Optimisation and Extension of Octree Based Occupancy Mapping Using Stereo Cameras

Miao, Yu

Award date:
2022

Awarding institution:
University of Bath

Optimisation and Extension of Octree Based Occupancy Mapping Using Stereo Cameras

submitted by

Yu Miao

for the degree of Doctor of Philosophy

of the

University of Bath

Department of Mechanical Engineering

September 2021

Abstract

With the advent of 3D sensors, point clouds are becoming increasingly popular in robotic perception. Using point clouds, mapping algorithms can generate 3D environment models. A widely used one is OctoMap, informing a robotic platform which parts of the environment are free and which are not using the octree structure.

The generation of point clouds from sensor data and the operation of OctoMap are governed by different parameters, the correct selection of which significantly affects the process and the quality of the final map. Unfortunately, research in the process to identify the parameter set to achieve best occupancy mapping performance remains limited. The current work aims to fill this gap with a two-step principled methodology that first identifies the most significant parameters by conducting Neighbourhood Component Analysis on all parameters and then optimise those using grid search with the area under the Receiver Operating Characteristic curve.

In addition, the map update policy in OctoMap has limitations. All the nodes containing endpoints will be assigned with the same probability regardless of the points being noise and the probability of one such node can only be increased with a single measurement. Moreover, potentially occupied nodes with points inside but traversed by rays cast from the sensor to endpoints will be marked as free. To overcome these limitations, the current work presents a mapping method using the context of neighbouring points to update the nodes containing points, with the occupancy information of a point represented by the average distance from the point to its k-Nearest Neighbours (k-NN). A relationship between the distance and the change in probability is defined with the Cumulative Density Function of average distances, potentially decreasing the probability of a node despite points being present inside.

This study is conducted on 20 data sets collected with specially designed targets in two outdoor environments, providing precise ground truths for evaluation purposes. Point clouds are created by applying StereoSGBM on the images from a stereo camera and poses are produced by ORB-SLAM. Using the proposed method, a clear indication can be seen that the mapping parameters are more important than other parameters. Through grid search, improvement in occupancy map quality can be achieved over default parameters. Moreover, the k-NN mapping method can also achieve improvement over the performance of OctoMap.

Acknowledgements

I would like to express my sincere gratitude to Dr Ioannis Georgilas and Dr Alan Hunter for their guidance, patience and valuable advice throughout the entire PhD. Thanks to Prof Andrew Plummer for his help and support.

I would also like to thank Shengjun Xu for his ideas and help throughout various stages of my PhD.

I am grateful to my parents for their support and understanding. Many thanks to my friends for their support and encouragement.

Contents

List of Figures

List of Tables

Notation

Roman Symbols

$\mathbf{K}$	Camera matrix
$\mathbf{n}$	Normal vector of the spherical surface or planes
$\mathbf{p}$	Image patch
$\mathbf{R}_\gamma$	Rotation matrix for orientation of the image patch
$\mathbf{S}$	Matrix defined for binary tests
$\mathbf{S}_\gamma$	Steered version of $\mathbf{S}$
$\mathbf{x}$, $\mathbf{y}$	Points randomly picked up by a Gaussian distribution around image patch centre
${}^{b}\bar{\mathbf{P}}$	Homogeneous coordinates in the cube frame
${}^{b}_{c}\mathbf{T}$	Transformation matrix from the camera frame to the cube frame
${}^{s}_{c}\mathbf{T}$	Transformation matrix from the camera frame to frame $O_s - x_s y_s z_s$
${}^{b}_{r}\mathbf{T}$	Transformation matrix from frame $O_r - x_r y_r z_r$ to the cube frame
${}^{c}\mathbf{P}$	Coordinates in the camera frame
${}^{c}\tilde{\mathbf{P}}$	Normalised coordinates in the camera frame
${}^{c}_{w}\mathbf{R}$	Rotation matrix from the world frame to the camera frame
${}^{c}_{w}\mathbf{T}$	Transformation matrix from the world frame to the camera frame
${}^{c}_{w}\mathbf{t}$	Translation matrix from the world frame to the camera frame
${}^{p}\bar{\mathbf{P}}$	Homogeneous coordinates in the pixel frame

${}^{r}_{s}\mathbf{T}$	Transformation matrix from frame $O_s - x_s y_s z_s$ to frame $O_r - x_r y_r z_r$
${}^{s}_{c}\mathbf{R}$	Rotation matrix from the camera frame to frame $O_s - x_s y_s z_s$
${}^{s}_{c}\mathbf{t}$	Translation matrix from the camera frame to frame $O_s - x_s y_s z_s$
${}^{w}\bar{\mathbf{P}}$	Homogeneous coordinates in the world frame
$\tilde{x}$, $\tilde{y}$	Normalised image coordinates
B_s	Block size for matching two images
C	Truncation value for the pre-filtered image pixels
c'_x, c'_y	Horizontal and vertical coordinates of the principal point of the right camera
c_x, c_y	Horizontal and vertical coordinates of the principal point of the single camera or left camera
d	Disparity in metres
d_m	Maximum allowed difference in the left-right disparity check
d_n	Difference between maximum disparity and minimum disparity
d_p	Disparity in pixels
d_v	Maximum variation in disparity
d_w	Maximum size of regions to consider noise speckles
$d_{\min}$	Minimum possible disparity value
E	Intensity centroid
f	Focal length in metres
f_p	Focal length in pixels
f_x, f_y	Horizontal and vertical focal lengths in pixels
k	Number of neighbours
$l_{\max}$, $l_{\min}$	Upper and lower bounds in log-odds
l_{in}, l_{out}	Log-odds values to update free nodes within and outside the range s_c

l_{occ}, l_{free}	Log-odds values to update occupied and free nodes
M	Moment of image patches
m	Occupancy map
m'	Occupancy map to store log-odds values
m'_{out}	Nodes traversed by measurement z_{out} but not traversed by z_{in}
m_{in}	Nodes traversed by measurement z_{in}
m_{out}	Nodes traversed by measurement z_{out}
m_r	Ground truth
m'_r	Copy of ground truth
$mode$	Option to run the full-scale two-pass dynamic programming algorithm
N_0	Number of points in the original point cloud
N_b	Number of the combinations of ORB parameters
N_d	Number of data sets
N_e	Expected number of points at distance s_d
N_f	Number of features in each image
N_k	Number of the combinations of k-NN mapping parameters
N_l	Number of levels in the scale pyramid
N_o	Number of the combinations of OctoMap mapping parameters
N_p	Number of the combinations of point cloud generation parameters
N_s	Number of points in the down sampled point cloud
N_t	Number of the combinations of parameters for each data set
N_{FN}	Number of false negatives
N_{FP}	Number of false positives
N_{TN}	Number of true negatives

N_{TP}	Number of true positives
O	Geometric centre of the image patch
$O_c' - x_c'y_c'z_c'$	Right camera frame
$O_i' - x_i'y_i'$	Right image plane frame
$O_p' - u'v'$	Right pixel frame
$O_b - x_by_bz_b$	Cube frame
$O_c - x_cy_cz_c$	Camera frame or left camera frame
$O_i - x_iy_i$	Image plane frame
$O_p - uv$	Pixel frame or left pixel frame
$O_r - x_ry_rz_r$	Reference frame created based on the cube
$O_s - x_sy_sz_s$	Reference frame created based on the frame $O_r - x_ry_rz_r$ and plane κ_2
$O_w - x_wy_wz_w$	World frame
P	Scene point in 3D space
p_m'	Probability for a "miss" outside range s_c
P_1, P_2	Parameters for smooth control
p_h	Probability for a "hit"
P_l, P_r	Projections of point P in left and right image plane frames
p_m	Probability for a "miss"
p_t	Threshold for occupancy
p_u, p_l	Upper and lower bounds on the probability of a point
$p_{\max}$, $p_{\min}$	Upper and lower clamping thresholds
R	Correlation coefficient
r	Distance to the optical centre
r_u	Margin in percentage

S	Point set
s	Average of distance from a point to its k-NN
s_b	Length of the baseline
s_c	Range for how long individual beams are inserted
s_d	Distance between two planes
s_f	Scale factor between levels in each image
s_l	Side length of the cube or the resolution of the map
t	Time
T, T'	Values of parameters
t_i	Initial threshold to extract FAST corners
T_s, T'_s	Steps of parameters
$T_{\max}$, $T_{\min}$	Maximum and minimum values of a parameter
$t_{\min}$	Lower threshold to extract FAST corners
u_d, v_d	Unrectified pixel coordinates
$u_{\max}$, $v_{\max}$	Maximum values in axes u and v
V	Voronoi diagram
x_t	Pose at time t
$x_{1:t}$	Poses to time t
z_t	Observation at time t
$z_{1:t}$	Observations to time t
z_{in}, z_{out}	Measurements within and outside range s_c

Greek Symbols

α	Rotation angle about z_c axis
χ	Threshold value
δ_x^r, δ_y^r	Radial distortions

δ_x^t, δ_y^t	Tangential distortions
η_1, η_2	Tangential distortion coefficients
γ	Orientation of the image patch
κ_1 , κ_2	Planes contains the cube centre and the camera centre
μ	Average
ω_x, ω_y	Horizontal and vertical scale factors
ϕ	Azimuthal angle
σ	Standard deviation
θ	Polar angle
ζ_1, ζ_2, ...	Radial distortion coefficients

Acronyms

AUC	Area Under the Curve
BA	Bundle Adjustment
BRIEF	Binary Robust Independent Elementary Features
CDF	Cumulative Density Function
FAST	Features from Accelerated Segment Test
FCAUC	FDR-Controlled AUC
FDR	False Discovery Rate
FN	False Negative
FOV	Field of View
FP	False Positive
FPR	False Positive Rate
GEV	Generalised Extreme Value
GP	Gaussian Process
k-NN	k-Nearest Neighbours

KDE	Kernel Density Estimation
KLD	Kullback-Leibler Divergence
LIDAR	Light Detection and Ranging
MLR	Multiple Linear Regression
MSE	Mean Squared Error
NCA	Neighbourhood Component Analysis
ORB	Oriented FAST and Rotated BRIEF
PCC	Person Correlation Coefficient
PCL	Point Cloud Library
RNN	Recurrent Neural Network
ROC	Receiver Operating Characteristic
ROCS	ROC Surface
ROV	Remotely Operated Underwater Vehicle
SIFT	Scale-Invariant Feature Transform
SLAM	Simultaneous Localisation and Mapping
SNN	Shallow Neural Network
SURF	Speeded-Up Robust Features
TDR	True Discovery Rate
TN	True Negative
ToF	Time-of-Flight
TP	True Positive
TPR	True Positive Rate
UAV	Unmanned Aerial Vehicles
UGV	Unmanned Ground Vehicle
USV	Unmanned Surface Vehicle

VUS	Volume Under the Surface

Chapter 1

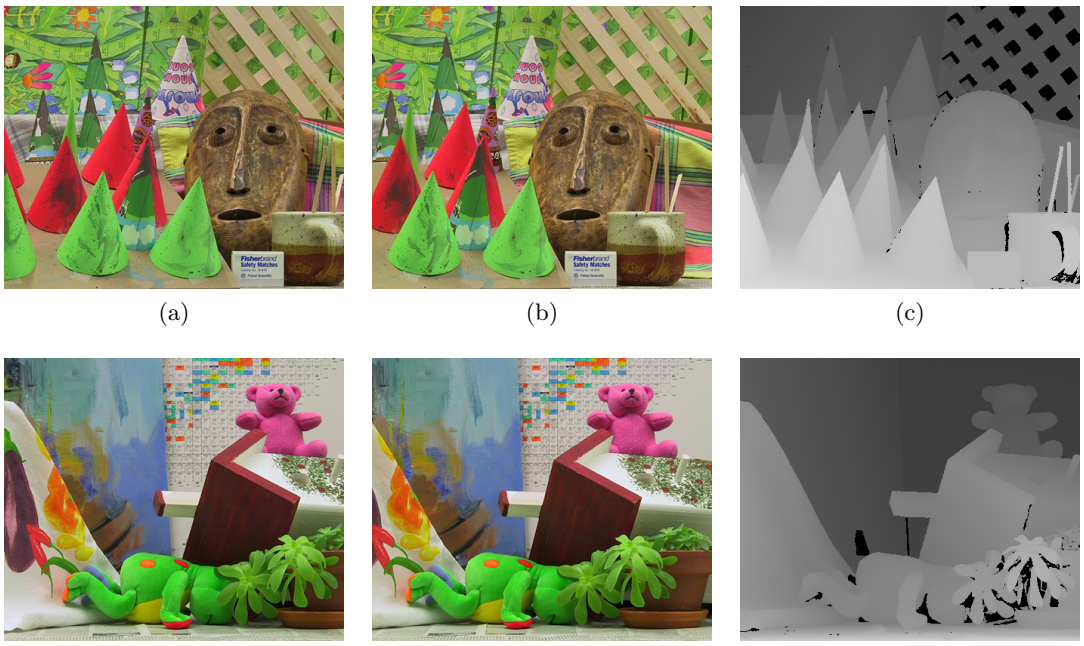
Introduction

In robotics, occupancy maps have a wide range of applications, including spatial representation of the real world [1], navigation [2, 3], motion planning [4, 5], and autonomous driving [6]. In most applications, occupancy maps are generated from point clouds. Over the past few years, sensors like LIDAR can produce high-quality point clouds to represent the 3D world [7, 8]. However, such sensors are normally expensive, which has restricted their applications. Recently, point clouds are becoming increasingly popular in the research community as cheaper solutions become available. A RGB-D camera [9] can simultaneously produce both colour and depth images, the latter of which can be used for point cloud reconstruction. Similarly, point clouds can be reconstructed from a stereo camera [10] with the disparity map derived by left and right images. One popular occupancy mapping algorithm generating occupancy maps from point clouds is OctoMap [11], using the octree structure [12] and its cubic nodes to model the 3D world. Given the usefulness and the wide adoption of OctoMap as a mapping algorithm, a further in-depth study of the octree based occupancy mapping is necessary.

