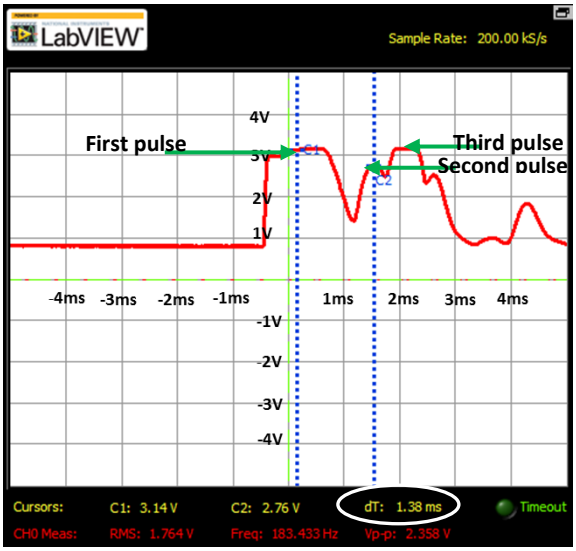
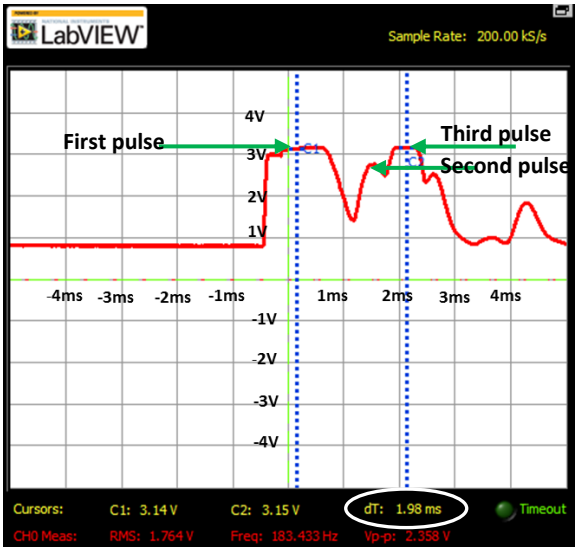


Figure 5.33: Test Case 1 configuration





(a): TOF of the first pulse Figure (b): TOF of the second pulse

Figure 5.34: Analogue envelope of MB1340 for test case 1

There are three pulses on the scope. The first pulse is the source burst. The second pulse is the signal reflected from the object because the distance from the sensor to the object is shorter than that to the ground. The third pulse is the normal reflection from the ground because this signal has the highest impulse response, which is the desired pulse. dT on the above graph is the time of flight (TOF) of the returned pulse, which is the time interval between the two blue dashed line cursor. dT is measured by two dashed line cursors, the first dashed line cursor is placed in the middle of the source pulse, the second dashed line cursor is placed in the peak of the corresponding returned pulse. The measurements are taken from the middle of the source burst's waveform rather than the rising edge of the source burst. Because several tests show that measuring from the rising edge gets a longer distance than the true distance, while measuring from the middle of the source burst's waveform gives the right value, so it is reasonable to measure from the middle of the source burst's waveform, please see the tests recorded in Appendix C.

In test Case 1 of experiment 2, **dT** is 1.38ms for the second pulse and 1.98 ms for the third pulse, respectively. The measured distance between the sensor and the object is 23.67 cm, and the distance between the sensor and the ground is 33.96 cm. On the basis of these results, it is found that the early reflection near the edge of the obstacle comes from the objects rather than the ground, so the scanner has begun measuring the object's depth before it is vertically above the object. Therefore, the measurement is much bigger than the true object. To avoid this, it requires a small beam angle, and the study of the beam angle can be found in Section 5.10.4.

In test case 2 below, the ultrasonic range finder MB1340 was placed at G to measure the distance to the object, GD=24.4cm, see Figure 5.35. MB1340's output is shown in Figure 5.36.

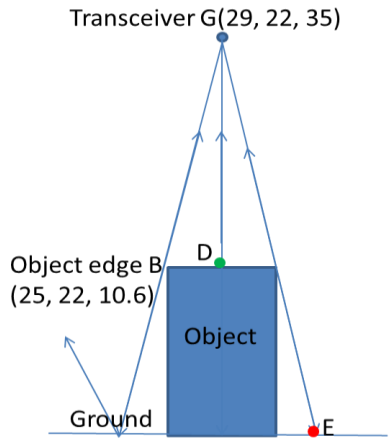


Figure 5.35: Testing case 2

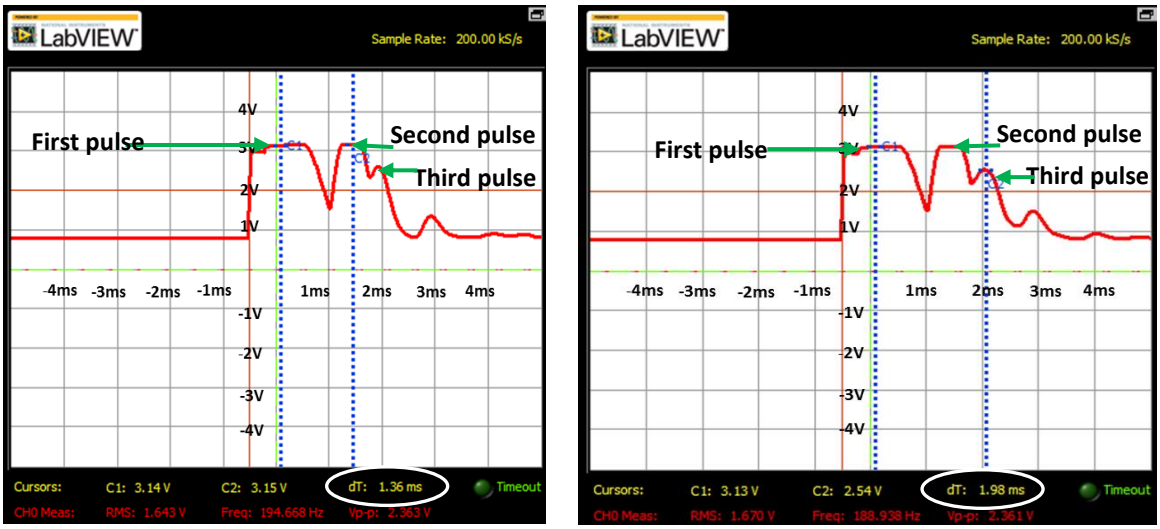


Figure 5.36: Analogue envelope output of MB1340 for case 2

The first pulse is the source pulse. The second pulse is the normal reflection signal reflected from the object because it has the highest impulse response and the distance from the sensor to the object is shorter than that to the ground, the early reflection pulse should come from the object, and it is the desired pulse. The third pulse is the signal reflected from the ground. **dT** on the above graph is the TOF of the returned pulse, 1.36ms for the second pulse, and 1.98 ms for the third pulse. The measured distance between the sensor and the object is 23.32 cm, and the distance between the sensor and the ground is 31.73 cm.

Both testing results show that the desired distance can be measured by finding the highest response signal (the normal reflection signal) of the analogue envelope output in the time domain, which is the most appropriate signal for distance measurement. See Table 5.2 for

recorded data. This could be much more accurate for distance measurement than only considering the early reflection signal. Base on this finding, one way to obtain a more accurate estimate is to design a data processing program that analyses the received signal, detects peaks.

Table 5.2: Measured Distance and calculated distance

	Sensor position	Travelled distance of the first pulse (cm)	Travelled distance of the second pulse (cm)	Distance to the object (cm)	Distance to the ground (cm)
Case 1	(21, 22, 35)	23.67	33.96	25.59	35
Case 2	(29, 22, 35)	23.32	33.61	24.4	35.32

5.10.4 Experiment 3: Effects of the Beam Angle

Another way to avoid big error is to use a narrow beam ultrasound sensor. MB7060 has a beam angle of 11°, which has the smallest beam angle among all 3 sensors. The comparison of the three sensors is shown in Table 5.1. The results of MB7060, HC-SR04, and MATLAB simulator scans along axis Y=22 cm from X=18 cm to 40 cm are shown in Figure 5.37.

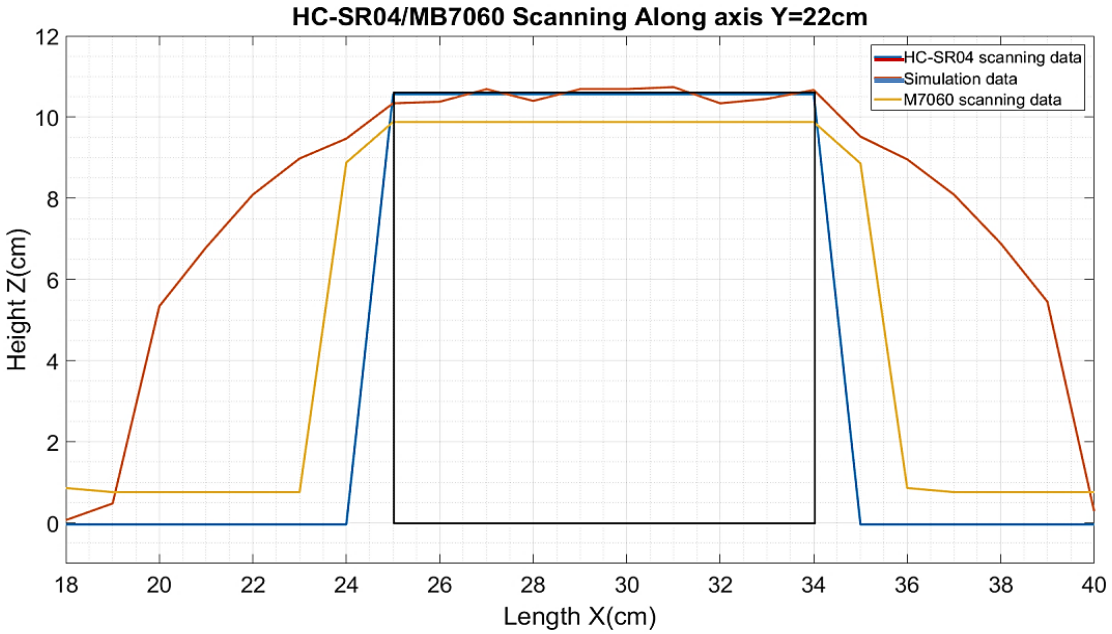


Figure 5.37: Experiment results and simulation results

The results obtained from the sensor with the smallest beam angle (MB7060) are close to the true object and the simulation result. This indicates that a narrow beam sensor will produce more accurate results. However, MB7060 has the accuracy of 1cm [164], so the 10.5cm cuboid is measured as 10 cm.

5.10.5 Experiment 4: Effect of the Height of the Measurement Plane

This experiment aims to validate the simulation results discussed in section 5.6.2, in particular, to investigate the relationship between measurement heights and accuracy of results. Hence, the same experiment shown in Fig 5.30 was repeated at different measurement heights, with the same resolution. Since the MB7060 sensor has a range from 25 cm to 765 cm [164] and the height of the object is 10.6 cm, so the scanning height cannot be lower than 35cm. The results of measurements at heights of 35 cm, 38 cm, 41 cm, and 44 cm are shown in Figure 5.38.

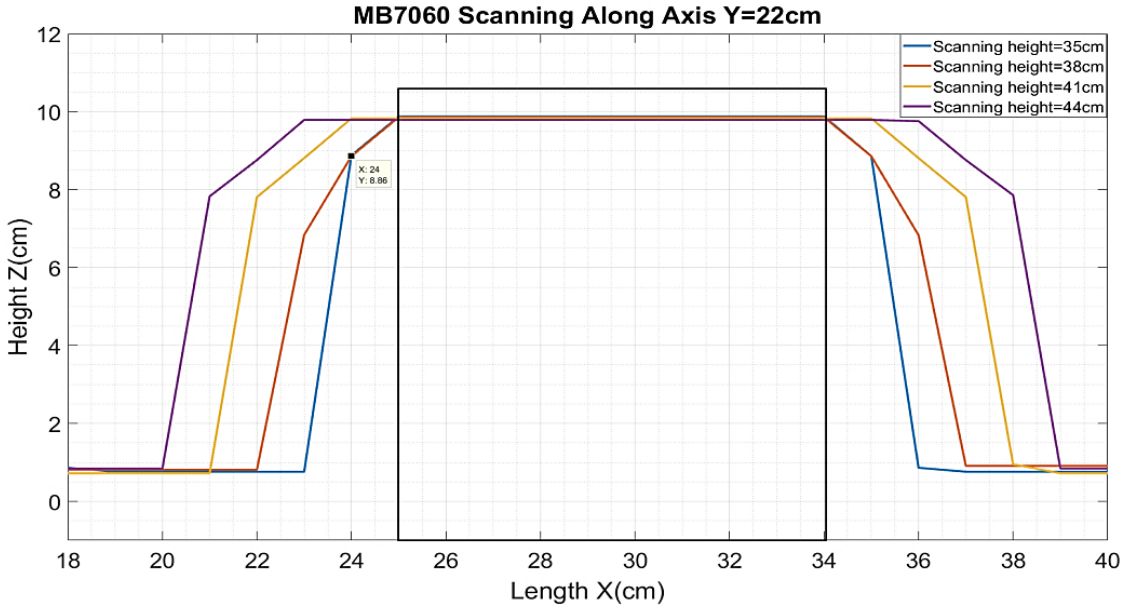


Figure 5.38: MB7060 result for different heights

The above results show that the error increases as the height of the measurement plane are increased, which verifies the simulation in Section 5.6.2. This result shows the mapping height should be carefully considered when undertaking the aerial mapping and keeping the scanner close to the clutter will give a better result.

5.10.6 Experiment 5: Gap Distance of Two Neighbour Obstacles

This experiment aims to study narrow gap detection for the MB7060 ultrasound sensor. The schematic of mapping is shown in Figure 5.39, and the parameters of the experiment are configured as follows:

- The height of the measurement plane was 35 cm.
- The gap distance between 2 obstacles was varied from 1 to 5 cm at intervals of 1 cm.
- The two obstacles were identical; their size was 9 cm×7 cm×10.5 cm.

- The sampling interval is 1cm.

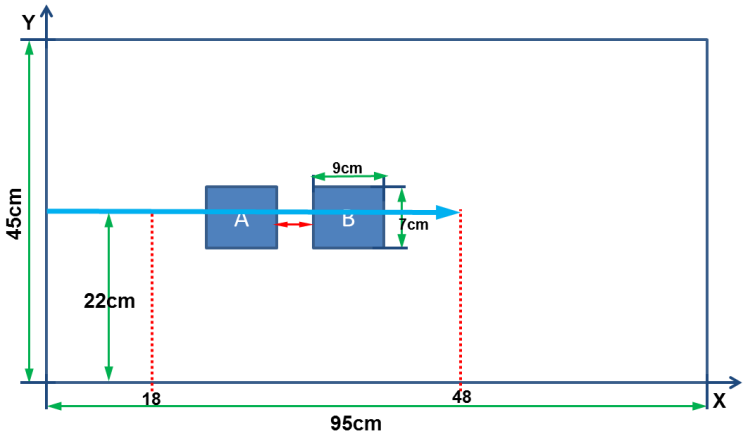


Figure 5.39: Schematic two nearby objects

The corresponding results are shown in Figure 5.40.

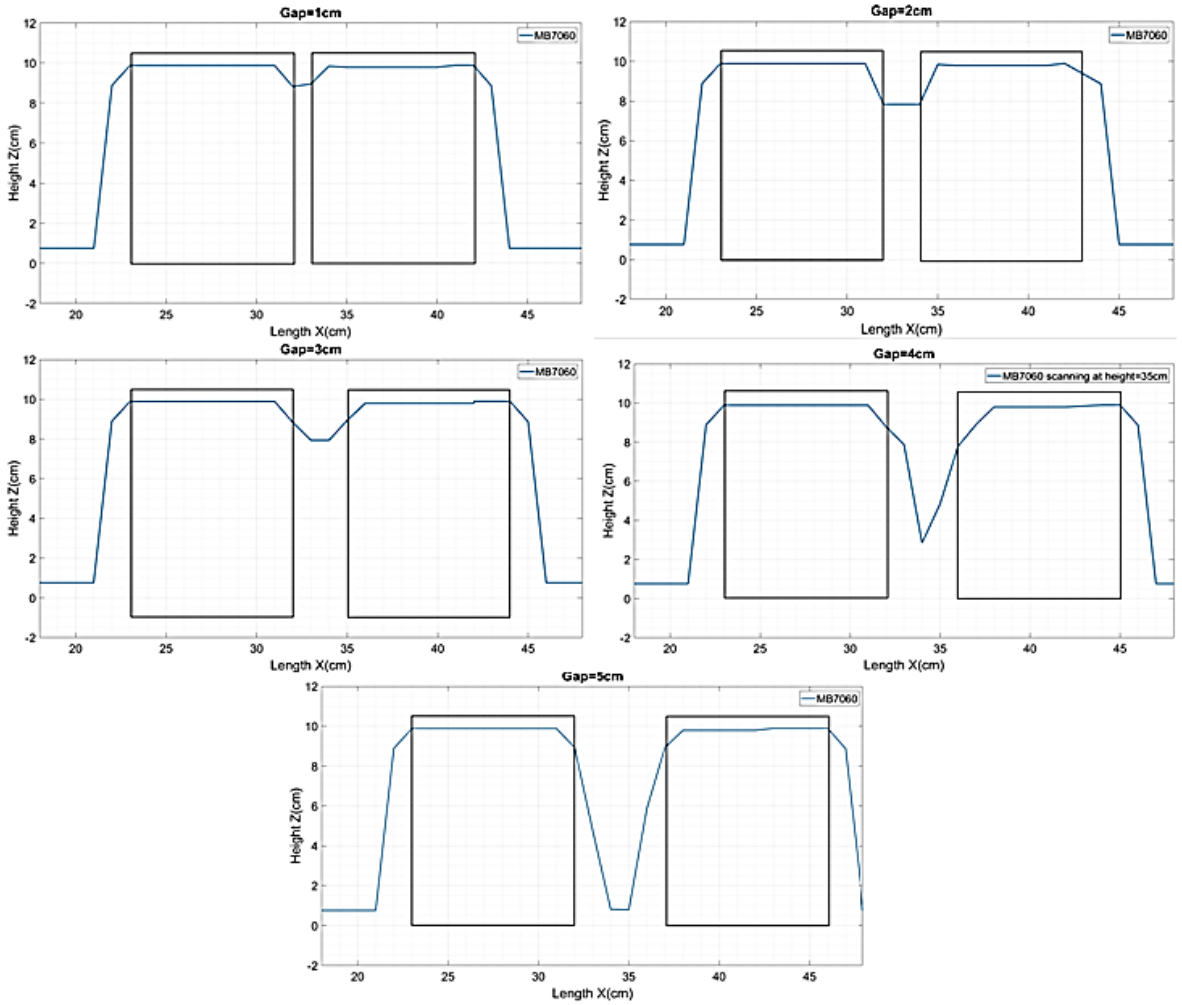


Figure 5.40: Mapping results of different gaps

Above results illustrate that if the measurement plane of MB7060 is 35 cm, the gap cannot clearly be observed until the gap distance between two adjacent obstacles larger than or equal

to 5cm. The relationship of the beam angle, gap distance, obstacle's height, and measurement height is given by Equation (5.4)

$$D = 2 \times (H_{measurement} - H_{Object}) \times tan(\frac{\theta}{2}) \quad (5.4)$$

Where D is the minimum gap distance, allowing the sensor to fully detect the gap. $H_{measurement}$ is the height of the measurement plane. H_{Object} is the height of the object. θ is the beam angle. The way how this equation is derived is shown in Figure 5.41.

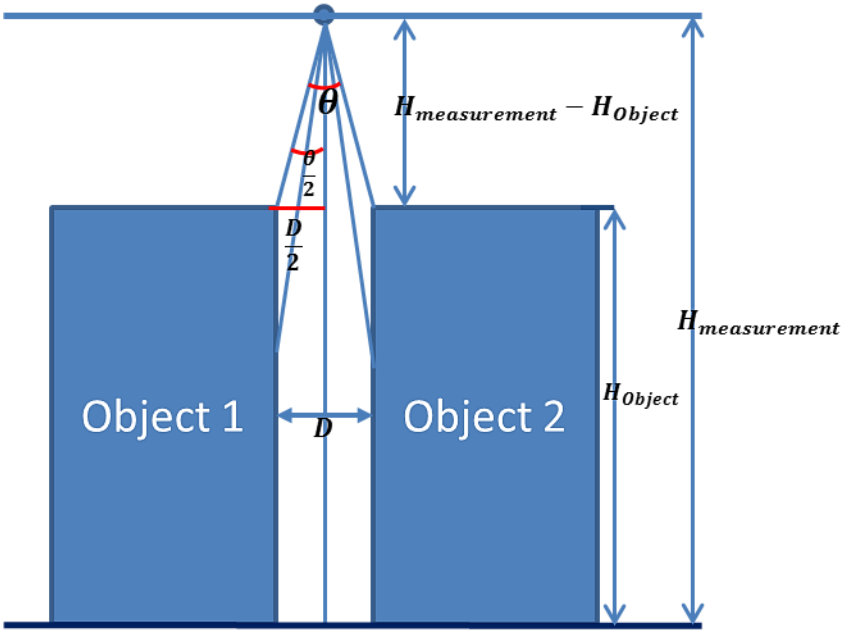


Figure 5.41: Estimation of the gap distance

Further development of the data processing in the future work needs to deal with the depth data to determine if the case is a big object or two separate objects that very close or even estimate the gap distance.

5.10.7 Experiment 6: Aerial mapping for a Pond-like Environment with Four Obstacles

This experiment aims to study the aerial mapping results of a simple representation of a pond-like environment (see Figure 5.42) using the MB7060 ultrasound sensor. The schematic of the environment is shown in Figure 5.43 and the parameters of the experiment as follows:

- The height of the measurement plane was 35cm.
- Four same-sized obstacles were placed on the bottom. The size of each obstacle was 9 cm×7 cm×10.5 cm.

- The sampling interval was 1 cm.

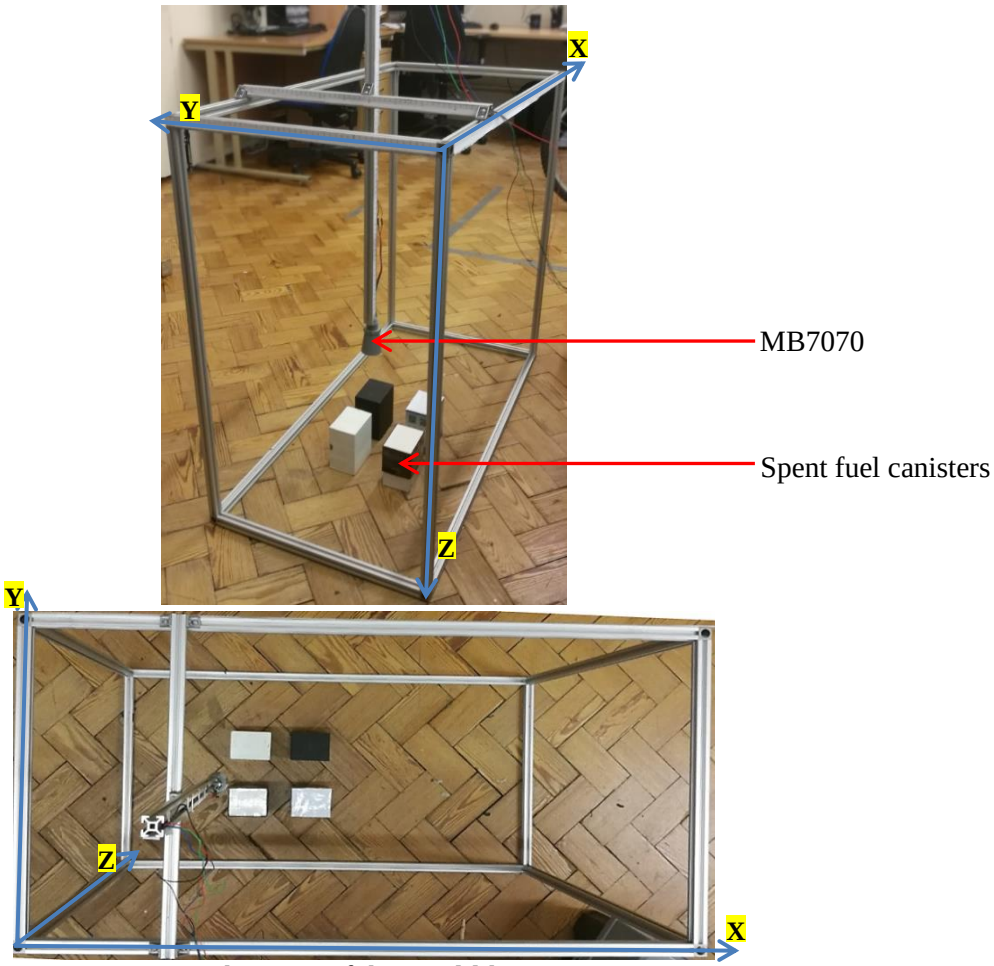


Figure 5.42: The setup of the pond-like environment

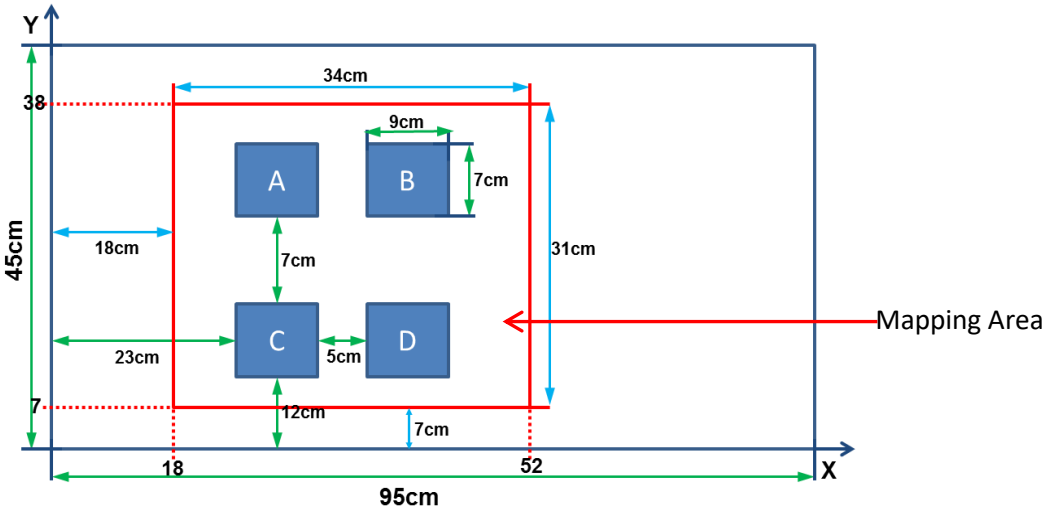


Figure 5.43: Schematic of the environment

The depth measurement results are shown in Figure 5.44 and 5.45 below.

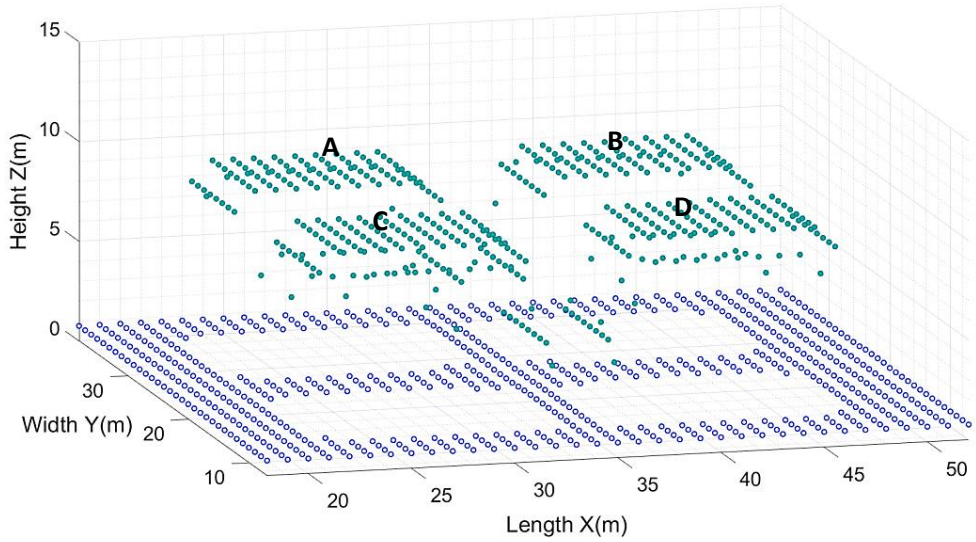


Figure 5.44: Aerial mapping results

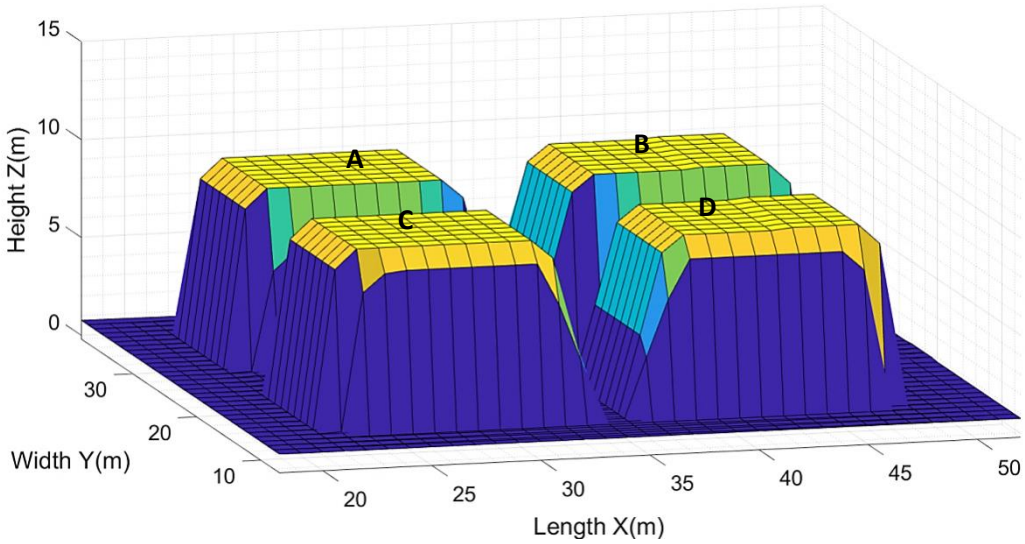


Figure 5.45: Surface plot reconstructed objects

The green dots are the depth measurements of the detected upper surfaces. The blue dots are the depth measurements of the ground of the pond. These results are very similar to the simulation results in Section 5.7, the only difference being that the boundary points of obstacles' upper surfaces have a lower height. This is caused by the beam angle. The beam angle of MB7060 is 11° so it can observe the obstacle when it is not vertically above the edge of the obstacle, which causes the boundary points. This is discussed in Section 5.6.

5.11 Summary and Conclusion

This chapter has provided some simulation results for different obstacle placements and layouts in storage ponds. It shows that aerial mapping is capable of providing a good amount of details about the underwater environment. The experiments validate some simulation cases.

A key outcome of this chapter is that the height of the measurement plane should be considered carefully to obtain better results. This suggests that at least two aerial mappings are needed. The first aerial mapping aims to extract the height of obstacles. The second phase aerial mapping will conduct in a lower measurement plane right above obstacles by knowing the height of each obstacle. Another key outcome of this chapter is the need for a small beam sensor such as the 11 ° used in mapping experiment to prevent the measurements is much bigger than the actual object. However, the problem that a small beam angle brings is if there are sloping objects, the sloping object cannot be detected, a solution to this is to design a sensor array with multiple narrow beam sensors, this has been discussed in Section 3. 7.3.

Chapter 6

Review of Path-planning Algorithms

6.1 Introductions

Path-planning is a crucial part of Robotics, which aims to find a path that enables a robot to move from the start position to the goal position while avoiding any obstacles in its environment [56]. Path-planning algorithms assume that obstacles are defined by line segments in 2D cases and stack of small planes in 3D cases, and the mobile robot is a moving particle.

Path-planning plays an important role in the navigation of autonomous mobile robots, which enables the identification and selection of a suitable path for the robot to traverse in the workspace area. Depending on whether the workspace is dynamic or static (, the path-planning algorithms can be categorised into online path-planning or offline path-planning. The online path-planning computes the trajectory to the goal during robot motion, whereas offline path-planning computes the trajectory before the motion begins [20]. A map acquisition procedure is required for the offline planner. While, The online planner does not need an extra offline map acquisition step as it maps the environment and finds the path at the same time, as in SLAM discussed in section 2.5.2. This research will only focus on offline path-planning.

This chapter reviews some popular mobile robot path-planning algorithms and their applications in 2D and 3D scenarios and discusses their advantages and disadvantages and identifies the appropriate approaches for underwater navigation problem. These include classic path-planning algorithms:

- Visibility graph (VG) [85],
- Probabilistic roadmap (PRM) [67],
- Artificial potential field (APF) [68, 73],

- Voronoi diagram (VD) [62, 74],
- Cell decomposition (CD) [61, 69],
- Rapid explore random tree (RRT) [66],
- Dijkstra's Algorithm [105,106], A*[107, 108, 109]

And bio-inspired path-planning algorithms:

- Genetic algorithm (GA) [71,114]

6.2 Basic Concept and Terminology

A complete path-planning usually consists of two phases: the pre-processing phase and a query phase. The pre-processing phase builds the *workspace* with the concept of the *configuration space* (*C*-space) [58]. The definition of *the workspace* is the geometry model of the working environment of the mobile robot and the *C-space* describes the pose of the mobile robot and the specification of the *workspace*. The *C-space* is defined by $C = C_{free} \cup C_{obs}$, where C_{free} is the free space and C_{obs} is the obstacle occupied space. The mobile robot can be regarded as a point or small sphere that has three degrees of freedom without considering its size, and the obstacles are enlarged according to the size of the robot [57, 59], which simplifies the path-planning problem, see the workspace and *C-space* translation in Figure 6.1. The query phase aims to find a suitable path on the built map so that the robot traverses in the workspace area without colliding with other obstacles.

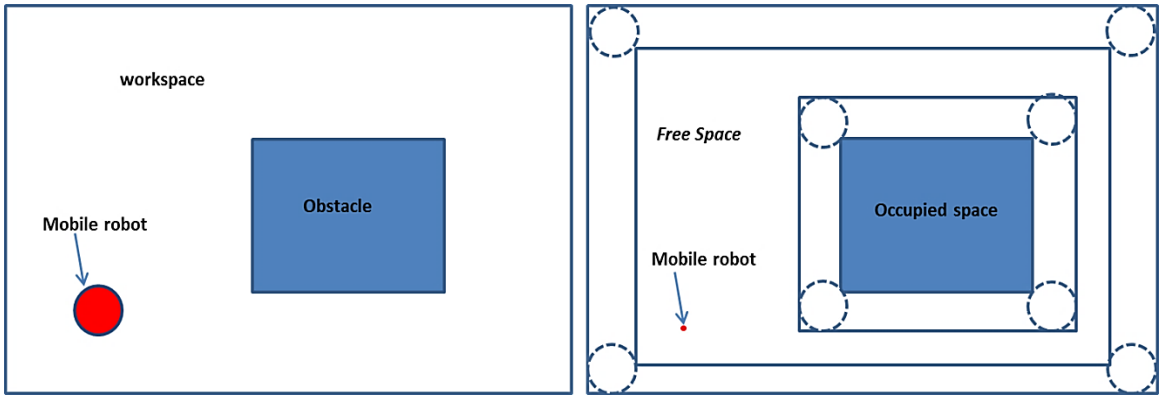


Figure 6.1: Workspace *C-space* translation

6.3 C-Space Modelling Techniques

To simplify the path-planning problem, it assumes that the workspace already being translated to *C-space*. The next step is to apply modelling techniques to represent the *C-space*

in the pre-processing phase because the *C-space* needs to be represented in a suitable way so that can be used in the query phase. *C-space* modelling techniques represent the *C-space* by a collection of grids [60], cells [61], nodes [62, 66] (nodes are defined as points in the workspace), or connecting lines [63, 64, 65]. For example, a 2D environment can be decomposed to small cells by Cell Decomposition algorithm and a 3D environment can be decomposed to small cubic grids by Gird Decomposition algorithm. Different path-planning algorithms model the *C-space* differently.

Cell Decomposition (CD) algorithm divides the workspace into simple, connected cell regions [61]. There are no strict constraints for the shape of cells. They could be square, triangular, or polygonal [69]. As a result, the *C-space* is modelled as free cell regions and obstacle cell regions, as shown in Figure 6.4.

A roadmap (RM) is defined as the network in the *C-Space*. It is obtained by the connection between randomly placed nodes in the free *C-Space*. The Road map (RM) algorithms such as Probabilistic Roadmap (PRM), Voronoi Diagram (VD), and Visibility Graph (VG) model the *C-space* by a set of nodes and connecting lines. As a consequence, the *C-space* is abstracted to a skeleton road map of discrete nodes, as shown in Figure 6.7. The rapidly-exploring random tree (RRT) also models the *C-space* as a set of nodes and connecting lines, although it is not a roadmap approach.

Some graph searching algorithms such as A* and Dijkstra's algorithm needs to utilise with *C-space* modelling techniques such as Grid Decomposition (GD) to compose a complete path-planning algorithm. GD is a 3D *C-space* modelling technique. It divides the environment into a finite number of equally sized 3D cubes by the method of equal division, resulting in a mesh. As shown in Figure 6.2: equally-spaced orthogonal horizontal and vertical dash planes segment the whole 3D environments. As a result, the whole environment is abstracted into a finite number of equally sized cells. Cells being occupied or partially occupied by obstacles are defined as inaccessible cells. Cells that are completely unoccupied are defined as accessible cells.

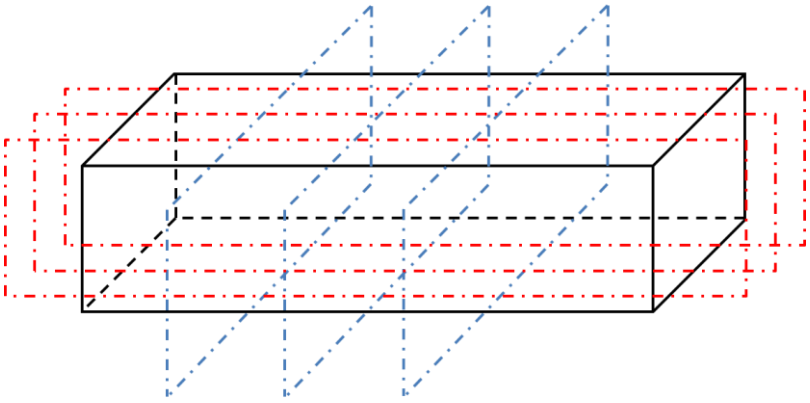


Figure 6.2: 3D space division

However, decomposing a whole environment by GD is time-consuming, so it is common to reduce computation time by using the octree model. An octree is a tree data structure that divides an environment into 8 equal octants [60].

The whole region is regarded as the root cube. The algorithm (which is recursive) then divides the root cube is into eight sub-cubes called ‘octants’, each octant is classified as ‘free’, ‘occupied’, or ‘mixed’ according to the spatial relation of the octant with respect to the given environment [77]. The ‘free’ octant is defined as the sub-cube is an **empty** space, the ‘occupied’ octant is defined as the sub-cube is **fully** occupied with obstacles, the ‘mixed’ is defined as the sub-cube is **partially** occupied by obstacles. If the decomposed cube is ‘free’ or ‘occupied’, then stops. If it is a ‘mixed’ node, it requires recursively decomposition until the desired level is reached [77]. An example of how the octree algorithm segments a 3D space is shown in Figure 6.3.

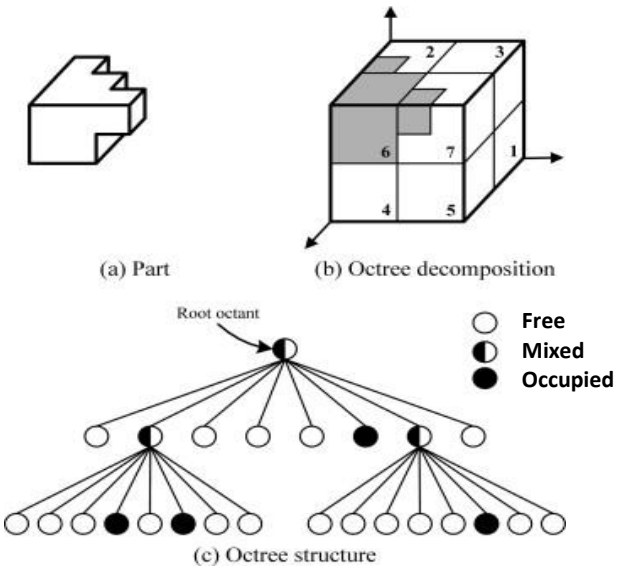


Figure 6.3: Octree decomposition model [77]

6.4 Path-planning Algorithms

6.4.1 Cell Decomposition

Cell decomposition (CD) is a path-planning algorithm based on a 2D *C-space* modelling approach. There are two types of CD: vertical cell decomposition (VCD) and approximate cell decomposition (ACD). VCD divides the workspace into several trapezoidal cell regions by projecting vertical lines cross each vertex of the obstacles [65]. Every trapezoidal cell is assigned with a number. As a result, the environment is segmented into free regions (blank area indicated with numbers in Figure 6.4) and forbidden regions (blue area in Figure 6.4). The midpoint of each vertical line is regarded as the waypoint on the path from the source to the goal. The complete path is created by connecting the adjacent waypoints, the start point, and the goal. The information derived from the connected waypoints enables a graph to be formed. Path algorithms, such as Dijkstra's (see section 6.3.7), can be applied to find the shortest path. It is important to note that the found path is usually not the shortest optimal path because the waypoint is only the midpoint of each vertical line. An example of VCD is shown in Figure 6.4.