1.1 Introduction to OctoMap

Point clouds are required to generate maps using OctoMap. Cheaper solutions to generate point clouds include RGB-D cameras and stereo cameras. A RGB-D camera is using structured light or time-of-flight (ToF) to acquire the depth information. However, this type of sensor suffers from characteristic problems such as noise and ambiguity, and non-systematic errors, e.g., scattering and motion blur [13, 14]. To achieve satisfactory performance, the operation of a ToF sensor requires a very controlled light environment, which has restricted its applications in outdoor environments with com-

plex lighting conditions [15]. Compared with RGB-D cameras, a stereo camera can generate high-resolution disparity maps which can be used for measurements in both indoor and outdoor environments [14, 16]. Figure 1-1 shows an example of benchmarking stereo images with disparity maps as ground truths. Using left and right images algorithms such as StrereoSGBM in OpenCV [17] can produce disparity maps, from which point clouds can be reconstructed with a binocular camera model. The procedure of 3D reconstruction of point clouds will be introduced in Chapter 2.



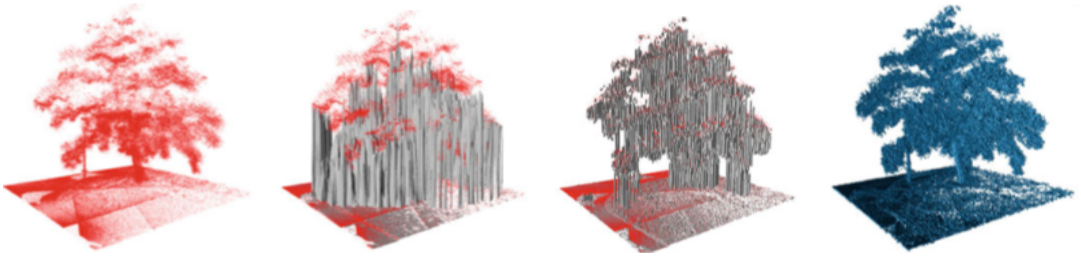
(a) (b) (c)

(d) (e) (f)

Figure 1-1: Benchmarking images [18]. (a-c) Cones: left image, right image and ground truth. (d-f) Teddy: left image, right image and ground truth.

With point clouds, mapping algorithms can generate occupancy maps representing the real world. Compared with other common mapping approaches, OctoMap is an efficient and flexible framework for 3D environment mapping [11]. Figure 1-2 shows the representation of a tree scanned with a laser range sensor using common mapping approaches elevation maps [19], multi-level surface maps [20] and OctoMap. A clear indication can be seen that the volumetric representation of OctoMap is better than the other mapping approaches. More details are preserved by OctoMap, and occupied and free space are consistent with the original laser measurements.

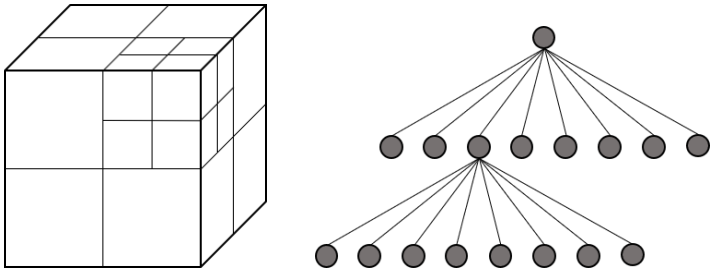
The octree [12] is implemented in the OctoMap to save memory and speed up the mapping process. An octree is an hierarchical data structure in which each internal



(a) (b) (c) (d)

Figure 1-2: 3D representation of a tree [11]. (a) Point cloud. (b) Elevation map. (c) Multi-level surface map. (d) OctoMap.

node has eight child nodes [12]. In OctoMap, 3D space can be represented by the cubic nodes in the octree. As shown in Figure 1-3, each node can be recursively subdivided into eight sub-nodes. The minimum size of the cube is equal to the resolution of an occupancy map. The recursive subdivision stops when a cubic cell reaches the minimum size. The nodes with the minimum dimension are called leaf nodes.



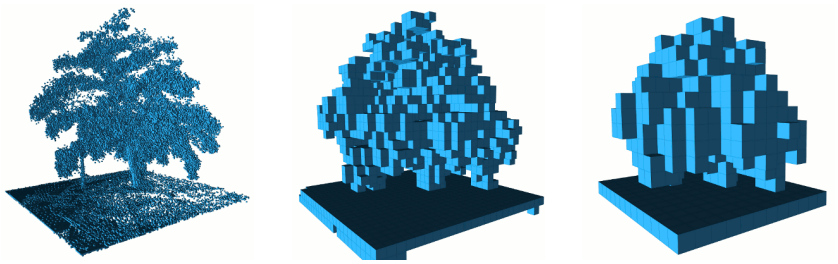
(a) (b)

Figure 1-3: Octree structure. (a) Recursive subdivision of a cube. (b) Octree.

OctoMap uses the octree and probabilistic estimation to model the 3D environment. A ray cast operation from the sensor to the endpoints in a point cloud will be performed to determine which nodes should be updated. The nodes containing endpoints will be updated with a high probability, while the nodes traversed by rays will be updated with a low probability. In an occupancy map, the state of a node can be occupied, free or unknown. If the space contained in a cubic volume has been explored, the node will be marked as either occupied or free depending on its corresponding probability and threshold. Otherwise, the node state will be marked as unknown. Due to the hierarchical structure of the octree, OctoMap has following benefits.

Multi-Resolution Queries An octree recursively subdivides a cubic cell into eight children. Conversely, eight child nodes can be pruned into one node. An occupancy

map can be queried at different levels. Figure 1-4 illustrates a map queried at different depths. The occupancy probability of a cubic volume is estimated at the leaf node level. The probability of a parent node can be acquired by recursive calculation from the probabilities of eight child nodes. Researchers have proposed approaches to get the probability of a node from its children, e.g., the average occupancy and the maximum occupancy [21].



(a) (b) (c)

Figure 1-4: Example of a map queried at different levels [11]. (a) Resolution 0.08 m. (b) Resolution 0.64 m. (c) Resolution 1.28 m.

Map Compression Since OctoMap uses the octree structure to represent 3D space, it is possible to compress a map to save memory. A clamping update policy proposed in [22] is implemented in OctoMap to set upper and lower bounds on the occupancy probability of a node. When the probability of a node reaches clamping thresholds, the state of this node will be considered as stable. If eight child nodes are all stable and their probabilities are equivalent to the same clamping threshold, then these nodes can be pruned into one node. When new measurements change the states of the children, the parent node will be redivided into eight nodes and the child nodes will be updated accordingly. If the user only focuses on the states of a node, i.e., occupied or free, then the values of probabilities are not important. In this case, the map can be further compressed with the maximum likelihood probabilities proposed in [23].

Octree Hierarchies OctoMap can display multiple submaps in one tree-structure with octree hierarchies. Figure 1-5 shows an example of an occupancy map with several objects at different resolutions in one tree. Background, table and objects are presented in yellow, magenta and cyan. The benefits of a hierarchical octree include independent maintenance and manipulation of submaps in terms of mapping parameters and movement, and preserving hierarchical dependencies among different submaps when one submap is moved.

Since OctoMap is memory-efficient and can quickly adapt to environmental changes, it

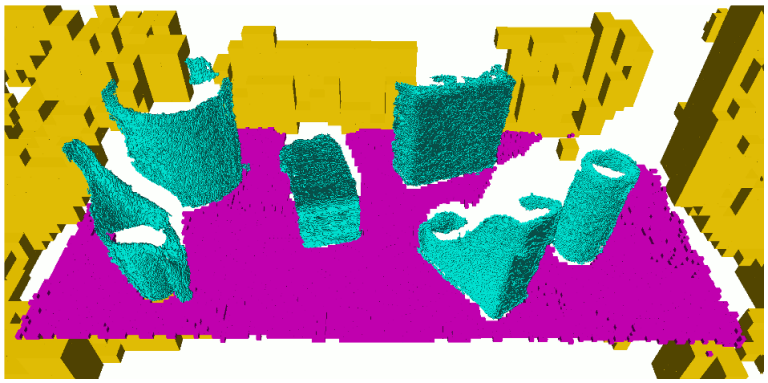


Figure 1-5: Example of a hierarchical octree model [11].

has a wide range of applications such as underwater inspection [24, 25], navigation [26] and autonomous driving [27].

In [28], an iterative path re-planning method based on OctoMap was introduced to adapt to the path in real time for the inspection mission of 3D underwater structures. OctoMap was used for maintaining a 3D map for quick access using sonar range measurements. Figure 1-6 shows an occupancy map with depth colour coded. The occupied nodes were used to reshape the nominal path which was then optimised to produce a smooth trajectory, based on which a path re-planning algorithm could be implemented. In [29], OctoMap was used to build the map of the inspection area. As shown in Figure 1-7, the underwater terrain was modelled by occupancy maps.

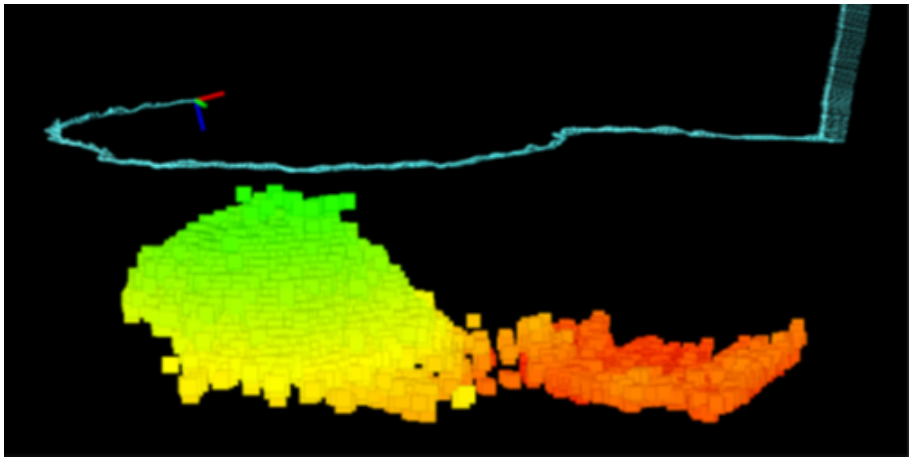


Figure 1-6: Example 3D occupancy grid map of an underwater environment obtained by Octomap during one of inspection trials [28].

OctoMap is also a popular mapping approach for navigation. A navigation method for uneven and unstructured indoor environments using OctoMap was proposed in [30].