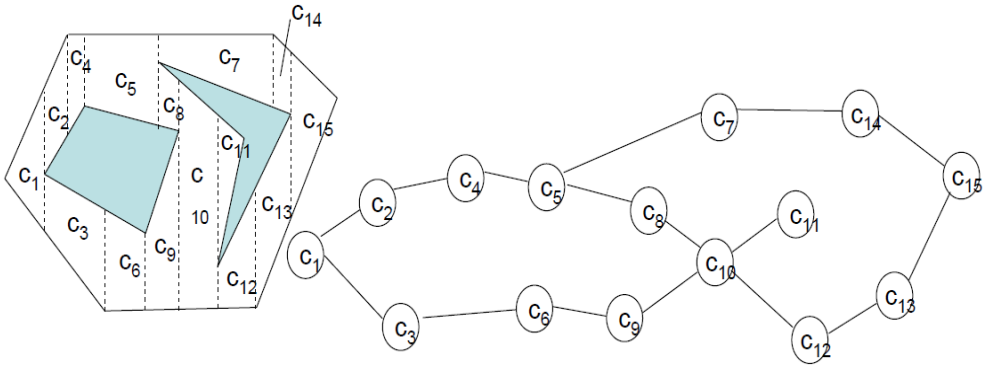
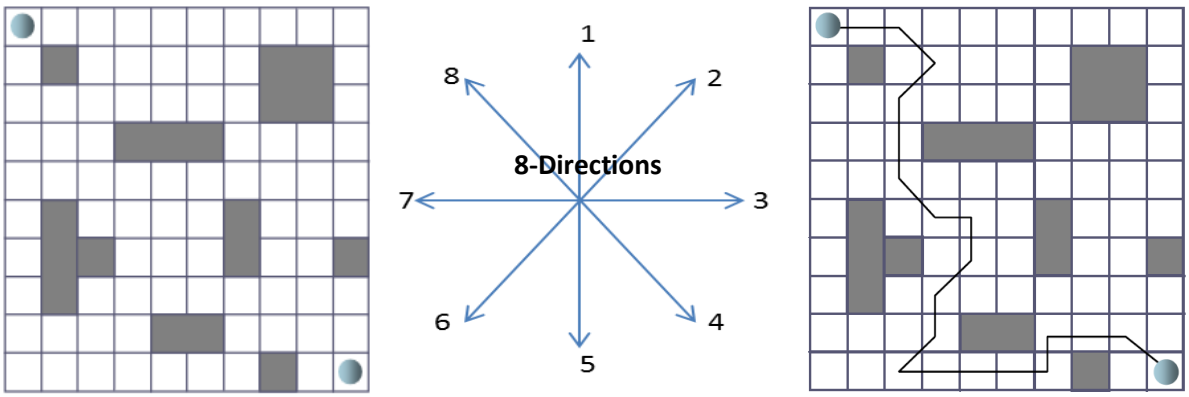


Figure 6.4: Vertical cell decomposition diagram [78]

VCD can only plan the forward path and not usable for 8-directional path-planning scene shown in Figure 6.5. While, the ACD method can plan movements in eight directions [79], which can avoid obstacles more effectively, as shown in Figure 6.5(c). Unlike VCD divides the *C-space* by projecting vertical lines cross each vertex of the obstacles, an ACD does not divide the *C-space* based on the vertex of obstacles, it divides the *C-space* into a finite number of occupied cells and free cells [79] by the method of Quadtree decomposition [76] (2D version of octree decomposition discussed above).



(a) (b) (c)

Figure 6.5: Approximate Cell Decomposition [79]

The CD methods can be applied to a 3D environment, and this is equivalent to the GD and Octree decomposition approaches described in Section 6.3. The GD is an option for solving the cluttered underwater environment navigation problem when it works along with searching algorithms like A* (see section 6.3.8). Williams and Jones [80] used octrees and connected graphs to represent the working environment of small autonomous aircraft to check overhead power lines. Their work reduced the usage of computer memory and increased searching speed.

6.4.2 Voronoi Diagrams

The Voronoi diagram is a path-planning algorithm based on the roadmap *C-space* modelling approach. This approach divides the *C*-space into convex polygons whose edges are a set of waypoints (nodes) with equal distances to their closest obstacles [74], see Figure 6.6. These waypoints are known as Voronoi vertices (the blue dots in Figure 6.6). The lines that connect the Voronoi vertices make up the path. The problem is that in most cases, the starting position and the target position are not Voronoi vertices. In this situation, choose its closest Voronoi vertex as the starting position and the goal, and calculate a path between them, then connect them to the true start and end nodes (see the dash black line in Figure 6.5). After that, apply Dijkstra's algorithm to compute the shortest path from $\boldsymbol{q_{start}}$ to $\boldsymbol{q}_{goal}$. A Voronoi diagram is shown in Figure 6.5. This approach will guarantee a path away from obstacles, but on the other hand, the obtained path could be conservative [82].

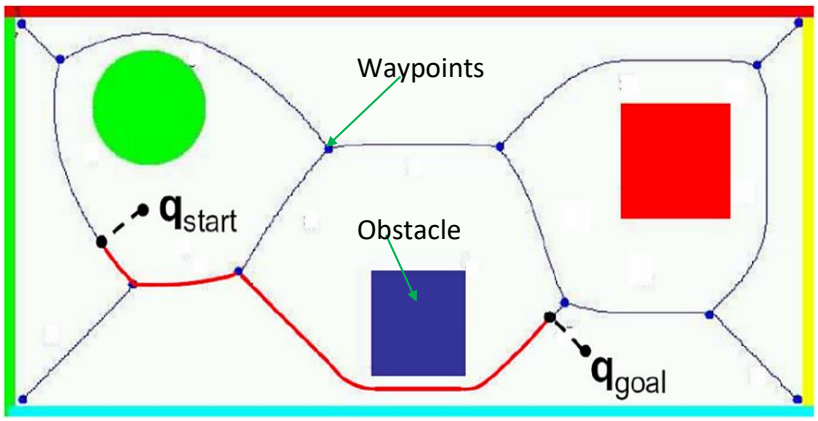
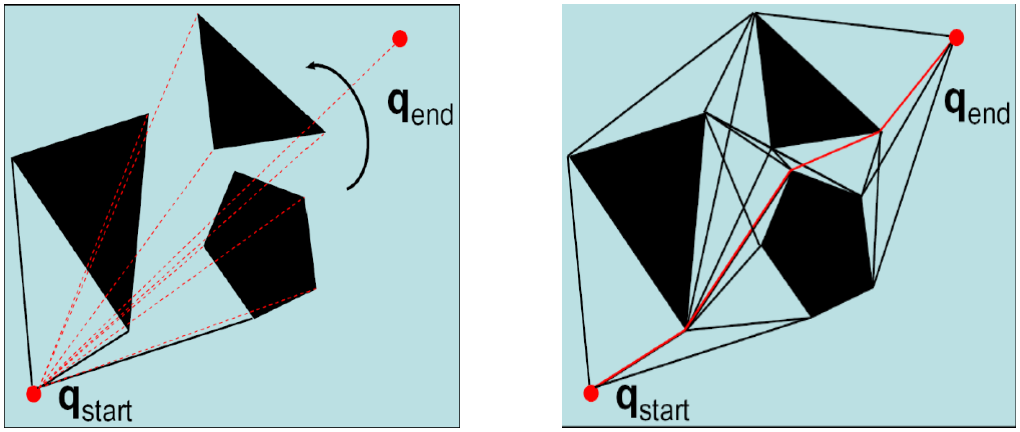


Figure 6.6: Voronoi diagram [81]

The VD can be extended to model a 3D environment, in [83] VD works along with the Geography Information System (GIS) to obtain an optimised 3D route. However, most Voronoi-based methods have the difficulty of calculating the Voronoi Diagram by studying lines and polygons, finding the vertices and nodes, and creating a tree to find the path in 3D [110].

6.4.3 Visibility Graph

The visibility graph (VG) is a popular path-planning algorithm based on the roadmap *C-space* modelling algorithm. In the VG, the vertices of obstacles, the start point, and the goal point are abstracted to a finite number of nodes. Each node represents a point location, and the line that connects every two nodes represents a connection path between them. If the line does not pass through any obstacles, the line is visible and considered as a feasible path and remains in the graph (solid black line in Figure 6.7(a)). Otherwise, it is an infeasible path that needs to be removed from the graph (dashed red line in Figure 6.7(a)). The same process above is repeated for the remaining nodes until the goal node. VG constructs a roadmap that connects the free space between obstacles, thus converting the continuous *C-space* into a skeleton graph-like structure. Finally, Dijkstra's searching algorithm was used to find the shortest path that connects the start and the goal (see Section 6.4.7). An example of the VG approach is shown in Figure 6.7.



(a) (b)

Figure 6.7: Visibility graph diagram [81]

The concept of VG can be extended to a 3D environment, which using the planes instead of the lines, paper [85] explains how VG works in 3D cases. Omar [84] proposed a method that first converts the 3D case to a 2D case, and then, finds a path by the traditional VG algorithm. After that, add the altitude to the path and convert it to a 3D path. Omar [84] demonstrated a 3D VG path-planning in a simulator, which shows the performance of obstacle avoidance in an environment with a set of regularly shaped cuboids. However, Omar's [84] simulations do not consider sloping obstacles.

6.4.4 Probabilistic Roadmap

The probabilistic roadmap (PRM) is another path-planning algorithm based on the roadmap *C-space* modelling algorithm. The basic idea of this algorithm is to generate sampling nodes in the *C-space* randomly and connect them. This algorithm abstracts continuous space as discrete nodes and connecting lines [67]. PRM probably satisfies probabilistic completeness. As long as the start goal path exists, if the number of sampling points increases without boundary, the probability that the algorithm finds a path is one [112]. Nodes are added up to a pre-specified maximum. The nodes are then checked to determine whether they lie in the free area or within obstacles. Nodes within obstacles are removed. The remaining nodes are connected to create local paths. If the created path crosses obstacles, it is removed. The final remaining connecting lines construct the skeleton graph-like roadmap.

The PRM path-planning algorithm consists of a learning phase and a query phase. The learning phase is to model the environment by randomly generated sampling nodes, and the query phase is to search the roadmap obtained by the learning phase to find a sequence of lines connecting these nodes that contribute to a collision-free path for the robot [90]. Graph

searching algorithms such as Dijkstra's algorithm or A* algorithm is often used in the query phase [91]. An example of the probabilistic roadmap is shown in Figure 6.8.

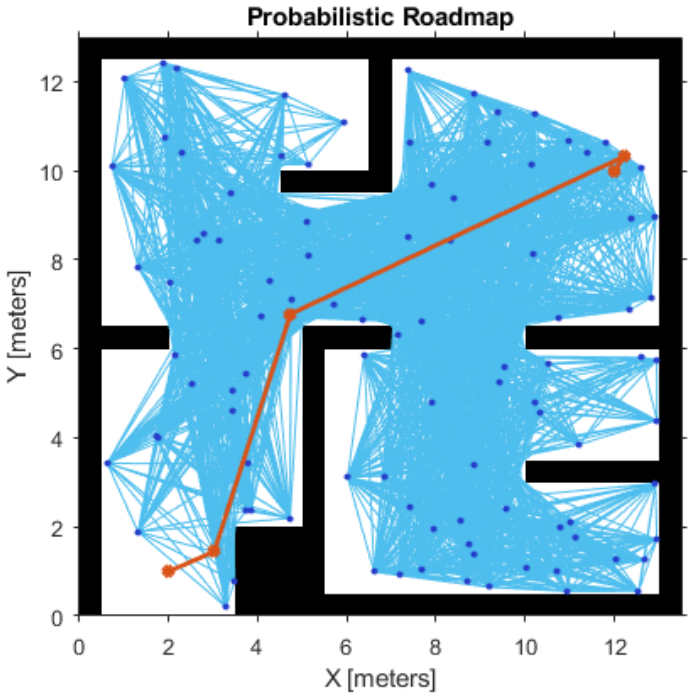


Figure 6.8: Probabilistic roadmap example [175]

PRM has two sub-algorithms, K-PRM [92] and S-PRM [93]. Each step, K-PRM chooses the nearest K neighbour nodes for connection, and S-PRM chooses nodes within a circle/sphere region for connection. This algorithm has also been applied to 3D environments e.g. [174]. Hrabar [88] applied this algorithm to simulate an unmanned aerial vehicle (UAV) flying in an urban canyon environment. However, his research relies on accurate and complete information about the environment, which would be a challenge in the cluttered underwater environment.

The main disadvantage of this technique is that the number of sampling nodes needs to consider carefully, it probably cannot find a path in a narrow space if the number of sampling node is too small. Besides this, since the path points are randomly generated, this method cannot guarantee to obtain the shortest solution [89].

6.4.5 Rapidly- Explore Random Trees

Rapidly-exploring random tree (RRT) is a graph searching algorithm that need not model the *C-space*, which is proposed by LaValle [66]. The principle of this algorithm is to explore the unsearched area by randomly generating sampling nodes in the unsearched area. Its name comes from the expansion of the searching acts as a structure of a tree.

RRT starts by regarding the start point as the initial root node of a tree. Then, randomly creates a node and tries to connect the new node to the root. This process continues with attempting finds a node in the expanding tree which nearest to the random node and extends the node in the direction of the random node for a small distance to obtain the new node. If the new node insides an obstacle, then remove it and create another new random node until the new node does not inside an obstacle. Then, add the node to the exploring tree [66]. The above procedures will be repeated until the target is reached. The expansion of the exploring tree is shown in Figure 6.9.

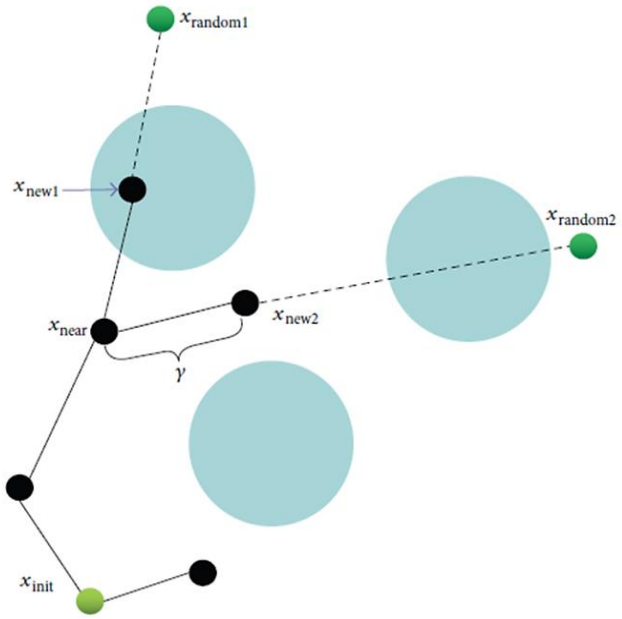


Figure 6.9: An example of expanding explore tree [91]

First of all, generating a random node $x_{random1}$ on the *C-space*, and then find the nearest node on the expanding tree, which is x_{near} , x_{near} is the nodes in the expanding tree, which is created in the same sampling process previously. After that, a new node x_{new1} is added at a small distance from the tree along the line to $x_{random1}$. However, x_{new1} is placed within the obstacle (green circle), so it cannot be added to the expanding tree. Repeating the same procedures above to generate a random node $x_{random2}$ and adding a new node x_{new2} at a small distance from the tree along the line to $x_{random2}$. Since x_{new2} is not placed within an obstacle, it can be added to the expanding tree.

Advantages of RRT are low time-complexity, fast searchability, and probabilistic completeness [66]. It could work along with other searching algorithms like A* to improve its performance [119] (see section 6.4.9).

6.4.6 Artificial Potential Fields

The artificial potential field (APF) method is based on the idea of assigning a virtual potential field to the *C-space.* It was originally proposed by Khatib in 1985 [96]. This method assumes that the robot is moving forward to a goal with the aid of a potential field. This approach regards the goal as an attractive source which produces an attraction potential field to pull the robot closer and regards the obstacles as repulsive sources which produce repulsion potential fields to push the robot away. Consequently, these potential fields are combined to form a composite force that drives the robot from the start position towards the goal by following the negative gradient.

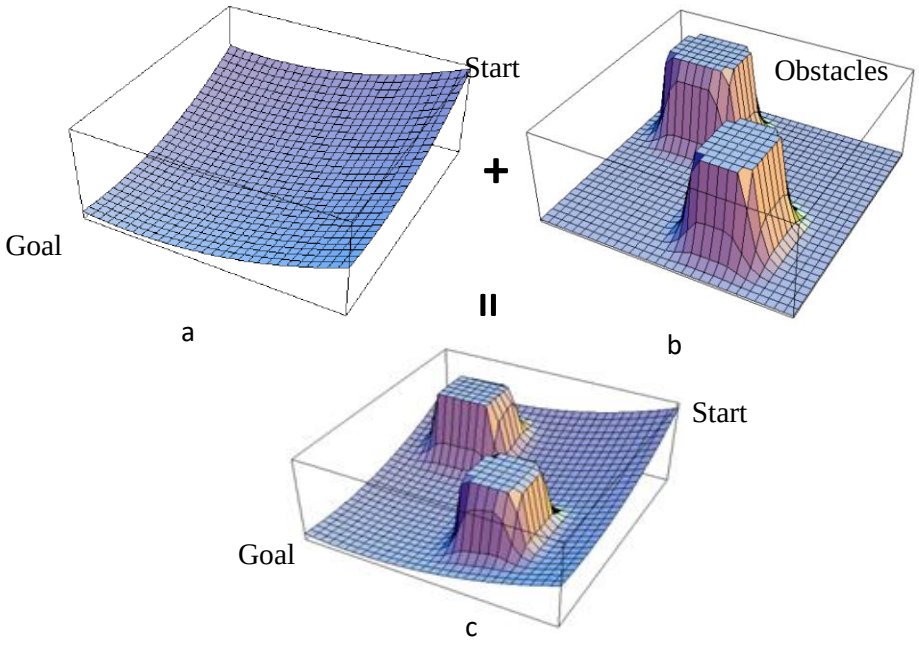


Figure 6.10: (a) gradient map of the attractive potential field; (b) the repulsive potential field; (c) the composite potential field are shown [95]

Figure 6.10(a) shows a negative gradient potential field from the start (highest potential) to the goal (lowest potential). There are no objects in this environment, and the mobile robot will follow a trajectory between the start and the goal based on the gradient. Imagine the mobile robot is a rolling ball following a "downhill" path. Figure 6.10(b) shows a positive gradient potential field of obstacles. Imagine obstacles are small mountains on the hill, and the rolling ball was forced away and never be able to climb. Figure 6.10(c) shows the composite gradient potential field obtained by superposing (a) and (b). The basic potential function is the sum of the attraction potential field and the repulsion potential field [96]:

$$U(q) = U_{att}(q) + U_{rep}(q) \tag{6.1}$$

$$F(q) = F_{att}(q) + F_{rep}(q) \tag{6.2}$$

Where $U_{att}(q)$ is the attraction potential and $U_{rep}(q)$ is the repulsion potential. $F_{att}(q)$ is the active force and $F_{rep}(q)$ is the repulsive force.

The attraction potential field function is assumed given by a parabolic function [96]:

$$U_{att}(q) = \frac{1}{2}\varepsilon\rho^2(q, q_{goal}) \tag{6.3}$$

Where ε is the attraction index, $\rho(q, q_{goal})$ is the distance between the goal and the mobile robots. The attraction force $F_{att}(q)$ is the gradient of $U_{att}(q)$, so the derivative of the attraction potential function, which is given by:

$$F_{att}(q) = -\nabla U_{att}(q) = -\varepsilon * \rho(q, q_{goal}) \tag{6.4}$$

The repulsion potential fields function is also a parabolic function [96]:

$$U_{rep}(q) = \begin{cases} \frac{1}{2}\sigma * \left(\frac{1}{D(q)} - \frac{1}{\rho 0}\right)^2, & D(q) < \rho 0 \\ 0, & D(q) \geq 0 \end{cases} \tag{6.5}$$

Where $\rho 0$ is the repulsion radius (robot beyond this range will not receive any repulsion force produced by the repulsion source). $D(q)$ is the distance between the robot and the obstacle. σ is the repulsion index. The repulsion force $F_{rep}(\mathrm{q})$ is the gradient of the repulsion potential, which represents Force Inducting an Artificial Repulsion from the Surface (FIRAS) [96]:

$$F_{rep}(\mathrm{q}) = -\nabla Urep(q) = \begin{cases} \sigma * \left(\frac{1}{D(q)} - \frac{1}{\rho 0}\right) * \frac{1}{D^2(q)}\nabla D(q), & D(q) < \rho 0 \\ 0, & D(q) \geq 0 \end{cases} \tag{6.6}$$

Artificial potential field path-planning is suitable for both 2D and 3D scenarios. However, there are three main drawbacks with this approach:

1. The ‘collision’ problem. Because the intensity of the attractive force is proportional to the distance, if the distance between the goal and the robot is great, the attraction force will be strong and overwhelm repulsion forces, which might cause the robot to collide with obstacles [97].

2. The ‘goals non-reachable with obstacle nearby’ problem. If an obstacle is placed near the goal, the mobile robot near the goal will receive a strong repulsion force, causing the mobile robot unable to reach the goal [97, 98].

3. The ‘trapped in a local minimum’ problem. When the attraction force and the repulsion force are equal or almost equal but opposite in direction, the composite potential force is zero, which will cause the robot to be trapped in, or oscillate about, a local minimum [97, 99].

6.4.6.1 Improved Artificial Potential Fields Function to Address the Collision Problem

There are improvements to the basic artificial potential field approach, which attempt to mitigate or eliminate the drawbacks stated above. To solve the problem where the attraction force overwhelms a repulsion force, the attraction potential function 6.3 and attraction force function 6.4 can be modified as follows [73, 97]:

$$U_{att}(q) = \begin{cases} \frac{1}{2}\varepsilon\rho^2(q, q_{goal}) & \rho(q, q_{goal}) \leq d^*_{goal} \\ d^*_{goal}\varepsilon\rho - \frac{1}{2}\varepsilon(d^*_{goal})^2 & \rho(q, q_{goal}) > d^*_{goal} \end{cases} \quad (6.7)$$

$$F_{att}(q) = -\nabla U_{att}(q) = \begin{cases} \varepsilon * \rho(q, q_{goal}) & \rho(q, q_{goal}) \leq d^*_{goal} \\ \varepsilon * d^*_{goal} & \rho(q, q_{goal)} > d^*_{goal} \end{cases} \quad (6.8)$$

A new element d^*_{goal} is assigned to the equation, which is the threshold distance between the robot and the goal. If the distance between the robot and the goal is smaller than the threshold distance, the attraction function does not change. If the distance between the robot and the goal is greater than the threshold distance, the attraction potential function becomes $d^*_{goal}\varepsilon\rho - \frac{1}{2}\varepsilon(d^*_{goal})^2$ and the attraction force becomes $\varepsilon * d^*_{goal}$ which is a constant value. This new equation ensures that once the distance is greater than the threshold distance, the attraction force will not increase anymore, so will not have the problem that at a far distance, the attraction overwhelms the repulsion.

6.4.6.2 Improved Artificial Potential Fields Function to Address the Goal Unreachability Problem

The original repulsion potential Function 6.5 and repulsion force Function 6.6 was modified as follows to address the inability to reach the goal problem [98]:

$$U_{rep}(q) = \begin{cases} \frac{1}{2}\sigma\left(\frac{1}{\rho(q,q_{obs})} - \frac{1}{\rho 0}\right)^2 \rho^n(q, q_{goal}) & \rho(q, q\_obs) \leq \rho 0 \\ 0 & \rho(q, q_{obs}) > \rho 0 \end{cases} \quad (6.9)$$

$$F_{rep} = -\nabla U_{rep}(q) \quad (6.10)$$

$$F_{rep} = F_{rep1} \cdot \overrightarrow{n_{or}} + F_{rep2} \cdot \overrightarrow{n_{rg}} \quad (6.11)$$

$$F_{rep1} = \sigma\left(\frac{1}{\rho(q,q_{obs})} - \frac{1}{\rho 0}\right) * \frac{\rho^n(q,q_{goal})}{\rho^2(q,q_{obs})} \quad (6.12)$$

$$F_{rep2} = \frac{n}{2} * \sigma\left(\frac{1}{\rho(q,q_{obs})} - \frac{1}{\rho 0}\right) * \rho^{n-1}(q,q_{goal}) \quad (6.13)$$

The distance between the robot and the goal $\rho^n(q,q_{goal})$ is added to the repulsion potential field function. The gradient of the new repulsion potential field function $U_{rep}(q)$ creates two repulsion forces F_{rep2} and F_{rep1}, whose directions are from the obstacle to the robot ($\overrightarrow{n_{or}}$) and from the robot to the goal ($\overrightarrow{n_{rg}}$), respectively. As the robot approach to the goal, F_{rep1} getting smaller and smaller and the combination of F_{rep1} and F_{rep2} leads to a lower repulsion force. As a result, mobile robots are able to reach their goal.

6.4.6.3 Introduce Simulated Annealing to Address the Local Minimum Problem

The local minimum problem is the main drawback of the APF path-planning algorithm [99]. One solution is to introduce simulated annealing (SA) into the method [99]. The concept of SA comes from annealing in metallurgy. The concept of SA in path planning can be interpreted as having a probability of accepting a worse next position (e.g. a longer route with a larger potential) when exploring a space [101]. In terms of using in SA-APF path-planning, when the robot falls into a local minimum, the SA part starts to work. It accepts the new solution unconditionally if the new position has a lower potential $U(q)$. Otherwise, with a probability of $e^{\frac{-\Delta}{kT}}$, to accept a new position has a higher potential $U(q)$[99], where Δ is the potential diffidence between the new position and the previous position, k is Boltzmann constant, and T denotes the temperature. After successfully escaping from a local minimum, the SA part will deactivate and the APF part will activate. The procedures of the simulated annealing algorithm for escaping the local minimum are described below:

Algorithm 1 SA-APF path planning algorithm

```
procedure SET x=S                          ▷ Start-point for simulated annealing
    Set T = T0                             ▷ Choose the annealing strategy, T0 is high
    while T ≥Tf and not escaped do
        x' = x + Δx                        ▷ Pick random neighbour
        U(x') = f(x')                      ▷ Calculate the potential at x'
        ΔU = U(x') − U(x)                  ▷ Calculate the potential difference
        if ΔU ≤ 0 then
            x = x'
        else
            if Rand(0,1) ≤ e^(−Δ/(KT)) then
                x = x'                     ▷ Accept x' with with certain probability
            end if
        end if
        T = Tnew                           ▷ Assign a new cooling temperature
    return x                               ▷ The new neighbour x
```

The SA algorithm cannot guarantee escape from a local minimum before temperature cooling because it randomly generates the next movement nodes. With the help of the SA algorithm, the local minimum problem in APF path-planning is alleviated, but not guaranteed to be eliminated.

The APF and SA-APF algorithm can be easily extended to the 3D case without needing to model to the *C-space*. There are some applications of artificial potential field methods in practice. Prahlad Vadakkepa [102] implemented APF in a micro-robot soccer environment. Their approach was tested in different scenarios related to ball tracking and ball kicking. Tan et al. integrated APF and A* and applied them to rotary-wing flying robots [103]. Cui [104] proposed the APF approach to resolve the underwater docking problem. Matthias [117] applied the APF approach to address a real-time obstacle avoidance problem.

6.4.7 Dijkstra's Algorithm

Dijkstra's algorithm is a node-based graph searching algorithm for finding the shortest paths among nodes in a graph [105]. In this algorithm, the graph/map needs to be abstracted to a finite number of nodes, and arcs connect adjacent nodes. Usually, it is used along with *C-space* modelling techniques [115] such as PRM, CD, and VD.

After the *C-space* is abstracted into a finite number of nodes and connecting lines, Dijkstra's algorithm can be used to find out the shortest path. In the beginning, the algorithm will set the current node as the start node, the distance to the start node to zero and the distance to the remaining nodes to infinity. Then, estimate the cost to its adjacent unvisited neighbour nodes,

and compare the newly estimated distance to the current assigned value, if the new estimated distance is smaller than the current value, update the value, otherwise keep the same. After all neighbour nodes have been checked, mark the current node as visited and remove it from the unvisited set, and select the unvisited neighbour node that assigned with the smallest distance as the new current node. The above process will repeat for each neighbour node until the goal node is visited. The nodes that have been visited contribute to the shortest path. Dijkstra's algorithm is explained with the aid of Figure 6.11 below:

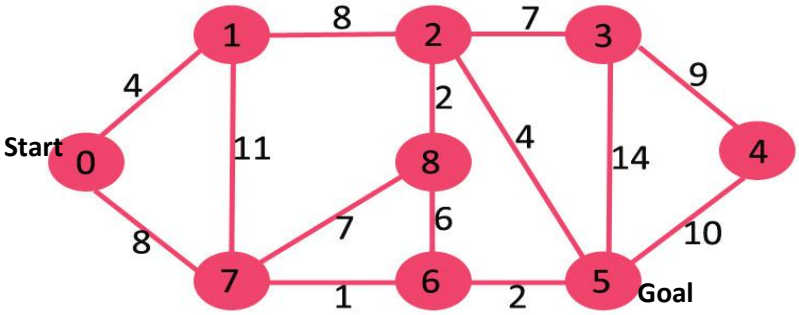


Figure 6.11: Dijkstra's algorithm

Nodes in Figure 6.11 represent waypoints in the free space and connecting lines between adjacent nodes in Figure 6.11 represent accessible paths. Node 0 is the start, node 5 is the goal. In the beginning, node 0 is the current node, and its adjacent nodes are node 1 and node 7, which are at distances of 4 and 8, respectively. Node 1 will be selected as the new current node, and node 0 is marked as visited. Node 1's adjacent nodes are 2 and 7, the distance to node 2 is 4+8=12; the distance to node 7 is 4+11=15 which is greater than the previous distance 8, so the previous distance remains the shortest distance. Since node 7 is closer to node 0 than node 2, node 7 will be selected as the new current node, and node 1 is marked as visited.

Node 7's adjacent nodes are 8 and 6. Node 6 has the shortest distance, so it is selected as the new current node, and node 7 is marked as visited. For the current node (node 6), its adjacent nodes are node 8 and node 5, among them node 5 is the goal, and it has the short distance (distance to node 5 is 11, while the distance to node 8 is 15), therefore, the search ends. The shortest path consists of node 0, node 7, node 6, and node 5.

Dijkstra's algorithm will search every sampled node in all directions to guarantee the shortest path, but the searching process is time-consuming. To improve efficiency, the A* algorithm was developed.

6.4.8 The A* Algorithm

The A* algorithm is a node-based search algorithm and usually works along with some *C-space* modelling methods such as grid decomposition (GD). GD divides the environment into a finite number of free grids/nodes and obstacle grids/nodes, and the A* algorithm finds a collision-free path on the grid map. Peter Hart, Nils Nilsson and Bertram Raphael [107] first described the algorithm in 1968. It can be regarded as an extension of Dijkstra's algorithm [106]. The A* algorithm is not only widely used in mobile robotics [109] but also in modern computer games [108]. The basic principle of the A* algorithm is continuously searching the appropriate next free grid/node to approach the destination by considering the heuristic information and the condition of the current grid/node. The A* algorithm is based on Equation (6.14):

$$F(n) = g(n) + h(n) \tag{6.14}$$

Where ***F(n)*** is the cost function which estimates the **total** cost of moving from the start node to the goal node. ***h(n)*** is a heuristic function which estimates the cost of the shortest path from the current node to the goal node and ***g(n)*** is a movement function which estimates the cost of moving from the start to the current node. Usually, diagonal distance and Manhattan distance is used to estimate the movement cost movement ***g(n)***, and the Euclidean distance [100] is used to estimate the cost between the current node and the goal node, so the heuristic function ***h(n)*** can be expressed as:

$$h(n) = \sqrt{(x_{current} - x_{goal})^2 + (y_{current} - y_{goal})^2 + (z_{current} - z_{goal})^2} \tag{6.15}$$

Where $(x_{current}, y_{current}, z_{current})$ is the coordinate of the current position; $(x_{goal}, y_{goal}, z_{goal})$ is the coordinate of the goal position.

During the searching process, all accessible nodes are categorised into two lists: the open list and the closed list. The open list contains all accessible nodes that surround the current node and have not been checked. The closed list holds the nodes that have already been checked and have been to be added to the path. At the beginning of the search, the start node is allocated to the closed list. Its adjacent 26 neighbour nodes are placed on the open list. In 3D cases, the A* algorithm usually work with 3D environment modelling method such as GD see section 6.3, so one cell has 26 adjacent neighbour cells. The start node is the parent of its accessible adjacent nodes, and these are the children of the start node. If a node lies inside an

obstacle, then it is an invalid node. The selection of the next node is made by calculating the ***F(n)*** from each adjacent node to the goal. The node that has the lowest cost becomes the chosen node and is regarded as the new parent node. It will be removed from the open list and added to the closed list. The above procedures are repeated until reach the goal node. The shortest path is then obtained by backtracking from the goal node, initially to its parent, and from then to the parent's parent until the start node is reached. The A* algorithm is explained with the aid of Figure 6.12 below:

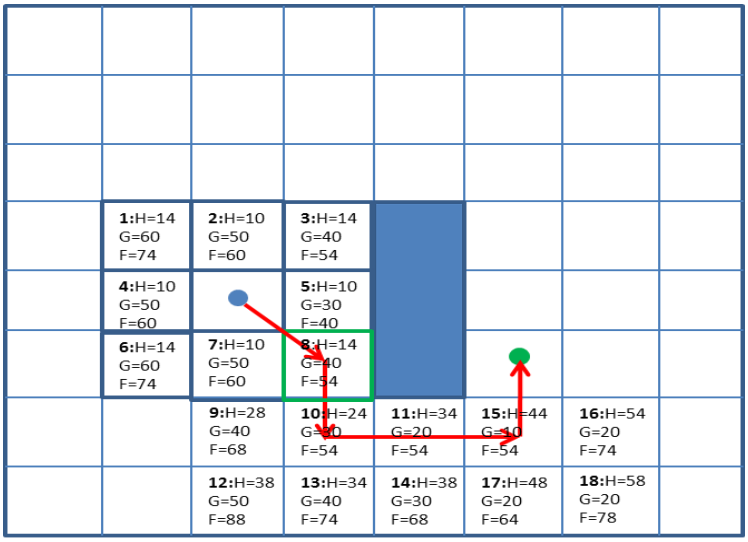


Figure 6.12: A* path-planning

The blue dot is the start node, which is placed on the closed list; nodes 1 to 8 are neighbouring nodes, which are placed on the open list; H indicates the distance cost from the current node to its neighbour nodes, 14 for diagonal distance cost, 10 for Manhattan distance cost; G indicates the distance cost from the neighbour node to the goal; F indicates the distance cost through the neighbour node to the goal, which is the sum of H and G. After applying the cost function ***F(n)*** to the neighbours, node 5 is found to have the least cost, so it is regarded as the next start node. The new searching starts at node 5 and repeats the same procedure above. However, the right of node 5 is an occupied area that is not accessible, so these will not be considered. The new starting node (node 5) has 4 neighbouring nodes (nodes 2, 3, 7, and 8) excluding its parent node and the inaccessible nodes. Among its neighbouring nodes, node 8 has the least cost, so it becomes the next start node. However, the updated cost of node 8 (F=60) is bigger than its initial cost (F=54) and its parent node is also the blue dot (initial starting node). Therefore, node 5 cannot be considered as the next start node, and node 8 is the next start node. After that, the above procedures are repeated until the goal node (green dot) is reached. The obtained path is indicated with red arrows in Figure 6.12.

The A* algorithm often combines with other path-planning algorithms such as RRT, PRM or with a *C-space* modelling algorithm like GD (as discussed above). This algorithm is capable of being used in higher dimensional environments as long as the environment can be abstracted as discrete grids/ nodes. Niu and Zhuo [157] introduced "cell" and "region" conception to represent 3D environments. Then, apply the A* algorithm.

6.4.9 RRT-A*

RRT-A* algorithm introduces the cost function of the A* algorithm to the RRT algorithm to optimise the performance [119]. As discussed in Section 6.2.4, the RRT algorithm can deal with the path-planning problem successfully. However, the obtained path is not optimal in terms of its length because the selection of nodes is random. The RRT-A* algorithm adds the cost function of A* to the process of nearest neighbour selection, which guides the path towards the target rather than being purely random.

6.4.10 Genetic Algorithms (GAs)

Genetic algorithms (GAs) are an adaptive heuristic search algorithm based on the evolutionary ideas of natural selection and genetics [113]. In terms of path-planning, 'Genes' are nodes that are waypoints on the path, and GAs use genetic operators to optimise the initial path. There are 5 genetic operators in GAs path-planning [114]:

- The cross operator: randomly choose two nodes from two paths; swap the remained path after the chosen node.
- The mutation operator: randomly choose one node from a path and replace it with a node that does not belong to any path.
- The mobile operator: random choose a node in the path and move it to a nearby position.
- The delete operator: randomly choose a node in the path, check its two adjacent nodes and connect them together. If remove the chosen node result in a shorter collision-free path, then delete this node from the path.
- The improve operator: this operator can only be used in the collision-free path. Select a node from the path and insert two new nodes in two sides of the selected node. Then, connect the 2 new nodes with a path. If the new path is feasible, remove the selected node.

The above genetic operators are applied to parent paths to generate optimised child paths. In GA, the parent path is defined as the initial path obtained from the previous path planning algorithm, which could be obtained by RRT or PRM. The parent paths should be line segments that connect the start and the goal via several intermediate nodes. Gorkem Erinc [116] and his colleagues use RRT to generate the parent paths and then apply GAs to optimise the parent path. The child path is the resultant path after encoded by genetic operators stated above. For example, consider the RRT path 1-3-5-7-9-11-13 is the parent path, where 1 is the start, 13 is the goal, and rest intermediate nodes 3, 5, 7, 9, 11 are the "genes" of this path; these "gene" can be mutated, crossed, and swapped with nodes to generate child paths.

GA is an optimization algorithm, so it needs the initial path. The main shortcoming of GAs is that those genetic operators stated above can be freely paired so that extremely large numbers of child path can be found and will create many infeasible paths [91].

6.5 Discussion and Summary

This chapter lists ten different path-planning algorithms, and a summary of the strength and weaknesses of each algorithm is shown in Table 6.1.

Table 6.1: Summary of path-planning algorithms (1)

Path-planning algorithms	Type	Able to extend to 3D	Advantages	Disadvantages
CD	Workspace modelling algorithm	Y	CD can be extended to 3D, which is known as GD. It can be applied to mobile robots and work with aerial mapping to create an occupancy grid map.	It needs to work along with searching algorithm, like A*.
VD	Workspace modelling algorithm, Roadmap	Y	The planned path keeps far away from nearby obstacles.	Difficult to apply in a 3D environment
VG	Workspace modelling algorithm, Roadmap	Y	Good performance in the regular and polygon-based environment.	The path keeps very close to obstacles. Difficult to be used in a cluttered environment.
PRM	Workspace modelling algorithm, Roadmap	Y	Guarantee to find a path as long as the sampling node can increase without boundary	Cannot guarantee the found path is optimal, difficult to find a path in a narrow gap, computation complexity
RRT	Roadmap, searching algorithm	Y	Has fast searchability, low time complexity, do not need a *C-space* modelling algorithm	Cannot guarantee the found path is optimal
APF	Potential field	Y	Do not need to work along with environment modelling algorithm, low time complexity, created path is smooth, real-time obstacle avoidance	Easy to fall in a local minimum

Table 6.2: Summary of path-planning algorithms (2)

Path-planning algorithms	Type	Able to extend to 3D	Advantages	Disadvantages
APF-SA	Potential field	Y	Do not need to work along with environment modelling algorithm, low time complexity, and can mitigate the local minimum problem	Still has the local minimum problem
A*	Searching algorithm	Y	It can find a relatively short and low-cost path, can combine with other environment modelling algorithm such as GD	Possible high time complexity
RRT-A*	Roadmap, searching	Y	Has fast searchability, low time complexity, improved the path by forcing the search direction towards to the goal	Cannot guarantee the found path is optimal
Dijkstra's	Searching algorithm	Y	Guarantee to find the shortest path	High time complexity
GAs	Bio-inspiration	Y	optimise the initial path	High time complexity and will create many unfeasible paths

GD method is suitable for the storage pond environment as it can be used to model the aerial survey results since the aerial mapping is conducted in the equal-spaced grid, see Section 3.1. The constraint of decomposition resolution of the GD depends on the aerial survey sampling resolution of the environment. To fill the aerial survey results to GD to create a 3D occupancy grid map, their resolution should be identical to each other to ensure data consistency. This method needs work along with other sampling searching algorithms such as the A* algorithm to find a collision-free path.

VD, VG, and PRM are typical roadmap methods. VD is difficult to apply in an irregular 3D environment because of that have the difficulty of calculating the Voronoi Diagram by studying lines and polygons, finding the vertices and nodes, and creating a tree to find the path in 3D [110]. The old storage pond is an irregular 3D environment, so VD will not be considered. Since the path obtained by the VG is as close as possible to the obstacle, a minor control error of the robot will cause a collision to obstacles. Hence, VG also will not be considered for the storage pond environment. The PRM is not a good approach for solving the path-planning problem in a storage pond either. Because sample nodes are generated randomly, the PRM is not good at finding a path in a narrow gap [90]. The highly disordered old pond might have a narrow gap between adjacent spent fuel containers, so the PRM is not a good option.

RRT and RRT-A* approach are good candidates for solving the path-planning problem in the storage pond. They can search the workspace regardless of the geometry of the environment,

which means they can work in irregular 3D environments [111]. In addition, it has a low time complexity compared to PRM.

APF and APF-SA do not need to model the geometry of the environment, and it can create a smooth path and achieve real-time obstacle avoidance [70]. The performance of APF and APF-SA is strongly affected by parameters of the APF function such as repulsion index and attraction index, so those parameters need to be chosen carefully. Besides the local minimum problem, it is a good option for solving the storage pond path-planning problem.

The A* algorithm is an excellent option for solving the path-planning problem in the storage pond. Because the A* algorithm can work along with the GD discussed earlier. Initially, the storage pond needs to be decomposed into a finite number of grids by utilising the GD method. Then, convert it to an occupancy grid map. After that, apply the A* algorithm to find out a low-cost collision-free path.

GA is an optimisation algorithm, and it can be used to optimise the original path. However, it will create a lot of unfeasible paths due to the random selection of genetic operators [91]. Therefore, GA will not be considered for the storage pond path-planning problem.