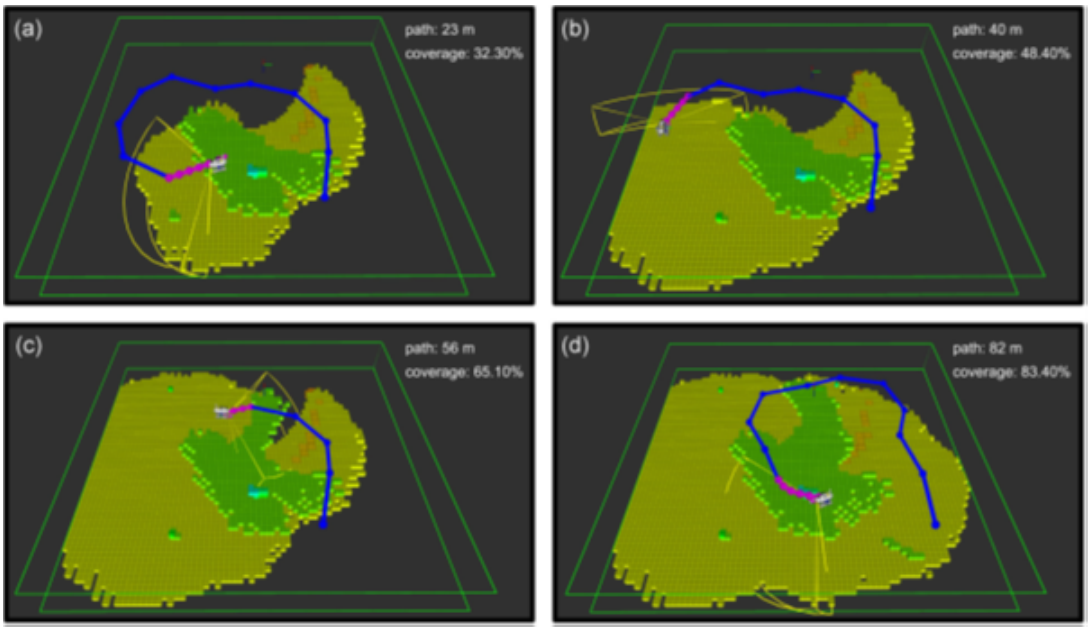
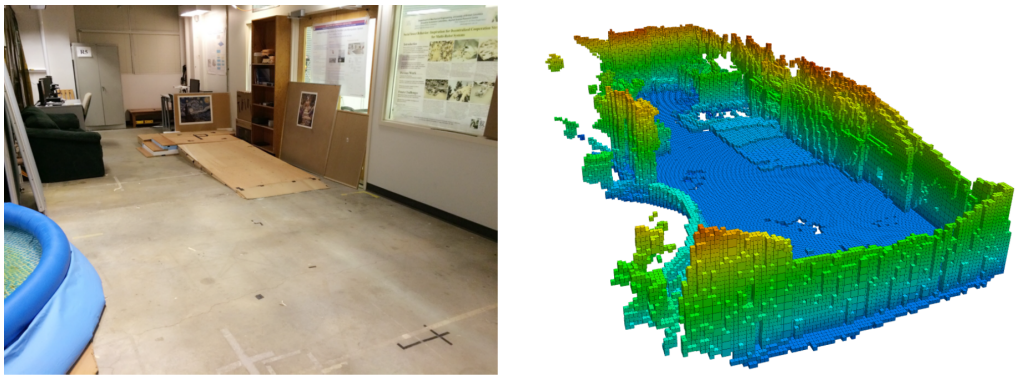


Figure 1-7: Four snapshots of an inspection survey [29]. Green polygons delimit the inspection area at depth 0m and the maximum depth. Yellow lines represent the field of view (FOV) of the vehicle. The blue line is the inspection path generated by the path planner. Purple spheres are the path created by the motion planner.

Figure 1-8 shows a wheeled robot navigating up the slope to the goal at the higher platform. The uneven environment was first modelled by OctoMap using the wheel odometry, 2D laser and RBD-D data. Then the map could be further used for indoor navigation. Figure 1-9 shows the representation of the real environment with OctoMap.

Figure 1-8: Navigation of the robot up the slope to the goal at the higher platform [30].

Robotic applications such as autonomous driving, navigation and localisation can be improved by dense semantic models [31]. As an important perception technology, se-



(a) (b)

Figure 1-9: 3D mapping of the real environment [30]. (a) Photo of the real environment. (b) 3D representation of the environment using OctoMap.

mantic segmentation is widely used in autonomous driving and other industries [32]. A semantic OctoMap mapping method based on CBAM-PSPNet was proposed in [32]. The images acquired by a RGB-D camera were processed by two threads. ORB-SLAM [33, 34] was running in the background to generate poses from images. CBAM-PSPNet was used for the semantic segmentation of the images. With segmentation information and the depth image, semantic point clouds could be constructed. Then the semantic point clouds were inserted into OctoMap to generate semantic occupancy maps. The maps generated by OctoMap and semantic OctoMap have been compared in Figure 5-4.

1.2 Motivation

The generation of point clouds from raw images derived by stereo cameras and the operation of OctoMap are governed by several parameters, the correct selection of which significantly affects the process and the quality of the final map. Although point cloud generation algorithms and mapping algorithms are normally initialised with their default parameters, there is no evidence to show these parameters are the optimal ones. To generate an occupancy map using OctoMap, the pose corresponding to each frame of point cloud is required. The poses corresponding to point clouds are not always obtained from sensors directly, but by implementing localisation algorithms, e.g., ORB-SLAM [33], on the raw images. In this case, parameters for producing poses are introduced. Considering occupancy mapping using point clouds has been widely used, the impact of the different parameters needs to be evaluated and well understood.

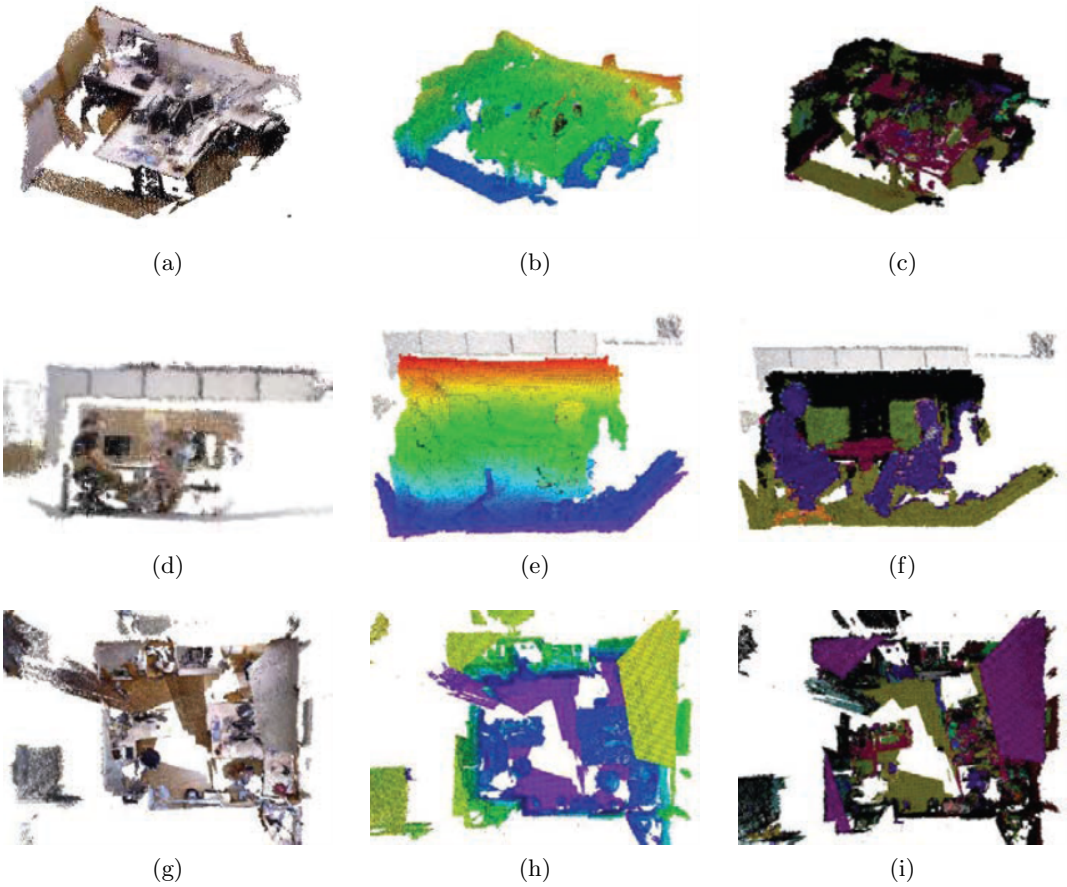


Figure 1-10: Comparison of mapping effects [32]. (a), (d) and (g) are the point clouds of three test scenes. (b), (e) and (h) are the occupancy maps derived by OctoMap. (c), (f) and (i) are the occupancy maps derived by semantic OctoMap.

OctoMap is effective in building occupancy maps using point clouds. As specified in the previous section, the ray cast operation determines the nodes to be updated and the probability of a node is updated accordingly depending on whether it contains endpoints or is traversed by rays. However, this update policy has limitations which may affect the mapping performance. All the nodes containing endpoints will be updated with the same probability even if the points inside are noise. The probability of one such node can only be increased. Moreover, a potentially occupied node with points inside but also traversed by rays will be assigned with a low probability, which may lead to this node being marked as free. Considering these facts, the update policy in OctoMap can be improved to achieve better mapping performance.

1.3 Contributions

This thesis aims to improve the performance of the octree based occupancy mapping by a systematic approach for parameter reduction and optimisation, and representing occupancy information with the average distance from a point to its k-Nearest Neighbours (k-NN). The main contributions are:

- a low-cost way to create targets and test scenes with the boxes of different textures, either a plain cardboard texture or Voronoi diagrams, arranged in I, O, T, L and S layouts inspired by the shapes of free tetrominoes in Tetris game [35] in different environments, either in front of buildings or in a parking lot, providing precise ground truths for evaluation purposes (Chapter 3, Publication 2);

- a two-step principled methodology that first identifies the most significant parameters by conducting Neighbourhood Component Analysis (NCA) [36] on all parameters and then optimise those using grid search with the area under the Receiver Operating Characteristic (ROC) curve (Chapter 4, Publication 2);

- a k-NN method for occupancy mapping using the context of neighbouring points to update the nodes containing points with a relationship defined between the average distance and the change in occupancy probability using the Cumulative Density Function (CDF) of average distances, potentially decreasing the probability of a node despite the points being present in the node. (Chapter 5, Publications 1 and 3)

Two journal articles and one conference paper were published.

1. Y. Miao, A. J. Hunter, and I. Georgilas, "An occupancy mapping method based on k-nearest neighbours," *Sensors*, vol. 22, no. 1, 139, 2022. (Chapter 5)

2. Y. Miao, A. J. Hunter, and I. Georgilas, "Parameter reduction and optimisation for point cloud and occupancy mapping algorithms," *Sensors*, vol. 21, no. 21, 7004, 2021. (Chapters 3 and 4)

3. Y. Miao, I. Georgilas, and A. J. Hunter, "A k-nearest neighbours based inverse sensor model for occupancy mapping," in *Proceedings of Annual Conference Towards Autonomous Robotic Systems*, 2019, pp. 75-86. (Chapter 5)

Data sets are available at `https://doi.org/10.15125/BATH-00594`, accessed on 10 September 2021 under the Creative Commons Attribution 4.0 license.

1.4 Thesis Outline

In this thesis, the study of the optimisation and extension of the octree based occupancy mapping mainly consists of two parts, i.e., parameter reduction and optimisation, and a k-NN method for occupancy mapping. After providing an introduction in this chapter, the thesis is organised as follows.

Chapter 2 examines the previous related research and presents the detailed background required by the work in the thesis. Initially research on SLAM data sets, OctoMap parameters and the extension of OctoMap has been introduced. Then the procedure of point cloud reconstruction from a stereo camera using the disparity map derived by left and right images is provided, followed by a brief introduction to ORB-SLAM which will be used to generate camera poses in the following chapters. The mathematical derivation of the update policy in occupancy mapping is introduced and a brief analysis of the key steps of OctoMap is given as well. The limitations of OctoMap update policy are further discussed.

Chapter 3 introduces the details of the 20 data sets collected in the outdoor environments with specially designed targets and the measured ground truths are given as well.

Chapter 4 presents the method for parameter reduction using NCA and optimisation with grid search. The details of parameters and performance metrics are first introduced, followed by the node classification procedure using the concept of pixel connectivity [37]. A simple mapping approach as a proxy for the cleanness of point cloud is given as well. The relationship between mapping parameters and performance measures is investigated with neural networks. The results derived by the two-step principled methodology are presented and discussed.

Chapter 5 gives a detailed description of the k-NN based inverse sensor model using the relationship between the average distance from a point to its k-NN and the occupancy probability. The relationship is defined with the CDF of average distances and the choice of the distribution is explained. The map update procedure using the proposed model is given as well. The proposed k-NN method and OctoMap are optimised using the methodology in Chapter 4 and then compared. A point cloud sampling approach based on the k-NN method is introduced in the end.