From these discussions above, the RRT (RRT-A*), APF (SA-APF), and A* appear to be most applicable to the storage pond problem. The next chapter will compare and discuss the performance of RRT (and RRT-A*), APF (and SA-APF), and A* in proposed 3D test pond cases by MATLAB simulations.

Chapter 7

Path-planning Simulations for Storage Ponds

In the previous chapter, ten different path-planning algorithms are carefully reviewed and discussed, five of which can be potentially used to solve the path-planning problem of closed underwater environments. To evaluate the best path-planning algorithm for storage pond navigation, path-planning simulators were developed for each candidate algorithms to investigate their feasibility and performance in different pond environments. The five path-planning algorithms are the artificial potential field algorithm (APF), the simulated annealing-artificial potential field algorithm (SA-APF), the A* algorithm (A*), the randomly exploring random tree algorithm (RRT), and the randomly exploring random tree-A* algorithm (RRT-A*). This chapter presents the MATLAB simulation results for various storage pond-like environments.

The following symbols are commonly used in this chapter: The black dot and the red dot represent the start and the goal, respectively.

7.1. Systematic Simulations of the Proposed Environment

To study the performance of each algorithm, the first set of simulations is systematic simulations of a pond environment in which cuboid clutter of varying sizes, distributed randomly. The assumptions and the geometry configurations of the storage pond-like environment are listed below:

- The pond has a length of 50m, a width of 25 m, and a height of 10 m.
- 32 different sizes, heights, and randomly placed cuboid obstacles inside the pond.
- The mobile robot is regarded as a particle that has three degrees of freedom without considering its size.
- The coordinate of the starting point is (1, 1, 10) m, its height is reduced in steps of 2 m. The coordinate of the goal point is (43, 19, 3) m, and maintains constant.

The proposed storage pond-like environment is shown in Figure 7.1, which can be seen to be an appropriate abstraction of many pond environments, see the pond figures in Section 3.4. The purpose of the first set of simulations is to compare the quality of the paths generated by the different algorithms and to investigate how the path varies for different path-planning algorithms when the start point varies in height. This is of interest because the difficulty of planning a collision-free path near the water surface and near the pond bottom is different, and they are typical work scene for an underwater robot. The mobile robot will be released on the surface of the water and deployed to navigate at the bottom of the pond to measure pH level, radiation level, and temperature in different places.

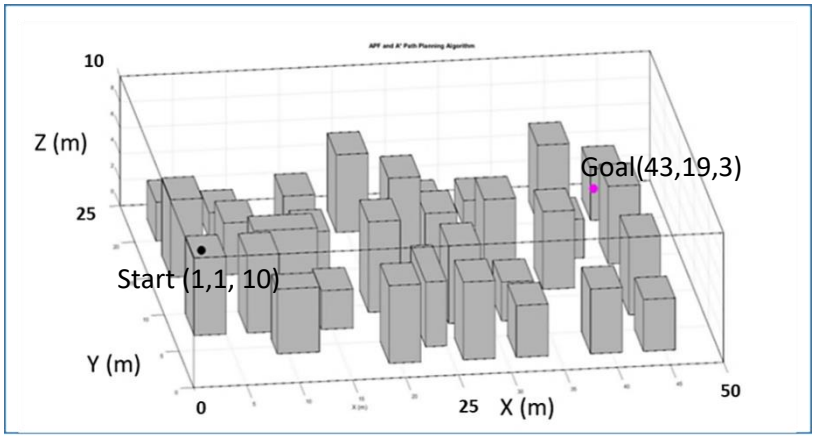
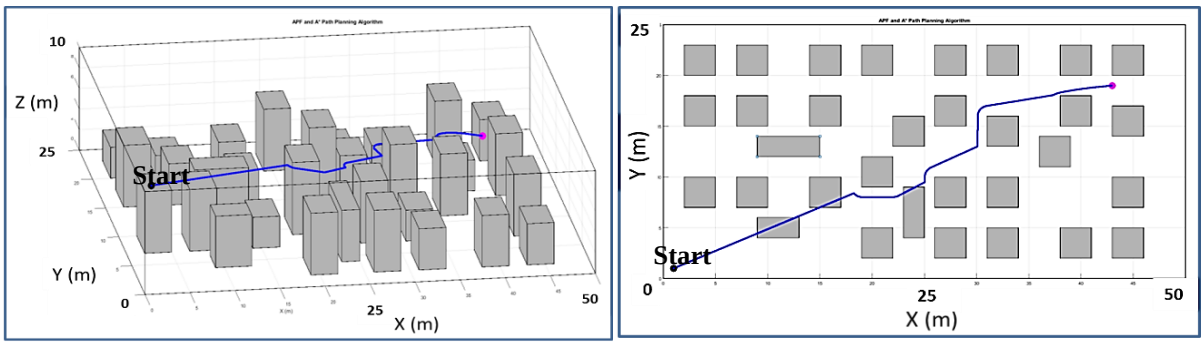


Figure 7.1: Pond environment

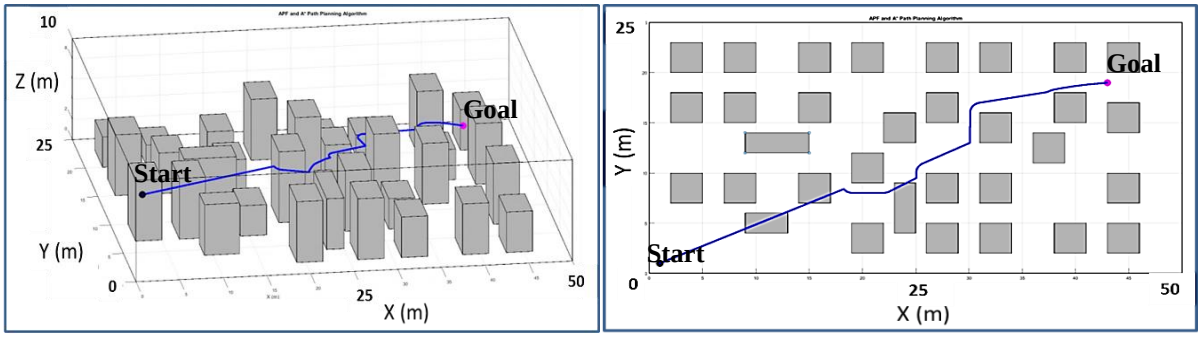
7.1.1 Performance of APF Simulation

Figures 7.2 to 7.6 show the collision-free path found by the APF simulator that maintains the goal's height but decreases the start's height from 10 meters to 2 meters. The APF simulator was built based on the improved attractive and repulsive functions given in Section 6.2.6.1 and 6.2.6.2.



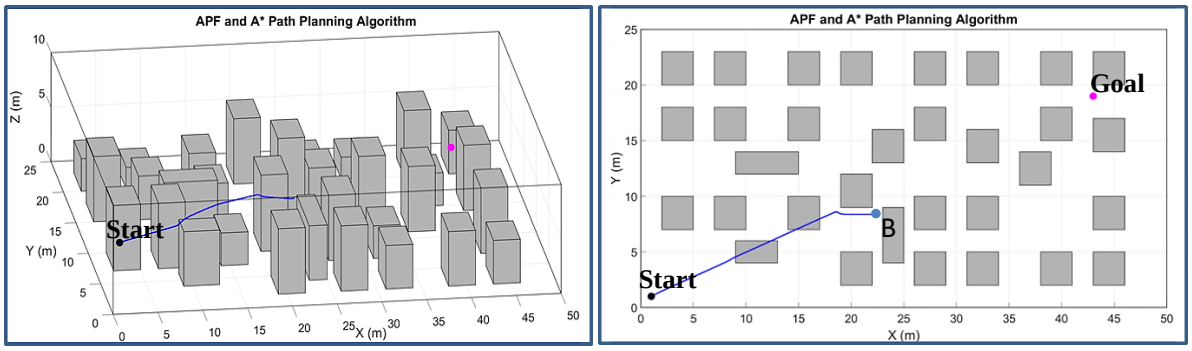
(a) 3D view (b) Plan view

Figure 7.2: APF path for starting position of (1, 1, 10)



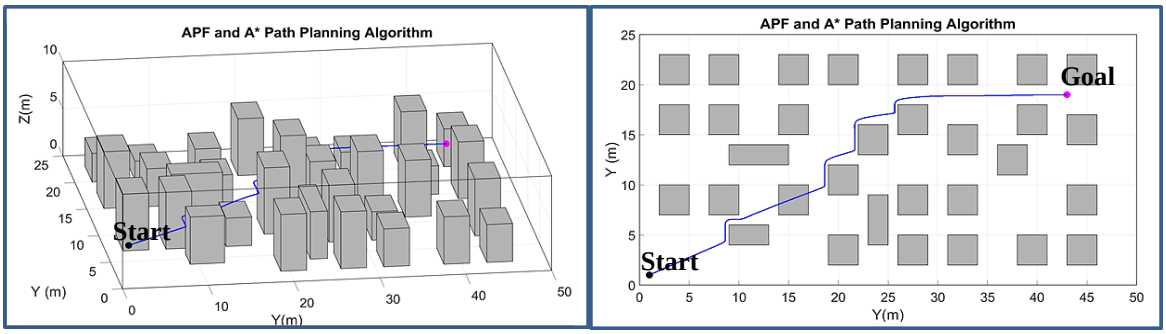
(a) 3D view (b) Plan view

Figure 7.3: APF path for starting position of (1, 1, 8)



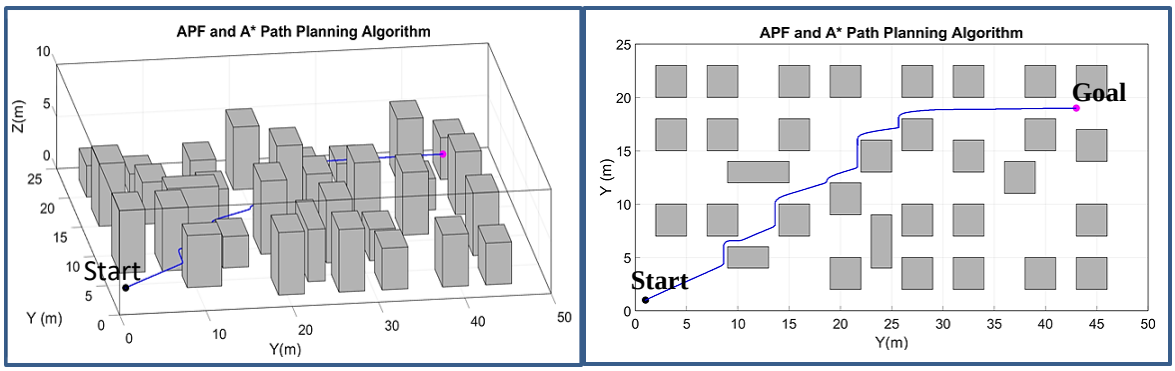
(a) 3D view (b) Plan view

Figure 7.4: APF path for starting position of (1, 1, 6)



(a) 3D view (b) Plan view

Figure 7.5: APF path for starting position of (1, 1, 4)



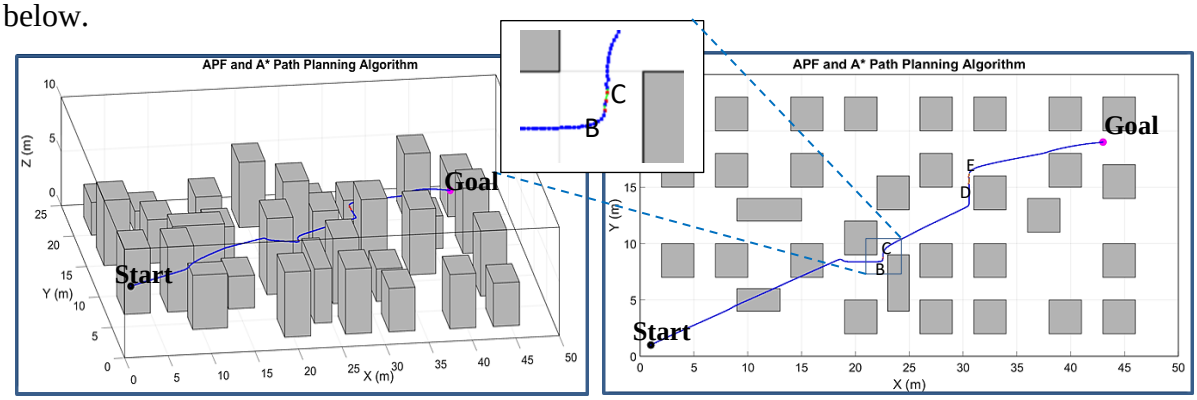
(a) 3D view (b) Plan view

Figure 7.6: APF path for starting position of (1, 1, 2)

The figures above show that a collision-free path from the start to the goal is successfully found for starting heights of 10m, 8m, 4m, and 2m. However, when the starting height is 6m, the path terminated at B (see Figure 7.4). This means the robot which starts at (1, 1, 6) will encounter a local minimum at B on the way to its goal (43, 19, 3). One way to avoid this is to change the attraction or repulsion coefficients, but it will not resolve this problem because local minima are not eliminated but moved. The above results show that unanticipated local minima can easily arise in complex environments. Because of this uncertainty, pure APF is not a good choice for the storage pond application. However, combining APF with another algorithm, such as simulated annealing (SA) could mitigate the local minimum problem.

7.1.2 Performance of SA-APF Simulations

The SA-APF algorithm provides a mechanism for robots to escape from local minima; see Section 6.4.6 for details. When no local minimum is encountered, it executes the APF part. When the path becomes trapped in a local minimum, it activates the SA algorithm. After escaping from the local minimum, the APF is executed again. The SA-APF simulation results with a starting point height of 10 m, 8 m, 4 m, and 2 m are identical to that of the APF simulation results in section 7.1.1 because the SA part only activated when encountering a local minimum. The SA-APF simulation result with a starting point height of 6 m is shown below.



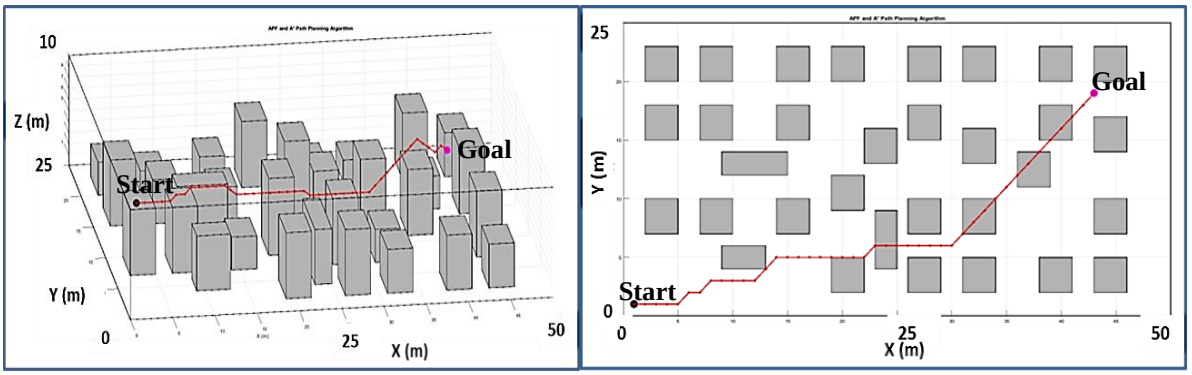
a) 3D view b) Plan view

Figure 7.7: SA-APF path for starting position of (1, 1, 6)

The green/red path in Figure 7.7 is the SA-APF path, which is activated at B and deactivated at C. With the help of the SA algorithm, the robot is able to escape from the local minimum and successfully reach the goal (the same in D and E). However, the SA algorithm can only mitigate the local minimum problem but cannot guarantee to solve all local minimum problems, as illustrated in section 7.2.1.

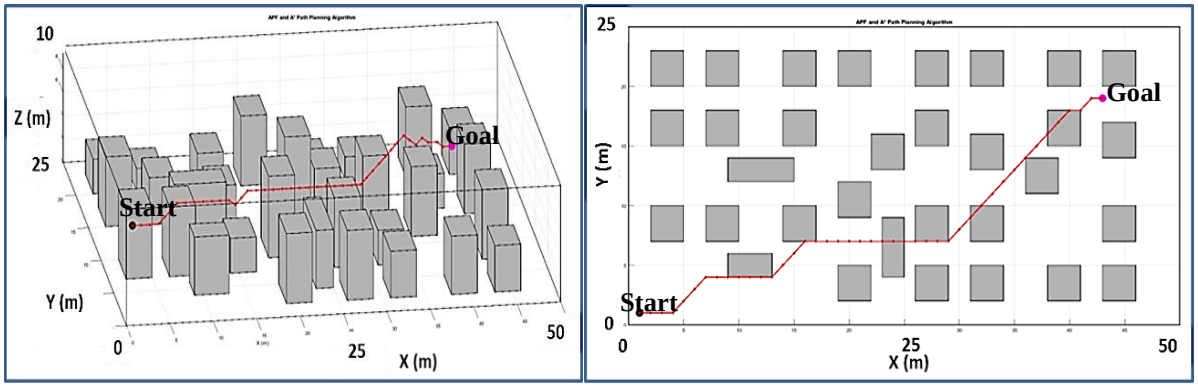
7.1.3 Performance of A* Simulation

The A* simulation results for scenarios described in Section 7.1.1 are shown in Figures 7.8 to 7.12.



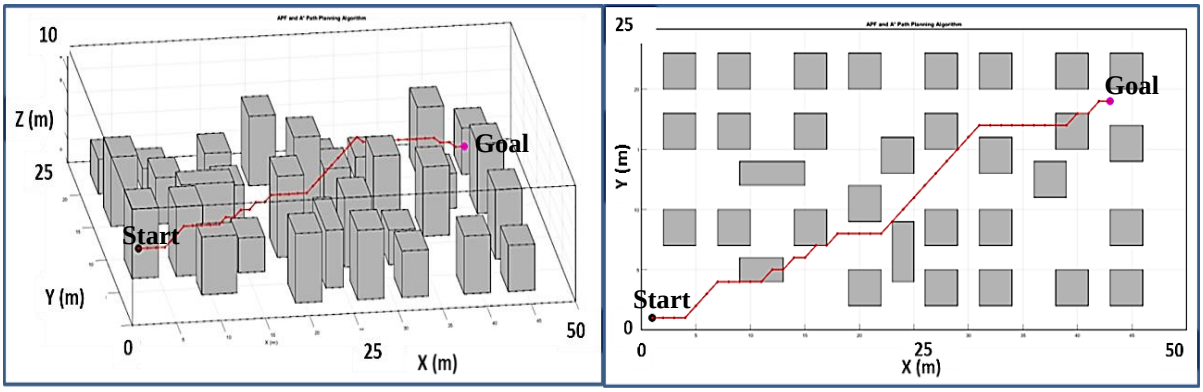
a) 3D view b) Plan view

Figure 7.8: A* path for starting position (1, 1, 10)



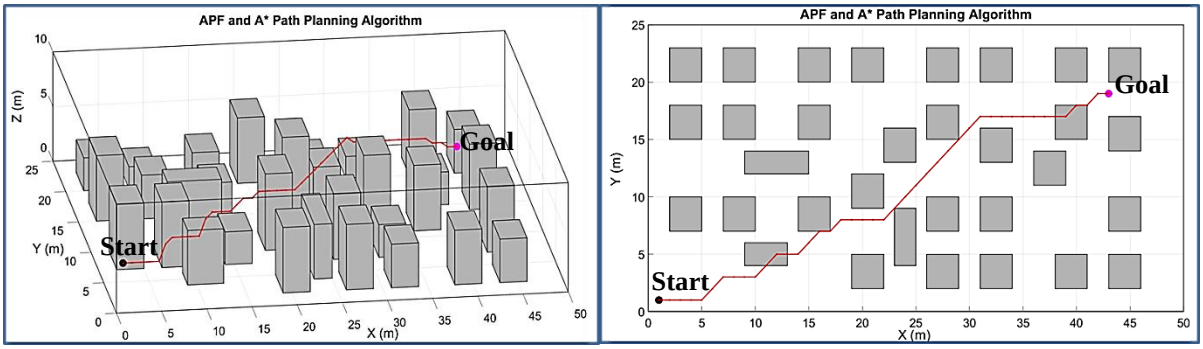
a) 3D view b) Plan view

Figure 7.9: A* path for starting position (1, 1, 8)



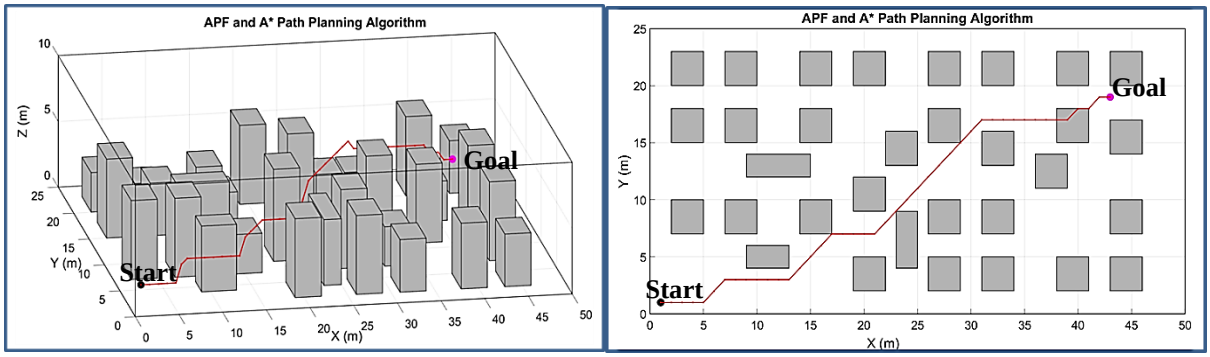
a) 3D view b) Plan view

Figure 7.10: A* path for starting position (1, 1, 6)



a) 3D view b) Plan view

Figure 7.11: A* path for starting position (1, 1, 4)



a) 3D view b) Plan view

Figure 7.12: A* path for starting position (1, 1, 2)

The A* algorithm successfully found a collision-free path that connects the start and the goal in all test cases, although the paths have a zigzag.

7.1.4 Performance of RRT and RRT-A* Simulations

The RRT and RRT-A* simulation results for the scenarios described in section 7.1.1 are shown in Figures 7.13 to 7.17.

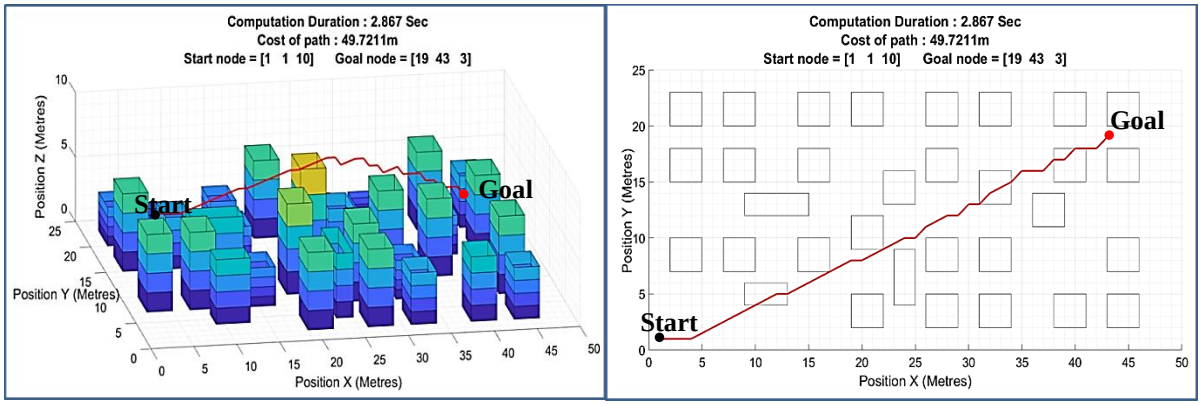


Figure 7.13a: RRT-A* path for starting position (1, 1, 10)

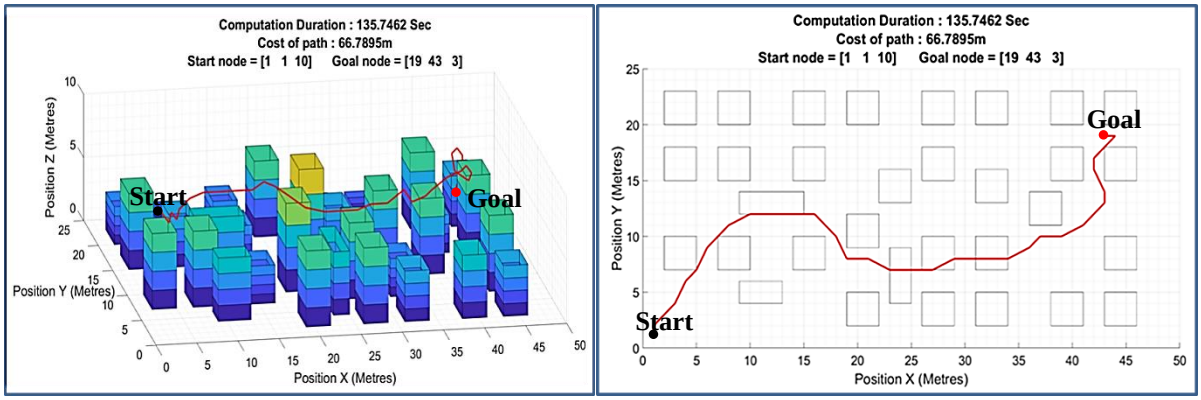


Figure 7.13b: RRT path for starting position (1, 1, 10)

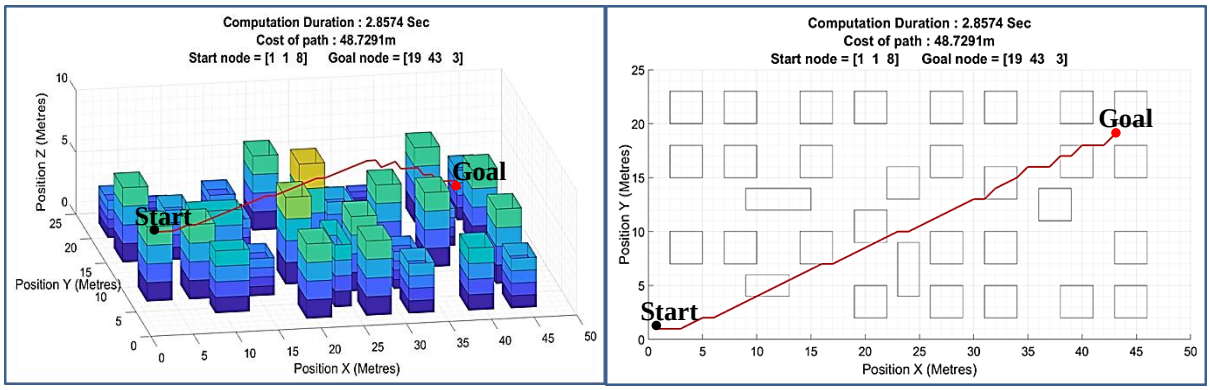


Figure 7.14a: RRT-A* path for starting position (1, 1, 8)

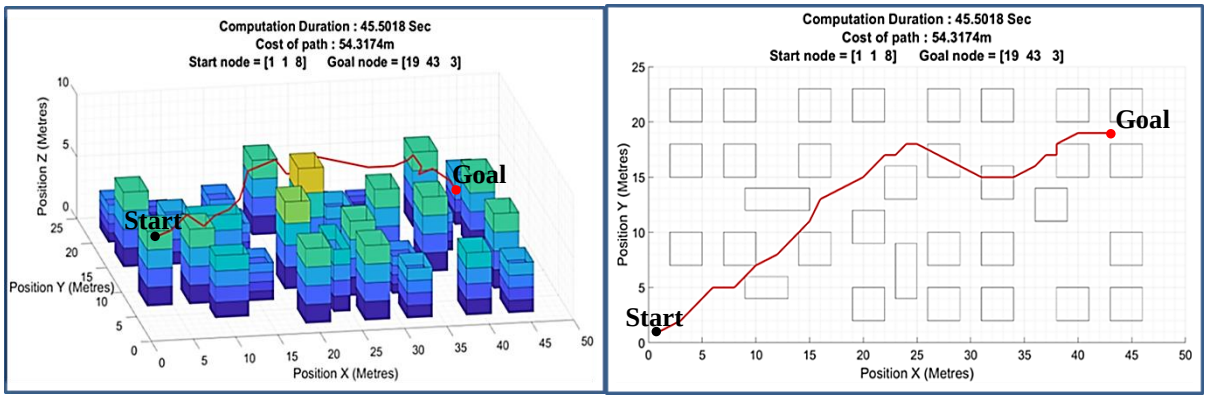


Figure 7.14b: RRT path for starting position (1, 1, 8)

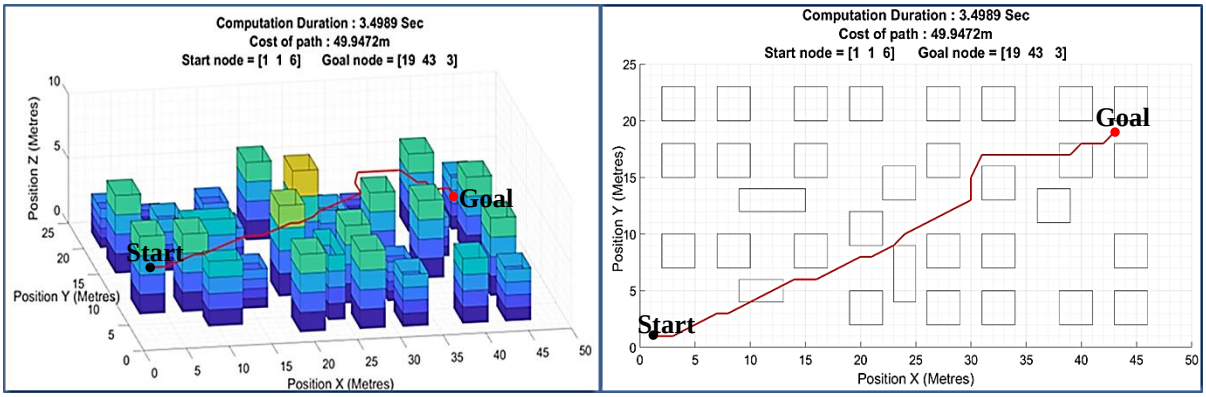


Figure 7.15a: RRT-A* path for starting position (1, 1, 6)

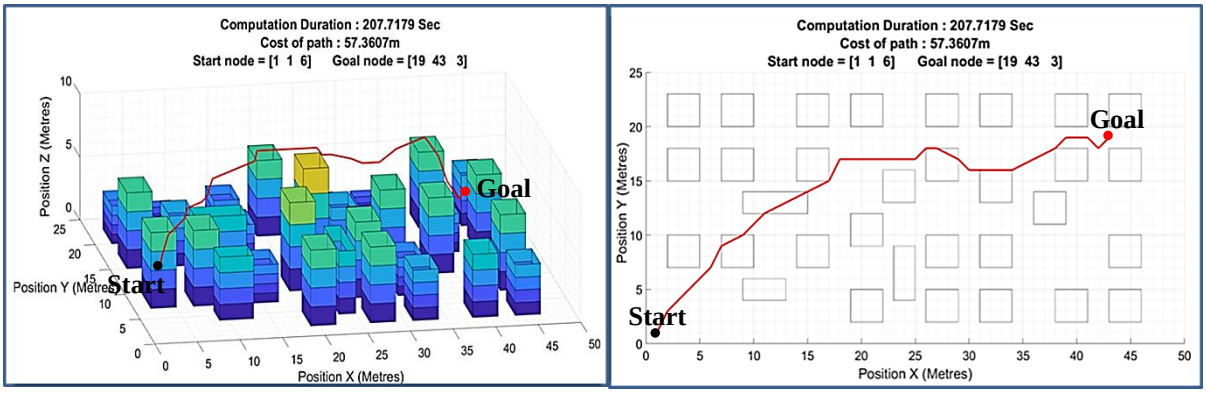


Figure 7.15b: RRT path for starting position (1, 1, 6)

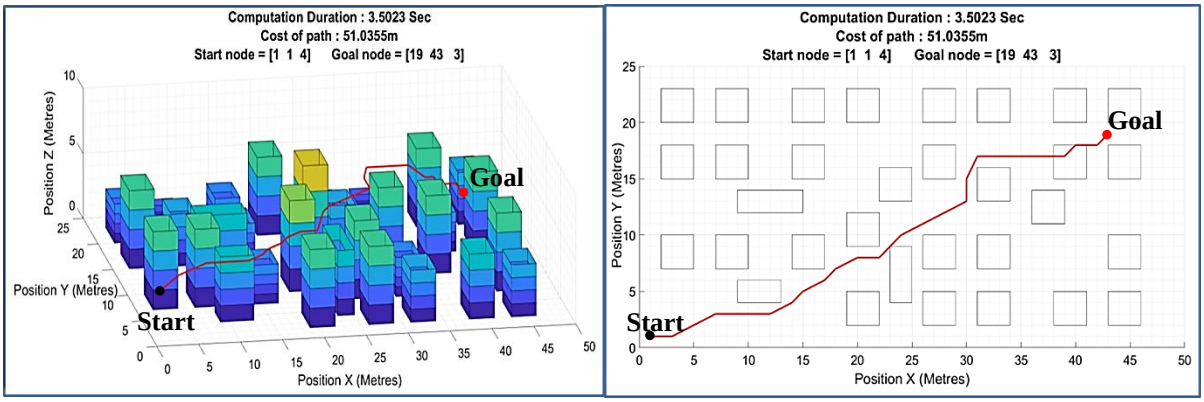


Figure 7.16a: RRT-A* path for starting position (1, 1, 4)

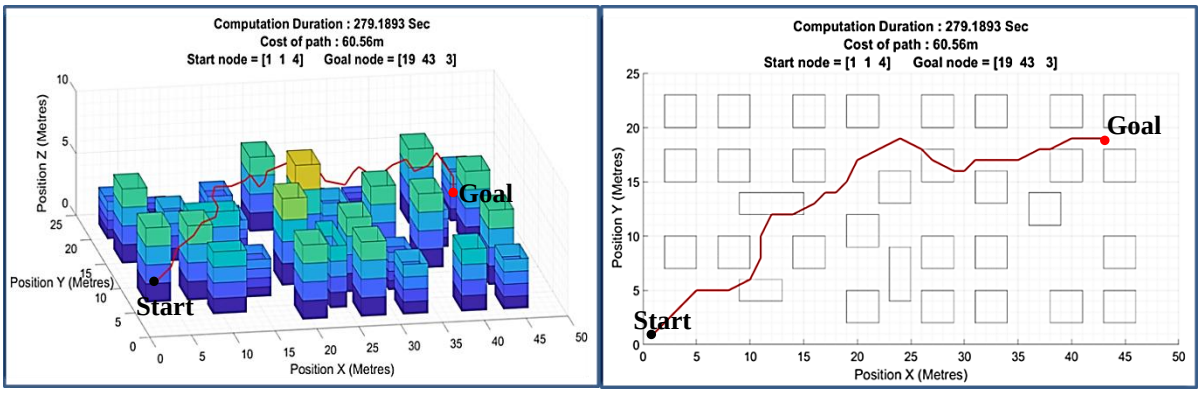


Figure 7.16b: RRT path for starting position (1, 1, 4)

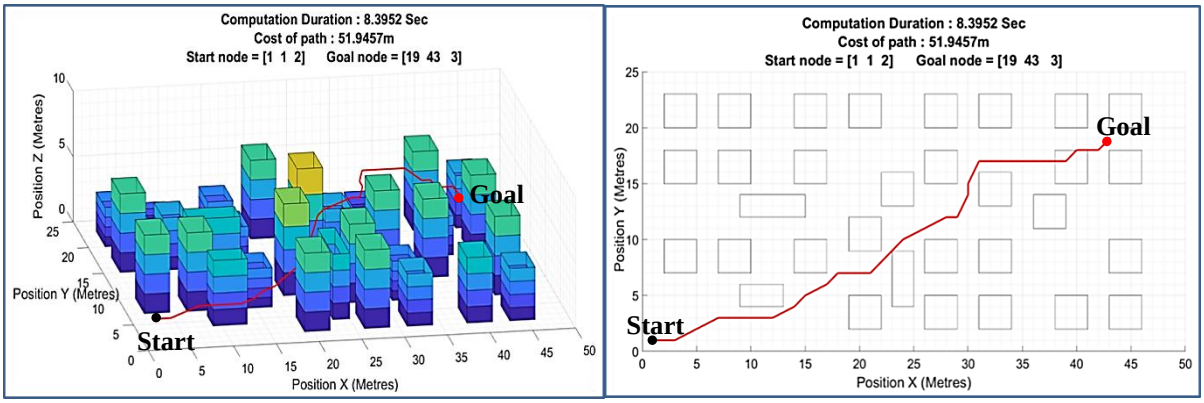


Figure 7.17a: RRT-A* path for starting position (1, 1, 2)

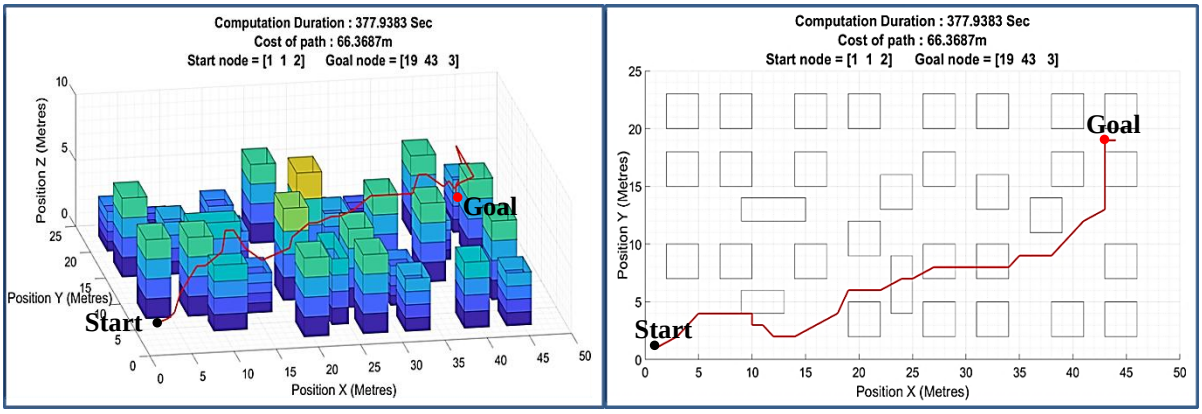


Figure 7.17b: RRT path for starting position (1, 1, 2)

In all test cases, both the RRT and RRT-A* algorithms are capable of finding a collision-free path that connects the start and the goal. However, the RRT-A* results are better than the RRT results because its paths are significantly shorter than the RRT path. Five repeating tests about the path length and simulation run time of RRT and RRT-A* are recorded in Table 7.1 and Table 7.2. The comparison of path length and simulation run time for five mentioned path-planning algorithms are shown in Table 7.3 and 7.4, respectively.

Table 7.1: Path length of RRT and RRT-A*

Test time	The path length of RRT 10m (m)	The path length of RRT-A*10m (m)	The path length of RRT 8m (m)	The path length of RRT-A*8m (m)	The path length of RRT 6m (m)	The path length of RRT-A*6m (m)	The path length of RRT 4m (m)	The path length of RRT-A*4m (m)	The path length of RRT 2m (m)	The path length of RRT-A*2m (m)
1	62.1867	49.7211	56.4328	48.7291	57.3607	49.9472	60.2855	51.0355	66.533	54.5975
2	64.0886	49.7211	54.3174	48.7291	56.1711	49.9472	63.6693	51.2955	66.3687	53.0355
3	66.3876	49.7211	59.8939	48.7291	69.3835	49.9472	60.56	51.2955	61.5568	52.5673
4	66.7895	49.7211	54.1656	48.7291	58.4201	49.9472	63.7067	51.2955	63.5201	53.5396
5	57.0363	49.7211	56.3526	48.7291	52.5946	49.9472	65.1507	51.5396	54.8567	53.864
Average (m)	63.2977	49.7211	56.2325	48.7291	58.7860	49.9472	62.6744	51.2923	62.5671	53.5208

Table 7.2: Simulation run time of RRT and RRT-A*

Test time	Run time of RRT 10m (m)	Run time of RRT -A*10m (s)	Run time of RRT 8m (s)	Run time of RRT -A*8m (s)	Run time of RRT 6m (m)	Run time of RRT-A*6m (s)	Run time of RRT 4m (m)	Run time of RRT-A*4m (s)	Run time of RRT 2m (s)	Run time of RRT-A*2m (s)
1	177.306	3.549	144.558	3.107	207.718	3.583	122.771	5.339	115.422	4.432
2	117.7517	2.856	45.502	3.133	105.972	3.398	117.532	3.051	377.938	4.734
3	114.654	3.168	126.466	3.183	109.156	3.356	279.189	3.548	183.078	5.418
4	135.7462	3.122	35.458	3.207	113.283	3.285	125.214	3.311	295.607	5.463
5	118.9751	2.867	28.019	2.857	123.956	3.499	133.994	3.502	106.928	8.395
Average(s)	132.886	3.112	76.006	3.097	132.017	3.424	155.74	3.75	215.795	5.688

Table 7.3: Path length of RRT, A*, APF/APF-SA, and RRT-A*

Starting Height (m)	Pythagorean distance (m)	The path length of RRT (m)	The path length of RRT-A* (m)	The path length of A* (m)	The path length of APF/SA-APF (m)
10	46.228	63.2977	49.7211	51.6807	49.0594
8	45.967	56.2325	48.7291	51.4450	48.8079
6	45.793	58.7860	49.9472	50.4094	49.0657
4	45.706	62.6744	51.2923	51.0450	51.8240
2	45.706	62.5671	53.5208	51.6807	51.8259

Table 7.4: Simulation run time of RRT, A*, APF/APF-SA, and RRT-A*

Starting Height (m)	Simulation run time of RRT (s)	Simulation run time of RRT-A* (s)	Simulation run time of A* (s)	Simulation run time of APF/SA-APF (s)
10	132.886	3.112	292	145
8	76.006	3.097	273	141
6	132.017	3.424	250	142
4	155.74	3.75	251	178
2	215.795	5.688	249	191

Data recorded in the above tables show that the RRT-A* algorithm can generate a short collision-free path and cost less simulation run time compared to the RRT algorithm. The reason for that is that the exploring tree grows randomly in the RRT algorithm. Hence, many nodes are sampled at unnecessary locations so it takes longer to find a collision-free path to the goal and also results in a longer path. The exploring tree generated by the RRT-A* algorithm is more disciplined because the A* algorithm will reduce the number of explored

nodes by choosing sampling nodes that have a smaller cost to the goal. Therefore, the path found using RRT-A* is shorter than the path found using RRT.