Chapter 6 concludes the thesis and makes suggestions for future work.

Chapter 2

Literature Review and Background

This chapter first examines the previous related work, including existing data sets, research on OctoMap parameters and the improvement on OctoMap. Then the background required by the research in this work is presented. To generate an occupancy map, point clouds and their corresponding poses are necessary. Monocular and binocular camera models are explained, followed by the procedure of 3D reconstruction of point clouds from disparity maps. A brief introduction to ORB features is given. Based on ORB features, ORB-SLAM is introduced as well, which will be used for generating poses in the following chapters. In the end, occupancy grid mapping and its variant OctoMap are presented. Finally, the research gap explored in the thesis is summarised.

2.1 Literature Review

2.1.1 Public Data Sets

Data sets are needed to demonstrate the ability of a mapping approach. Researchers have created a number of data sets which are publicly available for benchmarking. These data sets were recorded with various sensors such as IMUs, cameras, sonars and laser range finders in either indoor or outdoor environments. The ground truths of the data sets can be categorised into two types, poses or maps. This section presents an overview of existing public data sets.

In [11], three data sets collected in real world environments were used to test the accuracy of OctoMap. The occupancy maps shown in Figures 2-1 and 2-2 were modelled

by OctoMap using three different data sets. The nodes in the free space are not shown for clarity. The map for the corridor of building 079 at the Freiburg campus has been created, as shown in Figure 2-1. The corridor data set was recorded with a laser range finder. The observations outside the building through windows were removed by limiting the max range of the sensor to 10 m. The visualised maps of two fairly large outdoor data sets, i.e.,the Freiburg campus and the New College [38] are presented in Figures 2-2a and 2-2b.

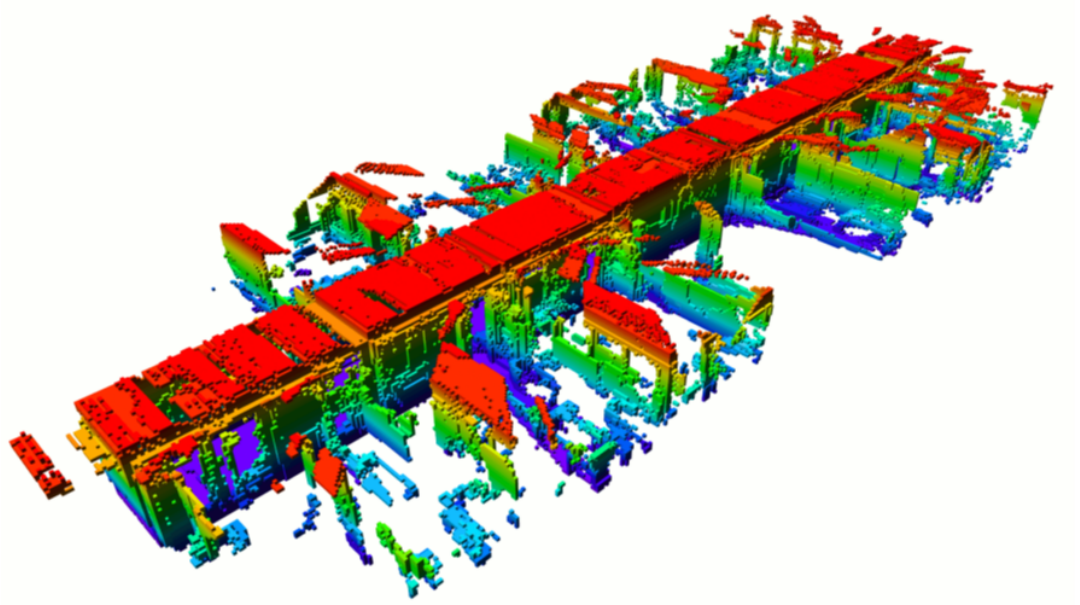
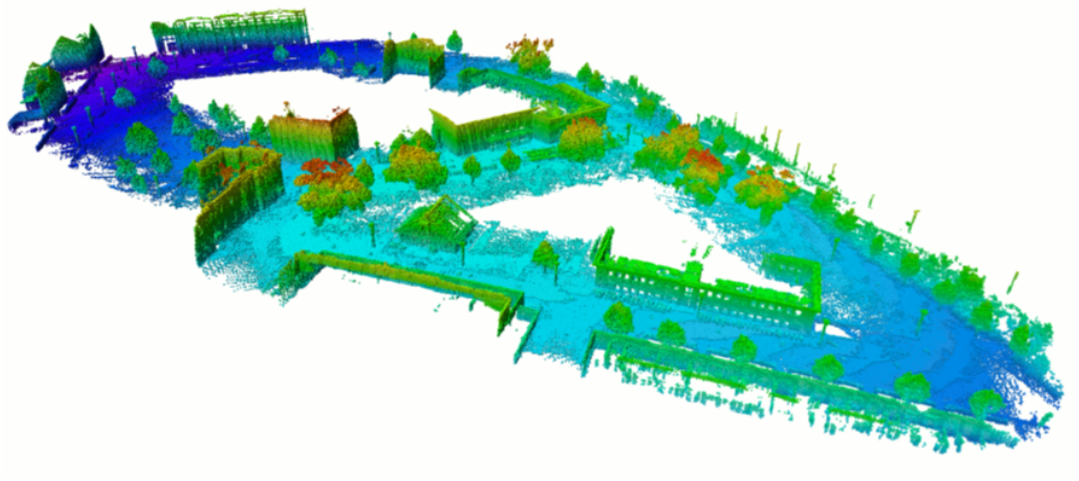


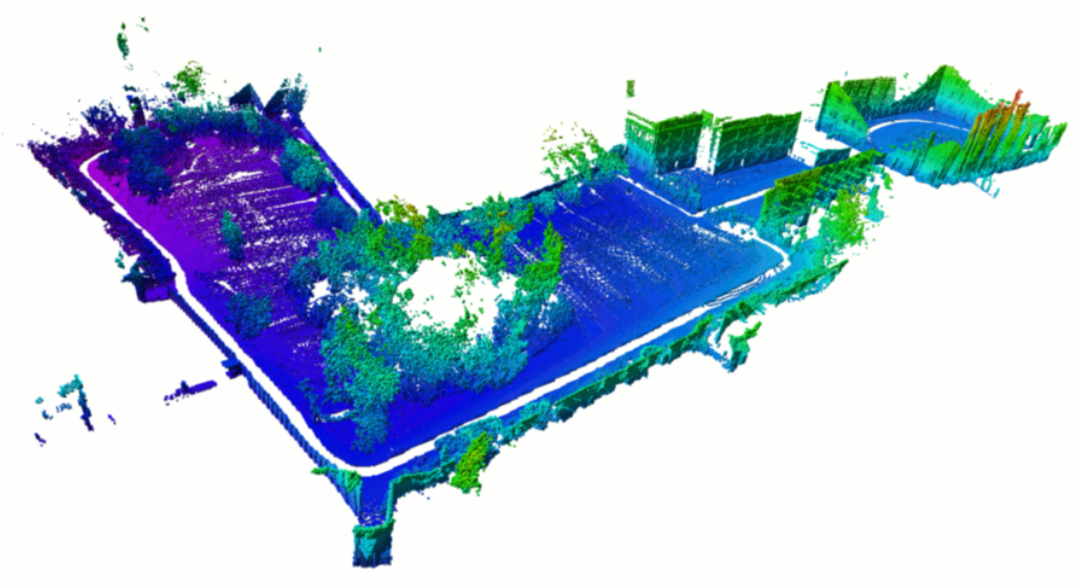
Figure 2-1: Occupancy map of FR-079 corridor data set (size of the scene: 43.7 m × 18.2 m × 3.3 m) [11].

Accuracy was used as the performance metric to evaluate OctoMap in [11]. The accuracy was defined as the percentage of correctly mapped nodes in all 3D scans from the sensor. A node with consistent occupancy states in the evaluated scans and the pre-built map generated by all or part of the scans would be counted as correctly mapped. Although this definition of accuracy can make comparison easy, it cannot illustrate how the model is right or wrong in absolute terms in a confusion matrix [39]. In addition, the map to be evaluated and the ground truth (pre-built map) are generated with the same data set, which may lead to a biased result.

Table 2.1 shows the overview of existing SLAM data sets. Thanks to the work in [40]. Normally, platforms used for data collection include vehicles, rovers, unmanned aerial vehicles (UAVs), unmanned surface vehicles (USVs), unmanned ground vehicles (UGVs), remotely operated underwater vehicles (ROVs), mobile robots and handheld. Types



(a) Freiburg campus.



(b) New College.

Figure 2-2: Resulting octree maps of two outdoor data sets [11]. (a) Freiburg campus data set (size of the scene: 292 m × 167 m × 28 m). (b) New College data set (size of the scene: 250 m × 161 m × 33 m).

of ground truths can be categorised into two groups, poses or maps, including depth maps, 3D point cloud maps and semantic segmentation. Various configurations of sensors can be used for recording data. Vision senors such as monocular cameras, stereo cameras and RGB-D cameras are widely used. Laser range finder is a good choice for long-distance observations. IMU and GPS are commonly used for deriving the position and orientation of a robotic platform.

Table 2.1: Overview of SLAM data sets.

Name	Year	Platform	Environment		Sensor setup								Ground Truth	
							Cameras			LIDAR						
			Indoor	Outdoor	IMU	GPS	Mono	Stereo	RGB-D	2D	3D	Radar	Sonar	Pose	Map
MADMAX [41]	2021	Hand Rover		✓	✓	✓	✓	✓						✓	
FinnForest [42]	2021	Vehicle		✓	✓	✓	✓	✓						✓	
UZH-FPV [43]	2019	UAV	✓	✓	✓	✓	✓							✓	
FMDataset [44]	2019	Hand	✓		✓				✓						
Rosario [45]	2019	Hand		✓	✓	✓		✓						✓	
AQUALOC [46]	2019	ROV		✓	✓		✓							✓	
ADVIO [47]	2018	Hand		✓	✓		✓							✓	
OxIOD [48]	2018	Hand	✓		✓									✓	
KAIST [49]	2018	Vehicle		✓	✓	✓	✓	✓			✓			✓	✓
TUM VI [50]	2018	Hand	✓		✓			✓						✓	
Multi Vehicle Event [51]	2018	Vehicle		✓	✓	✓		✓			✓			✓	✓
Collaborative SLAM [52]	2018	Hand	✓		✓				✓					✓	✓
VI Canoe [53]	2018	USV		✓	✓	✓	✓	✓						✓	
RPG Event [54]	2017	Hand	✓	✓	✓		✓							✓	
Underwater Cave [55]	2017	AUV		✓	✓		✓						✓	✓	
Zurich Urban MAV [56]	2017	UAV		✓	✓	✓	✓							✓	
Chilean Underground [57]	2017	Vehicle		✓				✓				✓		✓	
SceneNet RGB-D [58]	2017	Hand	✓						✓					✓	✓

Table 2-1 (continued): Overview of SLAM data sets.

Name	Year	Platform	Environment		Sensor setup								Ground Truth	
							Cameras			LIDAR		Radar	Sonar		
			Indoor	Outdoor	IMU	GPS	Mono	Stereo	RGB-D	2D	3D			Pose	Map
Symphony Lake [59]	2017	USV		✓	✓	✓	✓			✓				✓	
Sugar Beets [60]	2017	Mobile robot		✓		✓	✓		✓		✓			✓	
Oxford RobotCar [61]	2017	Vehicle		✓	✓	✓	✓	✓			✓			✓	
EuRoC [62]	2016	UAV	✓		✓		✓	✓						✓	✓
Cartographer [63]	2016	Hand	✓		✓					✓	✓				
TUM-Mono [64]	2016	Hand	✓	✓			✓								
Cityscape [65]	2016	Vehicle		✓		✓	✓								✓
Solar-Powered UAV [66]	2016	UAV		✓	✓	✓	✓							✓	
CoRBS [67]	2016	Hand	✓						✓					✓	✓
NCLT [68]	2016	Mobile robot		✓	✓	✓	✓			✓	✓			✓	
TUM-Omni [69]	2015	Hand	✓	✓			✓							✓	
ICL-NUIM [70]	2014	Hand	✓						✓					✓	✓
KITTI [71]	2013	Vehicle		✓	✓	✓	✓	✓			✓			✓	
7 Scenes [72]	2013	Hand	✓						✓					✓	✓
TUM-RGBD [73]	2012	Hand	✓		✓				✓					✓	
Devon Island Rover [74]	2012	Rover		✓		✓		✓			✓			✓	
Ford Campus [75]	2011	Vehicle		✓	✓	✓	✓	✓			✓			✓	
Marulan [76]	2010	UGV		✓	✓	✓	✓			✓				✓	
NewCollege [38]	2009	Mobile robot		✓	✓	✓		✓		✓				✓	

From Table 2.1, it can bee seen that the ground truths for most data sets are poses only. These data sets are not suitable for evaluating the quality of an occupancy map. Data sets, e.g, [65], use semantic segments as ground truths which are not suitable for judging mapping performance. Depth maps derived by stereo cameras are given as ground truths in several data sets such as [49] and [51]. However, as introduced in [51], the reported depths might have errors due to moving objects. Although some RGB-D data sets such as CoRBS [67] can give both poses and 3D point clouds as ground truths, they are only limited to indoor environments. In addition, the derivation of the ground truths in public data sets normally involves high-priced hardware and complicated procedure.

2.1.2 Optimisation of OctoMap Parameters

OctoMap is governed by several parameters which affect the mapping process and the quality of final maps. Although previous studies on parameter optimisation techniques have been reported in the literature, few of them focused on optimising OctoMap parameters and research in the process to identify the parameter set to achieve best occupancy mapping performance remains limited.

In most applications, OctoMap is implemented with default parameters. The default parameters were introduced in [11]; however, there is no clear evidence to show they are the optimal ones. In [11], the values of these parameters were experimentally determined to work best for mapping mostly static environments with laser range finders, while still preserving the ability to adapt to occasional changes. The impact of map resolution on memory usage has been investigated, as shown in Figure 2-3. Memory consumption would increase exponentially as the resolution becomes better. Clamping parameters were also analysed by Kullback–Leibler divergence (KLD) in terms of map accuracy and compression. The trade-off between map confidence and compression is shown in Figure 2-4. With a higher clamping threshold, a map would be further compressed but at the cost of losing map confidence. Although some of OctoMap parameters were claimed to be experimentally determined, how the experiments were conducted is not given in [11]. There is no systematic method provided to test the optimality of default parameters.

Other researchers usually focused on the resolution when OctoMap was implemented to make sure the desired resolution matched well with their application scenarios. In [77], the main parameters of the MoveIt! motion planning framework were investigated to identify the parameters that affect the overall performance of the system most. As one of OctoMap parameters, the resolution was tested since it influences both

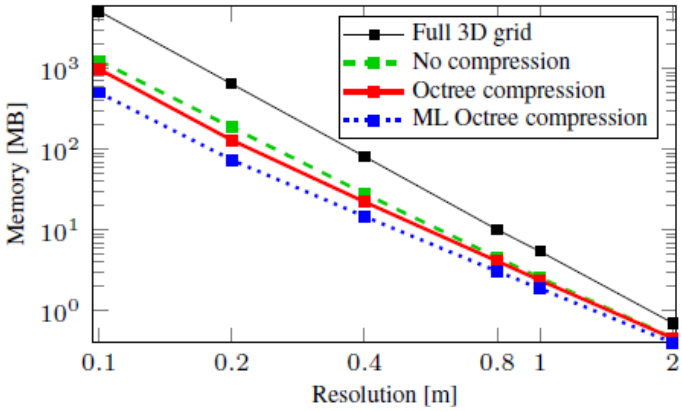


Figure 2-3: Effect of resolution on memory usage of the Freiburg campus data set in logarithmic scale [11].

planning time and the length of the planned movements. As an efficient 3D mapping framework, OctoMap has been tailored for field robots in outdoor environments in [78]. The clamping threshold was slightly increased to achieve better results. In [79], it has been mentioned that the change in the resolution and the initial probability of OctoMap could affect the final result. However, how to choose these parameters was not explained. Moreover, the resolution is not a parameter which directly determines the probability of a cubic node and other OctoMap parameters were often ignored when tuning parameters.

2.1.3 Extension of OctoMap

As introduced in Section 1.1, OctoMap has a wide range of applications due to its compact and flexible octree structure. Research has been done to extend OctoMap to improve its performance or expand its applications.

Since the outliers in the map generated by the raw OctoMap might affect the robot navigation, an improvement algorithm was proposed in [80] to remove the sparse outliers. The average distance from a point to its k-NN in the point cloud was computed. A Gaussian distribution derived by the average distances of all the points was used to remove the outliers in the point cloud. The processed point clouds were then used for building occupancy maps. The proposed method was tested on TUM data set with RGB-D SLAM. As shown in Figures 2-5 and 2-6, the outliers in point clouds and occupancy maps have been removed. Similarly, the Point Cloud Library (PCL) [8] provides several filters such as PassThrough filter, StatisticalOutlierRemoval filter and RadiusOutlier removal to remove the outliers in point clouds. The work in [80] is sim-

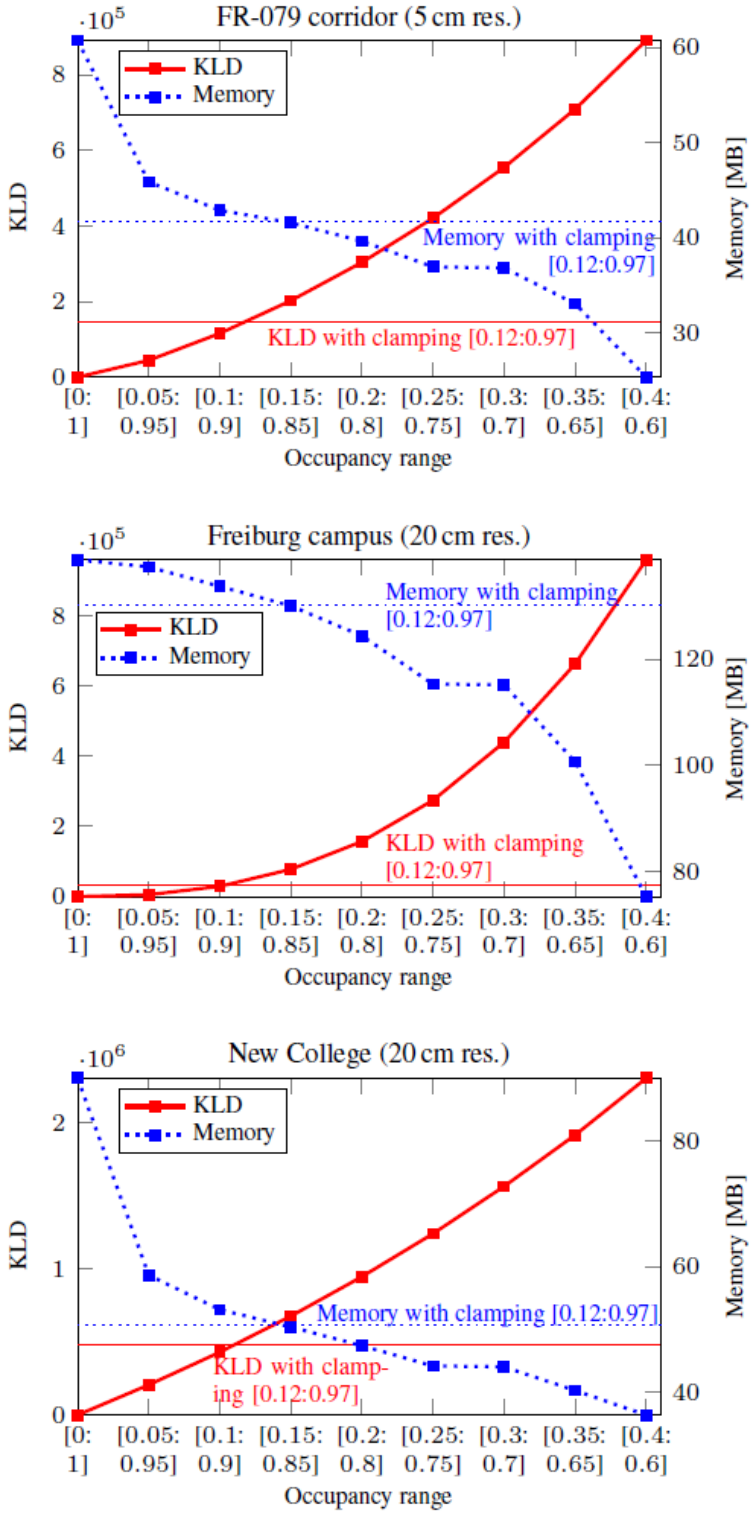
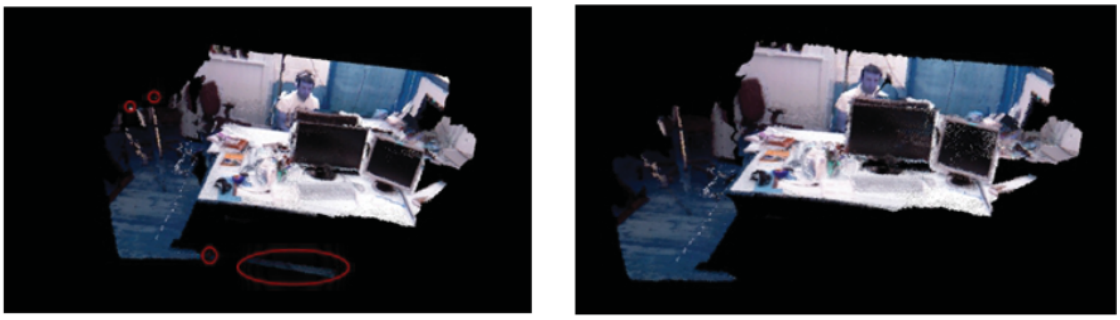


Figure 2-4: Effect of clamping ranges on map compression and accuracy in three data sets [11].

ilar to the StatisticalOutlierRemoval filter in PCL. Instead of building a map with the original point clouds, the quality of the map can be improved using the filtered point clouds. However, the details in environmental features may not be preserved when the outliers or clusters in point clouds are removed by filters. As shown in Figure 2-6, after implementing the removal algorithm, the number of the nodes in the occupancy map decreased from 16 536 to 13 510. Considering an extreme case, for the point clouds of perfect quality, some points will still be removed, which will lead to the loss in map details. In addition, the problem can be more severe when point clouds are down sampled for the use in fairly large scenes.



(a) (b)

Figure 2-5: 3D point clouds of the TUM sequence fr1/xyz [80]. (a) Raw point clouds. (b) Point clouds with outliers removed.

(a) (b)

Figure 2-6: Occupancy maps of the TUM sequence fr1/xyz [80]. (a) Raw occupancy map generated by OctoMap (16 536 nodes). (b) Occupancy map with outliers removed (13 510 nodes).

As introduced in Section 1.1, OctoMap has been extended with semantic segmentation, as shown in Figure 1-10. Similar work has been reported in [81]. Gaussian Processes

(GPs) multi-class classification was implemented for map interface to extend OctoMap to semantic OctoMap. An example of developed semantic OctoMap using the data set in [82] has been shown in Figure 2-7. Instead of using offline trained classifiers for semantic segmentation, a nonparametric and data-driven method to parse scenes was introduced in [31], which can be easily deployed to a large environment.

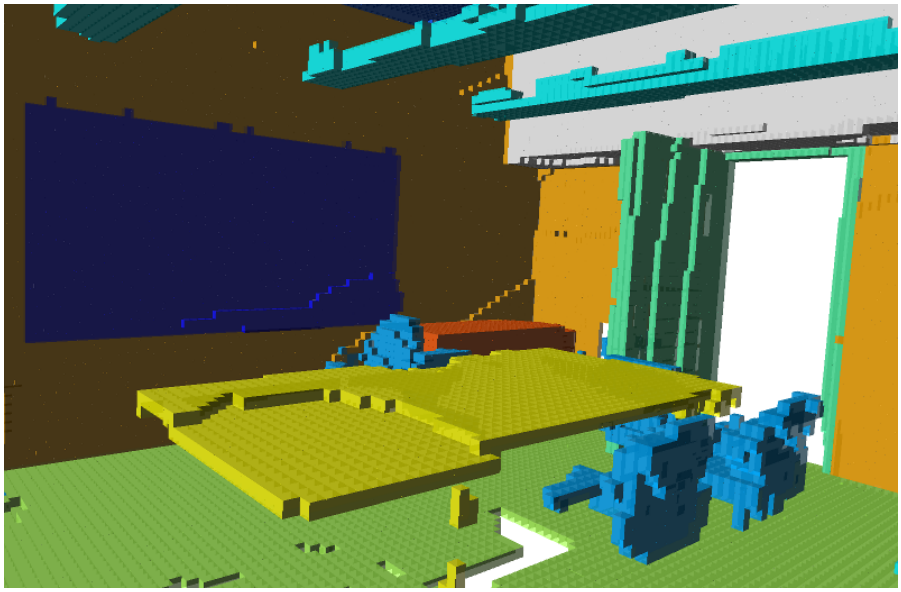


Figure 2-7: Semantic OctoMap built with the Stanford 2D-3D-Semantic data set (area 1, conference room 1) [81].