In the systematic simulations, A*, SA-APF, and RRT-A* all successfully planned a collision-free path with close path length, so they will be considered in the following simulations.

7.2. Simulations with the Goal inside a Hollow Canister

The results discussed in section 7.1 showed that SA-APF, A*, and RRT-A* all performed well. Therefore, the test case considered in this section will focus on these algorithms. As discussed in Section 5.8, in some legacy ponds, the spent fuel canisters are hollow with an open-top, and it is likely that pond managers would be interested to send the AUV into a hollow canister to take measurements. Hence, path-planning, when the target is inside a hollow canister, is of interest. Figure 7.18 shows a legacy pond that contains some open storage canisters.

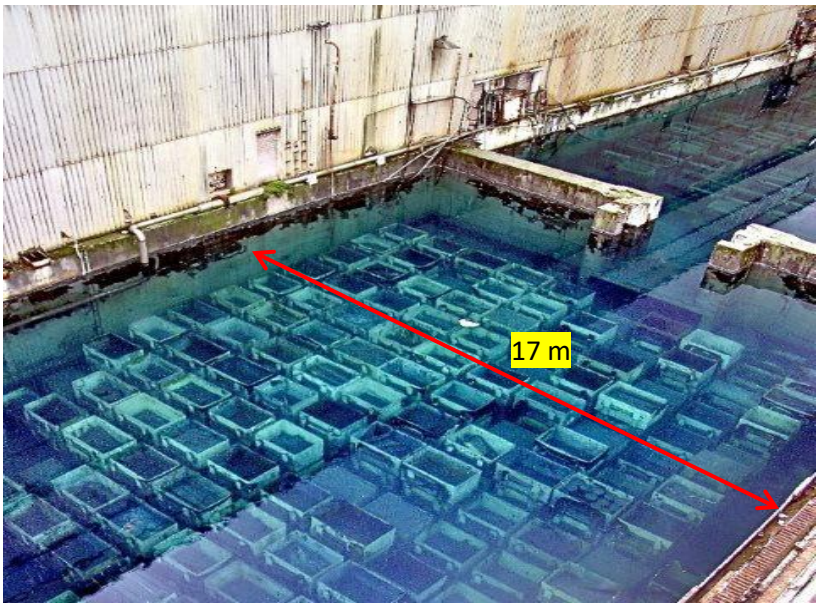


Figure 7.18: Open top nuclear storage canister []

The assumptions and the geometry configurations of the hollow canisters storage pond-like environment are listed below:

- The pond has a length of 50 m, a width of 25 m, and a height of 10 m.
- A 10m*10m*6m hollow canister was located in the middle of the pond.
- The thickness of the canister wall is 1m.
- The starting point at (6, 18, 1) (a point near the bottom of the pond and away from the hollow canister), the goal at (28, 10, 1) (a point near the bottom of the canister).

The proposed storage pond-like environment is shown in Figure 7.19. The purpose of the following simulations is to test if the selected path-planning algorithms are feasible in this case and how they perform.

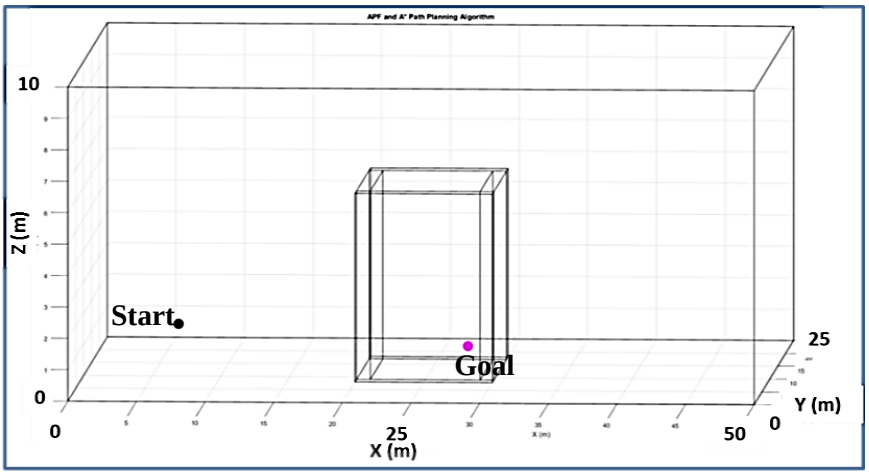
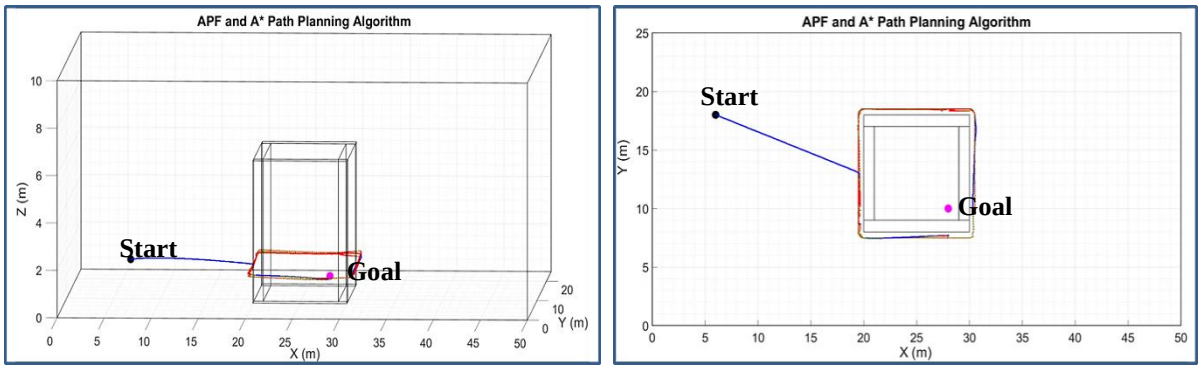


Figure 7.19: Hollow canister scene

7.2.1 Performance of SA-APF Simulation



(a) (b)

Figure 7.20: SA-APF path in a hollow canister scene (a) 3D view (b) Plan view

The above results show that the SA-APF simulator cannot find a feasible path that reaches the goal. As Figure 7.20 shown, the red path is the SA path which was employed to escape from the local minimum, but the red path loops around the outside of the hollow canister and cannot find a way to enter it. An analysis of this situation is discussed in section 7.2.4.

7.2.2 Performance of A* simulation

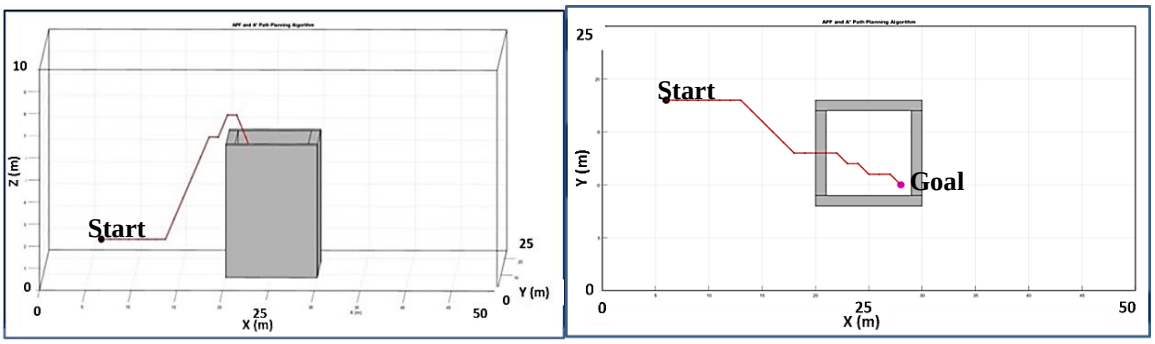


Figure 7.21: A* path in hollow canister scene

The above results show that the A* algorithm successfully finds a collision-free path that reaches the goal. The length of the path is 29.513 m.

7.2.3 Performance of RRT-A* simulation

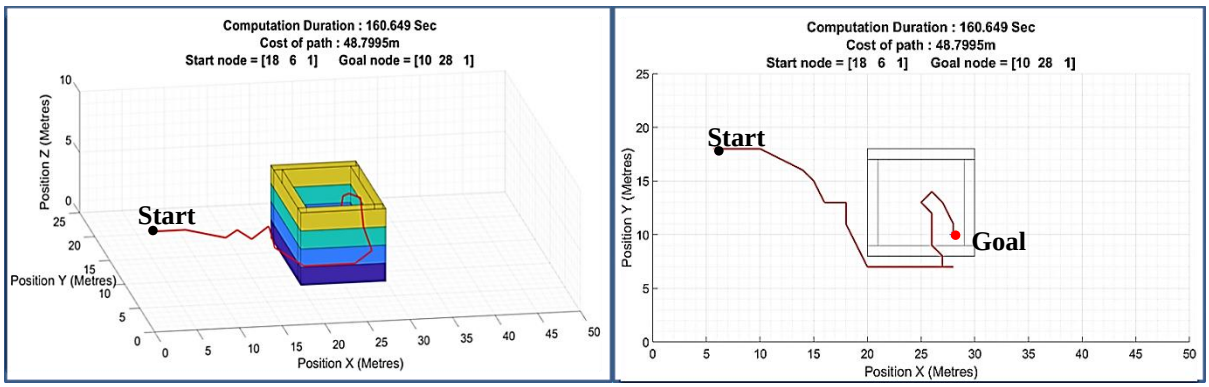


Figure 7.22: RRT-A* in hollow canister scene

The above results (one of the results of the five tests) show that the RRT-A* algorithm successfully finds a collision-free path that connects the start and the goal. However, sometimes the path takes some unnecessary steps which deviate markedly from the obvious path. Because the exploration of the next state is random, the RRT-A* algorithm generates a longer path, which is unintuitive, but successful. The path lengths of five repeating tests are shown in Table 7.5. Compared with the A* path (29.513m), the RRT-A* path is much longer.

Table 7.5: Path length and simulation run time of RRT-A*

Test time	Length of RRT-A* path (m)	Run time of RRT -A* (s)
1	48.7995	160.649
2	36.653	295.2312
3	32.231	322.7477
4	32.3592	541.007
5	38.6213	133.4345
Average(s)	37.7328	290.6139

7.2.4 Discussion of SA-APF

The reasons why the SA-APF algorithm cannot find a feasible path to reach a goal inside a hollow canister are explained with the aid of Figure 7.23. Near the starting position, the attraction force will significantly overwhelm the repulsion force, because the repulsion potential only exists around the hollow canister within a certain range. As a result, the path will be pulled closer to the bottom of the hollow canister. When the path gets close to the canister and with a height lower than the height of the canister, the path receives a horizontal repulsion force from the canister's wall. The combination of the repulsion force and attraction force will produce a downward composite force. The red arrow indicates the resultant force on the robot.

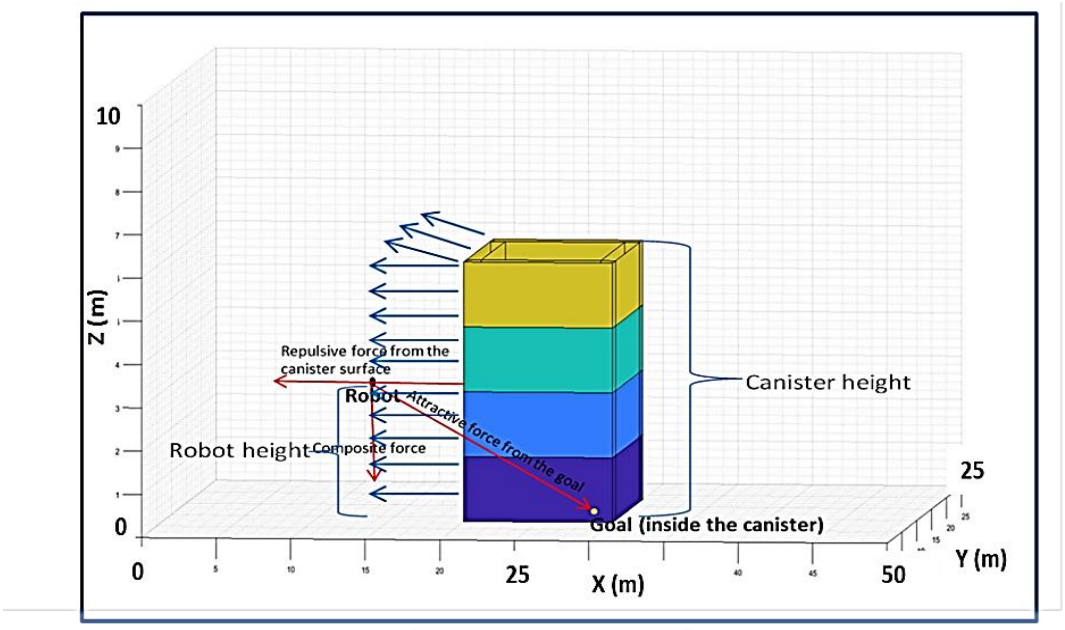
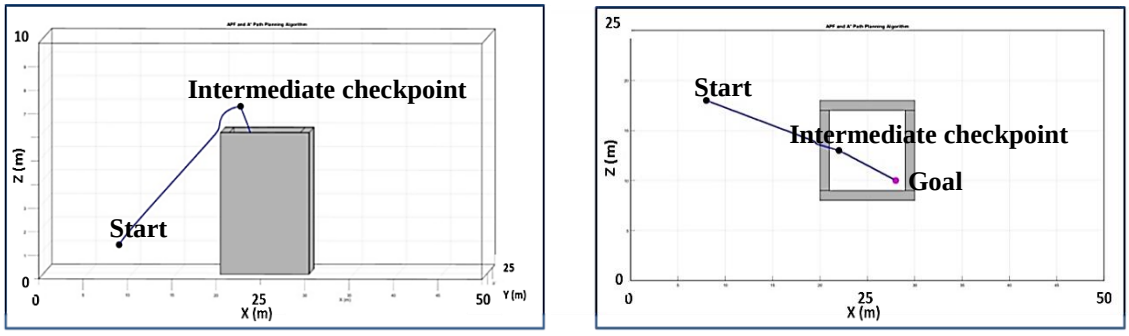


Figure 7.23: Force analysis

The force analysis in Figure 7.23 shows that the composite force always acts downwards. As a result, the AUV moves downwards and can never ascend to cross the canister's wall and reach the goal. Once the robot's height is lower than the hollow canister's height, the composite force will always act downwards, so even with the aid of the SA algorithm the path still cannot reach the goal. One solution is to place an intermediate checkpoint near the top of the hollow canister. The robot will first move to the intermediate checkpoint, then move to the final goal. As shown in Figure 7.24.



(a) (b)

Figure 7.24: Intermediate checkpoint (a) 3D view (b) Plan view

To find out **WHERE** the intermediate point should be placed to successfully find a collision-free path to the goal, the line-of-sight approach is introduced. Line of sight in this context is defined as the line that connects the intermediate checkpoint and the goal and not intersects the canister wall. In other words, the intermediate checkpoint can be observed from the goal. The following works validate that the line-of-sight approach can be used to find a feasible intermediate checkpoint.

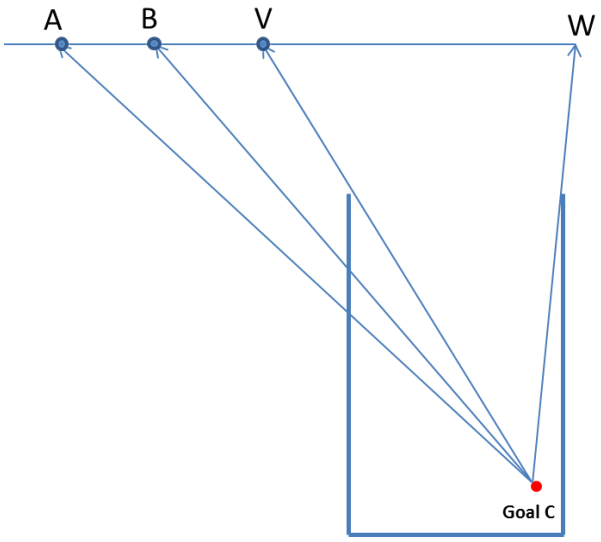


Figure 7.25: Line of sight

A, B, V, W are four intermediate checkpoints in the same horizontal plane. For points A and B, their line-of-sight AC and BC cross the canister wall, so they are not visible to goal C. For point V and W, their line-of-sight VC and WC do not cross the canister wall, so they are visible to goal C. Figure 7.26 is a test case of the scene shown in Figure 7.25.

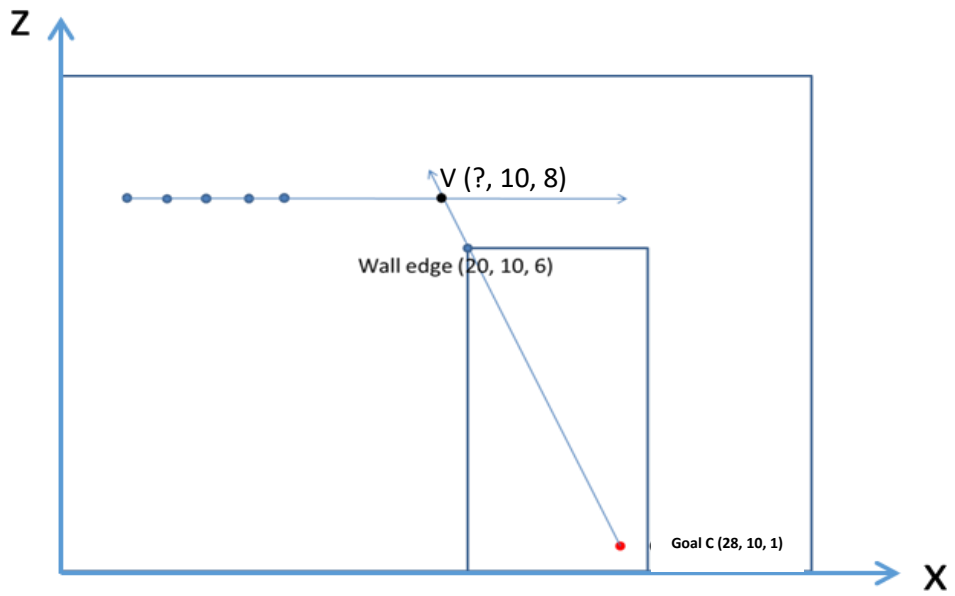


Figure 7.26: Testing case

According to the line-of-sight method, point V, wall edge, and the goal should be on the same straight line, the Equation of line VC is

$$z = -0.625x + 18.5$$

The estimated coordinate of point V is (16.8, 10, 8). Set V as the start and run the APF simulator. Results in Figure 7.27 validate that intermediate checkpoint V satisfies the line-of-sight can be used to find a collision-free path to the target.

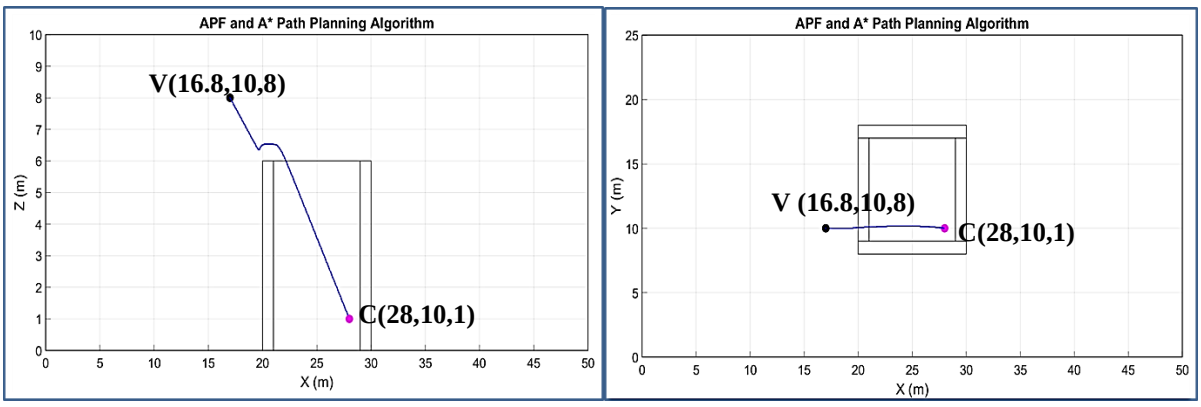


Figure 7.27: Intermediate goal V

As for point B (16, 10, 8), it does not satisfy the line-of-sight. Hence, it cannot be used to find a feasible path to the target, see the results in Figure 7.28: the path fails to reach the goal.

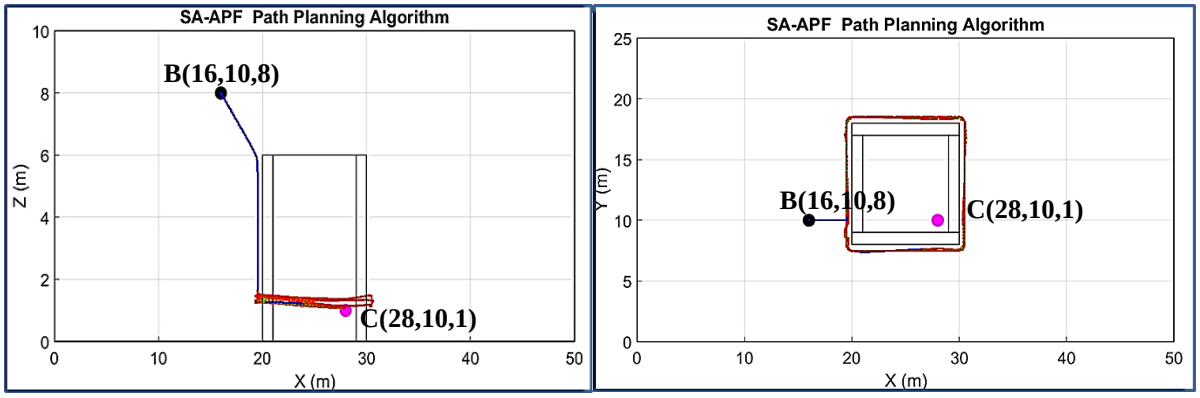


Figure 7.28: Intermediate point B

7.3 Discussion and Summary

This chapter investigated the performance of five path-planning algorithms used in the proposed pond-like geometry. The simulation results reported in irregular pond-like geometry test case show that pure APF easily falls to the well-known local minimum problem and pure RRT creates a much longer path. The remaining SA-APF, A*, and RRT-A* algorithms appear to be capable of planning a short and collision-free path.

The simulation results reported in the hollow canister geometry test cases show that SA-APF algorithm cannot guarantee to find a feasible path without placing an appropriate intermediate checkpoint. Both the RRT-A* and A* algorithm can find a feasible path to reach the goal placed at the bottom of the hollow canister, but A* path is significantly shorter than RRT-A* path. Simulation results in this chapter suggest the A* algorithm has better performance in the proposed pond cases among the path-planning algorithms considered in this chapter. Unlike APF and SA-APF, A* algorithm does not have the local minimum problem and, it is capable of planning a short collision-free path. Most importantly, as discussed in Section 6.5, A* algorithm can work along with the occupancy grid map obtained by aerial mapping, so it can be used for solving the path-planning problem in storage ponds. Hence, the focus of subsequent work will be A*algorithm.

Chapter 8

3D Grid Map Construction and A* Path-planning Algorithm

8. 1 Introduction

The chapter has a strong connection with outcomes from Chapter 2, 3, 6, and 7. Chapter 2 proposed an echo-sounding depth measurement technique. Chapter 3 proposed an echo-sounding based 'aerial mapping' approach to gathering information about the distribution of the material in a pond. The information enables a depth measurement topological height map of the enclosed underwater environment to be constructed. Chapter 6 has discussed the path-planning algorithms appropriate for storage pond environments. Chapter 7 compares the performance of path-planning algorithms in different environments by simulation and finds that the A* algorithm has better performance than other algorithms. This chapter presents a method for constructing a point cloud grid map based on data obtained by aerial mapping and then uses A* path-planning algorithm to find a collision-free path on the grid map. Rodenberg [128] presents a workflow for pathfinding through a point cloud representation of an indoor environment. Phung [129] developed a data processing system to reconstruct a 3D point cloud of the environment and generate waypoints for UAV.

This chapter presents a method that creates a 3D model based on the data obtained by a 2D aerial scan and used for planning a collision-free path. The whole process includes a complete pond topology acquisition, occupancy grid map reconstruction, and route planning.

8.1.1 General Description

Underwater path-planning refers to the complete pond topology acquisition and route planning process. Take the example of a storage pool scenario shown in Figure 8.1: the AUV works in a storage pond environment that contains randomly placed and oriented cuboids. It

tries to find a collision-free path to reach its desired destination point. To achieve this, it is necessary to create a model of the environment and to select an appropriate path-planning algorithm.

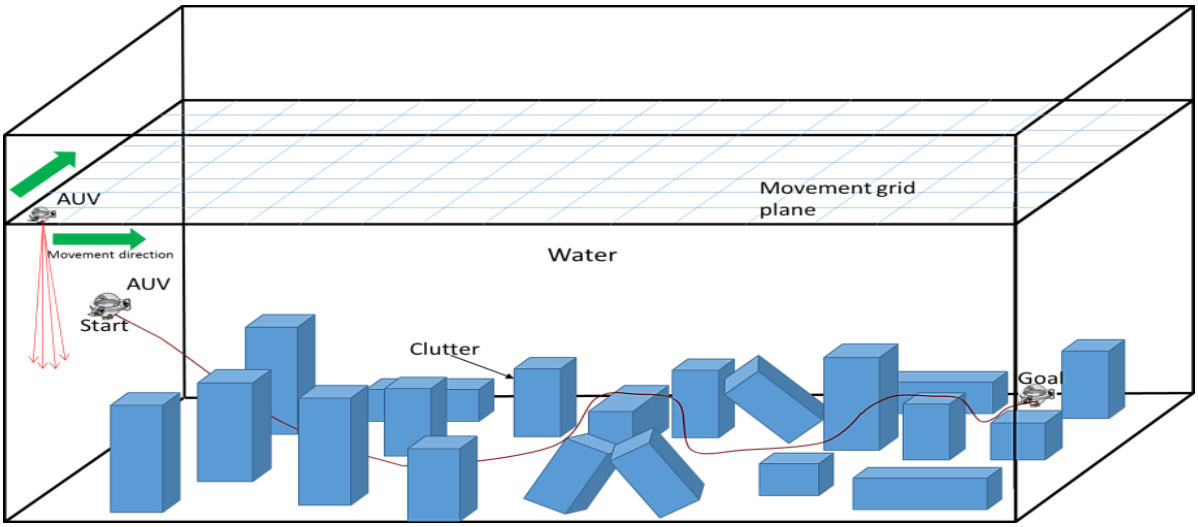


Figure 8.1: Survey grid of aerial survey

As discussed in Chapter 3, an effective way to model an unknown underwater environment is to perform an aerial survey on the horizontal measurement plane to obtain measurements of pond depth on a known two-dimensional grid of points. It assumes that the measurement plane is in the clear water above the clutter and parallel to the floor of the pond. Clearly, it is possible to use vertical planes to perform surveys, but this is complicated by the presence of clutter at the bottom of the pond.

The survey result is a matrix of tuples, whose components are measurement position and corresponding depth. This data is the 2D projection of the 3D clutter on to the measurement plane and can be used to construct a height map of the objects at the bottom of the pond. This provides useful data about the disposition of material within a pond. However, height maps do not provide adequate support for the path-planning needed for detailed exploration. What is needed is a 3D representation of the pond clutter. Therefore, constructing the point cloud model of each obstacle under certain restrictive assumptions about its shape is necessary. As discussed in Section 3.4, the shape of objects in storage ponds is assumed to be prismatic (i.e. objects have a polygonal base and a cross-section that does not vary with height, this includes both the cuboid and cylindrical containers that are typically used in most ponds). Although an aerial scan only gathers the information about objects' upper surfaces, the rest body of the object can be predicted by knowing the parameters of its upper surface, thus yielding a 3D point cloud to represent the object, this will be explained in Section 8.2.3.

Once the environment has been modelled as a point cloud, it can be converted to a 3D binary grid map (discussed in Section 8.2.4) to which the A* path-planning algorithms can be applied to find a collision-free path.

The work in this chapter aims:

(i) To develop a 3D environment construction algorithm to produce and initialise a 3D digital grid map based on raw depth measurements data and assumptions about the shape of the obstacles;

(ii) To apply the A* path-planning algorithm to the resulted 3D digital grid map in order to compute a collision-free path that connects the starting position of the robot and the proposed target position.

The novelty of this processing system is reflected in the following two features:

(i) The use of the raw upper surface's depth measurements data obtained by a grid aerial survey to build the required point cloud representation of the pond and its contents;

(ii) The initial depth measurements and the reconstructed point cloud cannot be used for path-planning directly. After processing, it can be used in path-planning, and this system integrates data acquisition and path-planning;

To guarantee that the A* path-planning algorithm can work in the 3D grid map, both processes must be performed in the same coordinate system and the depth measurement must lie on the user-specified aerial survey grid. The survey grid is the location of the mesh points in the measurement plane (as indicated in Figure 8.1). The A* algorithm can be used in a workspace that based on a discretized representation of the environment's geometry. In this system, the A* searching grid is a 3D grid based on the 2D survey grid but with the same interval in the z-axis. For objects standing upright of the sampled locations[4] will coincide with the survey grid position. However, for leaning objects they are not, the sampled locations do not lie on the survey grid, so it requires further data processing to make them consistent. The rest of this chapter explains how to establish a 3D grid map based on raw depth measurements and provides simulation results.

[4] Sampled locations refer to the coordinate of reflection positions.

8.1.2 Non-inclined objects

In the case of non-inclined objects, the coordinate of the scanner for each scanning is identical to the coordinate of the detected point. As a result, depth measurements at positions lie on the survey grid.

8.1.3 Single inclined object

In the single inclined object case, a correction needs to be applied to the survey matrix to compensate for the effects of object inclination. As a result, the correction process leads to depth measurements at positions that do not lie on the survey grid. Since the corrected survey matrix does not lie on the survey grid, it is necessary to process the corrected survey matrix to create a new matrix that lies on the survey grid. See Section 8.2 for details.

8.1.4 Multiple inclined objects

Situations, where ponds contain multiple inclined objects, are similar to the single inclined object case. The main difference is necessary to identify which fragment of the matrix corresponds to which inclined object and then process each fragment of the matrix separately. See Section 8.2 for details.

8.2 Establishment of the 3D Digital Grid Map

The raw depth measurements data is a height map of the environment, which cannot be used in path-planning directly. The procedures for conducting raw depth measurements data processing and establishing the 3D digital map are described as follows:

- *Boundary tracing algorithm*. This extracts objects' boundary points, estimates objects' occupied region and vertices, and calculates the equation of the upper surface;
- *Upper surface measurement points regeneration.* This generates new upper surface points that lie on the survey grid based on the equation of the upper surface which estimated from extracted vertices;
- *3D point cloud model construction*. This constructs the remaining part of the canister by interpolating points via the assumptions about the shape of the canister;
- *Grid method.* This converts the point cloud into a 3D digital grid map that can be used in the A* path planning algorithm.

The flow chart of the processing procedure is shown in Figure 8.2.

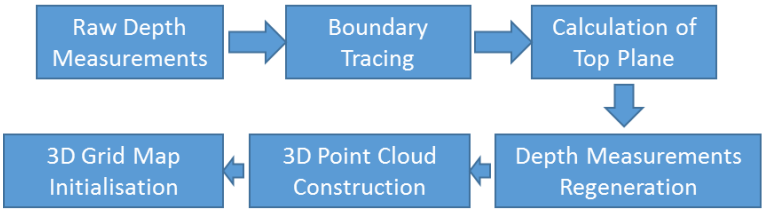


Figure 8.2: The processing flow chart

8.2.1 Boundary Tracing

Boundary tracing is a technique applied to digital images to extract the boundary of objects contained within the image [120]. A digital image is a set of pixels on a square tessellation, with each pixel having a certain value. Since the depth measurement data have similar features, image processing boundary tracing algorithms were used.

There are four common ways to trace a boundary in an image, the Square Tracing Algorithm [121], the Moore-Neighbour Tracing [121], the Radial Sweep [121], and Pavlidis' Algorithm [121]. This work adopts the radial sweep algorithm to extract boundary points from depth measurement data because this algorithm is simple and effective compared to others. Boundary tracing includes two steps:

- Extracting boundary points.
- Removal of the traced object.

It assumes that it is possible to use height difference to segment the data into objects and the pond ground, so there is a need for setting a threshold height. If the height is less than the threshold, it is the ground. Otherwise, it is the object. The threshold configuration is based on error statistics from the sensor datasheet. The height checking proceeds in sequence to find the first object point. The first object point is defined as the first point whose height is not zero or far from zero, which is also classified as the origin of the boundary. After the first object point has been found, the boundary tracing procedure is invoked. It uses the concept of chain code [115, 122], whose checking encoder moves along the boundary of the region to extract boundary points (in a clockwise or anticlockwise direction). It continues until the checking encoder returns to the starting position.

To find the next boundary point, the encoder will check the height of the eight directions points that surround the start point in clockwise order from the south direction until the next boundary point is found. Figure 8.3 shows the eight directions of the start point and the

checking orientation. (i,j) is used to represent eight directions of adjacent points relative to the start point.

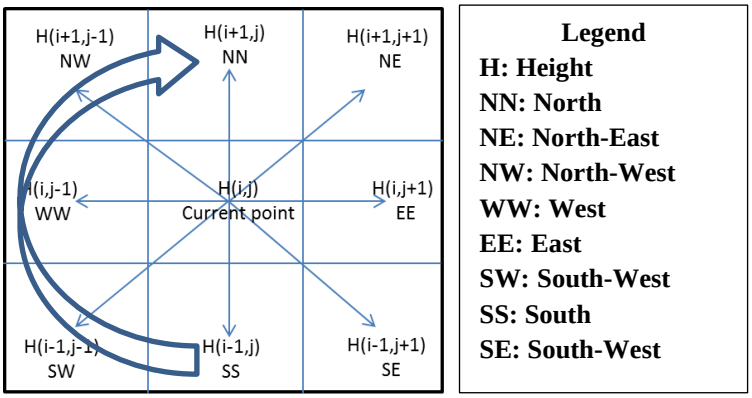


Figure 8.3: 8 Directions respect to the current point

Once the next boundary point is found, it is saved and treated as a new starting point, and a process similar to the above is repeated with one important difference. The difference is that rather than starting checking from the southern direction, checking commences from the direction of the previous starting point. This approach is demonstrated in Figure 8.4

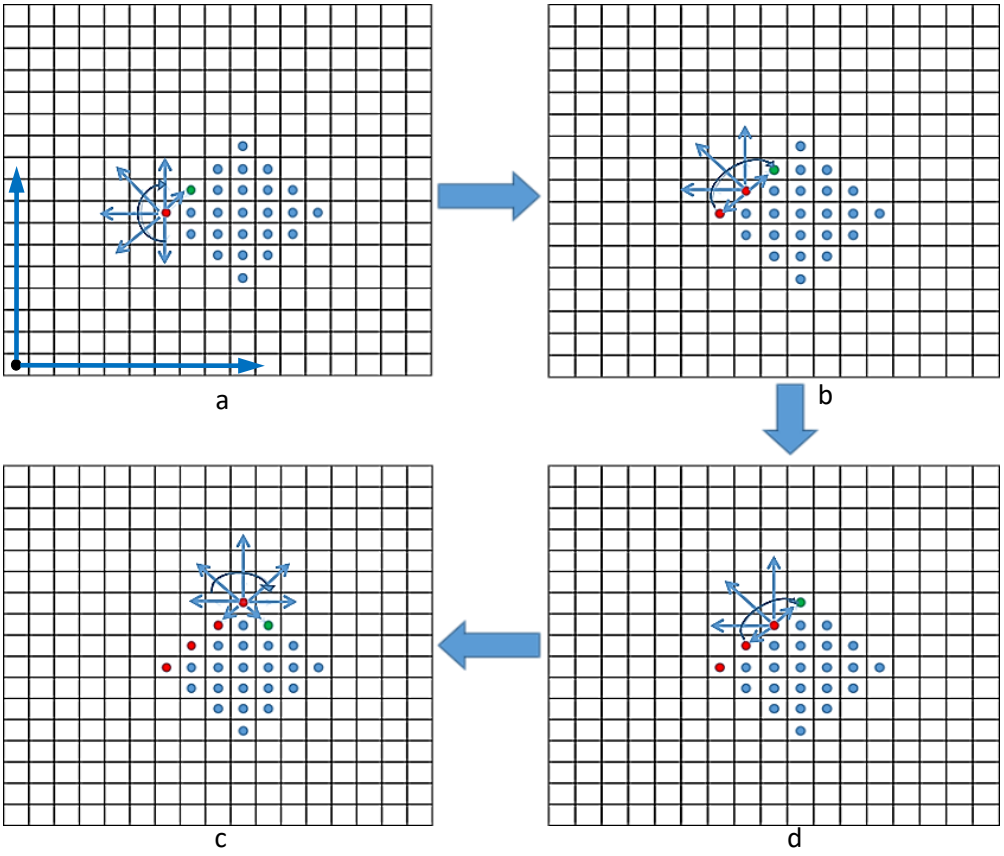


Figure 8.4: Boundary Tracing

In Figure 8.4, red dots represent the found boundary points and green dots represent the next boundary points to be found. The sample data is scanned from the bottom left (the black dot)

to top right for finding a starting point (by column). After finding a starting point (the red point in the upper left corner diagram of Figure 8.4), a clockwise boundary check starts from the point to the south of the starting point and then checks the cells in a clockwise direction until a non-floor point is found. This is the cell to the north-west of the starting point in the first grid of Figure 8.4. After that, regarding the found next boundary point as the new start, and repeating the boundary check. The first direction to be checked this time is the south-west point of the current start (see the upper right diagram of Figure 8.4). The boundary checking will terminate when the initial starting point (the first red dot) is found for a second time. Figure 8.4 demonstrates an example of the boundary tracing process on a sample point set. The most important thing in the boundary tracing procedure is the "sense of direction". The correct directions are with respect to the previous positions. Therefore, it is important to keep track of the current and previous orientation to make the right moves.

In practice, there will be more than one object in the pond environment, so the obtained depth measurement data will contain different sets of points (one set of points corresponds to one object). However, the problem is the boundary tracing will repeatedly trace the first object and cannot break out the loop. To escape from the loop, it needs to remove each traced object from the original data set so that it does not prevent tracking of the remaining objects. Once the removal process has finished, the scanning process restarts to find the next object point and trace its boundary. See the example in Figure 8.5.

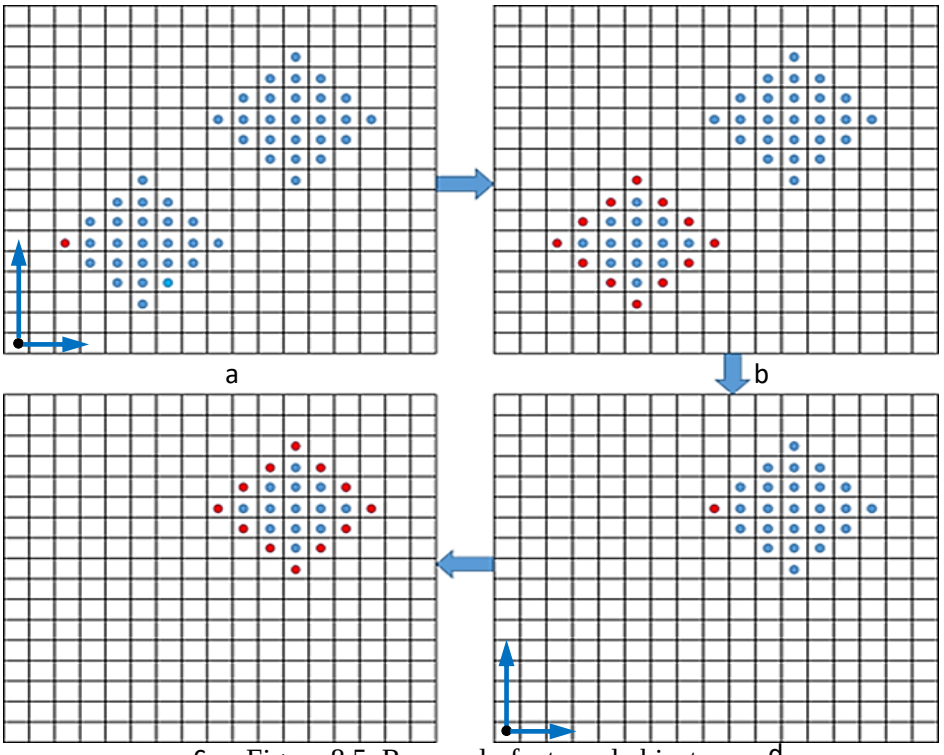


Figure.8.5. Removal of a traced object

There are two sets of points in the depth measurement data in the upper left diagram of Figure 8.5, so there are two objects. The red points are boundary points, the blue points are internal object points and the black dot highlights where the scanning starts. Figure 8.5 demonstrates how to extract boundary points when there are multiple objects in the depth measurement data. Firstly, the image scanning is done 'vertically' here i.e. on a column by column basis to find the first object point, and then the process traces the boundary of the first object (the upper right diagram of Figure 8.5) in order. After that, remove the first object from the original data by resetting the height of each first object's point as zero and then restart the boundary tracking (the lower right diagram of Figure 8.5). However, one problem remains. When two objects overlap or touch each other, the points of their contact point(s) will be common. As a result, this algorithm will treat them as one object. It may be possible that some form of segmentation [155] could be applied to separate the two objects.

8.2.2 Modified Depth Measurement Data

Sometimes, the measurement points do not lie in the user's specified survey grid as the inclined case mentioned in 8.1.2, so it is necessary to calculate the plane equation to create modified depth measurement points that lie in the user's specified survey grid.

A plane in 3D space is defined by three points that do not all lie on the same line, or by a point and a normal vector to the plane. To calculate the vector equation of a plane, it is necessary to know the coordinates of at least three points that are not on the same line but on the same plane. However, the selection of points needs to be careful because it cannot assume that any three arbitrary points taken from the plane are able to represent the plane. The chosen points should contain special characteristics about the plane, so the most south, north, west, and east vertices are the best option. Since the boundary points in the depth measurement data can be extracted by boundary tracing, those vertices can be found by comparing x-axis value and y-axis values of boundary points. Once the coordinates of three vertices[5] points are known, the vector plane equation can be calculated. The general form of the vector equation of a plane is:

$$\vec{n} \cdot (\vec{p} - \overrightarrow{p1}) = 0 \tag{8.1}$$

$\vec{n}$ is the normal vector of the plane. The normal vector $\vec{n}$= (a, b, c) of a plane can be calculated by the *Cross Product* of any 2 vectors in the plane. $\overrightarrow{p1}$=(x_1, y_1, z_1) is one of the

[5] Vertices refer to the most northerly, southerly, westerly, and easterly of the boundary points.

known vertices points. $\vec{p}$ =(x, y, z) is any point on this plane. This equation can be transformed into the general planar equation, and the height of any point on the plane is given:

$$(a, b, c) \cdot (x - x_1, y - y_1, z - z_1) = 0$$

$$ax + by + cz - (ax_1 + by_1 + cz_1) = 0$$

$$z = \frac{(ax_1 + by_1 + cz_1) - by - ax}{c} \tag{8.2}$$

For the case that depth measurements of reflection positions that do not lie on the survey grid such as tilted objects, Equation (8.2) can be used to calculate the height of the points whose position lie on the survey grid and the object's top plane. Thus, a modified measurement data is obtained, where points have been interpolated on to the survey grid. Since there are numerous objects, the above processes are undertaken for each object.

8.2.3 Environment Construction by Point Cloud

After acquiring a modified measurement data, the next step is to further process to create a complete 3D point cloud model of the environment. This procedure is called environment construction. As discussed in Section 3.4, objects are assumed to be prismatic, so its side surfaces are perpendicular to its upper surface. This is a crucial assumption for constructing the object because the key idea of constructing the environment by Point Cloud is to add new points to the modified measurement points in the perpendicular direction of the upper surface to fill the space between the object's upper surface and the pond ground. Added points must lie on the survey grids. Thus, the approach generates a 3D point cloud model of obstacles by a group of points under certain assumptions about the shape of the objects. This technique is known as linear height interpolation [123]. Figure 8.6 shows a schematic diagram of the perpendicular height interpolation for an inclined cuboid.

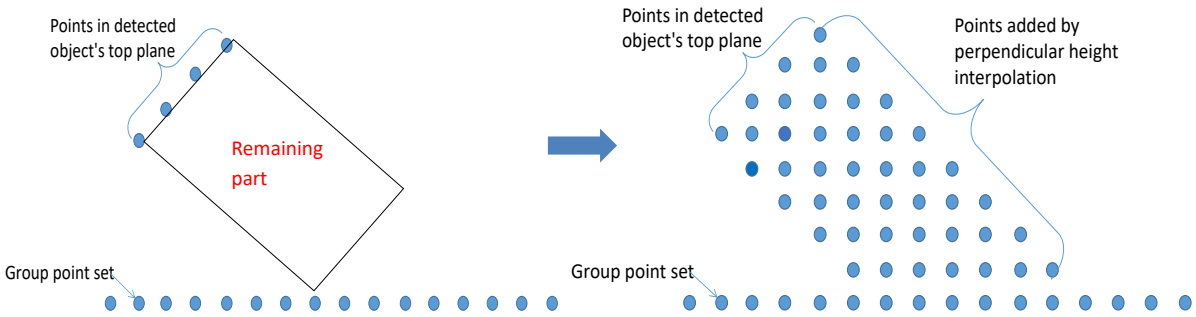


Figure 8.6: A schematic of the linear perpendicular height interpolation

An example of using linear perpendicular height interpolation to generate a 3D point cloud model of the detected object is shown below. It is explained with the aid of Figure 8.7.

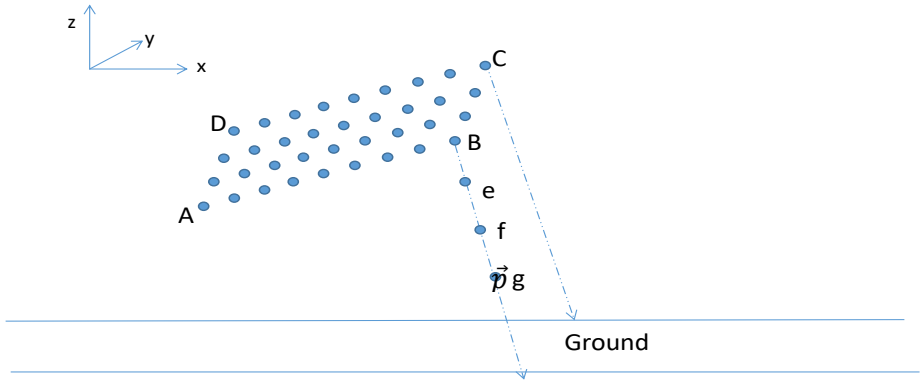


Figure 8.7: A schematic of the linear perpendicular height interpolation

Assume the resolution of the survey grid is **i**×**i**, and measurement points all lie on the survey grid. Point A, B, C, D are four most south, north, west, north vertices from the measurement data. The coordinates of A, B, C and D are (x_a, y_a, z_a), (x_b, y_b, z_b), (x_c, y_c, z_c), (x_d, y_d, z_d), respectively. **e**, **f**, **g** are the interpolated points, they must satisfy the condition that they all lie on the survey grid and on the vector p that is perpendicular to the upper plane and passes through B. The first step is to calculate the equation of vector $\vec{p}$. The coordinate equation of a point on the vector is given by:

$$\overrightarrow{AB} = (x_b - x_a, y_b - y_a, z_b - z_a)$$

$$\overrightarrow{AD} = (x_d - x_a, y_d - y_a, z_d - z_a)$$

$$\vec{n} = \overrightarrow{AB} \times \overrightarrow{AD} = (x_n, y_n, z_n)$$

$$\vec{p} = \vec{B} + t \cdot \vec{n} = (x_b, y_b, z_b) + t \cdot (x_n, y_n, z_n) \tag{8.3}$$

$$(x, y, z) = (x_b + t \cdot x_n, y_b + t \cdot y_n, z_b + t \cdot z_n)$$

$$x = x_b + t \cdot x_n \tag{8.4}$$

$$y = y_b + t \cdot y_n \tag{8.5}$$

$$z = z_b + t \cdot z_n \tag{8.6}$$

e is a point on vector p. Assume the z coordinate of e (z_e) is known, so the x, y coordinate of **e** can be calculated by following equations:

$$e = (x, y, z_e)$$

$$z_e = z_b + t \cdot z_n \tag{8.7}$$

$$t = \frac{z_e - z_b}{z_n} \tag{8.8}$$

$$y_e = y_b + \frac{z_e - z_b}{z_n} \cdot y_n \tag{8.9}$$

$$x_e = x_b + \frac{z_e - z_b}{z_n} \cdot x_n \tag{8.10}$$

The remaining question is to determine which grid cell that point e is located in. To do so, it is necessary to find its nearby grid point on the survey grids to determine the grid cell where point **e** is located. As Figure 8.8 shows, the blue dot is point e, its four nearby survey grids vertex highlighted in the red dot, so grid cell that point e locates is found.

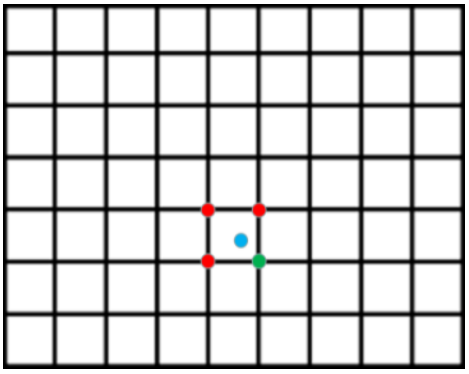


Figure 8.8: Determine the grid cell

After processing the new topological depth measurements data by the procedures given above, a 3D point cloud model of detected objects is obtained. Each point is assigned to the corresponding grid voxel in the 3D grid map.

8.2.4 Grid Method to Establish 3D Occupancy Grid Map

Since the 3D model of the environment is represented by discrete points in the survey grid, a 3D digital map can be created based on the occupancy grid method [124]. The basic idea of the occupancy grid method is to create an environment map as an evenly spaced field of binary numbers: obstacle regions and free regions are encoded by 0 and 1, respectively. First of all, divide the whole environment into a finite number of equal-sized cells without considering obstacles. See Figures 8.9 and 8.10.

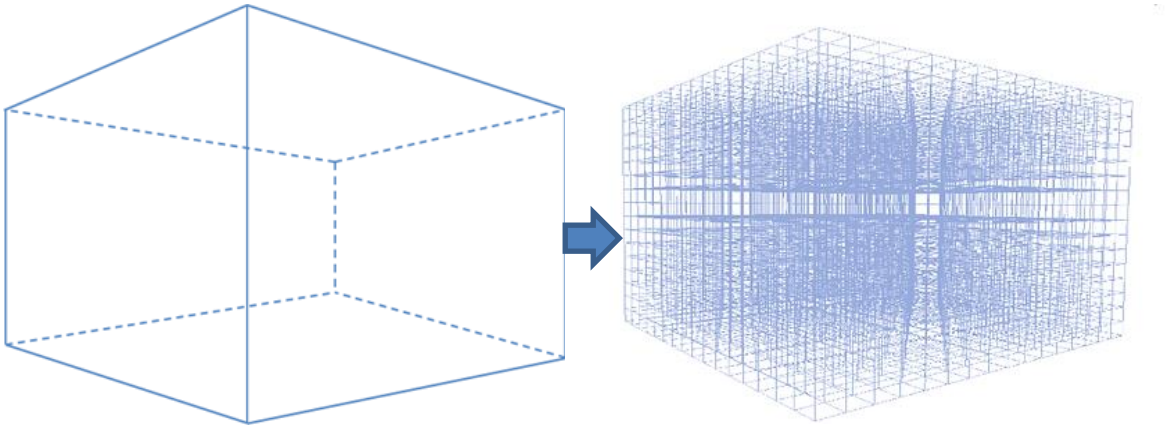


Figure 8.9: The whole environment Figure 8.10: The meshed environment

Then, model obstacles by a finite number of discrete points based on the algorithm introduced in section 8.2.1 to 8.23. After that, align each point to its corresponding survey grid and convert it to the occupancy grid map. To convert the point cloud model to a 3D digital grid map, the first step is to define cells occupied by objects as inaccessible cells and cells not occupied by points as accessible cells. The accessible cells are outside obstacles and the inaccessible cells are inside obstacles. Next, the 3D matrix that will hold the digital map is initialised by encoding each inaccessible cell with '1' and accessible cell with'0'. As a result, the binary matrix of the 3D digital grid map of the environment is obtained. It can then to be used in path planning algorithms, e.g. A* algorithm. Figure 8.11 shows the conversion of a 2D environment to a 2D occupancy grid map, red dots represent object regions and green dots represent non-object regions.

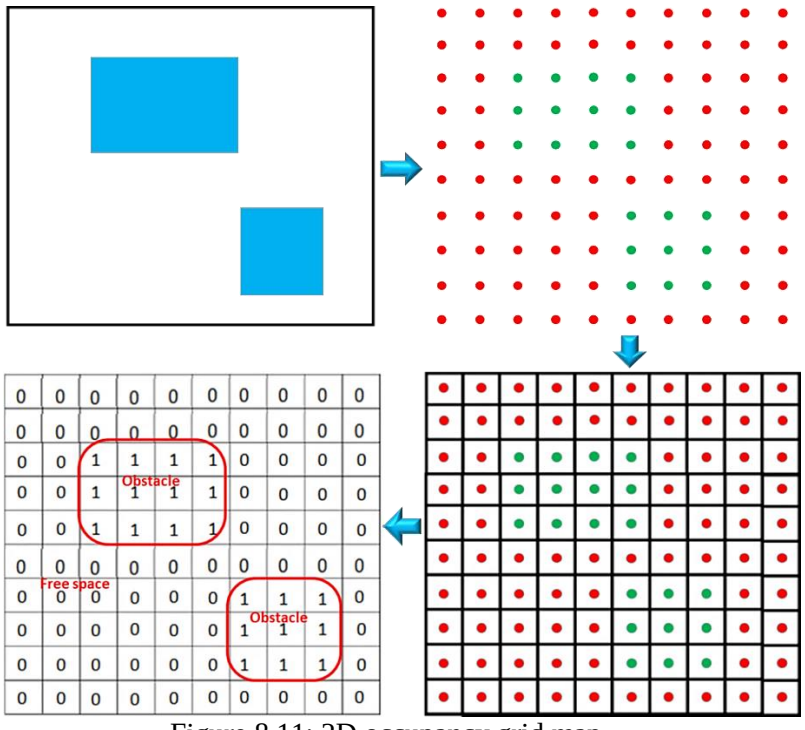


Figure 8.11: 2D occupancy grid map

8.3 Path-planning on 3D Digital Grid Map Based on A* Algorithm

Once a 3D digital grid map has been constructed, the remaining work is to import the grid map data to the A* path-planning algorithm and find a collision-free path that connects the start point and the goal. The grid map is the workspace, each grid cell is regarded as a node, and the binary 0 and 1 indicate each whether a node is accessible or inaccessible, respectively. The A* algorithm begins by considering the start cell's [6] accessible neighbour nodes. Figure 8.12 shows A* path-planning on the 2D occupancy grid map in Figure 8.11.

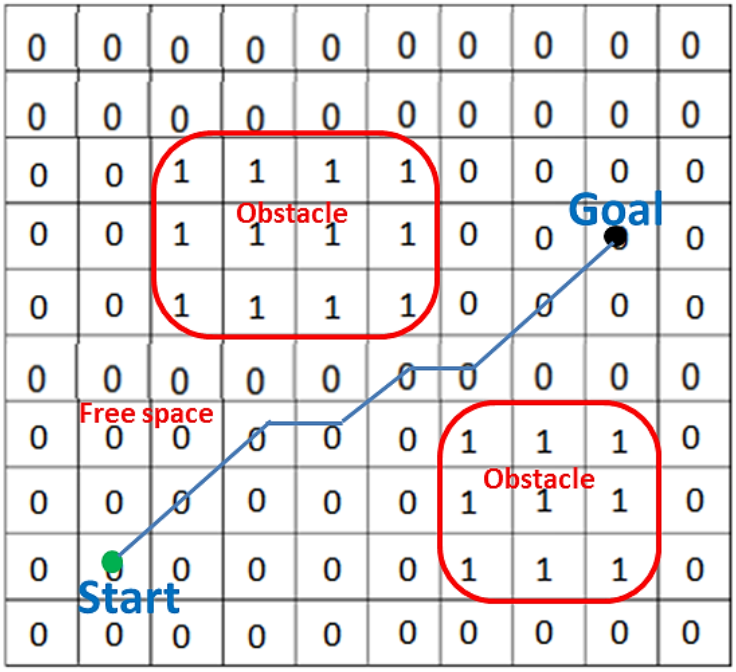


Figure 8.12: A* path-planning on a 2D occupancy grid map

8.4 The Simulation Experiments for Pond-Like Environment

To validate the algorithm explained in Section 8.2 and 8.3, a simulation of the aerial survey data processing and route planning is undertaken. The simulator was built with MATLAB. There are several assumptions in the simulation experiment:

1. The mobile robot is regarded as a moving particle so its size is not considered and has three degrees of freedom.
2. The dimension of the pond-like environment is 50m×25m×10m.

[6] The start-cell contains the start point.

3. The aerial survey grid is identical to the A* searching grid in the XOY plane.
4. The aerial survey sampling grid resolution is 0.5 m ⌧ 0.5 m.
5. A* path planning algorithm will be used in the survey grid.
6. All obstacles are prismatic.

8.4.1 The Simulation for Pond-Like Environment Case 1

The first simulation (case 1) uses a very simple pond environment that contains three objects (one cuboid, one hexagonal prism, and one pentagonal prism). This is shown in Figure 8.13. The first step is to perform an aerial survey to acquire depth measurements data about the bottom information of the pond environment. The measurements data are shown in figure 8.14.

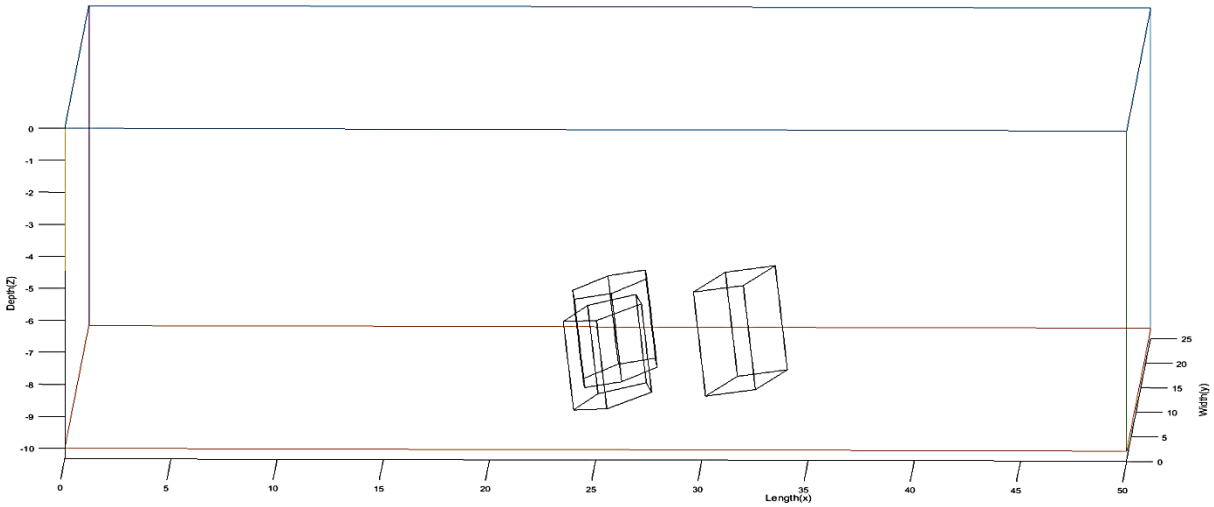


Figure 8.13: Pond-like environment Case 1

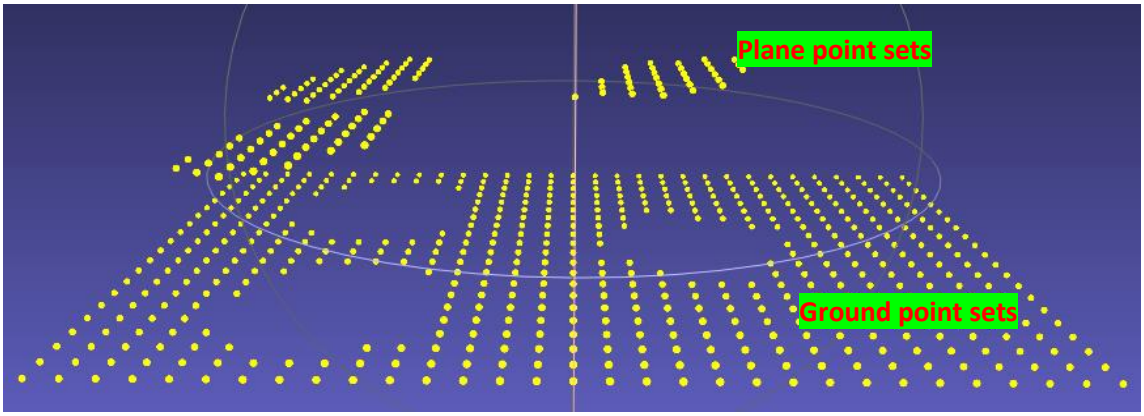


Figure 8.14: Raw depth measurements of the pond environment Case 1

In the second stage, the measurements data is processed by the boundary tracing algorithm to estimate the vertices of the upper surfaces of objects (see Appendix D). These vertices are

used in the calculation of the plane equation and new measurement points. After that, linear height interpolation is applied to generate new points based on the survey grid, representing the object, as discussed in Section 8.2.3, resulting in a 3D point cloud model of the environment shown in Figure 8.15.

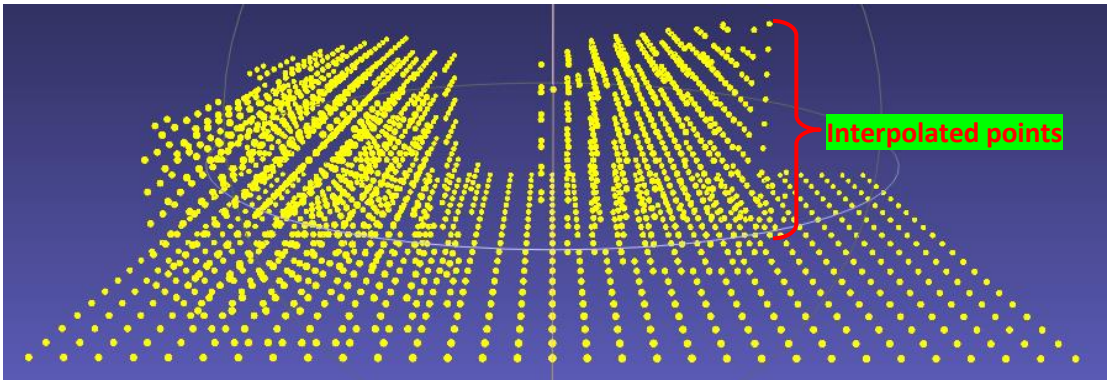


Figure 8.15: Height interpolated point cloud for case 1

The next step initialises the digital map matrix (as explained in section 8.2.4) with occupied cells being assigned a binary '1' (yellow dot), and unoccupied cells being assigned a binary '0' (red dots). The output of this stage is the 3D digital grid map shown in Figure 8.16.

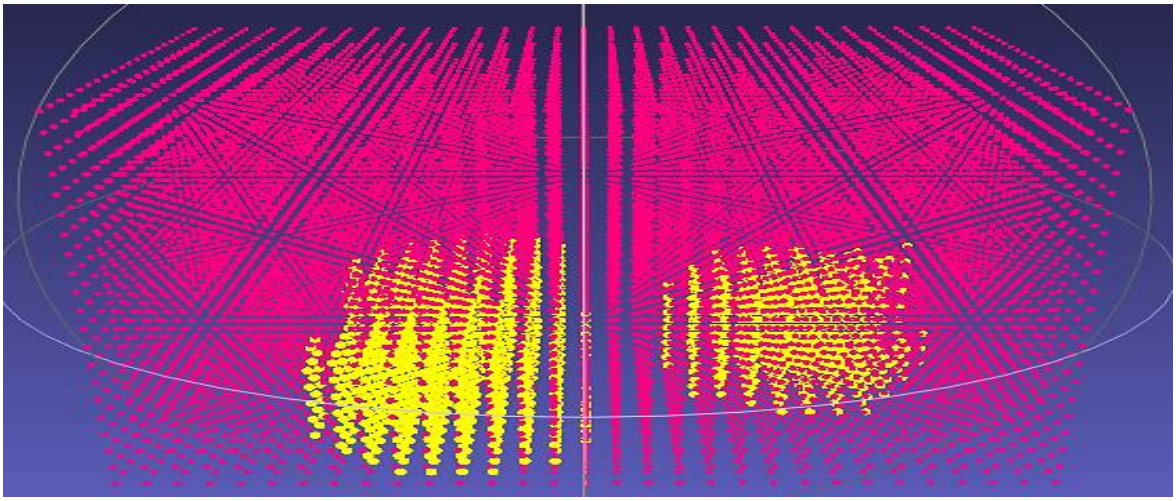
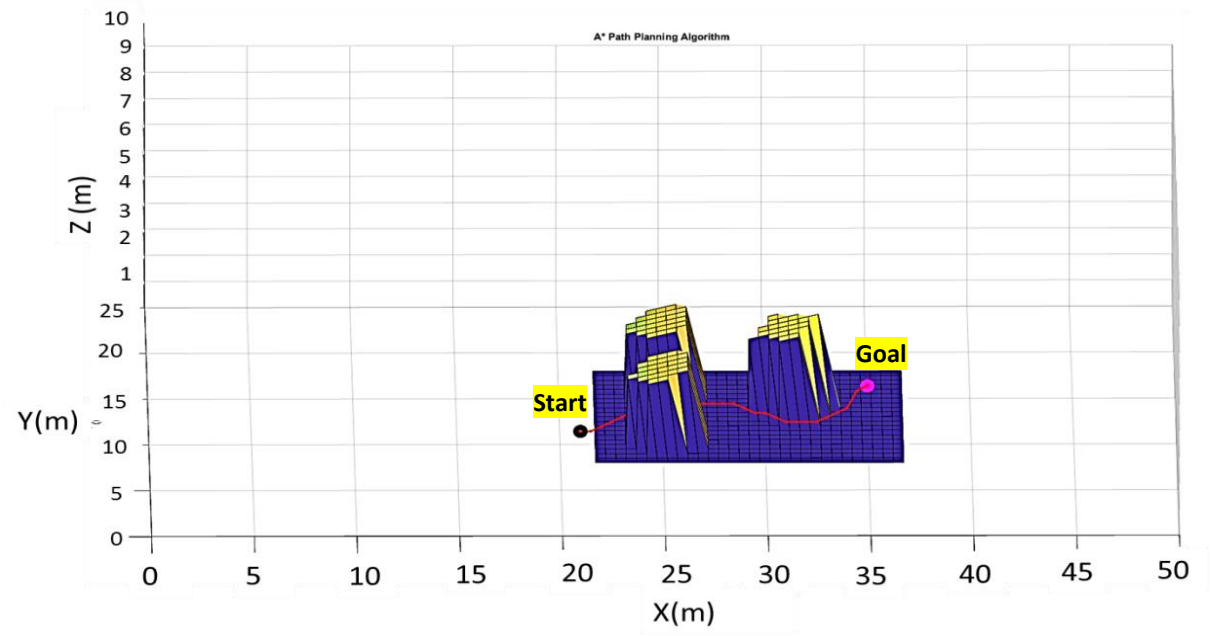
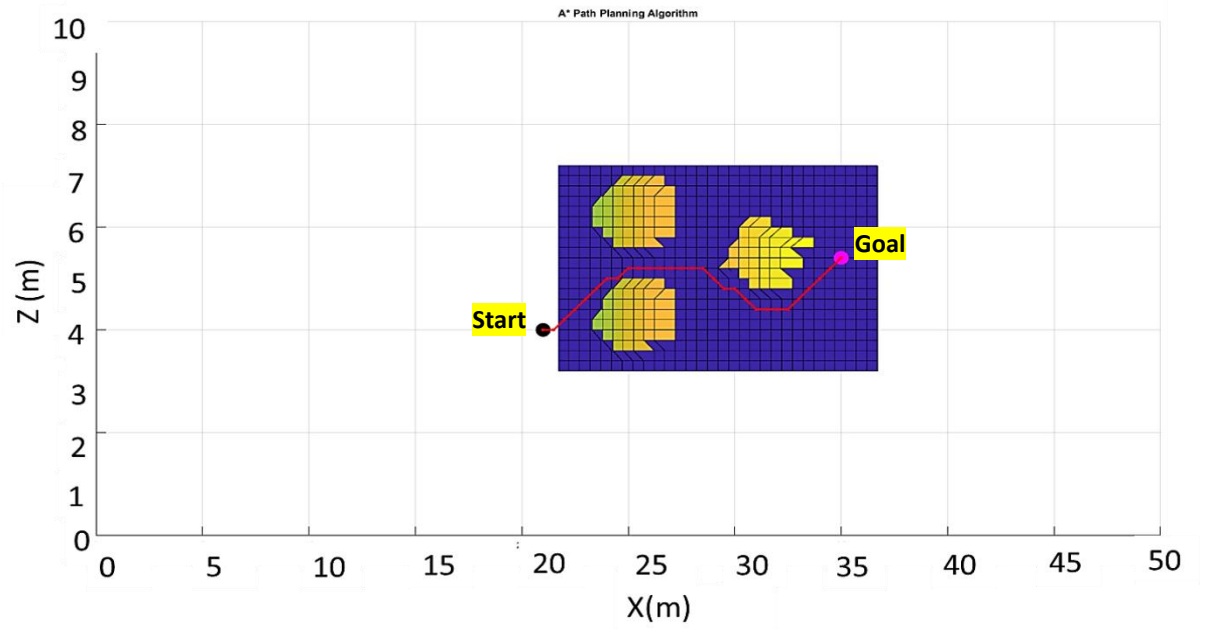


Figure 8.16: 3D grid map for case 1

Finally, the 3D digital grid map is imported to the A* path-planning simulator as the workspace. It is assumed that the starting point is (21, 4, 1) and the ending point is (35, 5, 3). The corresponding path planned by the A* algorithm on the 3D grid map (Figure 8.16) is shown in Figure 8.17.



(a)



(b)

Figure 8.17: A* path: (a) 3D view for case 1. (b) Plan view for case 1.

8.4.2 The Simulation for Pond-Like Environment Case 2

Case 1 above only represents a simple pond-like environment that only has few spent fuel canisters. Case 2 below represents a complex pond-like environment (old pond) that has 18 different sized and randomly placed spent fuel canisters. Some are upright, some are tilted, and some are lying on their sides. The 3D grid map construction and path-planning simulation of the pond-like environment Case 2 are shown in Figure 8.18 to 8.22 below.

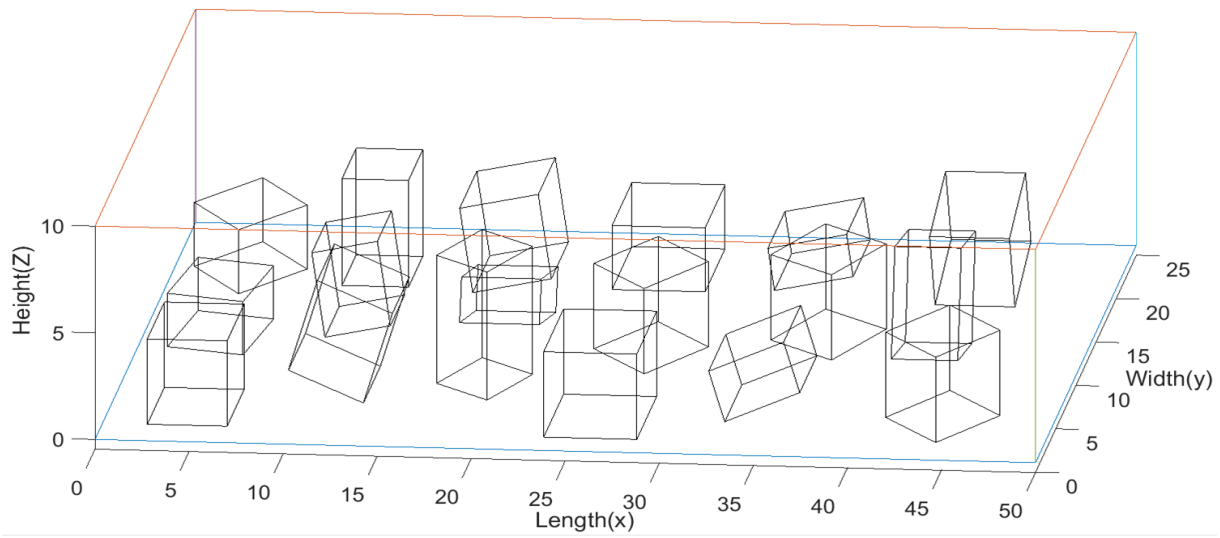


Figure 8.18: Pond-like environment Case 2

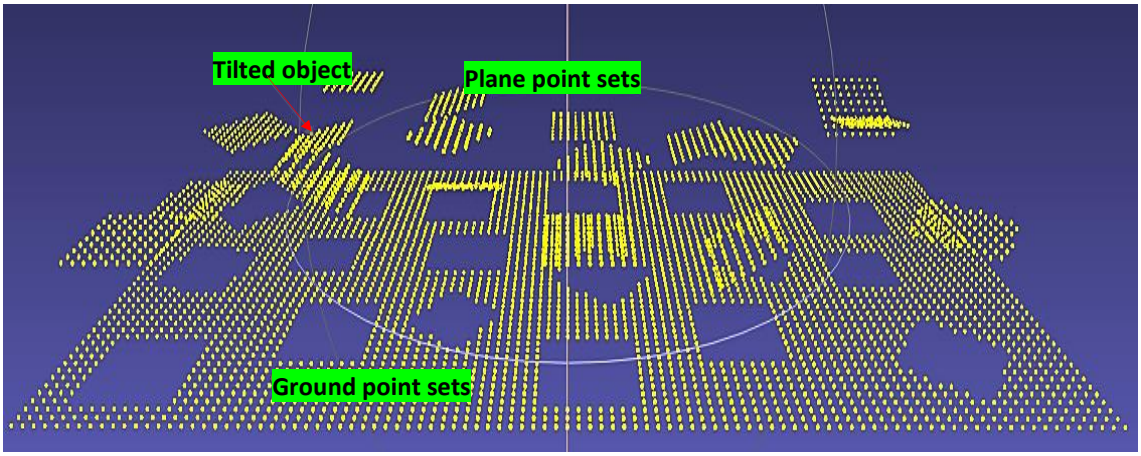


Figure 8.19: Raw point cloud of pond-like environment Case 2

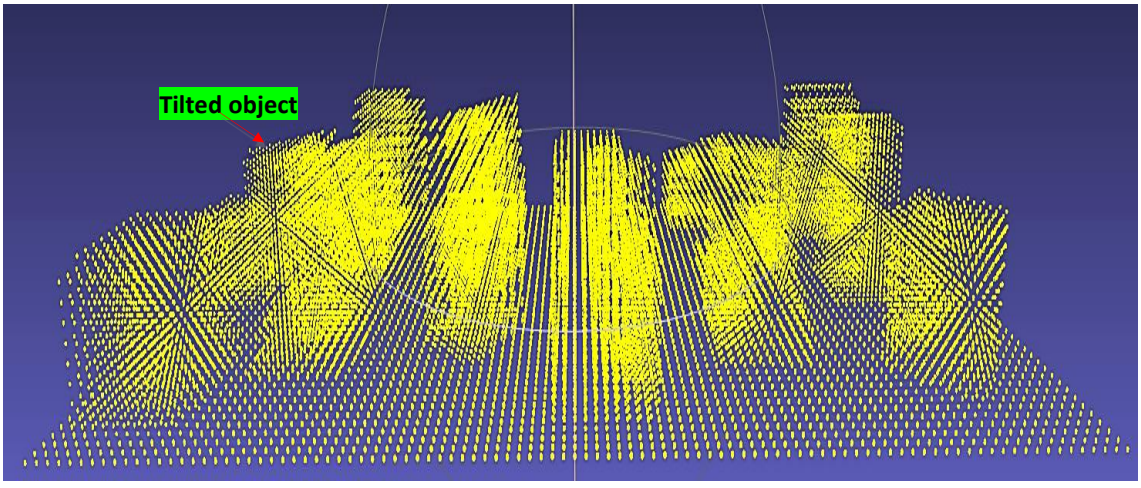


Figure 8.20: Height interpolated point cloud of Case 2

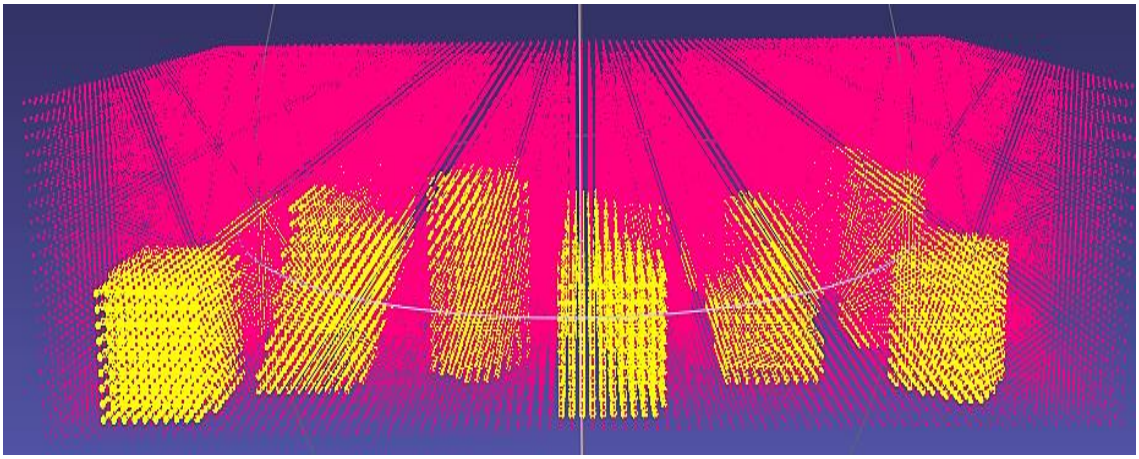
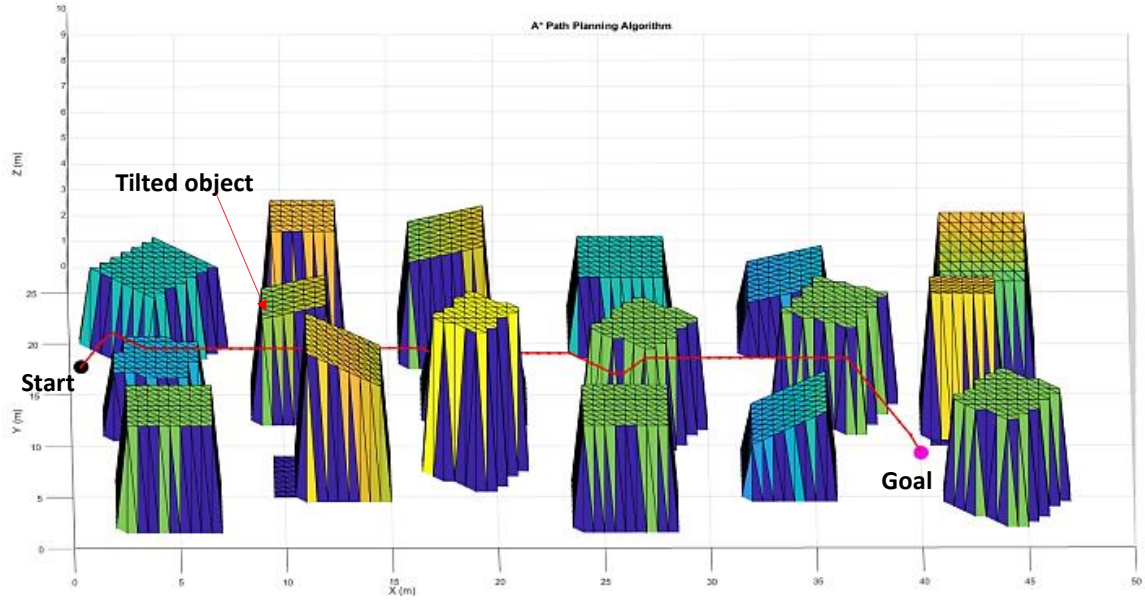
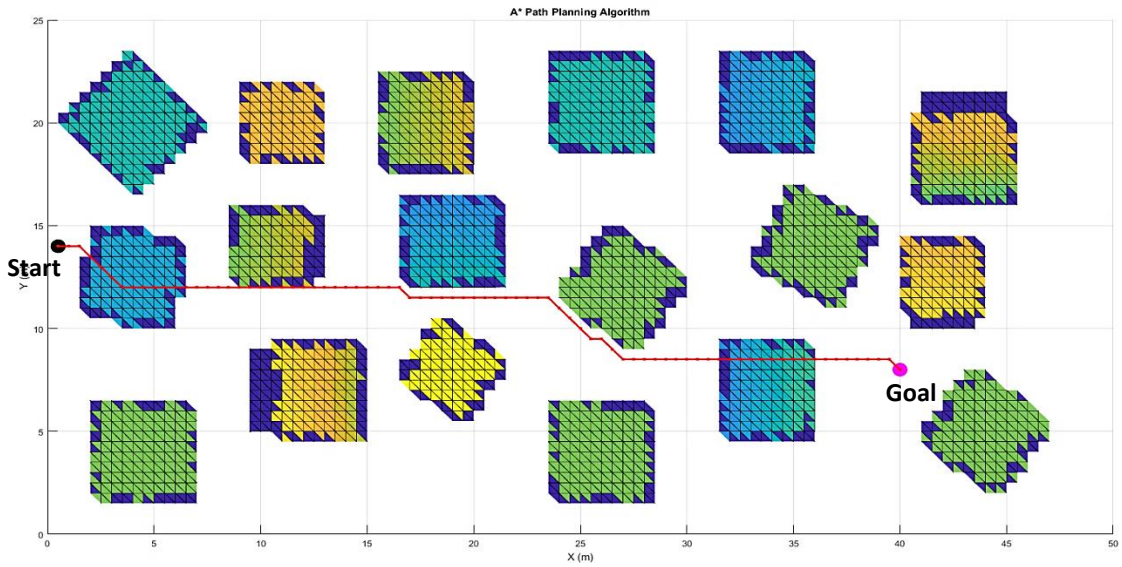


Figure 8.21: 3D grid map of Case 2



(a)



(b)

Figure 8.22: A* path: (a) 3D view for case 2. (b) Plan view for case 2

As the figures above show, after processing the measurements data (Figure 8.14 and 8.19) with the data processing approach, a well-formatted 3D digital grid map (Figure 8.16 and Figure 8.21) can be obtained. A* algorithm can plan a collision-free trajectory (red line in Figure 8.17 and Figure 8.22) on this map. Above results demonstrate the navigation approach works successfully. For more results, see Appendix E.