Recently, machine learning is getting more and more popular. To refine occupancy maps, Recurrent-OctoMap is proposed in [83], with each cell in OctoMap modelled as a Recurrent Neural Network (RNN). Unlike most existing semantic mapping approaches which only focused on improving semantic understanding of single frames, the learning approach in this work aims to fuse semantic observations. In [84], learning-aided 3D occupancy mapping is introduced to build 3D maps from sparse and noisy range sensor data. By expanding the method leveraging Bayesian kernel inference in [85], the occupancy states of the unobserved regions can be predicted to build dense occupancy maps.

Previous studies reported in the literature also focused on the improvement of the efficiency of occupancy mapping. In [86], a computationally efficient probabilistic map update policy utilising the sparse nature of the environment is proposed. The efficiency of the OctoMap framework can also be improved by the Fast Line Rasterisation Algorithm [87].

2.2 Background

2.2.1 Camera Model

To reconstruct point clouds from images, how the objects in the real world are projected into the image needs to be understood. Figure 2-8 shows the transformation from world coordinates to pixel coordinates. The parameters for imaging process can be categorised into extrinsic parameters and intrinsic parameters. Extrinsic parameters transform world coordinates to camera coordinates and intrinsic parameters transform camera coordinates to pixel coordinates. Different coordinate systems will be introduced in the following section.

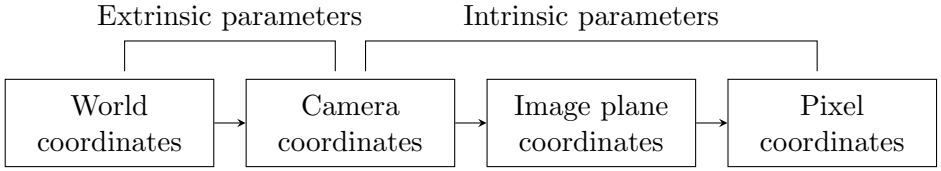


Figure 2-8: Coordinates transformation in imaging process.

Pinhole Camera Model.

The ideal pinhole camera model [88] in figure 2-9 presents the relationship of four coordinate systems, i.e., world frame $O_w - x_w y_w z_w$, camera frame $O_c - x_c y_c z_c$, image plane frame $O_i - x_i y_i$ and pixel frame $O_p - uv$. O_c is the optical centre of the camera, z_c is the principal axis of the camera and perpendicular to the image plane. $O_i - x_i y_i$ and $O_p - uv$ are located in the image plane. O_i is the principal point, which is the intersection of the principal axis and the image plane. The distance between O_c and O_i is the focal length of the camera, which is denoted as f.

For any point (x_c, y_c, z_c) in the camera frame, its projection to the image plane is:

$$\begin{bmatrix} x_i \\ y_i \end{bmatrix} = \frac{f}{z_c} \begin{bmatrix} x_c \\ y_c \end{bmatrix} . \tag{2.2.1}$$

Two scale factors ω_x and ω_y will change the metric units to pixels:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \omega_x x_i + c_x \\ \omega_y y_i + c_y \end{bmatrix} , \tag{2.2.2}$$

where (c_x, c_y) are the coordinates of the principal point in the pixel frame. Let $f_x = \omega_x f$ and $f_y = \omega_y f$, where f_x and f_y are the focal lengths in pixels. Then the transformation

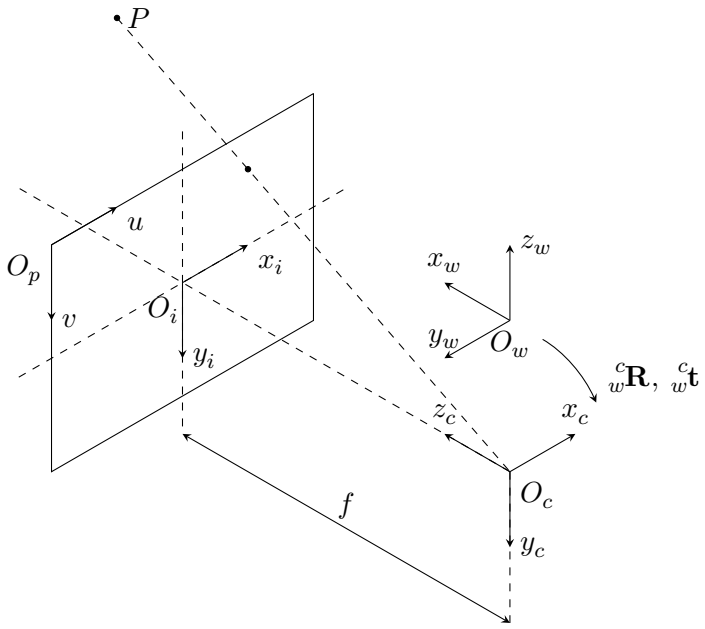


Figure 2-9: Pinhole camera model.

from the camera frame to the pixel frame can be denoted as:

$$z_c \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} \coloneqq \mathbf{K}{}^c\mathbf{P}\,, \tag{2.2.3}$$

where $\mathbf{K}$ is the camera matrix, also known as camera intrinsic parameters, and ${}^c\mathbf{P}$ is the coordinates in the camera frame. Let $\tilde{\mathbf{P}}$ and $\bar{\mathbf{P}}$ denote normalised and homogeneous coordinates of $\mathbf{P}$, respectively. The transformation from camera coordinates to pixel coordinates in (2.2.3) in can be rewritten as:

$${}^p\bar{\mathbf{P}} = \mathbf{K}{}^c\tilde{\mathbf{P}}\,, \tag{2.2.4}$$

where ${}^p\bar{\mathbf{P}}$ represents homogeneous coordinates in the pixel frame. World coordinates can be converted to camera coordinates by a rotation matrix ${}^c_w\mathbf{R}$ and a translation matrix ${}^c_w\mathbf{t}$. Then the transformation from the world frame to the pixel frame can be denoted as:

$${}^p\bar{\mathbf{P}} = \frac{1}{z_c}\mathbf{K}\left(\begin{bmatrix} {}^c_w\mathbf{R} & {}^c_w\mathbf{t} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_w & y_w & z_w & 1 \end{bmatrix}^\top\right)_{(1:3)} \coloneqq \frac{1}{z_c}\mathbf{K}({}^c_w\mathbf{T}\,{}^w\bar{\mathbf{P}})_{(1:3)}\,, \tag{2.2.5}$$

where ${}^w\bar{\mathbf{P}}$ is the homogeneous coordinates in the world frame and ${}^c_w\mathbf{T}$ is the trans-

formation matrix from the world frame to the camera frame. A transformation from homogeneous coordinates to non-homogeneous coordinates is hidden in (2.2.5).

Lens Distortion

Without rectification, the images derived by a camera are normally distorted. Images can be rectified by considering radial distortion and tangential distortion [89]. The light rays will bend more near the edges of the lens than at the optical centre, which will cause radial distortion. Tangential distortion occurs when the lens and the image plane are not strictly parallel. With a good camera calibration and rectification procedure, the distortion in the image can be eliminated.

The radial distortion can be approximated as:

$$\begin{bmatrix} \delta_x^r \\ \delta_y^r \end{bmatrix} = \begin{bmatrix} (\zeta_1 r^2 + \zeta_2 r^4 + \dots)\tilde{x} \\ (\zeta_1 r^2 + \zeta_2 r^4 + \dots)\tilde{y} \end{bmatrix}, \tag{2.2.6}$$

where δ_x^r and δ_y^r are radial distortions, ζ_1, ζ_2, $\dots$ are coefficients for radial distortion, $\tilde{x}$ and $\tilde{y}$ are in normalised image coordinates, $\tilde{x} = \dfrac{u_d - c_x}{f_x}$, $\tilde{y} = \dfrac{v_d - c_y}{f_y}$, (u_d, v_d) are unrectified coordinates, and r is the distance from the point to the centre and $r^2 = \tilde{x}^2 + \tilde{y}^2$. Normally, two coefficients are sufficient for radial distortion compensation.

The tangential distortion can be denoted as:

$$\begin{bmatrix} \delta_x^t \\ \delta_y^t \end{bmatrix} = \begin{bmatrix} 2\eta_1 \tilde{x}\tilde{y} + \eta_2(r^2 + 2\tilde{x}^2) \\ \eta_1(r^2 + 2\tilde{y}^2) + 2\eta_2 \tilde{x}\tilde{y} \end{bmatrix}, \tag{2.2.7}$$

where δ_x^t and δ_y^t are tangential distortions, η_1 and η_2 are the coefficients of tangential distortion.

With accurate calibration, the camera model for radial and tangential distortion corrections can be expressed in the following form:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} f_x(\tilde{x} + \delta_x^r + \delta_x^t) \\ f_y(\tilde{y} + \delta_y^r + \delta_y^t) \end{bmatrix} + \begin{bmatrix} c_x \\ c_y \end{bmatrix}. \tag{2.2.8}$$

3D Reconstruction from Binocular Vision

Figure 2-10 shows the binocular camera model. O_c and O_c' are the optical centres of left and right cameras, respectively. $O_c O_c'$ is called the baseline of which the length is s_b. Let $O_c' - x_c' y_c' z_c'$, $O_i' - x_i' y_i'$ and $O_p' - u'v'$ denote the right camera frame, the right

image plane frame and the right pixel frame. Two cameras are identical in an ideal binocular camera model and two lenses have exactly the same focal length f. P is a scene point in 3D space and its projections onto the image planes of two cameras are P_l and P_r. The coordinates of the projection points in two images are different. The x-coordinates of P_l and P_r in the respective image plane frames created on two cameras are x and x'.

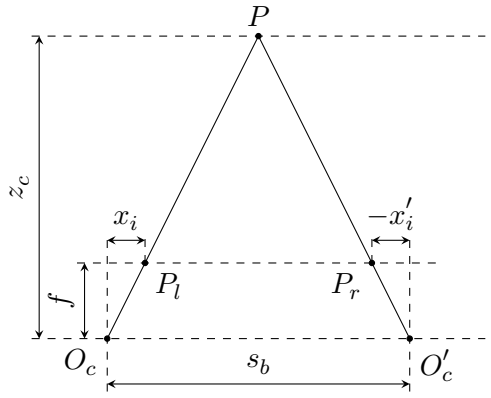


Figure 2-10: Binocular camera model.

The distance from point P to the baseline can be calculated by the projection relationship in Figure 2-10, and is equal to its z-coordinate in the left camera frame:

$$z_c = \frac{f s_b}{d}\,, \tag{2.2.9}$$

where $d = x_i - x_i'$, also known as the disparity.

In computer vision, the disparity is measured in pixels and referred to as the coordinate difference of a point between the left and right images. With (2.2.1) and (2.2.3), the following relationship can be derived:

$$\frac{x_i}{f} = \frac{u - c_x}{f_x}\,. \tag{2.2.10}$$

Let f_p denote the focal length in pixels of two lens and normally $f_p = f_x = f_y$. The coordinates of P_l and P_r in their corresponding pixel frames are denoted as (u, v) and (u', v'). The coordinates of the principal points of two cameras are (c_x, c_y) and (c_x', c_y'). Then the distance from point P to the baseline can be calculated by:

$$z_c = \frac{f_p s_b}{d_p}\,, \tag{2.2.11}$$

where d_p is the disparity in pixels and $d_p = u - u' - c_x + c'_x$ and in the ideal model $c_x = c'_x$.