8.5 Errors in Measurement

Measurement error is an important factor that needs to be considered in pond data acquisition and path planning. In practice, the depth measured by the acoustic sensor has errors. The accuracy might vary from a few centimetres up to ten centimetres depending on the quality of the sensor. Assume the errors satisfy Gaussian distribution. Appendix F shows a simulation of the mapping results after adding Gaussian error on the measurement. Measurement errors in height will cause the obtained 3D point cloud model of the tilted object to be shifted in the horizontal plane with respect to the real position. As a result, the measured object will be either smaller or bigger than the true object. This section aims to study the effects of measurement errors and to investigate how to mitigate the collision problem caused by measurement errors. Assume that the noise has a Gaussian distribution and its standard deviation is σ. The discussion starts with the 2D case; see Figure 8.23.

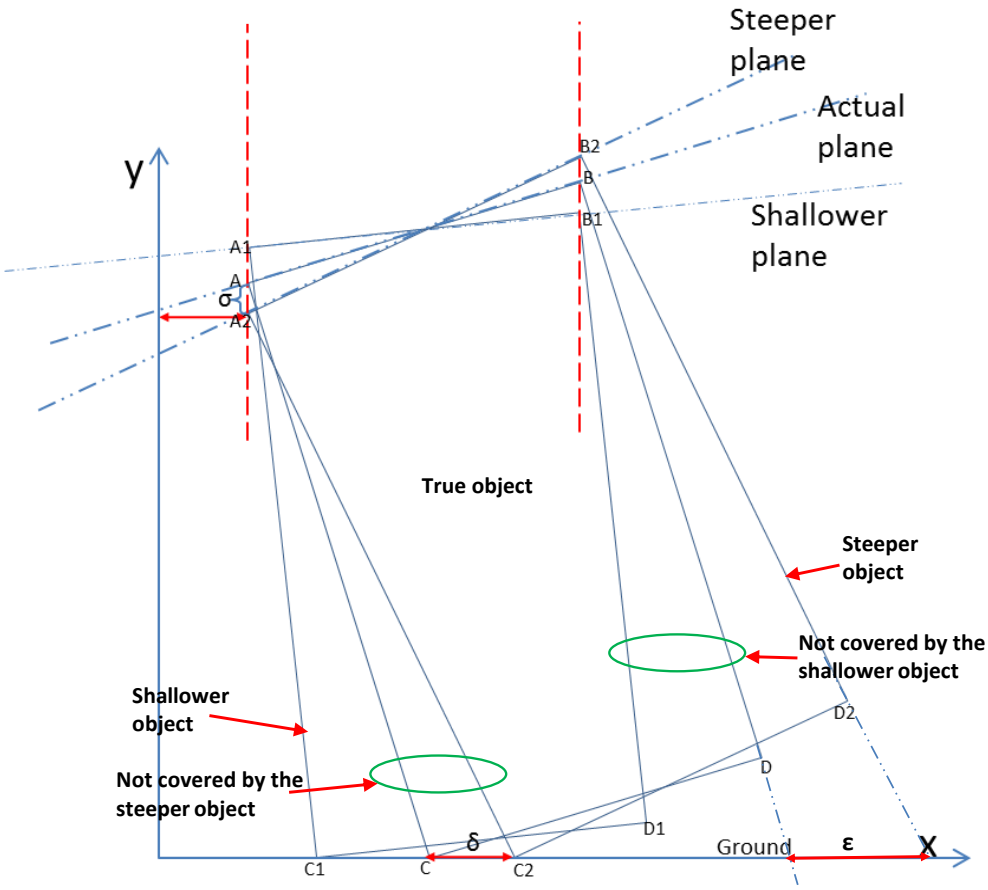


Figure 8.23: Object created by measurement errors in height

A and B are the mean values of multiple measurements of the vertices of the true object's upper surface obtained by the sensor (assume there are 10 measurements). AB is the line of the measured upper surface plotted by the mean value. A1 is the largest positive distribution of the measured vertex A in the Gaussian distribution based on the mean value; A2 is the largest negative distribution of the measured vertex A in the Gaussian distribution based on the mean value; B2 is the largest positive distribution of the vertex B in the Gaussian distribution based on the mean value; B1 is the largest negative distribution of the vertex B in the Gaussian distribution based on the mean value.

A1B1 is the line that connects A1 and B1, which represents the upper plane of the shallower object A1C1D1B1 (created by the shallower plane, the construction of the object based on the upper surface is explained in section 8.2.3). A2B2 is the line that connects A2 and B2, which represents the upper plane of the steeper object A2C2D2B2 (created by the steeper plane). δ is the maximum separation caused by 2σ in the x-axis (left-hand side); ε is the maximum separation caused by 2σ in the x-axis (right-hand side). Assume A=(x_a, y_a), B=(x_b, y_b), so A1=$(x_a, y_a + 2\sigma)$, B1=$(x_b, y_b - 2\sigma)$, A2=$(x_a, y_a - 2\sigma)$, B2=$(x_b, y_b + 2\sigma)$.

It can be observed in Figure 8.23 that neither the shallower object nor steeper object can completely enclose the true object (ABCD), which may cause a collision problem when used for planning a collision-free path in practice. Therefore, it is necessary to create a larger bounding box that can completely enclose the real object based on the measurement data.

8.5.1 Bounding Box Expansion in 2D

To ensure the bounding box can completely enclose the true object, it is necessary to expand the measured object by adding 2σ in height and adding δ and ε in width. Calculations procedure of δ and φ are shown in Appendix F.

$$\delta = x_{c2} - x_c = \frac{4\sigma(y_a - 2\sigma)}{x_b - x_a} - 2\sigma * \frac{y_b - y_a}{x_b - x_a} \tag{8.11}$$

$$\varepsilon = x_{D2} - x_D = \frac{4\sigma(y_b - 2\sigma)}{x_b - x_a} - 2\sigma * \frac{y_b - y_a}{x_b - x_a} \tag{8.12}$$

So the boundary vertex of the bounding box's upper plane is given by:

$$\text{Left boundary vertex} = (x_{a'} - \delta, y_{a'} + 2\sigma) \tag{8.13}$$

$$\text{Right boundary vertex} = (x_{b'} + \varepsilon, y_{b'} + 2\sigma) \tag{8.14}$$

Since the object is assumed to be prismatic and the boundary vertex A'' and B'' of the bounding box's upper plane is given by above equations 8.13 and 8.14, the boundary vertex C' and D' of the bounding box's bottom plane can be calculated by the perpendicular interpolation explained in 8.2.3. The resulting bounding box $A''C'D'B''$ (named as the noise correlated object) is shown in Figure 8.24, which encloses the true object.

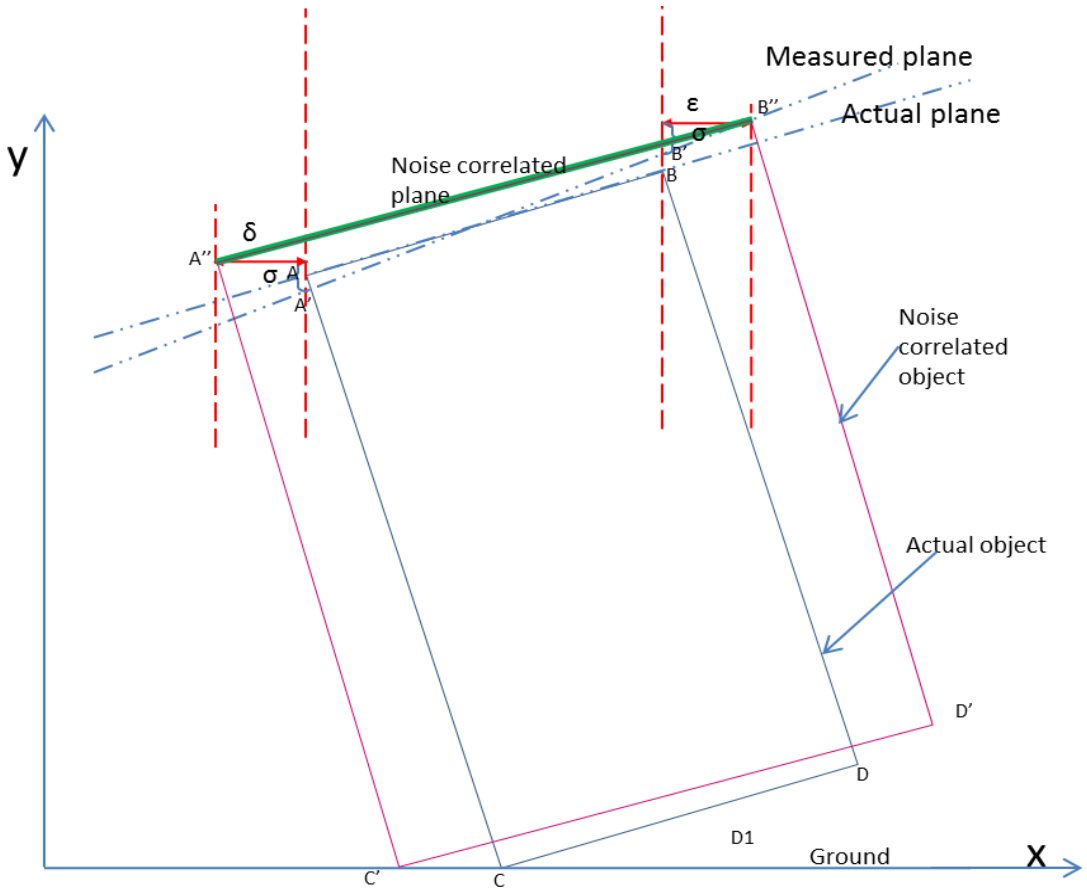


Figure 8.24: Bounding box

In the following simulation experiment, assuming:

- a standard deviation of the sensor reading is 4 cm, which is representative of sensor noise,
- the coordinates of the two vertices of the actual object's upper surface are A (6 m, 4 m) and B (11 m, 5 m), area of the true object is 20.8m^2,
- the sensor readings are subject to noise with a Gaussian distribution,
- there are 40 pairs of samples of measurement boundary vertices A' and B' of the upper plane;

After using the approach explained in Section 8.2 to calculate the vertices of the measured objects and bounding box, the vertices can be used to calculate the area of the box. In this simulation experiment, the area of the bounding box, the true object, and the measured object of each sample are recorded in Figure 8.25. The x-axis is the order of samples. The Y-axis is the corresponding calculated area. For example, the area of the measured object calculated by the first pair of samples of measured boundary vertices A' and B' of the upper plane is 20.85m^2, and the area of the corresponding bounding box is 22.2 m^2.

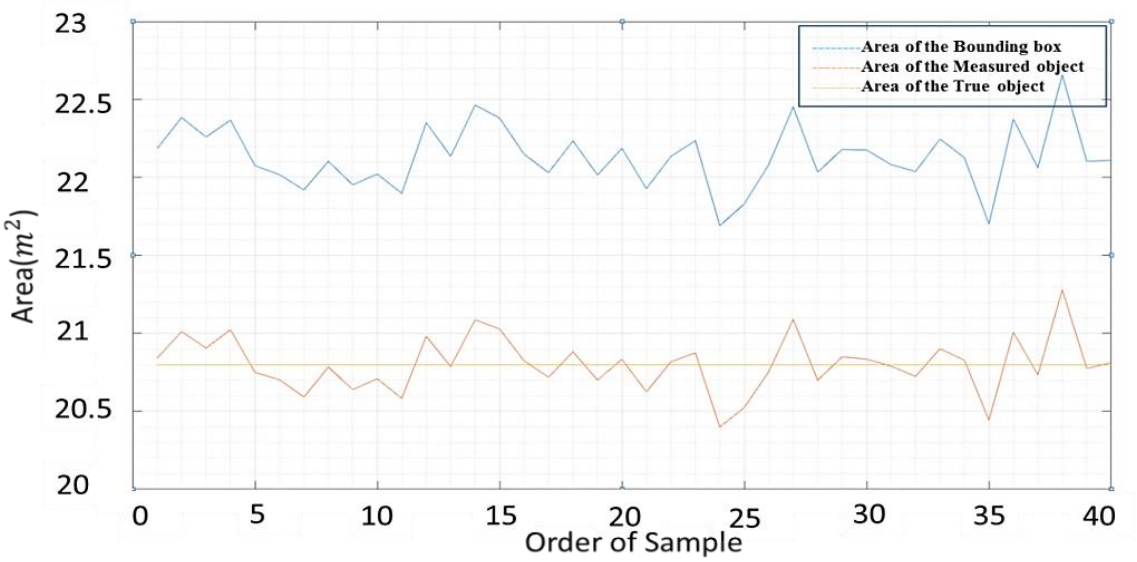


Figure 8.25: The area of the bounding box, the actual object, and the measured object of each sample

From Figure 8.25, it is easy to find that the area of the bounding box is guaranteed larger than the area of the true object, which means the bounding box completely encloses the true object. Because the method that creates the bounding box is based on extending the vertice of the original object, so it will guarantee that the bigger area encloses the smaller area. The above results validate the statement that after applying the bounding box expansion, the bounding box can completely enclose the true object. Therefore, there is no need to worry about that the measured object is smaller than the true object and causes a collision problem in practice.

8.5.2 Bounding Box Expansion in 3D

The above approach can be used in the 3D case in the same manner because a 3D object can be described as in terms of a stack of 2D cross-sections of a 3D object. For example, Figure 8.26 is the plan view of a 3D object, and it can be regarded as the Y-axis accumulation of 2D cross-section shown in Figure 8.27. δ and ε for each 2D cross-section can be calculated by Equation 8.11 and 8.12, thereby find the bounding box of each cross-section.

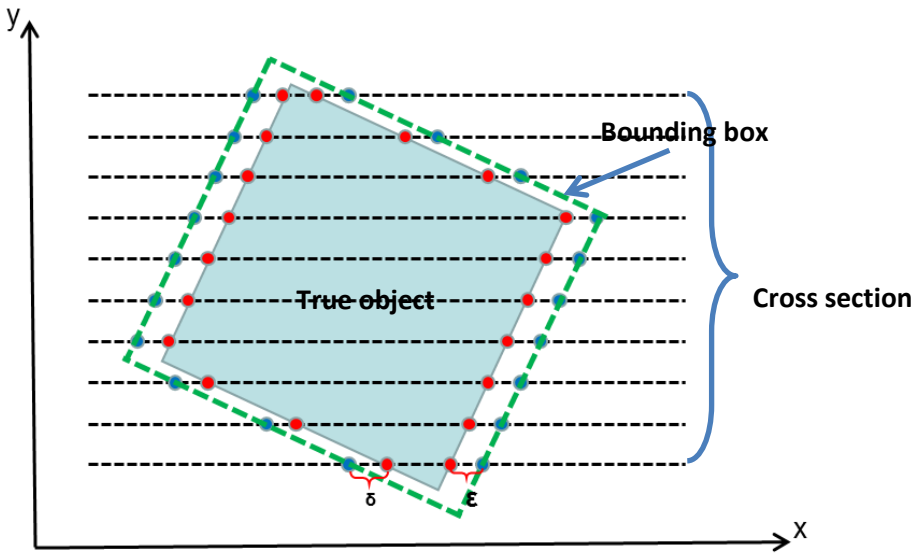


Figure 8.26: Plane view of a 3D object

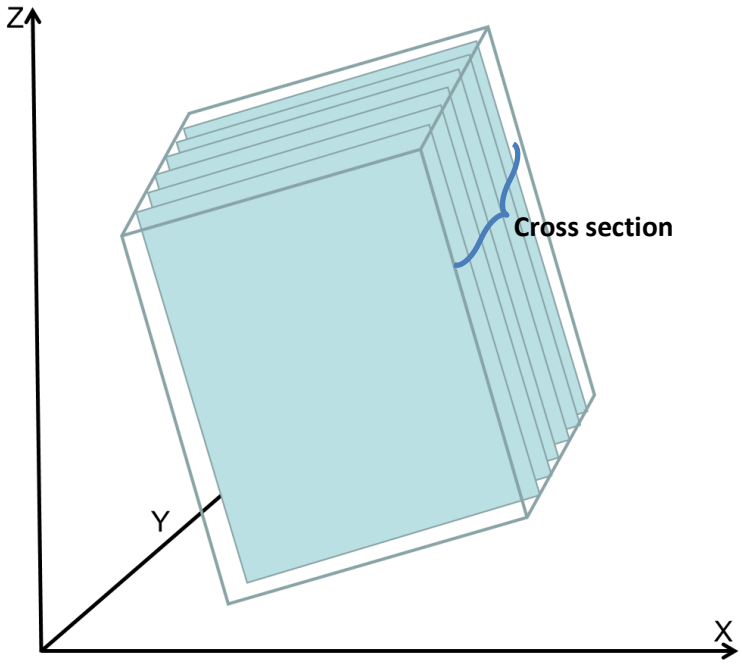


Figure 8.27: Cross-section

The 3D bounding box approach has been applied to Case 1 shown in Figure 8.10, and the resulting point cloud of the bounding box of the hexagonal prism is shown in Figure 8.28. The yellow object is the hexagonal prism and those green dots represent the boundary of the bounding box that encloses the object.

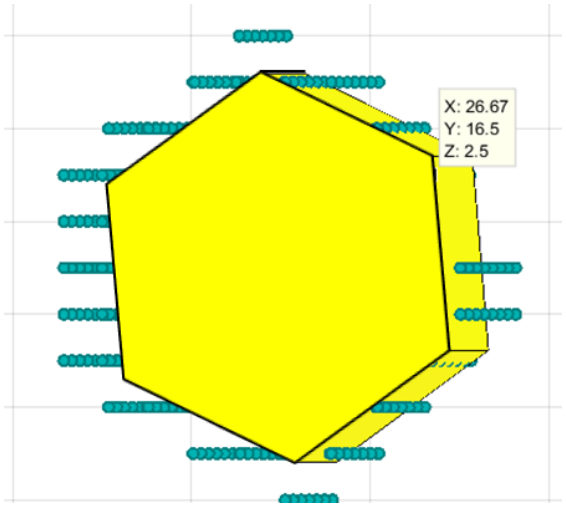


Figure 8.28: Point cloud of the hexagonal prism's bounding box

Both paths calculated by the A* algorithm in the workspace of the measurement object and the workspace of the bounding box for case 1 (see Figure 8.13) are shown in Figure 8.29. The old path is the path calculated for the measured object, and the new path is the path calculated for the bounding box. The advantage of the new path is that it avoids the collision problem caused by the measurement errors discussed at the beginning of Section 8.5. The bounding box method prevents the path from being too close to obstacles.

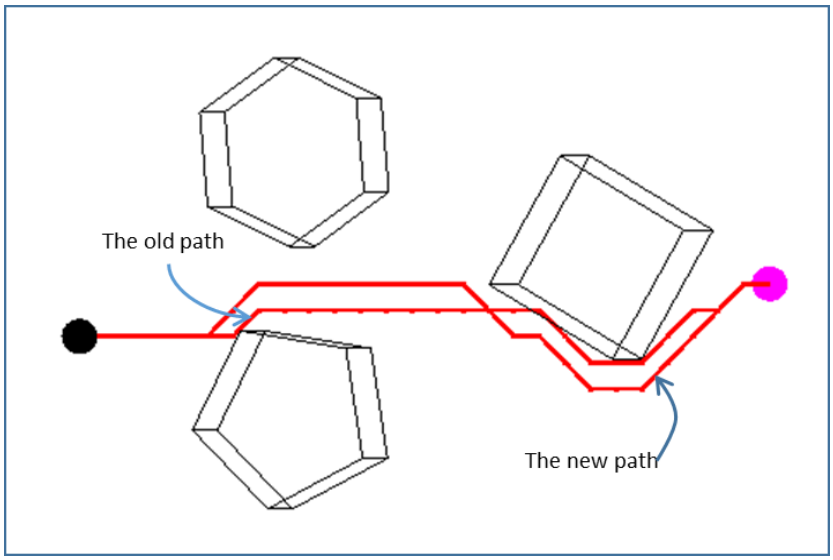


Figure 8.29: Planned path

8.5.3 Localisation Error

The positional error in the X and Y axis poses the same problem as the measurement error in Z-axis. The positional error is caused by the localisation error of the µAUV during aerial mapping mentioned in section 3.1. Although this thesis does not focus on the localization, it

is worth to mention that in principle the bounding box can be used to solve the collision problem caused by the localization error. After estimating the standard deviation of the localisation error, the problem can be solved by the same bounding box approach discussed above. As shown in Figure 8.30, it needs to trace the boundary of the measured object, find its vertices and then expands the bounding box accordingly

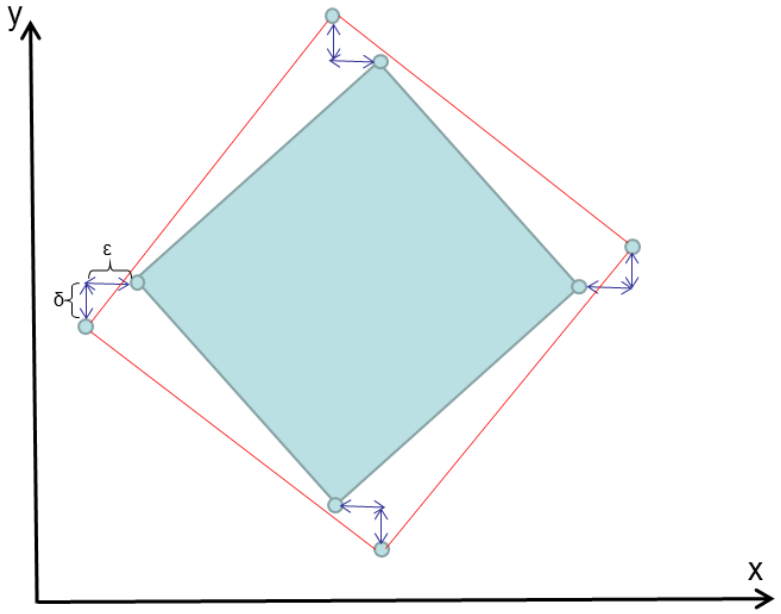


Figure 8.30: Bounding box for the positional error in X and Y-axis

The final type of error is the sampling error caused by sampling resolution, which is already discussed in Section 5.3.

8.6 Summary

This chapter explains how to process the depth measurements obtained by aerial mapping to establish a 3D grid map for path planning. This data processing system includes boundary tracing, new depth measurement data, linear height interpolation, and grid map construction. In a simulation study, the A* path planning algorithm is used in 3D grid maps obtained using the above procedure, which demonstrates this processing approach could be capable of being used in the pond mapping application. This data processing system integrates environment acquisition and path-planning, which is a key step for underwater navigation. This chapter also introduced and validated a bounding box approach to avoid the measurement object is smaller than the true object due to measurement errors in practice.

Chapter 9

Multiresolution Hierarchical Data Sampling and Path-Planning

9.1 Introduction

The complexity of the clutter distribution in legacy ponds is unknown, so in a disorganised pond, there will be some areas devoid of clutter, some areas accumulate of clutter, as the legacy pond shown in Figure 1.2 Section 3.4. In such the circumstance, a basic fixed-scale aerial mapping is inefficient and might miss information for narrow areas, thus there is a need for an approach that can obtain a map with a different level of details. The basic mapping is only a start. Once there is enough information the pond's about material distribution, further investigation of certain parts of the pond is necessary. This is where the need for higher resolution mapping.

To carry out detailed exploration in some specific areas, a multi-resolution hierarchical mapping and path planning approach is introduced. As the name suggests, this algorithm has two stages: Multi-resolution hierarchical data sampling and Multi-resolution hierarchical path-planning. Subbarao [126] develops a Quadree-planning algorithm based on the A* algorithm to speed up the path-planning process considerably. Tisotras [127] also proposed a Multi-resolution hierarchical path-Planning for small UAVs using wavelet decompositions.

In the first stage, the multi-hierarchy depth measurement map can be obtained by a multi-hierarchy aerial survey, and its corresponding 3D digital grid map can be obtained by the same method stated in section 8.2. The multi-hierarchy sampling method is based on the hypothesis that some regions in the 2D height map are less discriminable at certain resolutions than at others. Therefore, in this particular region, it requires a higher sampling resolution to acquire denser data than others.

In the second stage, the Multi-resolution A* algorithm is introduced, which corresponds to the multi-resolution digital grid map obtained by the multi-hierarchy sampling in the first stage. The Multi-resolution A* algorithm has at least two search radii (for finding its neighbour points), one for the low-resolution area, and one for the high-resolution area. The shift between the searching radii depends on the current region. Once the exploring reaches the high-resolution area, it will execute the smaller search radius. Otherwise, it will execute the greater search radius. The Multi-resolution A* path-planning algorithm enables the robot to search in the obtained hierarchical grid map in different resolutions. This chapter presents a Multi-resolution hierarchical data sampling and path-planning approach for autonomous agents.

9.2 Hierarchical Data Sampling

The data sampling process contains two phases. In the first phase, the environment is sampled at a low resolution. As a result, the low-resolution point cloud data of the environment is obtained. In the second phase, the data obtained at the low sampling resolution will be analysed to identify the obstacle areas, and then a higher sampling resolution will be implemented to these regions. Thus, the Multi-resolution point cloud data is obtained. Take the pond-like environment shown in Figure 9.1 as an example.

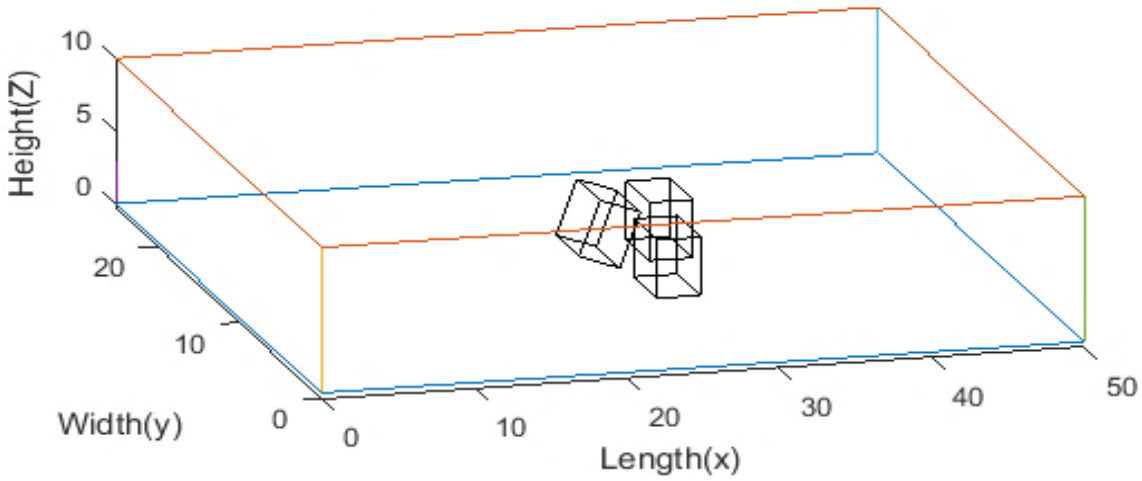


Figure 9.1: Geometry model of multiple obstacles

There are three cuboid obstacles placed in the 50 m×25 m×10 m pond, one of the obstacles leans against its nearby obstacle. In the first phase, the environment is sampled with a resolution of 1m×1m (low-resolution). The depth measurements obtained by the first phase data sampling (aerial survey) and its corresponding surface plot are shown in Figures 9.2 and 9.3, respectively.

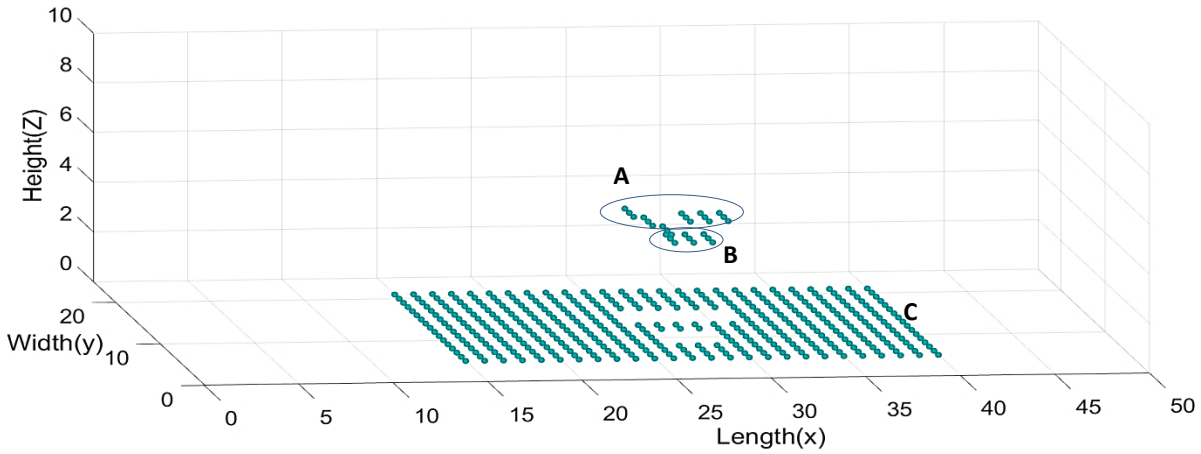


Figure 9.2: Depth measurement points obtained by first phase data sampling

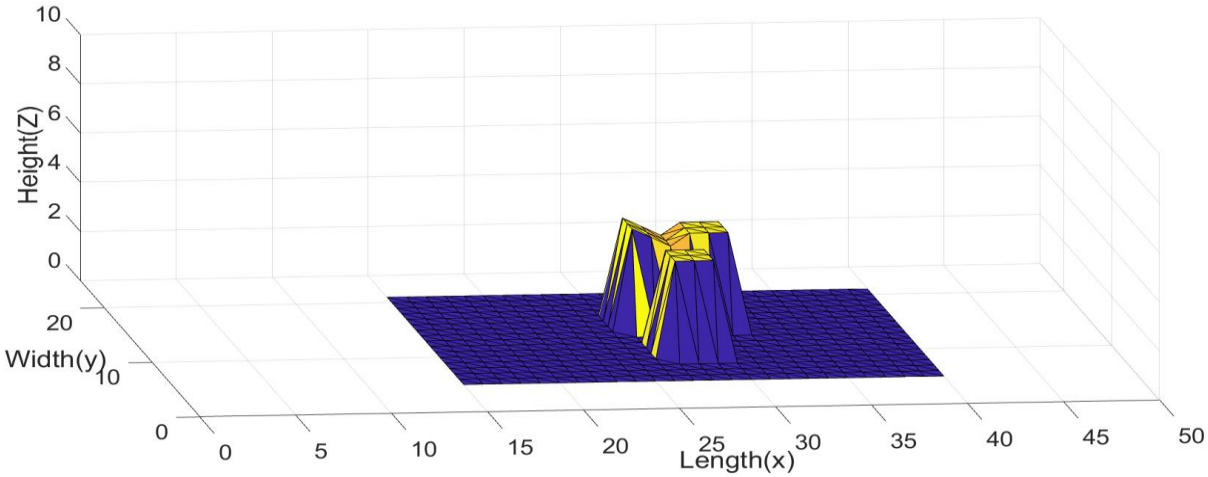


Figure 9.3: Reconstructed objects obtained by surface plot

In the second phase, the first job is to identify the occupation area of clutter and then apply the high-resolution second phase aerial survey to the occupation area of clutter. From the measurement results obtained by the first aerial survey shown in Figure 9.2 and 9.3, it is easy to identify that central areas of the pond accumulate clutter, and other areas devoid of clutter. Therefore, the central areas of the pond need a higher sampling resolution to obtain detailed information. Use the bounding box introduced in Section 8.5 to estimate the occupation regions of clutter and then apply a higher sampling resolution aerial survey to those regions. After that, the height interpolation algorithm (see Section 8.2.3) is applied to support the construction of a 3D point cloud map, resulting in a hierarchical 3D point cloud map. Assume that the sampling resolution for the second phase aerial survey in this example is 0.5 m×0.5 m.the corresponding point cloud of the environment shown in Figure 9.1 obtained by the multi-resolution survey is shown in Figure 9.4. It is easy to find the point cloud model in Figure 9.4 clearly shows three objects, while, depth measurements data in Figure 9.2 only shows two objects. The hierarchical adds sampling adds more details to the map.

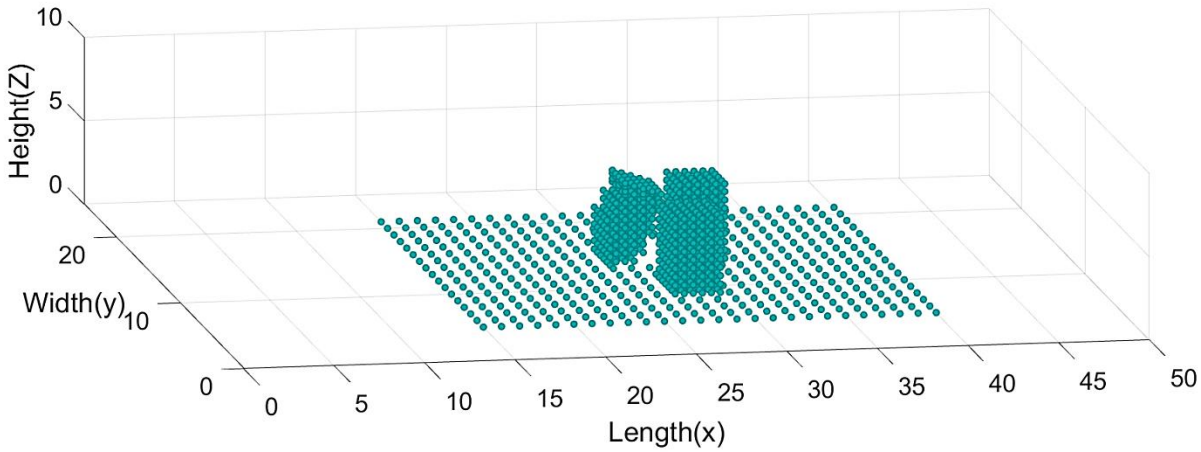


Figure 9.4: Hierarchical 3D point cloud map

Finally, the same step described in Section 8.2 is followed, which convert the 3D point cloud map to a 3D occupancy grid map.

9.3 Multiresolution A* Path-Planning

Multi-resolution A* Path-Planning algorithm is similar to the basic A* path-planning algorithm discussed in section 6.4.8. The difference is that, rather than searching in a constant radius, the search radius is variable. The evaluation function of a node n becomes:

$$f(n) = D(p,r) + h(n) \qquad 9.1$$

where $D(p,r)$ is the cost of the path from its parent node p to node n within search a radius r. $h(n)$ represents the heuristic estimate of the cost of the remaining path from n to the goal. If the grid map has two resolutions **a**×**a** and **b**×**b**, the Multi-resolution Path-Planar should have two searching radii: $\sqrt{2}a$ and $\sqrt{2}b$. Figure 9.5 shows the searching circle of the hierarchical 3D point cloud map shown in Figure 9.4.

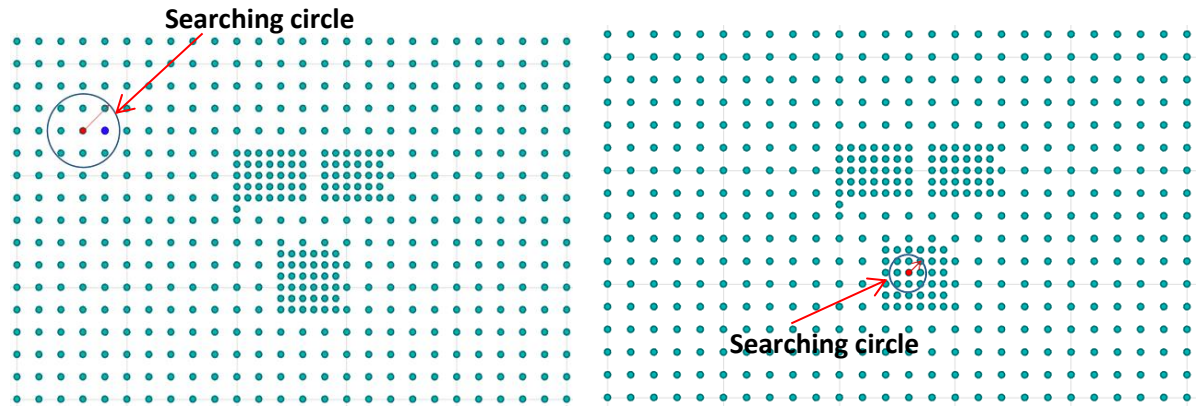


Figure 9.5: Multi-resolution search

It should be noted that when the edge between the **a×a** area and the **b×b** area is reached, the search radius will change. The following simulations successfully show Multi-resolution A* path-planning in different testing cases.

In the first testing case, assume the start is (16, 15, 1), the goal is (35, 15, 2), the planned path in the 3D grid map is shown in Figure 9.4:

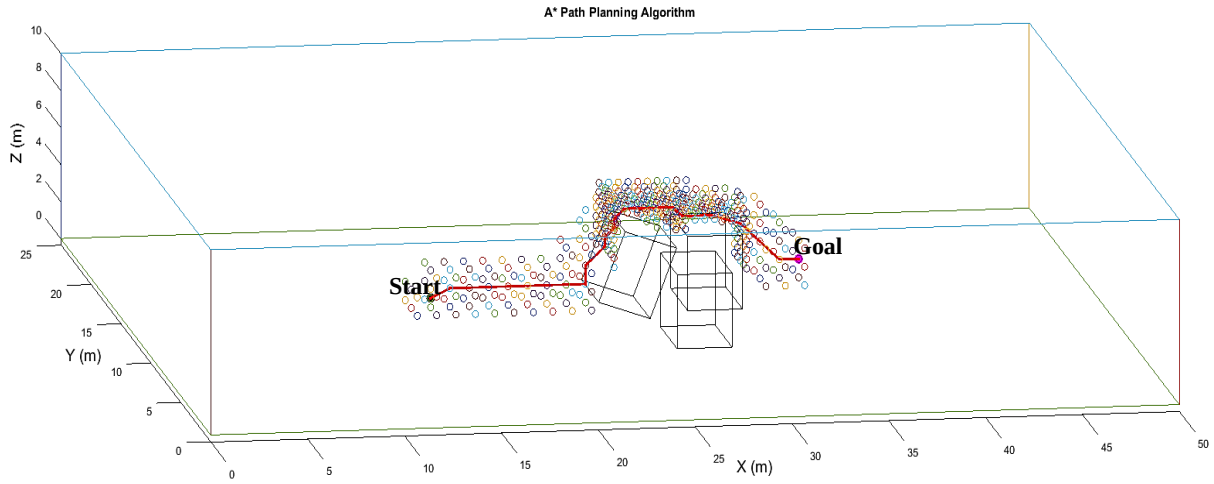


Figure 9.6: A* path on hierarchical 3D point cloud map

In the second testing case, assume the start is (16, 11, 2), the goal is (28.5, 15, 0.5) which located in the shadow area between the tilted object and its next object, the planned path in the 3D grid map is shown in Figure 9.4:

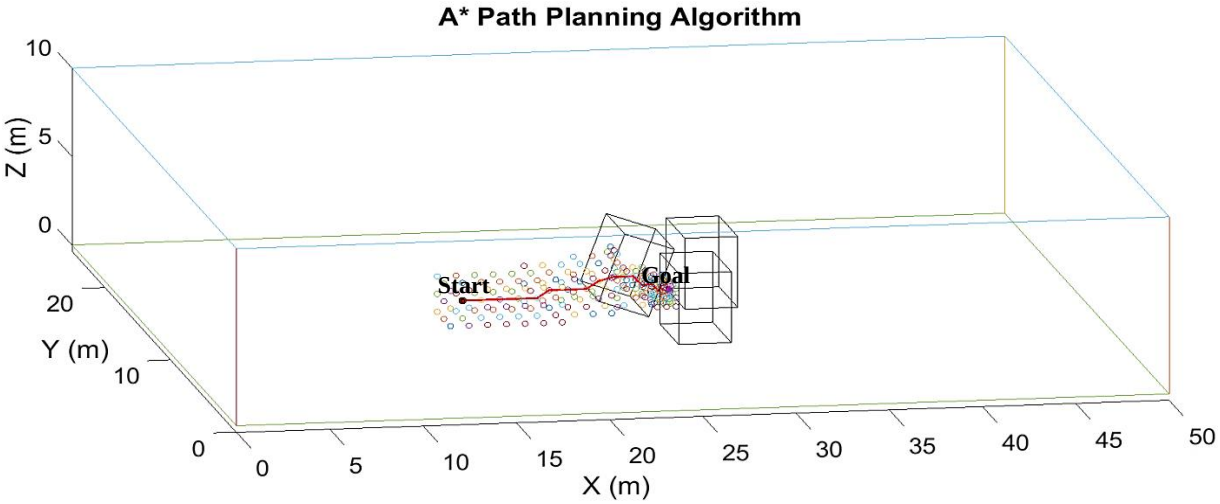


Figure 9.7: 3D view of A* path on hierarchical 3D point cloud map

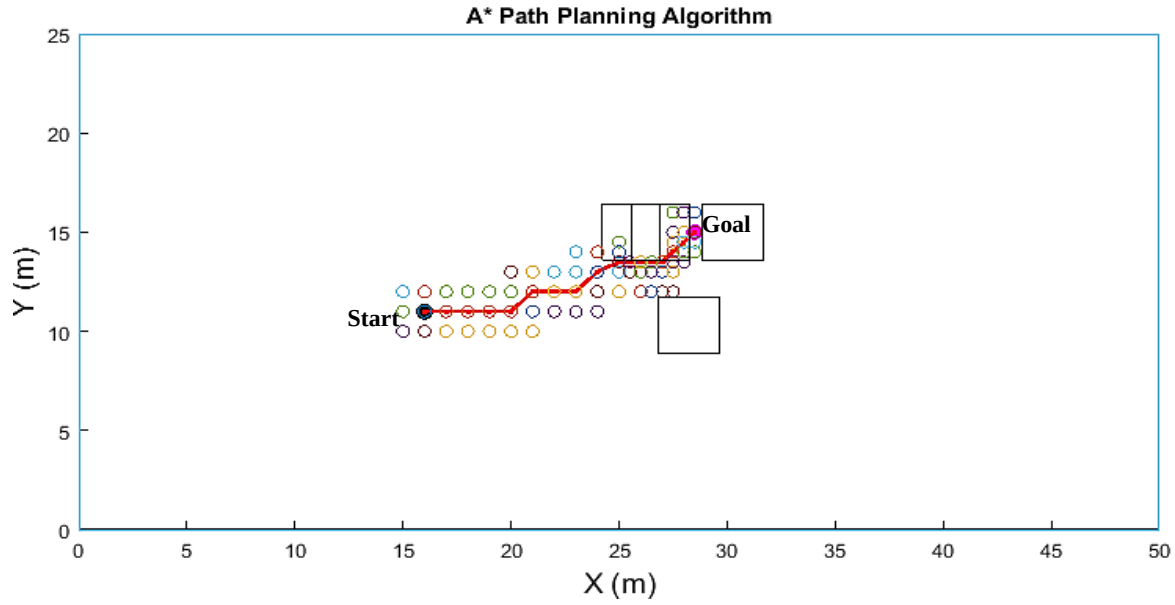


Figure 9.8: Plan view of A* path on hierarchical 3D point cloud map

Both results show that the Multi-resolution A* path-planning works as expected. Figure 9.9 shows a hierarchical 3D point cloud map of an even more cluttered and disordered environment. Figure 9.10 and Figure 9.11 show the corresponding collision-free path generated by the Multi-resolution A* path-planning algorithm.

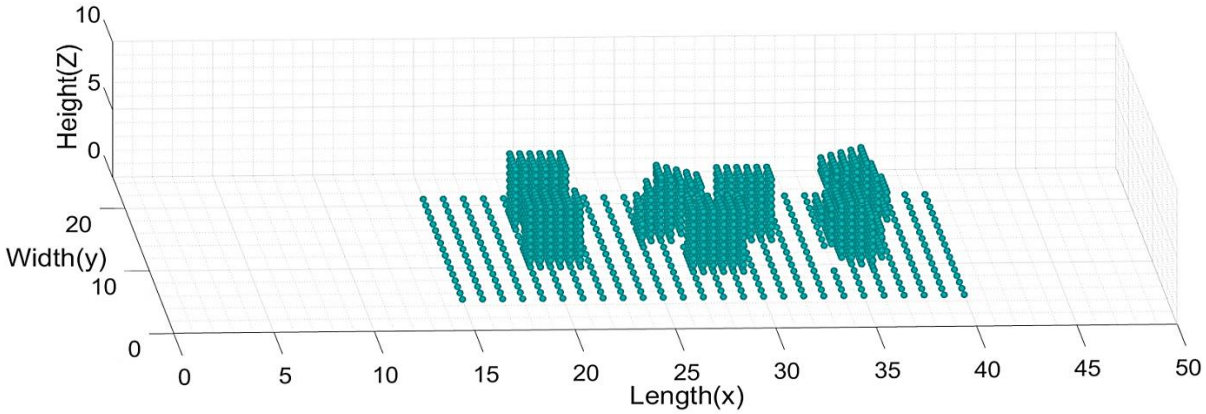


Figure 9.9: Hierarchical 3D point cloud map

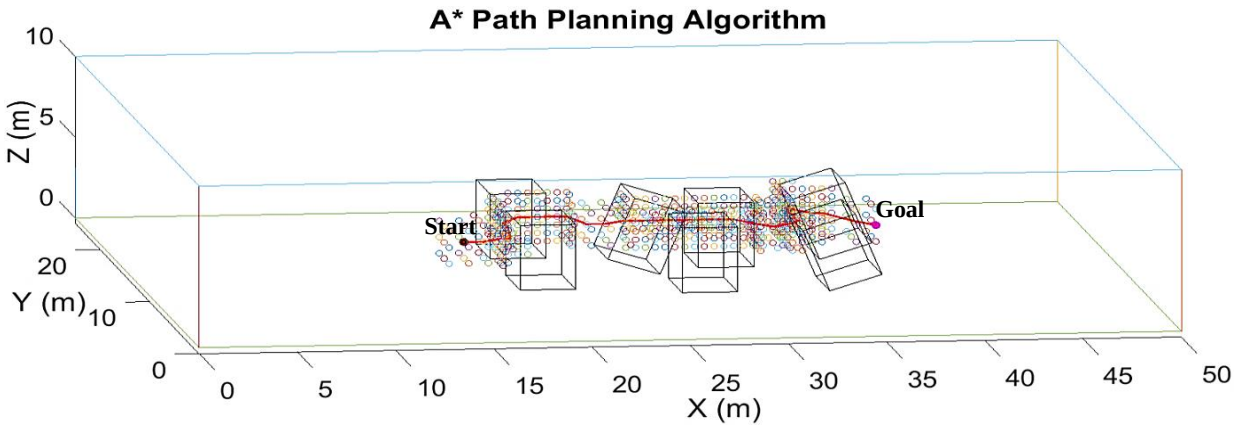


Figure 9.10: Multi-resolution A* path-planning for a cluttered environment

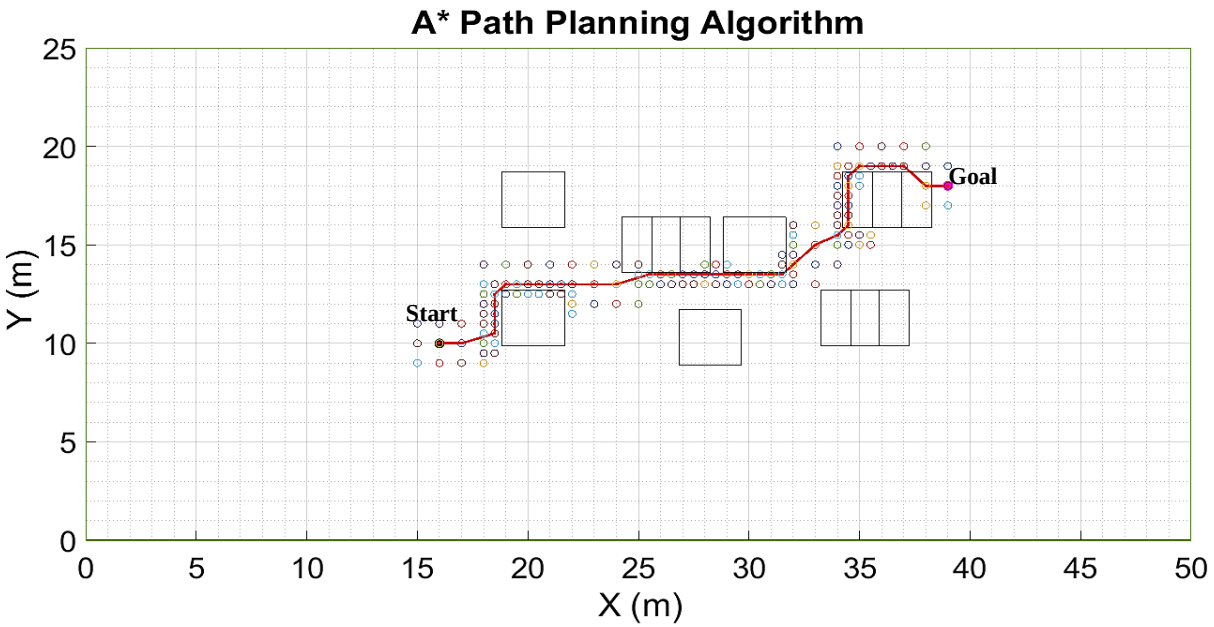


Figure 9.11: Plan view of multi-resolution A* path-planning for cluttered environment

Compared to the basic A* path-planning algorithm mentioned Chapter 8, it can work in the multi-resolution hierarchical map, which is more flexible.

9.4 Summary

This chapter has proposed a multi-resolution aerial mapping and path-planning algorithm that executes a low-resolution first phase aerial mapping procedure throughout the whole pond. Post-processing of results is then used to identify cluttered areas in order that a second, higher-resolution phase of aerial mapping is carried out in the identified cluttered areas. This method is suitable for an environment that needs a different level of detail in a different part of the environment.

Chapter 10

Conclusion and Future Work

This chapter briefly summarises the work presented in this thesis, draws conclusions from each chapter, and outlines potential future work directions that are worth carrying on.

10.1 Thesis Summary and Conclusions

A comprehensive review of underwater navigation strategies and mapping techniques was provided in Chapter 2. This concludes that echo-sounding is a good option for mapping a cluttered pond with a £300 low-cost sensor-limited µAUV.

Chapters 3, 4 and 5 are devoted to the development of aerial mapping for underwater map acquisition that could be implemented via echo-sounding using cheap and simple sensors. Due to the problems of conducting experiments in water pond as discussed in Section 3.5, the work is decided to start with computer simulation. In the aerial mapping simulation, the ray tracing algorithm is introduced to simulate the propagation of acoustic waves as rays.

The echo sounding method relies on a perpendicular reflection from an object arriving back at the sensor so that TOF can be measured, and the distance between the sensor and the detected point obtained. The outcome of the echo-sounding aerial mapping is a height topological map of the upper surface of the canisters that placed the pond. Throughout the study, it was found that this requirement cannot be met for sloping surfaces if their slope angle is bigger than the beam angle of the sensor. Methods for addressing this problem were developed. For example, using three narrow beam sensors to create a sensor array and ensure that the beam of each sensor does not overlap with each other to create a wide beam.

An extensive simulation in Chapter 5 study finds that the accuracy of mapping results depends on resolution, as expected, and also on the height of the survey plane. A series of experiments were designed to validate the echo-sounding aerial mapping and study the

mapping results under different scenes, different beam angles, and different survey heights. It demonstrates that aerial mapping can provide a good amount of information.

Chapter 6 reviewed several notable path-planning algorithms and also discussed their application to the pond environment. It was found that RRT, A*, APF, SA-APF, and RRT-A* could be used for storage pond path-planning. Chapter 7 studied the performance of those algorithms in a number of storage pond clutter layouts by MATLAB simulation. The simulation shows that APF can fall into a local minimum as a result of the high clutter density, and the RRT algorithm generates a much longer path. The other algorithms, the A*, SA-APF, and RRT-A*, can plan a collision-free path. Simulations of paths requiring entry into hollow canisters showed that SA-APF cannot find a path to the goal unless an intermediate checkpoint was introduced. This was done separately. Both the A* and RRT-A* algorithms can plan a collision-free path in these tests, but in all testing cases, the path planned by the A* algorithm is slightly shorter than that planned by the RRT-A* algorithm. It found that the A* algorithm has the best performance, while others are less successful because their paths are longer or even fail to reach the goal like the APF-based algorithm failed to reach the goal in the hollow canister simulation case.

Chapter 8 proposed a novel approach that utilises depth measurement data to construct an occupancy grid map which can be imported into the A* path-planning algorithm to find collision-free paths. This was achieved by introducing a perpendicular interpolation algorithm to add new points to fill the space between the ground and the object's top surface point sets. This enables a 3D point cloud representing to be generated based on the assumption that the shape of spent fuel containers is prismatic. After the basic point cloud has been obtained, the whole workspace is decomposed to the grid, and then each point in the 3D point cloud is aligned to the corresponding aerial mapping grid. Finally, the point cloud map is converted to the occupancy grid map.

Once the occupancy grid map has been imported into the A* algorithm, the path planner can search on. Simulation test cases demonstrated that this approach works in pond-like environments successfully. Since the chosen sensor is low cost, the sensor error cannot be ignored. In case the sensor measurement error causes an accidental collision with the obstacle, the bounding box is used to expand the occupied space of the obstacle. The contribution of this part of the research is the development of an approach to constructing an occupancy grid

map based on the raw depth measurement data obtained by the low-cost range finder via simple echo sounding.

Chapter 9 introduced multi-resolution aerial mapping with the A* path-planning method, which can be used to increase the accuracy of survey results for highly cluttered zones. The idea of the method is to perform a low sampling resolution mapping in the first phase, then identify the rough area of obstacles and then perform high-resolution mapping in the second phase. It will process the obtained data in the same manner as Chapter 8. The final established occupancy grid map will have at least two grid resolutions and the A* algorithm is capable to find out collision-free paths. Chapter 9 also presented some simple simulations validating the multi-resolution aerial mapping and the A* path-planning.

The aim of this thesis was to investigate the navigation strategy and path-planning of μAUVs for monitoring and taking measurements in an enclosed cluttered underwater environment such as the nuclear storage pond. Throughout this thesis, it is clear that some elements of the initial objectives of the research have been accomplished successfully with optimistic results. An approach composed of environment acquisition, map reconstruction, and path-planning is developed. The work includes developed an echo-sounding based aerial mapping to gather the height map of the storage pond, a perpendicular height interpolation algorithm to create point cloud to represent the pond environment, and an occupancy grid map used for A* path-planning algorithm. This approach allows μAUVs with low-cost ultrasound ranger finders to map an unknown pond and plan a collision-free path in the obtained map about the pond. The simulations in Section 8.4 show the proposed approach work successfully. However, it is worth noticing that this work is only the start of research on the path-planning problem in nuclear-decommissioning environments, so there still a long way to go. One of the shortcomings of the work is the lack of practical work.

10.2 Future Research

Underwater navigation for small size and sensor-limited AUVs is a very challenging subject. The research conducted throughout this thesis leads to an approach taking simple aerial survey data and constructing a three-dimensional map from which an occupancy grid map can be derived and used for path-planning. However, it is only the start, and there are many unexplored areas that need to be addressed in the future. The work that is worth undertaking in future is discussed in this section.

10.2.1Vertical Survey

As explained in Section 8.2.3, this thesis uses the height interpolation approach to construct the point cloud model of an object based on the aerial mapping. However, for the overlap area discussed in Section 5.9, aerial mapping cannot detect the overlap area. A possible solution to this is to conduct vertical mappings to get side projection data of the environment. Future research should investigate the implementation of vertical mappings and how to integrate the vertical mapping data with aerial mapping data to construct the environment.

10.2.2 Further Experiments

Because of the current limitations of a lack of testing environments, the current aerial mapping validation experiments only conducted in an open-air testing rig that has several small-sized cuboids to represent well-organised spent fuel canisters. This thesis has studied many in-air aerial experiments, but more extensive in-air aerial experiments might be considered, such as more representatives of pond cases, sloping cases, and overlap cases. Meanwhile, the future practical experiments aim to investigate aerial mapping conducted by μAUV in a water tank that contains some canister-shaped obstacles. The future practical experiments aim to investigate the aerial mapping of irregular placed spent fuel canisters, especially the situation that objects leaning against each other and very close to each other and also the sloping obstacle issue. An aerial mapping depth analyser needs to be developed in order to analyse the raw depth measurement data, which enables the identification of some particular situation. For example, distinguishing a scene is two closely spaced but separate objects or one single object. This could potentially use a Neural Network to build a system to analyse this data.

10.2.3 Further Development of the A* Path-Planning Algorithm

According to Chapter 8, the planned collision-free paths obtained by A* algorithm are zig-zag and close to obstacles, which is not good for practical implementation. To solve these issues, the thesis introduced the bounding box approach to oversize the object. Other solutions are also of interesting, such as introducing an artificial potential field (APF) to the A* algorithm to smooth the path and keep the path away from obstacles, or improving the heuristic function in A* algorithm.

10.2.4 True Size µAUV Path-Planning

The current simulations based on the assumption of using a simple point with three degrees of freedom to model the movement of a µAUV. Therefore, the µAUV moves in a constant speed and maximum turning rate and its path can be parametrised with straight line segments and circular arc with any radius. However, a µAUV has constraints about the size, speed, degrees of freedom, and mobility in practice. They all need to be considered in practical implementations. Hence, the final topic of future work is taking speed control, motion control, and the size of µAUV into consideration. These would allow the implementation of µAUV in practice to validate the navigation and path-planning approach developed in this thesis.

Reference

[1] S. L. Sharma, A. Qavi, and K. Kumari, “Oil Pipelines/Water Pipeline Crawling Robot for Leakage Detection/Cleaning of Pipes,” *Global Journal of researches in engineering*, vol. 14, no. 1, 2014.

[2] S. Nawaz, M. Hussain, S. Watson, N. Trigoni, and P. N. Green, “An underwater robotic network for monitoring nuclear waste storage pools,” *Lect. Notes Inst. Comput. Sci. Soc. Telecommun. Eng.*, vol. 24, pp. 236–255, 2010.

[4] R. B. Wynn *et al.*, “Autonomous Underwater Vehicles (AUVs): Their past, present and future contributions to the advancement of marine geoscience,” *Mar. Geol.*, vol. 352, pp. 451–468, 2014.

[5] F. A. Azis, M. S. M. Aras, M. Z. A. Rashid, M. N. Othman, and S. S. Abdullah, “Problem identification for Underwater Remotely Operated Vehicle (ROV): A case study,” *Procedia Eng.*, vol. 41, pp. 554–560, 2012.

[6] World Nuclear Association “Radioactive Waste Management”, *World Nuclear Association*, 2015 updated. [Online]. Available:

http://www.world-nuclear.org/information-library/safety-and-security/safety-of-plants/fukushima-accident.aspx

[7] S. Stanley, T.A. York P.N. Green P.R. Green A. Phasouliotis Z. Qu S. Watson M. Hussain S. Nawaz N. Trigoni, “Acoustic Sensor Networks for Decommissioning,” Faculty *of Engineering and Physics Science, University of Manchester,* 2012.

[8] Sellafield Ltd, “Sludge removal marks significant step in Sellafield clean-up”, *Sellafield Ltd*, 2015.

[9] Picture from “News story Game changing progress in Sellafield pond,” *Sellafield Ltd and Nuclear Decommissioning Authority,* 2018. [Online]. Available:

https://www.gov.uk/government/news/game-changing-progress-in-sellafield-pond

[10] Z Shiller. Off-Line and On-Line Trajectory Planning. In: Carbone G., Gomez-Bravo F. (eds) Motion and Operation Planning of Robotic Systems. Mechanisms and Machine Science, 2015, vol 29. *Springer*, Cham.

[11] H. Durrant-whyte and T. Bailey, "*Simultaneous Localisation and Mapping (SLAM): Part I The Essential Algorithms*," IEEE Robotics & Automation Magazine (Volume: 13, Issue: 2, June 2006.

[12] S A Watson, "Mobile platforms for Underwater Sensor Network", Faculty of Engineering and Physics Science, PhD thesis, University of Manchester, 2012.

[13] M. Nancekievill *et al.*, "A Remote-operated System to Map Radiation Dose in the Fukushima Daiichi Primary Containment Vessel," *EPJ Web Conf.*, vol. 170, p. 06004, 2018.

[14] A. Griffiths, A. Dikarev, P. R. Green, B. Lennox, X. Poteau, and S. Watson, "AVEXIS—Aqua Vehicle Explorer for In-Situ Sensing," *IEEE Robot. Autom. Lett.*, vol. 1, no. 1, pp. 282–287, 2016.

[15] Figure Available at [Online]: http://uomrobotics.com/nuclear/roboticplatforms/aquatic/avexis/avexis150.html

[16] P. Saini, R. P. Singh, and A. Sinha, "Path loss analysis of RF waves for underwater wireless sensor networks," *2017 Int. Conf. Comput. Commun. Technol. Smart Nation, IC3TSN 2017*, pp. 104–108, 2018.

[17] W. B. Schulte III, "The frequency response, impulse response and transfer function of an ocean waveguide," *Naval postgraduate school, University of California*, June 2004.

[18] J. C. Kinsey, R. M. Eustice, and L. L. Whitcomb, "A survey of underwater vehicle navigation: recent advances and new challenges," *in Proceedings of the IFAC Conference of Manoeuvring and Control of Marine Craft*, vol. 88, 2006.

[19] H. Rice, S. Kelmenson, and L. Mendelsohn, "Geophysical navigation technologies and applications," *Position Location and Navigation Symposiu*, (IEEE Cat. No.04CH37556)pp. 618–624, 2004.

[20] G. Karras, S. Loizou, and K. Kyriakopoulos, "On-line state and parameter estimation of an under-actuated underwater vehicle using a modified dual unscented Kalman filter," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010.

[21] L. Huang, B. He, and T. Zhang, “An autonomous navigation algorithm for underwater vehicles based on inertial measurement units and sonar,” *CAR 2010 - 2010 2nd Int. Asia Conf. Informatics Control. Autom. Robot.*, vol. 1, pp. 311–314, 2010.

[22] G. Chen, J. Han, J. Wu, and Y. Ying, “The integrated navigation system of AUV ‘XUANWU’,” *Ocean. 2016 - Shanghai*, pp. 1–5, 2016.

[23] R. McEwen and H. Thomas, “Performance of an auv navigation system at arctic latitudes,” *IEEE J. Oceanic Eng.*, vol. 30, pp. 443–454, April 2005.

[24] L. Stutters, H. Liu, C. Tiltman, and D. J. Brown, “Navigation technologies for autonomous underwater vehicles,” *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.*, vol. 38, no. 4, pp. 581–589, 2008.

[25] L. Paull, S. Saeedi, M. Seto, and H. Li, *“AUV Navigation and Localization: A Review”, IEEE Journal of oceanic engineering*, vol. 39, no. 1, January 2014.

[26] K. Vickery, “Acoustic positioning systems. A practical overview of current systems,” *Proceedings of the Workshop on Autonomous Underwater Vehicles*, 1998.

[27] The Pythagorean Theorem. [Online]. Available: https://www.mathplanet.com/education/pre-algebra/right-triangles-and-algebra/the-pythagorean-theorem

[28] Ernest Jacobs, Elizabeth W. Ralston, “Ambiguity Resolution in Interferometry,” *IEEE Transactions on Aerospace and Electronic Systems*, Volume: AES-17, Issue: 6, Nov. 1981.

[29] J. Melo and A. Matos, “Survey on advances on terrain based navigation for autonomous underwater vehicles,” *Ocean Eng.*, vol. 139, pp. 250–264, 2017.

[30] F. C. Teixeira and A. M. Pascoal, “Geophysical Navigation of Autonomous Underwater Vehicles Using Geomagnetic Information,” *IFAC Proc. Vol.*, vol. 41, no. 1, pp. 178–183, 2008.

[31] W. Uddin, “Airborne LIDAR Digital Terrain Mapping for Transportation Infrastructure Asset Management,” *5th International Conference on Managing Pavements*, 2001.

[32] Figure obtained from Riegl, “Riegl Lms-Q780,” 2015. [Online]. Available at http://www.riegl.com/uploads/tx_pxpriegldownloads/DataSheet_LMS-Q780_2015-03-24.pdf

[33] H. Wenquan, *"*Research on Analyze Accuracy of Lidar Data in Surveying Projects*," The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*. Vol. XXXVII. Part B4. Beijing 2008.

[34] 2GRobotics.com, "SONAR VS. LASER," *2GRobotics.com.*

[35] T. S. Khye, N. C. Sim, S. S. Houyou, "Comparative Analysis of Radar and Sonar Principles", *Defense science and technology agency*.

[36] D. Kohanbash, "*LIDAR vs RADAR: A Detailed Comparison*", 2017. [Online]. Available at: http://robotsforroboticists.com/lidar-vs-radar/

[37] C. De Moustier and H. Matsumoto, "Seafloor acoustic remote sensing with multibeam echo-sounders and bathymetric sidescan sonar systems," *Mar. Geophys. Res.*, vol. 15, no. 1, pp. 27–42, 1993.

[38] Figure availiable at [Online]: https://palanquee.fr/StarFish-450F-poisson-tracte

[39] N. F. Jansen, "Short Range Object Detection and Avoidance," *Traineesh. Rep.*, no. 916, p. 17, 2010.

[40] H.R. Everett, D.E. DeMuth, and E.H. Stiz. "Survey of collision avoidance and ranging sensors for mobile robots". Technical Report 1194, Naval Comand Control and Ocean Surveillance Center,RDT&E Division, San Diego, California, December 1992.

[41] "Storage of water reactor spend fuel in water pools," *International Atomic Energy Agency*, 1982.

[42] T. A. York, "Acoustic sensor networks for decommissioning," *Meas. Control*, vol. 45, no. 2, pp. 48–54, 2012.

[43] "Speed of sound - temperature matters, not air pressure," [Online]. Available: http://www.sengpielaudio.com/SpeedOfSoundPressure.pdf

[44] David Oliva Elorza, "Room acoustics modelling using the ray tracing method: implementation and evaluation", University of Turku Department of Physics, 2005.

[45] L. L. Beranek, "Analysis of Sabine and Eyring equations and their application to concert hall audience and chair absorption," *J. Acoust. Soc. Am.*, vol. 120, no. 3, pp. 1399–1410, 2006.

[46] J. Kirschner, "Surface Acoustic Wave Sensors (SAWS)," *Microelectromechanical Syst.*, pp. 1–11, 2010.

[47] Lambert, J H. “Photometria, sive de Mensura et gradibus luminis, colorum et umbrae”. 1760

[48] Y. W. Lam, “A comparison of three diffuse reflection modeling methods used in room acoustics computer models,” *J. Acoust. Soc. Am.*, vol. 100, no. 4, pp. 2181–2192, 2005.

[49] G. D. Jones, “Underwater Distance Measurement Sensor for a Micro-Autonomous Underwater Vehicle,” Third Year Project Report, the University of Manchester, 2013.

[50] W. Gu, M. Aminikashani, P. Deng, and M. Kavehrad, “Impact of Multipath Reflections on the Performance of Indoor Visible Light Positioning Systems,” *J. Light. Technol.*, vol. 34, no. 10, pp. 2578–2587, 2016.

[51] M. Levoy and T. Whitted, “The Use of Points as a Display Primitive,” *Computer Science Department University of North Carolina Chapel Hill*, 1985.

[52] A. Sveier, “Primitive Shape Detection in Point Clouds,” Norwegian University of science and technology, 2016.

[53] B. Braden, “The Surveyor’s Area Formula,” *The College Mathematics Journal*, Vol. 17, NO. 4, 1986.

[54] P. Tsiotras, D. Jung, and E. Bakolas, “Multiresolution hierarchical path-planning for small UAVs using wavelet decompositions,” *J. Intell. Robot. Syst. Theory Appl.*, vol. 66, no. 4, pp. 505–522, 2012.

[55] “Photographs of Sellafield nuclear plant prompt fears over radioactive risk,” The Guardian.[Online]. Available: https://www.theguardian.com/environment/2014/oct/29/sellafield-nuclear-radioactive-risk-storage-ponds-fears

[56] L. Yang,1,2 J. Qi,1 D. Song, J. Xiao, J. Han, and Y. Xia “Survey of Robot 3D Path-planning Algorithms”, *Journal of Control Science and Engineering*, Vol. 2016, Article ID 7426913, 22 pages, 2016.

[57] L. V. J. and S. A. A., “Path-Planning Strategies for a Point Mobile Automaton Moving Amidst Unknown Obstacles of Arbitrary Shape,” *Algorithmica*, pp. 403–430, 1987.

[58] Tomás Lozano-Pérez. “Spatial Planning: A Configuration Space Approach,” *IEEE Transactions on Computers* C-32.2 (1983), pp. 108–120.

[59] C. Goerzen, Z. Kong, and B. Mettler, "*A survey of motion planning algorithms from the perspective of autonomous", UAV guidance*, J Intell Robot Syst. vol. 57, no. 65–100. 2010.

[60] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Auton. Robots*, vol. 34, no. 3, pp. 189–206, 2013.

[61] S. M. LaValle. "Cell Decompositions in Planning Algorithms," *Cambridge University Press*, New York, NY, USA. ch. 6, pp. 264–273. 2006

[62] T. Van Der Putte, "Using the discrete 3D Voronoi diagram for the modelling of 3D continuous information in geosciences," *Delft University of Technology*.

[63] T. Lozano- Perez and M. A. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles", *Contmum. ACM, 22*, pages 560-570, 1979.

[64] M. N. Bygi, "3D Visibility Graph," *Department of Computer Engineering Sharif University of Technology*, 2007.

[65] E. Welzl. "Constructing the visibility graph for n-lines segments in $O(n^2)$ time," *Information Processing Letters*, 20: pp.167-171, 1985.

[66]S. M. LaValle, "Rapidly-exploring random trees a new tool for path-planning," *Tech. Rep. 98-11, Computer Science Department,Iowa StateUniversity, Ames, Iowa, USA.* 1998.

[67] L.E. Kavraki, M.N. Kolountzakis, J.-C. Latombe, "Analysis of probabilistic roadmaps for path planning," *IEEE Transaction on robotics and automation*, vol. 14, no. 1, February, 1998.

[68] Zhang Y, Valavanis K P. A 3-D potential panel method for robotmotion planning[J]. *Robotica*, 15(4): 421-434, 1997.

[69] L. Zhang, Y. J. Kim, and D. Manocha, "A hybrid approach for complete motion planning," *IEEE Int. Conf. Intell. Robot. Syst.*, pp. 7–14, 2007.

[70] H. Seki, Y. Kamiya, and M. Hikizu, "Real-Time Obstacle Avoidance Using Potential Field for a Nonholonomic Vehicle," *Fact. Autom.*, 2012.

[71] N. Achour and M. Chaalal, "Mobile Robots Path-planning using Genetic Algorithms," *The Seventh International Conference on Autonomic and Autonomous Systems*, 2011.

[72] C. H. Fan, W. D. Chen, Y. G. XI," A neural networks-based approach to safe path of mobile robot in unknown environment", *Acta Automatica Sinica*, 2004.

[73] K. F. Uyanik, “A study on Artificial Potential Fields,” KOVAN Research Lab. Dept. of Computer Eng. Middle East Technical Univ. Ankara, Turkey, 2011.

[74] P. Bhattacharya and M. L. Gavrilova, “Roadmap-Based Path Planning,” *IEEE Robot. Autom. Mag.*, vol. 15, no. 2, pp. 58–66, 2008.

[75]N. M. Amato and Y. Wu, “A randomized roadmap method for path and manipulation planning,” *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, pp. 113–120, 1996.

[76] A. D. Angelo, “A Brief Introduction to Quadtrees and Their Applications,” *28th Canadian Conference on Computational Geometry*, 2016.

[77] J. Yun, S. Lee, and Y. Jung, “Development of a gap searching program for automotive body assemblies based on a decomposition model representation,” *Adv. Eng. Softw.*, vol. 81, no. C, pp. 7–16, 2015.

[78] Figure from: H. Choset, “Robotic Motion Planning: Cell Decompositions”, *The Robotics Institute Carnegie Mellon University*.

[79] M. Seda, “Roadmap Methods vs. Cell Decomposition in Robot Motion Planning,” *Proc. 6th WSEAS Int. Conf. Signal Process. Robot. Autom.*, pp. 127–132, 2007.

[80] M. Williams and D. I. Jones, “A rapid method for planning paths in three dimensions for a small aerial robot,” *Robotica*, vol. 19, no. 2, pp. 125–135, 2001.

[81] H. Choset et al., “Principles of Robot Motion: Theory, Algorithms, and Implementations,” *The MIT Press*, Cambridge, Massachusetts, 2005.

[82] G. Dudek, M.Jenken, Computational Principles of Mobile Robotics, second edition, pp184, section 6.3.2, *Cambridge university press*, 2010.

[83] L. Liu and S. Zhang, “Voronoi diagram and GIS-based 3D path-planning,” *17th Int. Conf. Geoinformatics*, 2009.

[84] R. Omar, “Path-planing for unmanned aerial vehicles using visibility line-based methods,” Thesis submitted for the degree of Doctor of Philosophy at the University of Leicester by Department of Engineering, March, 2011.

[85] Bygi, Mojtaba Nouri, and Mohammad Ghodsi. "3D visibility graph." *Computational Science and its Applications, Kuala Lampur,* 2007.

[86] T. Lozano-Pérez and M. A. Wesley, “An algorithm for planning collision-free paths among polyhedral obstacles,” *Commun. ACM*, vol. 22, no. 10, pp. 560–570, 2002.

[87] F. Schøler, Flemming. “3d path planning for autonomous aerial vehicles in constrained spaces,” PhD Diss., Section of Automation & Control, Department of Electronic Systems, Aalborg University, 2012.

[88] Hrabar. S. E. “Vision-based 3D navigation for an autonomous helicopter”. *University of Southern California*, 2006.

[89] Y. Chen, X. Zhao, and J. Han, “Review of 3D Path-planning Methods for Mobile Robot,” *Robot*, vol. 32, no. 4, pp. 568–576, 2010.

[90] L.E. Kavraki, P. Svestka, J.-C. Latombe, M.H. Overmars “Probabilistic roadmaps for path-planning in high-dimensional configuration spaces”, *IEEE Transactions on Robotics and Automation,* Volume: 12 , Issue: 4 , Aug 1996.

[91] L. Yang, J. Qi, D. Song, J. Xiao, J. Han, and Y. Xia, “Survey of Robot 3D Path-planning Algorithms,” *J. Control Sci. Eng.*, vol. 2016, pp. 1–22, 2016.

[92] Karaman S, Frazzoli E. “Sampling-based algorithms for optimal motion planning [J]”. *The International Journal of Robotics Research*, 2011, 30(7):846-894.

[93] LAValle S M. “Planning algorithms[M]”. *Cambridge university press*, 2006.

[94] K. Yang and S. Sukkarieh, “3D Smooth Path-planning for a UAV in Cluttered Natural Environments”, *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2008.

[95] Figure from: H. Choset, “Robotic Motion Planning: Potential Functions”, *The Robotics Institute Carnegie Mellon University*.

[96] O. Khatib, “Real-time obstacle avoidance for manipulators and mobile robots,” *Proceedings - IEEE International Conference on Robotics and Automation*. pp. 500–505, 1985.

[97] J. Guo, Y. Gao, and G. Cui, “Path-planning of mobile robot based on improved potential field,” *Information Technology Journal*, vol. 12, no. 11. pp. 2188–2194, 2013.

[98] S.S.Ge, Y.J.Cui. “New Potential Functions for Mobile Robot Path-planning”. *IEEE Transactions on Robotics and Automation,* vol. 16(5), pp.615-620, 2000.

[99] Q. Zhu, Y. Yan, and Z. Xing, “Robot path-planning based on artificial potential field approach with simulated annealing,” *Proc. - ISDA 2006 Sixth Int. Conf. Intell. Syst. Des. Appl.*, vol. 2, pp. 622–627, 2006.

[100] D. J. Peuquet, “An algorithm for calculating minimum Euclidean distance between two geographic features,” *Comput. Geosci.*, vol. 18, no. 8, pp. 989–1001, 1992.

[101]S. Kirkpatrick, C. D. Gellat, and M. P. Vecchi," Optimization by simulated annealing," *Science*, vol. 220, No. 4598, May 1983.

[102] P. Vadakkepat, Tong Heng Lee, and Liu Xin, “Application of evolutionary artificial potential field in robot soccer system,” Proceedings Joint 9th IFSA World Congress and 20th NAFIPS International Conference, pp. 2781–2785, 2002.

[103] J. Tan, L. Zhao, Y. Wang, Y. Zhang, and L. Li, “The 3D path-planning based on A∗ Algorithm and artificial potential field for the rotary-wing flying robot,” *Proc. - 2016 8th Int. Conf. Intell. Human-Machine Syst. Cybern. IHMSC 2016*, vol. 2, pp. 551–556, 2016.

[104] P. Cui, W-s. Yan, X-x Guo, “Path-planning for Underwater Docking Based on Modified Artificial Potential Field”, *International Conference on Advanced Robotics and Mechatronics (ICARM)*, 2016.

[105] E. W. Dijkstra, “A note on two problems in connexion with graphs,” *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.

[106] H. Reddy, "Pathfinding—Dijkstra’s and A* Algorithm’s." *International Journal in IT and Engineering* (2013): 1-15.

[107] P.E. Hart, N. J. Nilsson, B. Raphael, “A Formal Basis for the Heuristic Determination of Minimum Cost Paths”, *IEEE Transactions on Systems Science and Cybernetics*, Volume: 4, Issue: 2, July 1968.

[108] X. Cui and H. Shi, “A*-based pathfinding in modern computer games,” *Int. J. Comput. Sci.* vol. 11, no. 1, pp. 125–130, 2011.

[109] H. Pan, C. Guo, and Z. Wang, “Research for path-planning based on improved astart algorithm,” *ICCSS 2017 - 2017 Int. Conf. Information, Cybern. Comput. Soc. Syst.*, pp. 225–230, 2017.

[110] S. Garrido and L. Moreno, “Mobile Robot Path Planning using Voronoi Diagram and Fast Marching,” *International Journal of Robotics and Automation (IJRA)*, Volume (2 , Issue (1), 2011.

[111] M. Elbanhawi and M. Simic, “Sampling-based robot motion planning: A review,” *IEEE Access*, vol. 2, pp. 56–77, 2014.

[112] Rami Al-Hmouz, Tauseef Gulrez, and Adel Al-Jumaily, “Probabilistic Road Maps with Obstacle Avoidance in Cluttered Dynamic Environment,” *Proceedings of Intelligent Sensors, Sensor Networks and Information*, pp. 241–246, 2005.

[113] Mitchell Melanie, “An Introduction to Genetic Algorithms”, The MIT Press, 1999.

[114] Y. Hu and S. X. Yang, “A Knowledge Based Genetic Algorithm for Path-planning of a Mobile Robot”, *IEEE International Conference on Robotics Automation*, 2004.

[115] H. Wang, Y. Yu, and Q. Yuan, “Application of Dijkstra algorithm in robot path-planning,” *2011 2nd Int. Conf. Mech. Autom. Control Eng. MACE 2011 - Proc*, pp. 1067–1069, 2011.

[116] G. Erinc and S. Carpin, “A genetic algorithm for nonholonomic motion planning,” *IEEE International Conference on Robotics and Automation*, 2007.

[117] M. Nieuwenhuisen, M. Schadler, and S. Behnke, “Predictive Potential Field-Based Collision Avoidance for Multicopters,” *ISPRS - Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.*, vol. XL-1/W2, no. September, pp. 293–298, 2013.

[118] S. LaValle and J. Kuffner, “Rapidly-exploring random trees: Progress and prospects,” *Work. Algorithmic Found. Robot.*, p. 293, 2001.

[119] J. Li, S. Liu, B. Zhang, and X. Zhao, “RRT-A∗ Motion planning algorithm for non-holonomic mobile robot,” *Proc. SICE Annu. Conf.*, pp. 1833–1838, 2014.

[120] Y.-H. Tseng, H.-C. Hung, “Extraction of building boundary lines from airborne LIDAR point clouds”, *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences Congress*, vol. XLI-B3, July 2016. 6

[121] P.R. Reddy, V. Amarnadh, and M. Bhaskar, “Evaluation of stopping criterion in contour tracing algorithms”. *International Journal of Computer Science and Information Technologies*, vol. 3, pp. 3888–3894, 2012.

[122] W. Shahab, H. Al-Otum, and F. Al-Ghoul, “A modified 2D chain code algorithm for object segmentation and contour tracing,” *Int. Arab J. Inf. Technol.*, vol. 6, no. 3, pp. 250–257, 2009.

[123] D. Kidner, M. Dorey, and D. Smith, "What’s the point? Interpolation and extrapolation with a regular grid DEM." *Fourth International Conference on GeoComputation, 1999*.

[124] A. Bücken and S. Thrun, “Integrating grid-based and topological maps for mobile robot navigation” *Artif. Intell.*, no. 3 , pp. 944–951,August 1996.

[125] F. Moments, R. Resolving, and A. Trigonometric, “Forces, centre of gravity, reactions and stability,” Royal Academy of Engineering.

[126] S. Kambhampati and L. S. Davis, “Multiresolution Path-planning for Mobile Robots,” *IEEE J. Robot. Autom.*, vol. 2, no. 3, pp. 135–145, 1986.

[127] P. Tsiotras, D. Jung, and E. Bakolas, “Multiresolution hierarchical path-planning for small UAVs using wavelet decompositions,” *J. Intell. Robot. Syst. Theory Appl.*, vol. 66, no. 4, pp. 505–522, 2012.

[128] O. B. P. M. Rodenberg, E. Verbree, and S. Zlatanova, “INDOOR A∗ PATHFINDING THROUGH AN OCTREE REPRESENTATION of A POINT CLOUD,” *ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci.*, vol. 4, no. 2W1, pp. 249–255, 2016.

[129] M. D. Phung, C. H. Quach, D. T. Chu, N. Q. Nguyen, T. H. Dinh, and Q. P. Ha, “Automatic interpretation of unordered point cloud data for UAV navigation in construction,” *14th Int. Conf. Control. Autom. Robot. Vision, ICARCV 2016*, pp. 13–15, 2017.

[130] D. Badouel, “An efficient ray-polygon intersection, in: Graphics Gems”, *Academic Press*, pp 390-393. 1990.

[131] J. Vince, “Geometry for computer graphics: Formulae, Examples and Proofs,” *Springer Science & Business Media*, 2006.

[132] Z. Xiangyang, C. Ke’an, and S. Jincai, “On the accuracy of the ray-tracing algorithms based on various sound receiver models,” *Appl. Acoust.*, vol. 64, no. 4, pp. 433–441, 2003.

[133] T. Paterson, "Malaysia Airlines: World's only three Abyss submarines readied for plane search". *Telegraph.co.uk*. 23 March 2014.

[134] H. Bray, "Bluefin robot joins search for missing Malaysian plane - The Boston Globe". *BostonGlobe.com*. Retrieved 2017-02-28.

[135] B. Neuberger, “An Exploration of Commercial Unmanned Aerial Vehicles (UAVs) Through Life Cycle Assessments,” Rochester Institution of Technology RIT Scholar Works, 2017.

[136] J. Vincent, “Welcome to the automated warehouse of the future,” *TheVerge,* 2018. [Online]. Available: https://www.theverge.com/2018/5/8/17331250/automated-warehouses-jobs-ocado-andover-amazon.

[137] M. d. Berg, O. Cheong, M. V. Kreveld. “Computational Geometry,” *Springer*, 3rd edn., chapter 9. 2008.

[138] L. Stutters, H. Liu, C. Tiltman, and D. J. Brown, “Navigation technologies for autonomous underwater vehicles,” *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.*, vol. 38, no. 4, pp. 581–589, 2008.

[139] P. Ridao, M. Carreras, D. Ribas, and R. Garcia, “Visual inspection of hydroelectric dams using an autonomous underwater vehicle,” *J. Field Robot.*, vol. 27, no. 6, pp. 759–778, 2010.

[140] Y. Peng, P. N. Green, “Acoustic side scan on an enclosed underwater environment”, *UK Robotics and Automation System (UKRAS) conference*, Loughborough, 2019.

[141] Y. Peng, and P. N. Green. "3D Digital Grid Map Initialization and Path Planning for Autonomous Robot Navigation." *Proceedings of the 2019 2nd International Conference on Service Robotics Technologies*. ACM, 2019.