Then the coordinates of a point in the camera frame can be computed by its corresponding coordinates (u, v) in the pixel frame. This procedure is called 3D reconstruction, which can be written as:

$$\begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & -c_x \\ 0 & 1 & 0 & -c_y \\ 0 & 0 & 0 & f_p \\ 0 & 0 & \frac{1}{s_b} & -\frac{c_x - c'_x}{s_b} \end{bmatrix} \begin{bmatrix} u \\ v \\ d_p \\ 1 \end{bmatrix}. \tag{2.2.12}$$

Then the coordinates of the point in the camera frame are:

$${}^{c}\mathbf{P} = \begin{bmatrix} x_c & y_c & z_c \end{bmatrix}^\top = \frac{1}{W} \begin{bmatrix} X & Y & Z \end{bmatrix}^\top. \tag{2.2.13}$$

For the case of a stereo camera, the StereoSGBM algorithm [17] can generate the disparity map of left and right images. Then point clouds can be reconstructed by the stereo camera model from this disparity map. The parameters in the StereoSGBM algorithm in OpenCV are as follows:

- $d_{\min}$ is the minimum possible disparity value. It is normally set to 0 and should be adjusted accordingly when rectification algorithms can shift images.
- d_n is the difference between maximum disparity and minimum disparity. It must be a number greater than 0 and divisible by 16.
- B_s is the block size for matching two images. It must be an odd number no less than 1.
- P_1 is the first parameter for disparity smoothness control. A reasonably good sample in OpenCV is $P_1 = 8N_cB_s^2$, where N_c is the number of image channels.
- P_2 is the second parameter for disparity smoothness control. A reasonably good sample in OpenCV is $P_2 = 32N_cB_s^2$.
- d_m is the maximum allowed difference in the left-right disparity check. A non-positive value will disable the check.
- C is the truncation value for pre-filtered image pixels.
- r_u is the margin in percentage.

- d_w is the maximum size of smooth disparity regions to consider noise speckles and invalidate. 0 will disable speckle filtering.
- d_v is the maximum variation in disparity. Normally, 1 or 2 for d_v is good enough.
- *mode* is an option to run the full-scale two-pass dynamic programming algorithm, which is set to false by default.

2.2.2 ORB-SLAM

ORB Feature

Image feature has a wide range of applications in computer vision. As an efficient alternative to SIFT (Scale-Invariant Feature Transform) or SURF (Speeded-Up Robust Features), ORB (Oriented FAST and Rotated BRIEF) was proposed in [90]. ORB is based on FAST (Features from Accelerated Segment Test) keypoint detector and BRIEF (Binary Robust Independent Elementary Features) descriptor.

Keypoint FAST [91] is effective in detecting corner points, which can be used for extracting features for a real-time system. Normally, there are a large number of FAST keypoints in a image. ORB uses Harris corner filter [92] to determine which points should be accepted. Compared with the FAST keypoint, ORB is scale invariant by implementing an image pyramid [93]. In addition, ORB also shows good rotational invariance by representing orientation information with intensity centroid as per [94]. The moment of a patch can be defined as:

$$M_{ij} = \sum_{x,y} x^i y^j I(x,y)\,, \tag{2.2.14}$$

where $i, j \in \{0, 1\}$. Then the centroid is:

$$E = \left(\frac{M_{10}}{M_{00}}, \frac{M_{01}}{M_{00}}\right)\,. \tag{2.2.15}$$

From the geometric centre O to the centroid E, the vector $\overrightarrow{OE}$ can be constructed. The orientation of the patch can be defined as:

$$\gamma = \text{atan2}(M_{01}, M_{10})\,. \tag{2.2.16}$$

Descriptor The BRIEF descriptor is a binary vector which consists of only 0 and 1 [95]. The vector is normally a 128 to 512 bits string. In ORB feature, the length of a

descriptor vector $n = 256$. Each keypoint in a smoothed image patch can be described by a binary test as:

$$\tau(\mathbf{p}; \mathbf{x}, \mathbf{y}) \coloneqq \begin{cases} 1 & \text{if } I_{\mathbf{p}}(\mathbf{x}) < I_{\mathbf{p}}(\mathbf{y}) \\ 0 & \text{if } I_{\mathbf{p}}(\mathbf{x}) \geq I_{\mathbf{p}}(\mathbf{y}) \end{cases}, \tag{2.2.17}$$

where $I_{\mathbf{p}}(\mathbf{x})$ is the intensity of image patch $\mathbf{p}$ at the point $\mathbf{x}$, $\mathbf{x}$ and $\mathbf{y}$ are the points around the centre of the patch randomly picked up by a Gaussian distribution. Then the feature can be defined as a vector with n binary tests:

$$f_n(\mathbf{p}) \coloneqq \sum_{1 \leq i \leq n} 2^{i-1} \tau(\mathbf{p}; \mathbf{x}_i, \mathbf{y}_i) \,. \tag{2.2.18}$$

Since the descriptor is sensitive to high-frequency noise, it is necessary to smooth image before performing binary tests. In ORB feature, a 31×31 image patch is smoothed by a 5×5 sub-window. Compared with BRIEF, the descriptor in ORB is invariant to in-plane rotation. In [90], an approach to steer BRIEF is presented based on the orientation of keypoints. A matrix is defined for n binary tests at location $(\mathbf{x}_i, \mathbf{y}_i)$:

$$\mathbf{S} = \begin{bmatrix} \mathbf{x}_1 & \dots & \mathbf{x}_n \\ \mathbf{y}_1 & \dots & \mathbf{y}_n \end{bmatrix} . \tag{2.2.19}$$

Then the steered version of $\mathbf{S}$ is:

$$\mathbf{S}_\gamma = \mathbf{R}_\gamma \mathbf{S} \,, \tag{2.2.20}$$

where $\mathbf{R}_\gamma$ is the rotation matrix derived by orientation in (2.2.16). Then the steered BRIEF descriptor is:

$$g_n(\mathbf{p}, \gamma) \coloneqq f_n(\mathbf{p}) \mid (\mathbf{x}_i, \mathbf{y}_i) \in \mathbf{S}_\gamma \,. \tag{2.2.21}$$

The angle is discretised with an increment of $\pi/15$ and a lookup table of BRIEF patterns is pre-computed. Then the correct set of points $\mathbf{S}_\gamma$ will be used to compute its descriptor when the keypoint orientation is consistent in different views.

Brief Introduction to ORB-SLAM

ORB-SLAM is considered to be the most complete feature-based monocular visual SLAM system and has been extended to stereo visual SLAM [33, 34, 96]. Figure 2-11 is the overview of ORB-SLAM system. It is a feature based SLAM and ORB features are used to detect keypoints in the mapping process. ORB-SLAM consists of three threads, tracking, local mapping and loop closing. They are parallel threads, so ORB-SLAM can run efficiently on a CPU without using a GPU. Bundle adjustment (BA) is performed in both local mapping and loop closing to reduce the projection error of the visual sensor.

A map will be updated when the loop is detected to reduce the cumulative error in the pose. Loop detection also enables ORB-SLAM to relocate a mobile robot after losing tracking. ORB-SLAM performs well in tracking and localisation. However, the map derived by ORB-SLAM is not suitable for collision avoidance since ORB-SLAM uses sparse feature points to build a map.

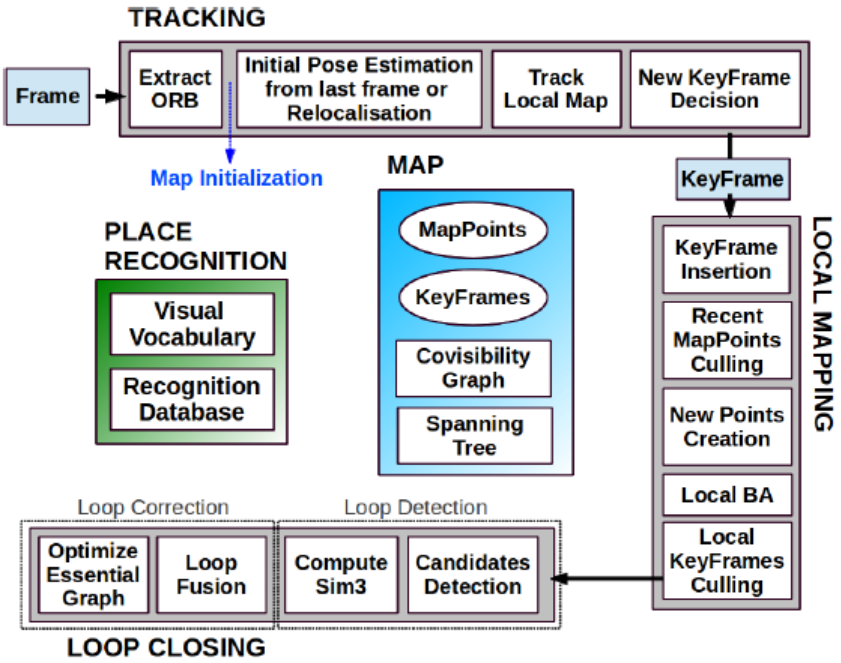


Figure 2-11: Overview of ORB-SLAM system [33].

In ORB-SLAM, the parameters are as follows:

- N_f is the number of features in each image.
- s_f is the scale factor between levels in the scale pyramid.
- N_l is the number of levels in the scale pyramid.
- t_i is the initial threshold implemented to extract FAST corners.
- $t_{\min}$ is the lower threshold to extract FAST corners. If no corners are detected with the initial threshold, threshold $t_{\min}$ will be imposed.

These parameters are all for ORB feature extraction. The corresponding default values are presented in Table 2-2.

Table 2-2: Default ORB parameters.

Parameter	Value
N_f	2000
s_f	1.2
N_l	8
t_i	12
$t_{\min}$	7

2.2.3 Occupancy Grid Mapping

In robotics, to navigate a robot the environment needs to be represented in a way which informs the robot which parts of the environment are free and which are not. Occupancy grid mapping presents the world as a field of random variables which are estimated by the posterior probability and arranged in evenly spaced grids [97], and was first introduced in [98]. The probability distribution of the map can be expressed by:

$$p(m \mid z_{1:t}, x_{1:t}) = \prod_i p(m_i \mid z_{1:t}, x_{1:t}) \,, \tag{2.2.22}$$

where m is the map, m_i is the cell, $z_{1:t}$ is the measurements up to time t and $x_{1:t}$ is the sequence of poses.

By implementing Bayes rules [99], the posterior probability of each cell is:

$$p(m_i \mid z_{1:t}, x_{1:t}) = \frac{p(z_t \mid m_i, z_{1:t-1}, x_{1:t}) p(m_i \mid z_{1:t-1}, x_{1:t})}{p(z_t \mid z_{1:t-1}, x_{1:t})} \,. \tag{2.2.23}$$

With Markov assumption [100], the above probability can be denoted as:

$$p(m_i \mid z_{1:t}, x_{1:t}) = \frac{p(z_t \mid m_i, x_t) p(m_i \mid z_{1:t-1}, x_{1:t-1})}{p(z_t \mid z_{1:t-1}, x_{1:t})} \,. \tag{2.2.24}$$

With Bayes rules:

$$p(z_t \mid m_i, x_t) = \frac{p(m_i \mid z_t, x_t) p(z_t \mid x_t)}{p(m_i \mid x_t)} \,. \tag{2.2.25}$$

Then (2.2.24) can be written as:

$$p(m_i \mid z_{1:t}, x_{1:t}) = \frac{p(m_i \mid z_t, x_t) p(z_t \mid x_t) p(m_i \mid z_{1:t-1}, x_{1:t-1})}{p(m_i \mid x_t) p(z_t \mid z_{1:t-1}, x_{1:t})} \,. \tag{2.2.26}$$

By implementing Markov assumption again, the probability can be denoted as:

$$p(m_i \mid z_{1:t}, x_{1:t}) = \frac{p(m_i \mid z_t, x_t)p(z_t \mid x_t)p(m_i \mid z_{1:t-1}, x_{1:t-1})}{p(m_i)p(z_t \mid z_{1:t-1}, x_{1:t})} \,. \tag{2.2.27}$$

The probability of its complementary event is:

$$p(\neg m_i \mid z_{1:t}, x_{1:t}) = \frac{p(\neg m_i \mid z_t, x_t)p(z_t \mid x_t)p(\neg m_i \mid z_{1:t-1}, x_{1:t-1})}{p(\neg m_i)p(z_t \mid z_{1:t-1}, x_{1:t})} \,. \tag{2.2.28}$$

The ratio of (2.2.27) and (2.2.28) is:

$$\frac{p(m_i \mid z_{1:t}, x_{1:t})}{p(\neg m_i \mid z_{1:t}, x_{1:t})} = \frac{p(m_i \mid z_t, x_t)p(m_i \mid z_{1:t-1}, x_{1:t-1})p(\neg m_i)}{p(\neg m_i \mid z_t, x_t)p(\neg m_i \mid z_{1:t-1}, x_{1:t-1})p(m_i)} \,. \tag{2.2.29}$$

That is:

$$\frac{p(m_i|z_{1:t}, x_{1:t})}{1 - p(m_i|z_{1:t}, x_{1:t})} = \frac{p(m_i|z_t, x_t)}{1 - p(m_i|z_t, x_t)} \frac{p(m_i|z_{1:t-1}, x_{1:t-1})}{1 - p(m_i|z_{1:t-1}, x_{1:t-1})} \frac{1 - p(m_i)}{p(m_i)} \,. \tag{2.2.30}$$

The above can be simplified by log-odds notation:

$$l(x) = \frac{p(x)}{1 - p(x)} \,. \tag{2.2.31}$$

Then the equation (2.2.30) can be expressed as:

$$l(m_i \mid z_{1:t}, x_{1:t}) = l(m_i \mid z_t, x_t) + l(m_i \mid z_{1:t-1}, x_{1:t-1}) - l(m_i) \,, \tag{2.2.32}$$

where $l(m_i \mid z_t, x_t)$ is the inverse sensor model, $l(m_i \mid z_{1:t-1}, x_{1:t-1})$ is the recursive term and $l(m_i)$ is the prior probability.