[142] D. R. Yoerger, M. Jakuba, A. M. Bradley, and B. Bingham, “Techniques for deep sea near bottom survey using an autonomous underwater vehicle,” *Int. J. Robot. Res.*, vol. 26, no. 1, pp. 41–54, 2007.

[143] M. Walter, F. Hover, and J. Leonard, “SLAM for ship hull inspection using exactly sparse extended information filters,” *in Proc. IEEE Int. Conf. Robot. Autom.*, May 2008, pp. 1463–1470.

[144] R. Eustice, O. Pizarro, and H. Singh, “Visually augmented navigation for autonomous underwater vehicles,” *IEEE J. Ocean. Eng.*, vol. 33, no. 2, pp. 103–122, Apr. 2008.

[145] GeoData Institute, “Geophysical Techniques,” GeoData Institute, [Online]. Available at: http://www.dunwich.org.uk/methodology/geophysical_techniques/

[146] L-3 Communications SeaBeam Instruments, “Multibeam Sonar - Theory of Operation,” L-3 Communications SeaBeam Instruments, 2000.

[147] H. Cho, B. Kim, and S. C. Yu, “AUV-based underwater 3-D point cloud generation using acoustic lens-based multibeam sonar,” *IEEE J. Ocean. Eng.*, vol. 43, no. 4, pp. 856–872, 2018.

[148] Project Oceanography, “Lesson I. Oceanographic Research, Gathering Data in the Field Underwater robots : Autonomous Underwater Vehicles (AUVs),” *Ocean Technology*, pp. 5–10, 1999.

[149] J. Melo and A. Matos, “Survey on advances on terrain based navigation for autonomous underwater vehicles,” *Ocean Eng.*, vol. 139, no. May, pp. 250–264, 2017.

[150] Price list available at [Online]: https://stema-systems.nl/equipment/multibeam-r2sonic-2020/

[151] A. K. Brown and Y. Lu, “Performance test results of an integrated GPS/MEMS inertial navigation package,” *Proc. 17th Int. Tech. Meet. Satell. Div. Inst. Navig. (ION GNSS)*, pp. 825–832, September 2004.

[152] Price list available at [Online]: http://www.cactusnav.com/echopilot-standard-transducer-only-skin-fitting-p-5900.html?gclid=Cj0KCQjwxMjnBRCtARIsAGwWnBNTlLW0EwhPW9-6JrSOhx5_TUHfK4SNuBrtCeKWMyZowH0MhqSxvAwaAjctEALw_wcB

[153] Price list available at [Online]: https://rowlandsmarine.co.uk/garmin-gt20-tm-transom-mount-transducer/

[154] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning." *The international journal of robotics research*, vol. 30, no.7. pp. 846-894, 2011.

[155] E. Grilli, F. Menna, and F. Remondino, “A review of point clouds segmentation and classification algorithms,” *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci. - ISPRS Arch.*, vol. 42, no. 2W3, pp. 339–344, 2017.

[156] S. Liao, “DJI drones helped track and stop the Notre Dame fire,” *The Verge*, 2019. [Online]. Available at: https://www.theverge.com/2019/4/16/18410723/notre-dame-fire-dji-drones-tracking-stopped-thermal-cameras.

[157] L. Niu and G. Zhuo, “An improved real 3d A∗ algorithm for difficult pathfinding situation,” in Proceeding of *the International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 37, 2008.

[158] P. Lilienfeld, “Optical Detection of Particle Contamination on Surfaces : A Review Optical Detection of Particle Contamination on Surfaces : A Review,” *Aerosol Science and Technology*, vol. 5, Issue 2, pp 145-165, Oct 1985.

[159] F. Hidalgo and T. Braunl, “Review of underwater SLAM techniques,” *ICARA 2015 - Proc. 2015 6th Int. Conf. Autom. Robot. Appl.*, pp. 306–311, 2015.

[160] “Ultrasonic Module HC-SR04”, [Online]. *ElectronicWings, sensor &module,* Available at: https://www.electronicwings.com/sensors-modules/ultrasonic-module-hc-sr04

[161] D. Wheatley and M. Gillings, “Digital elevation models,” *Spat. Technol. Archaeol.*, pp. 107–124, 2010.

[162] M. Meissner, “Wave-based method for simulating small room acoustics,” Advances in Acoustics. Polish Acoustical Society, Warsaw Division, pp 425-436. September, 2016.

[163] A. M. Bueno, M. Galindo, and Á. L. León, “Sound behaviour of concrete churches. The church of santa cruz de oleiros,” *Arch. Acoust.*, vol. 43, no. 2, pp. 297–306, 2018.

[164] “Maxbotix XL-MaxSonar-AE4 Sonar Range Finder MB1340,” [Online], *Pololu Robotics & Electronics*. [Online]. Available: https://www.pololu.com/product/1664.

[165] A. Alpkocak and M. K. Sis, “Computing impulse response of room acoustics using the ray-tracing method in time domain,” *Arch. Acoust.*, vol. 35, no. 4, pp. 505–519, 2010.

[166] D. Majersky, “Removal and Solidification of the High Contaminated Sludge into the Aluminosilicate Matrix Sial During Decommissioning activities,” *AllDeco, Tech. Rep.*, 2012.

[167] A. Pompei, M. A. Sumbatyan, and N. F. Todorov, “Computer models in room acoustics: The ray tracing method and the auralization algorithms,” *Acoust. Phys.*, vol. 55, no. 6, pp. 821–831, 2009.

[168] I. Wald, “Realtime Ray Tracing and Interactive Global Illumination,” *Saarl. Univ. Ger.*, vol. 48, no. 4, p. 242, 2004.

[169] Price list available at [Online]: https://odeon.dk/pdf/prices/ODEON_PriceList2019.pdf

[170] Price list available at [Online]: http://ease.afmg.eu/index.php/prices-vendors.html

[171] B. de Greve, “Reflections and Refractions in Ray Tracing,” *Stanford computer graphic laboratory*. October, 2004.

[172] G. N. Kumar and M. Bangi, “An Extension to Winding Number and Point-in-Polygon Algorithm,” *IFAC-PapersOnLine*, vol. 51, no. 1, pp. 548–553, 2018.

[173] Figure available at [Online]: https://www.gettyimages.fi/detail/news-photo/nuclear-reprocessing-on-september-7th-1988-cooling-pond-for-news-photo/113420561

[174] F. Yan, Y. S. Liu, and J. Z. Xiao, “Path planning in complex 3D environments using a probabilistic roadmap method,” *Int. J. Autom. Comput.*, vol. 10, no. 6, pp. 525–533, 2013.

[175] Figure available at [Online]: https://uk.mathworks.com/help/robotics/ug/probabilistic-roadmaps-prm.html

[176] the U.S. Nuclear Regulatory Commission, “Backgrounder on Storage of Spent Nuclear Fuel”, *the U.S. Nuclear Regulatory Commission*, October 2016. [Online]. Available:

https://www.nrc.gov/reading-rm/doc-collections/fact-sheets/storage-spent-fuel.html

[177] Atomic and E. Agency, “IAEA Nuclear Energy Series Decommissioning of Pools in Nuclear Facilities.” *International Atomic Energy Agency*. 2015.

[178] "Multilateration (MLAT) Concept of use", *International Civil Aviation Organization*, 2007.

[179] Figure available at [Online]:
https://upload.wikimedia.org/wikipedia/commons/e/ed/Principle_of_SBES.jpg

[180] Atomic and E. Agency, “Survey of wet and dry spent fuel storage.” IAEA-TECDOC-1100, July, 1999.

[181] W. R. Dawes, “Overview of radiation hardening for semiconductor detectors,” *Nucl. Inst. Methods Phys. Res. A*, vol. 288, no. 1, pp. 54–61, 1990.

Appendix A: Area Data for Different Sampling Resolution

Table of the measured area and true area

Sampling resolution (a*a) m	Measured area (m^2)	True area (m^2)
0.125	9.5968	10.2145
0.25	8.8842	10.2145
0.5	7.5198	10.2145
1	6.792	10.2145

Table of area difference between the true top plane and the measured top plane

Sampling resolution (a*a) m	Area difference in percentage (%)
0.125	6.0473
0.25	13.023
0.5	26.3811
1	33.5063

Appendix B: Recorded Aerial Mapping Data

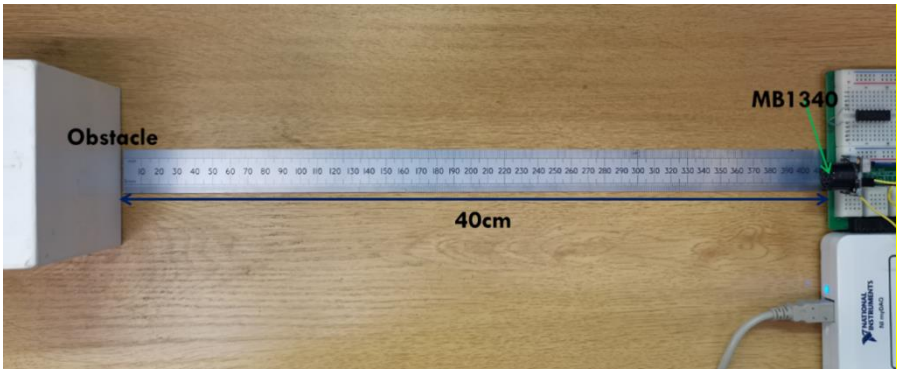
Recorded sampling position and height

Sample order	X	Y	Distance
1	20	11.5	10
2	20.5	11.5	10
3	21	11.5	10
4	21.5	11.5	10
5	22	11.5	NaN
6	22.5	11.5	NaN
7	23	11.5	NaN
8	23.5	11.5	NaN
9	24	11.5	NaN
10	24.5	11.5	NaN
11	25	11.5	NaN
12	25.5	11.5	5.906131
13	26	11.5	6.077141
14	26.5	11.5	6.248151
15	27	11.5	6.419161
16	27.5	11.5	6.590171
17	28	11.5	6.761181
18	28.5	11.5	6.932192
19	29	11.5	7.103202
20	29.5	11.5	7.274212
21	30	11.5	8.04657
22	30.5	11.5	8.04657
23	31	11.5	8.04657
24	31.5	11.5	8.04657
25	32	11.5	8.04657
26	32.5	11.5	10
27	33	11.5	10
28	33.5	11.5	10
29	34	11.5	10
30	34.5	11.5	10
31	35	11.5	10

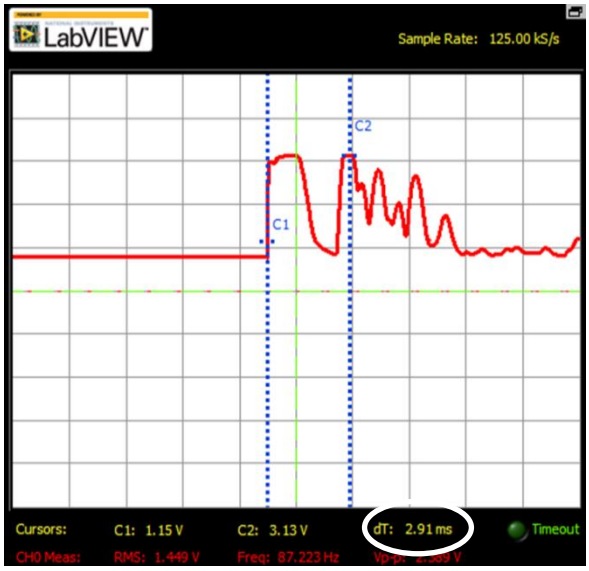
Estimated detected position and height

Sample order	X	Y	Height
1	20	11.5	0
2	20.5	11.5	0
3	21	11.5	0
4	21.5	11.5	0
5	22	11.5	NaN
6	22.5	11.5	NaN
7	23	11.5	NaN
8	23.5	11.5	NaN
9	24	11.5	NaN
10	24.5	11.5	NaN
11	25	11.5	NaN
12	23.49025	11.5	4.478243
13	23.93176	11.5	4.317546
14	24.37327	11.5	4.156849
15	24.81478	11.5	3.996152
16	25.25629	11.5	3.835455
17	25.6978	11.5	3.674759
18	26.13931	11.5	3.514061
19	26.58082	11.5	3.353364
20	27.02233	11.5	3.192667
21	30	11.5	1.98343
22	30.5	11.5	1.98343
23	31	11.5	1.98343
24	31.5	11.5	1.98343
25	32	11.5	1.98343
26	32.5	11.5	0
27	33	11.5	0
28	33.5	11.5	0
29	34	11.5	0
30	34.5	11.5	0
31	35	11.5	0

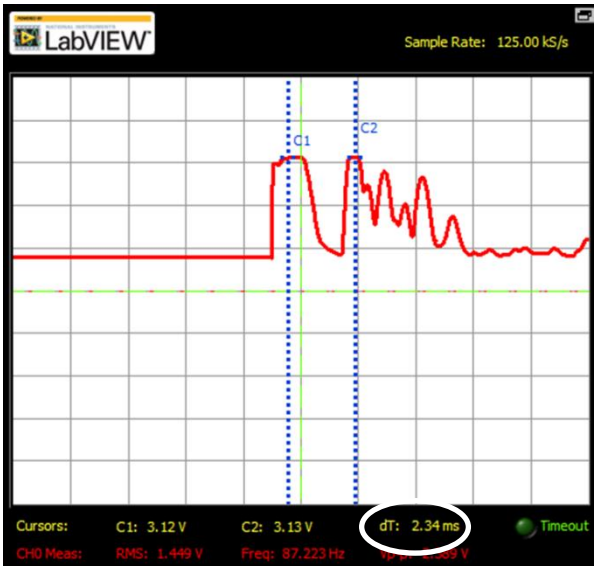
Appendix C: MB1340 Distance Measurement Tests



The distance tests are configured as the above figure. The MB1340 range finder is 40cm away from the obstacle. The measurement of TOF from the source burst's rising edge and the middle of the source burst's waveform are shown below, respectively.



Measuring from the rising edge



Measuring from the middle of the waveform

The TOF of measurement of the source burst's rising edge is 2.91 ms, so the calculated distance is 49.47 cm; The TOF of the middle of the source burst's waveform is 2.34 ms, so the calculated distance is 39.78cm. Obviously, measuring from the middle of the source burst's waveform has a more accurate result.

Appendix D: Boundary points and vertices

Boundary points =

{23.7735 11.0000 2.5566
24.2588 11.5000 2.6422
24.7442 12.0000 2.7277
25.2296 12.0000 2.8133
25.7149 11.5000 2.8988
26.2003 11.0000 2.9843
26.6857 10.5000 3.0699
26.2003 10.0000 3.0699
25.7149 9.5000 3.0699
25.2296 9.0000 3.0699
24.7442 9.0000 2.7277
24.2588 9.5000 2.6422
23.7735 10.0000 2.5566
23.2881 10.5000 2.4711

Boundary points of the first object

23.7735 15.5000 2.5566
23.7735 16.0000 2.5566
24.2588 16.5000 2.6422
24.2588 17.0000 2.6422
24.7442 17.0000 2.7277
25.2296 16.5000 2.8133
25.7149 16.5000 2.8988
26.2003 16.0000 2.9843
26.6857 16.0000 3.0699
26.2003 15.5000 3.0699
26.2003 15.0000 2.9843
25.7149 14.5000 2.9843
25.7149 14.0000 2.8988
25.2296 14.0000 2.8133
24.7442 14.5000 2.7277
24.2588 14.5000 2.6422
23.7735 14.5000 2.5566
23.2881 15.0000 2.4711

Boundary points of the second object

29.2881 13.5000 2.4711
29.2881 14.0000 2.4711
29.7735 14.5000 2.5566
30.2588 15.0000 2.6422
30.7442 15.0000 2.7277
31.2296 15.0000 2.8133
31.7149 15.0000 2.8988
32.2003 14.5000 2.9843
32.2003 14.0000 2.9843
32.6857 13.5000 3.0699
32.2003 13.0000 3.0699
32.2003 12.5000 2.9843
31.7149 12.0000 2.9843
31.2296 12.0000 2.8133
30.7442 12.0000 2.7277
30.2588 12.0000 2.6422
29.7735 12.5000 2.5566
29.2881 13.0000 2.4711}

Boundary points of the third object

Vertices points=

[23.2881 10.5000 2.4711
26.6857 10.5000 3.0699
25.2296 9.0000 3.0699
24.7442 12.0000 2.7277

Vertices of the first object

23.2881 15.0000 2.4711
26.6857 16.0000 3.0699
25.7149 14.0000 2.8988
24.2588 17.0000 2.6422

Vertices of the second object

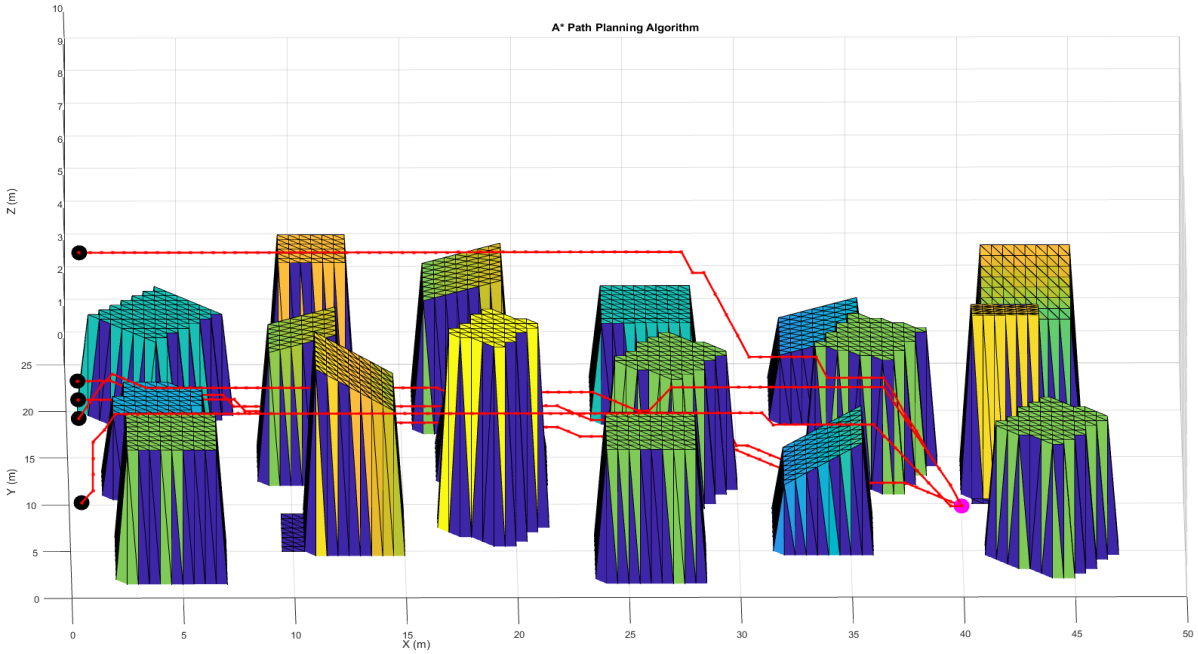
29.2881 13.5000 2.4711
32.6857 13.5000 3.0699
31.7149 12.0000 2.9843
30.2588 15.0000 2.6422]

Vertices of the third object

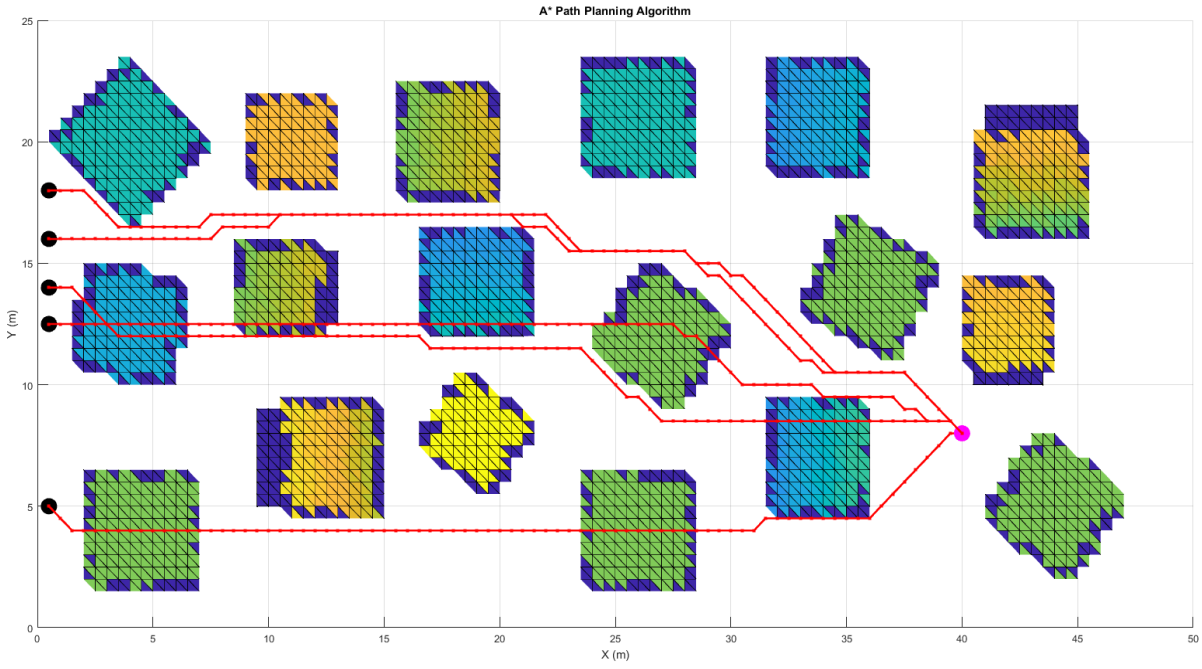
Based on the vertices points of each object's top plane, it is possible to calculate the plane equation. The plane equation can be used to calculate the height of the points whose position lie on the survey grid and the plane. The new points replace the original point one by one hence create the new point cloud. After obtaining the new plane point cloud, use the perpendicular height interpolation to fill the gap between obstacles' top plane point set and the ground point set. This generates the obstacle by a group of points under certain restrictive assumptions about the shape of the objects.

Appendix E: A* Path-Planning on Occupancy Grid Map

More test case with different start

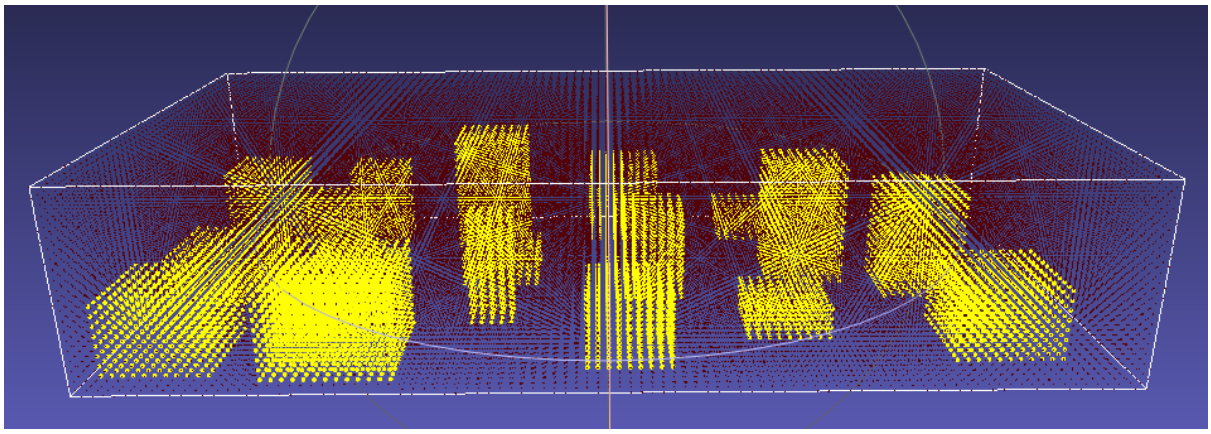


3D view of A* path

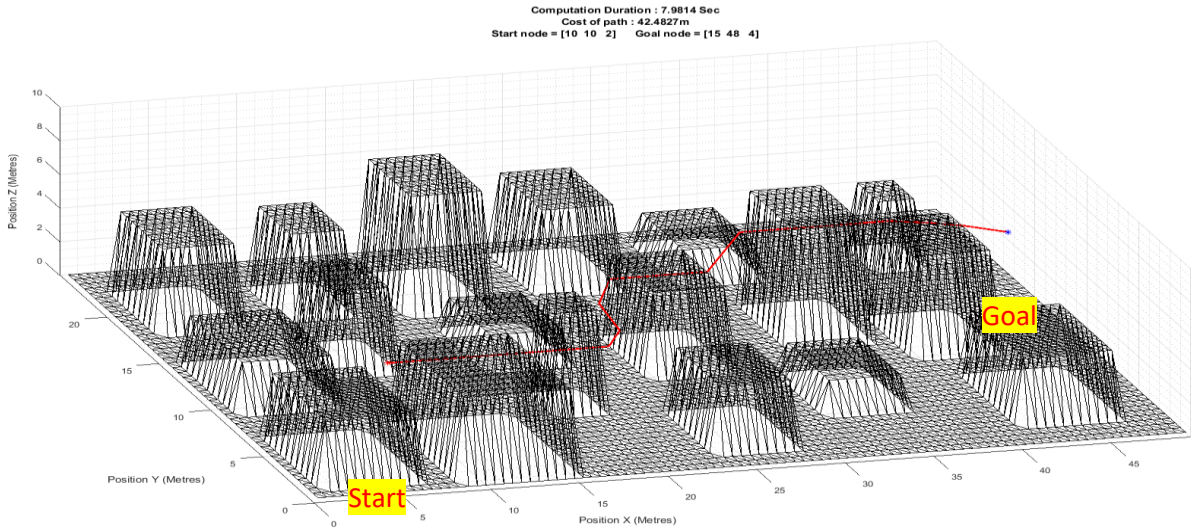


Plan view of A* path

More test case



Occupancy grid map

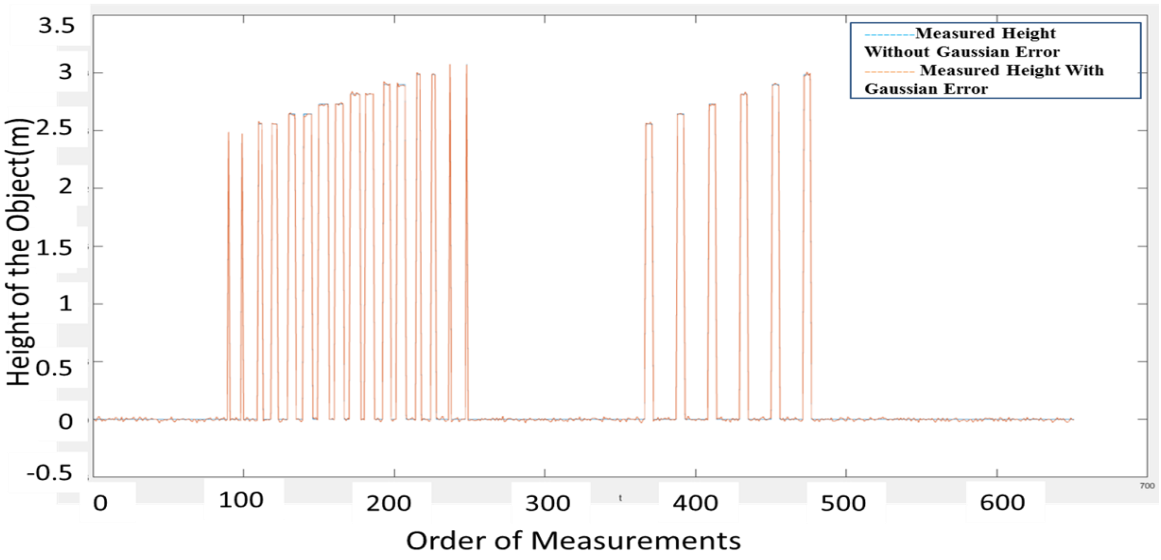


A* path on the occupancy grid map

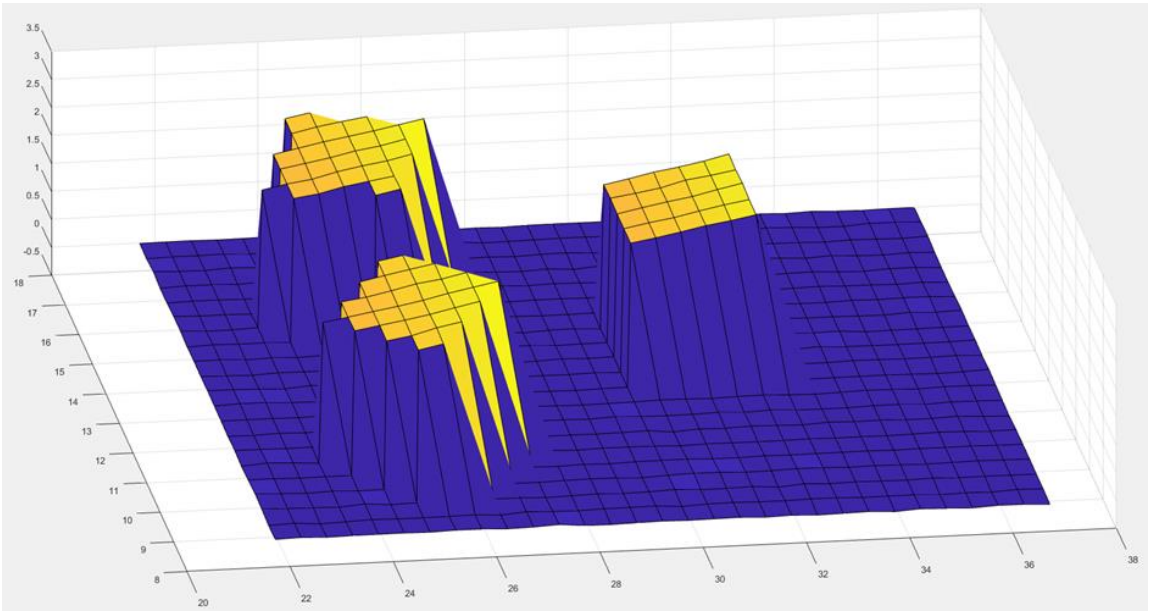
Appendix F: Mapping Result with Gaussian Error

The proposed object has an inclination angle of 10 degrees. An aerial mapping will provide a topological point cloud data of this object. In order to calculate the inclination angle of this object, we plan to find at least 3 vertices of the object from the point cloud and calculate the normal vector, thus calculate the inclination angle. Calculated inclined angle without Gaussian error is 9.9956 degree.

Test case 1: Inclination angle 10 degrees, Gaussian error 1%



Measured height with and without 1% Gaussian error

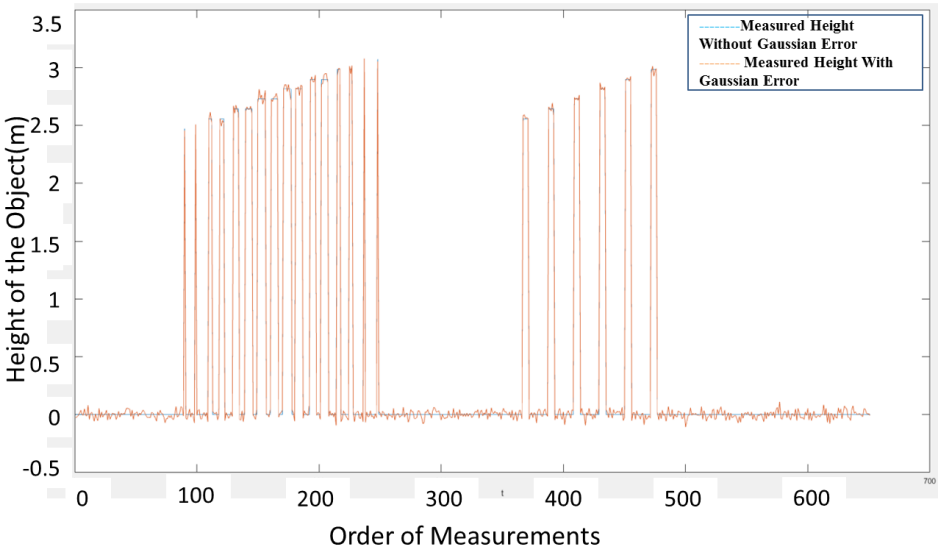


Surface plot of Gaussian error 1%

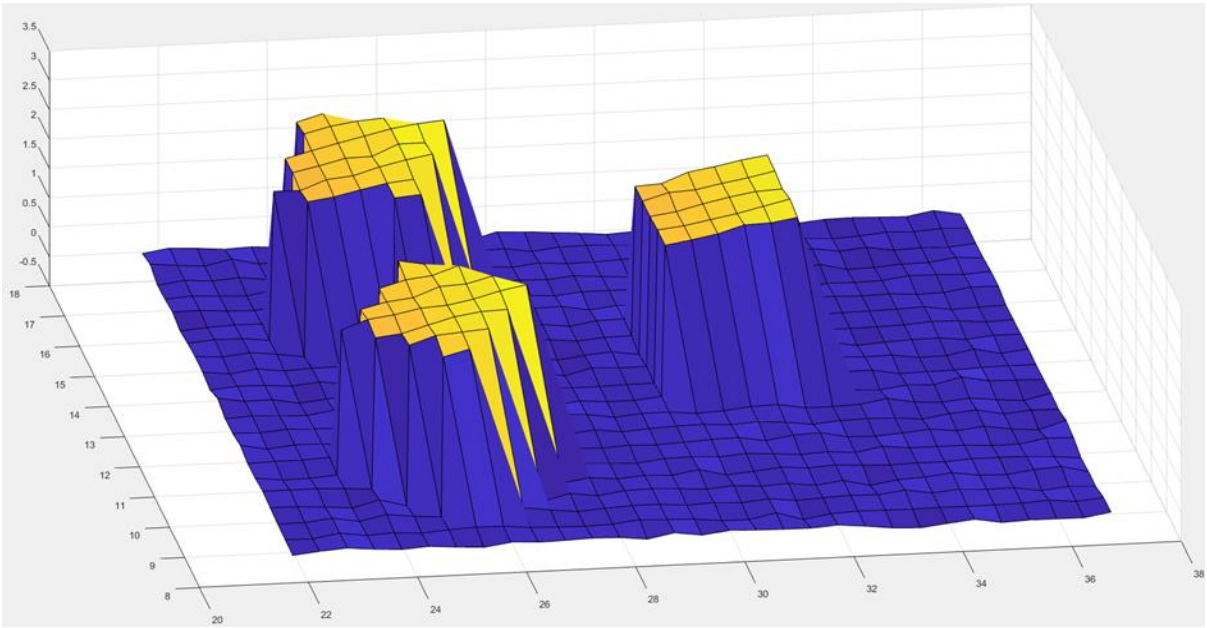
Table 1: Inclination angle 10 degrees, Gaussian error 1%

	Test 1	Test 2	Test 3	Test 4	Test 5
Normal vector	[0.8563,0,-4.8536]	[0.8444,0,-4.8536]	[0.8335,0,-4.8536]	[0.8359,0,-4.8536]	[0.8591,0,-4.8536]
Calculated inclination angle (degree)	10.0051	9.8694	9.7437	9.7713	10. 0379
Average angle (degree)	9.93982				
Standard deviation (degree)	0.14637				

Test case 2: Inclination angle 10 degrees, measurement Gaussian error 5%



Measured height with and without 5% Gaussian error

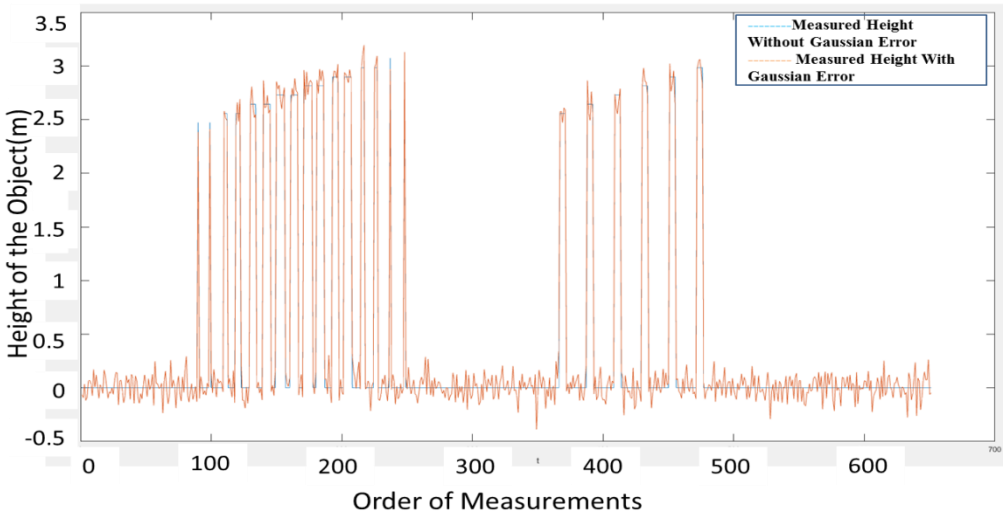


Surface plot of Gaussian error 5%

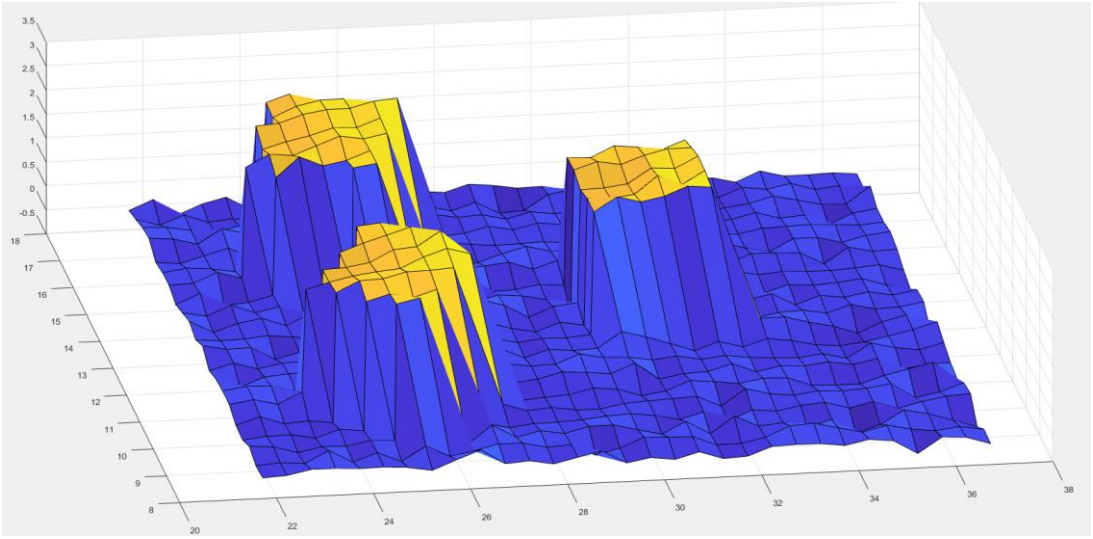
Table 2: Inclination angle 10 degrees, Gaussian error 5%

	Test 1	Test 2	Test 3	Test 4	Test 5
Normal vector	[0.9658,0, -4.8536]	[0.8443,0 -4.8536]	[0.8662,0 -4.8536]	[0.9135,0 -4.8536]	[0.8827, 0 -4.8536]
Calculated inclination angle (degree)	11.2536	9.8674	10.1189	10.6589	10.3079
Average angle (degree)	10.4414				
Standard deviation (degree)	0.5381				

Test case 3: Inclination angle 10 degrees, measurement Gaussian error 10%



Measured height with and without 10% Gaussian error



Surface plot of Gaussian error 5%

Table 3: Inclination angle 10 degrees, Gaussian error 10%

	Test 1	Test 2	Test 3	Test 4	Test 5
Normal vector	[0.7857, 0 -4.8536]	[0.8815, 0 -4.8536]	[0.7678, 0 -4.8536]	[0.9289, 0 -4.8536]	[0.6712, 0 -4.8536]
Calculated inclination angle (degree)	9.1956	10.2931	8.9887	10.8342	7.8729
Average angle (degree)	9.4369				
Standard deviation (degree)	1.161				

Appendix G: Calculation of δ

For Figure 8.20, the gradient of A2B2:

$$k = \frac{(y_b + 2\sigma) - (y_a - 2\sigma)}{x_b - x_a} = \frac{y_b - y_a + 4\sigma}{x_b - x_a}$$

A2C2 is perpendicular to A2B2, so the gradient of A2C2:

$$k' = -\frac{x_b - x_a}{y_b - y_a + 4\sigma}$$

Equation of A2C2 is

$$y = k'x + m$$

$$m = y_a - 2\sigma + \frac{x_b - x_a}{y_b - y_a + 4\sigma} * x_a$$

$$y = -\frac{x_b - x_a}{y_b - y_a + 4\sigma}x + y_a - 2\sigma + \frac{x_b - x_a}{y_b - y_a + 4\sigma} * x_a$$

C2 on the ground so $C2(x_{c2}, 0)$

$$x_{c2} = (y_a - 2\sigma) * \frac{y_b - y_a + 4\sigma}{x_b - x_a} + x_a$$

The gradient of AB:

$$k1 = \frac{y_b - y_a}{x_b - x_a}$$

AC is perpendicular to AB, so the gradient of AC:

$$k1' = -\frac{x_b - x_a}{y_b - y_a}$$

Equation of AC is

$$y = k1'x + m1$$

$$m1 = y_a + \frac{x_b - x_a}{y_b - y_a} * x_a$$

$$y = -\frac{x_b - x_a}{y_b - y_a}x + y_a + \frac{x_b - x_a}{y_b - y_a} * x_a$$

C on the ground so $C(x_c, 0)$

$$x_c = y_a * \frac{y_b - y_a}{x_b - x_a} + x_a$$

$$\delta = x_{c2} - x_c = \frac{4\sigma(y_a - 2\sigma)}{x_b - x_a} - 2\sigma * \frac{y_b - y_a}{x_b - x_a}$$

ε can be calculated by the same procedures above

$$\varepsilon = x_{D2} - x_D = \frac{4\sigma(y_b - 2\sigma)}{x_b - x_a} - 2\sigma * \frac{y_b - y_a}{x_b - x_a}$$