2.2.4 OctoMap

OctoMap integrates sensor readings with occupancy grid mapping in a method as per [98], which has been introduced in the previous section. In OctoMap, by incorporating pose x_t into measurement z_t, the probability of a node given sensor measurements $z_{1:t}$ can be denoted as:

$$l(m_i \mid z_{1:t}) = l(m_i \mid z_{1:t-1}) + l(m_i \mid z_t) \,, \tag{2.2.33}$$

where $l(m_i \mid z_t)$ is the inverse sensor model. A ray-cast operation will be performed from the sensor origin to the endpoints to determine which nodes should be updated.

The inverse sensor model in OctoMap is defined as:

$$l(m_i \mid z_t) = \begin{cases} l_{occ} & \text{if beam is reflected within volume} \\ l_{free} & \text{if beam traversed volume} \end{cases}, \tag{2.2.34}$$

where l_{occ} and l_{free} are the log-odds values to update occupied and free cells, respectively. The clamping update policy in [22] is applied in OctoMap to set limitations on the log-odds value:

$$l(m_i \mid z_t) = \max(\min(l(m_i \mid z_{1:t-1}) + l(m_i \mid z_t), l_{\max}), l_{\min}), \tag{2.2.35}$$

where $l_{\max}$ and $l_{\min}$ are lower and upper bounds in log-odds, respectively.

Based on the above OctoMap algorithm, the parameters which have an impact on the occupancy probability of a node are listed as follows:

- $p_{\max}$ is the upper clamping threshold. It is the upper bound of the occupancy probability.
- p_h is the probability for a "hit". If a node contains endpoints, a "hit" will be integrated to the node.
- p_t is the occupancy threshold. The node will be marked as occupied if its occupancy probability is greater than the threshold.
- p_m is the probability for a "miss". If a node is traversed by rays, a "miss" will be integrated to the node.
- $p_{\min}$ is the lower clamping threshold. It is the lower bound on the occupancy probability.

Default OctoMap parameters are presented in Table 2-3.

Table 2-3: Default OctoMap parameters.

Parameter	Value
$p_{\max}$	0.971
p_h	0.7
p_t	0.5
p_m	0.4
$p_{\min}$	0.1192

Through the literature review and the update policy in [11], the limitations of OctoMap

are summarised as follows:

- *Default parameters implemented in OctoMap may not be the optimal ones.* The operation of OctoMap is governed by several parameters, the choice of which will affect the performance of the mapping algorithm. There is no evidence to show the default parameters introduced in [11] are the optimal ones. Only clamping parameters are analysed in terms of map accuracy and compression.

- *The evaluation using accuracy cannot illustrate how the map is right or wrong.* In [11], point clouds are generated by a LIDAR sensor and the algorithm is evaluated using accuracy by comparing the map generated by the evaluated scans and the pre-built map generated with the same data set. However, accuracy cannot demonstrate how the map is right or wrong as a confusion matrix [39].

- *Ground truths and the maps to be evaluated are generated with the same data set, which may result in a biased evaluation.* On one hand, the measurements in the data set may contain noise and thus cannot represent the ground truth very well. On the other hand, the ground truth is better to be generated in a different way than with the data set itself. A better alternative would have been the use of a measured ground truth based on the measurements with a measuring device.

- *Parameters for point cloud generation and pose generation are not considered.* A potential limitation in [11] is that point clouds are not always obtained directly from 3D sensors, e.g. LIDAR, but using cheaper solutions, for example, a stereo camera. In this case the point clouds need to be reconstructed following an algorithm. As a result, parameters for point cloud generation affect the quality of point clouds and thus have a potential impact on mapping performance. Since it requires both point clouds and corresponding poses to build an occupancy map, parameters for pose generation should be considered as well.

- *The inverse sensor model in OctoMap has limitations when dealing with nodes containing points.* Normally, point clouds are either generated by sensors directly or by implementing algorithms on the original data from sensors. Due to sensor noise and the limitations of algorithms, a point cloud normally contains both points on the external surfaces of objects and noise points. One one hand, in OctoMap, the nodes containing endpoints are updated with the same probability regardless of the points being noise. On the other hand, with a single frame of point cloud, the probability of a node with points inside can be increased but never allowed to be decreased. Moreover, potentially occupied nodes containing points but traversed by the ray cast operation may be marked as free.

2.3 Research Gap

The literature review has shown that OctoMap is an important research area for robotic perception. Through review, the research gaps can be identified as:

- The derivation of the ground truths in existing public data sets normally requires high-priced hardware and complicated procedure. There is a lack of a low-cost and simple way to create data sets with precise ground truths.
- Few of previous studies focus on the process to identify the parameter sets for the best occupancy mapping performance.
- The extension of OctoMap in the previous work is mainly about improving map quality through pre-processing or expanding its applications, but the research in improving the inverse sensor model of OctoMap remains limited. This is another research gap.

2.4 Summary

This chapter summarises the existing work related to the contributions of the thesis. Public SLAM data sets and the research on the improvement and extension of OctoMap has been introduced. The background required by the following chapters are also given. To generate occupancy maps, point clouds and corresponding poses are required. Point clouds can be reconstructed from the disparity map using a stereo camera model. ORB-SLAM can produce the poses needed for occupancy mapping. The mathematical principles of occupancy grid mapping are introduced. The update policy and the limitations of OctoMap are explained as well.

Chapter 3

Test Scenes and Targets

This chapter is based on [101], and describes the test scenes and targets for the experiments in this thesis. Initially, the motivation for measured targets with rich image features is introduced. Then the setup of test scenes and targets is presented, followed by the procedure for data collection. The overview of each data set is provided and the corresponding camera trajectory is given as well. In the end, different types of ground truths are discussed and the ground truth for each data set is presented.

3.1 Data Sets

An overview of data sets has been presented in Table 2.1 and the limitations of existing data sets have been discussed in Section 2.1.1. A brief summary is given here. Most public data sets use poses as the ground truth. Although these data sets can be used by the map ground truth generated with part of the frames and the map for evaluation built with the other frames as per [11], it may lead to biased results. Depth maps given in the data sets recorded by stereo cameras normally suffer from moving objects. RGB-D cameras can be used for recording data sets, providing accurate poses and 3D point clouds as ground truths. However, their applications are limited to indoor environments. Moreover, the concrete experiments to generate ground truths in existing data sets require either expensive sensors or complicated procedure. In this section, a low-cost and simple way is used to collect data sets and generate ground truths.

3.1.1 Motivation for Reference Targets

In the most applications of occupancy mapping, the scene to be explored is normally relatively large and the objects are of irregular shapes. In such environments, it is

difficult to evaluate the performance of a mapping algorithm since an accurate ground truth for real world representation can hardly be obtained. In addition, visual-based mapping algorithms usually heavily rely on image features but objects in the scene can be lack of features. To make the evaluation reliable and create more test scenes, covering the surfaces of targets with artificial textures can be considered. For the ease of evaluation, a small measured scene in which objects are of regular shapes and rich features is needed to mediate between mapping algorithms and real applications.

Boxes are naturally to be a good choice of targets since they can be accurately measured. However, the colour of the external surfaces of plain boxes remains same. As a consequence, targets like boxes are lack of image features. To create more features, a Voronoi diagram can be used as the patterns on the coverings of boxes. It has a wide range of applications such as image segmentation, path planning and network coverage optimisation [102–104]. Given a set S of n points in the plane, a Voronoi diagram divides a plane into regions where all the points in each region is closer to one point of S than to any other point of S [105]. Given two points $P_i, P_j \in S$ $(i \neq j)$, the bisector of $\overline{P_i P_j}$ divides the plane into two halves. The half-plane containing P_i is the locus of points closer to P_i than to P_j. Let $H(P_i, P_j)$ denote this half-plane. Then the Voronoi polygon associated with P_i can be defined as:

$$V_i = \bigcap_{i \neq j} H(P_i, P_j)\,. \tag{3.1.1}$$

In fact, this polygon is the intersection of $n-1$ half-planes and all the points in this polygon is closer to P_i than other points of S. The Voronoi diagram then can be defined as the collections of convex polygons produced by all the points in S:

$$V = \bigcup_{i} V_i\,. \tag{3.1.2}$$

The distance between points was originally defined by Euclidean distance to characterise the loci of proximity [106]. In this thesis, the Euclidean plane is used to generate Voronoi diagrams, the polygons of which are then filled with random colours.

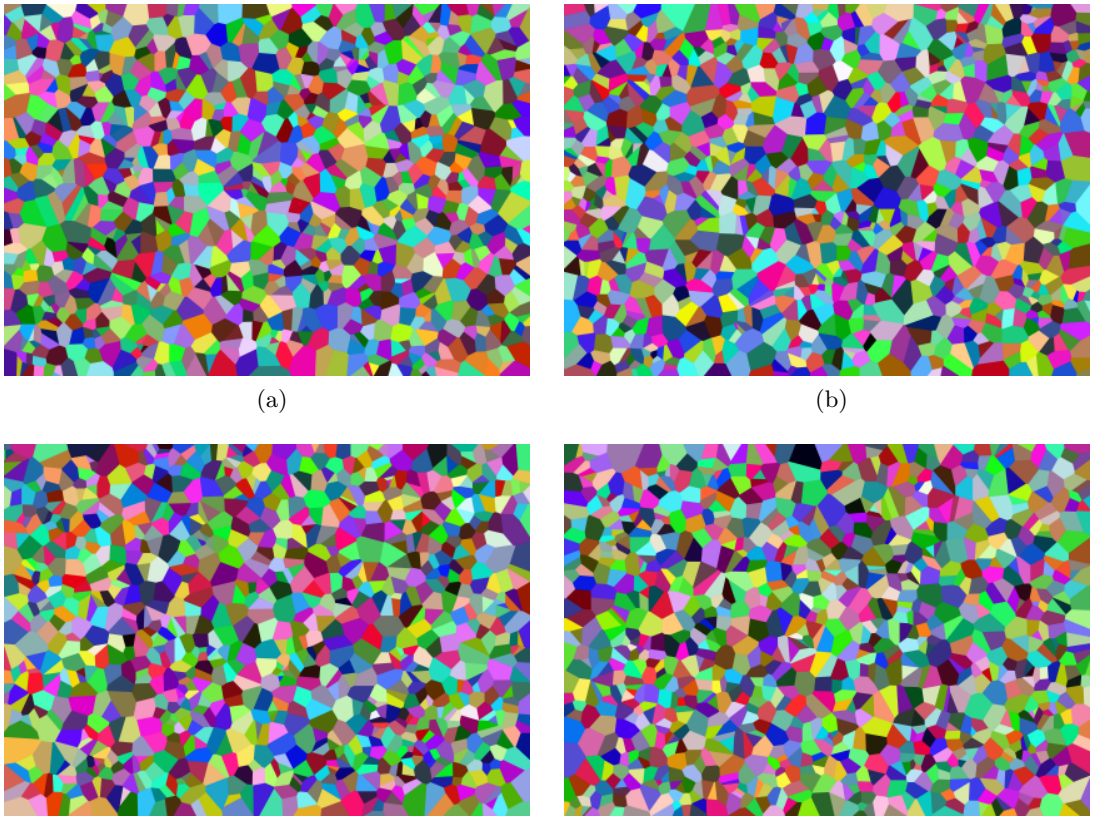
3.1.2 Experimental Setup

The experiments are conducted in two different environments, in front of buildings and in a parking lot. Boxes with different textures on the external surfaces are the targets to be explored. The boxes have a plain cardboard texture or are covered with Voronoi

diagrams to allow the investigation of the impact of the texture.

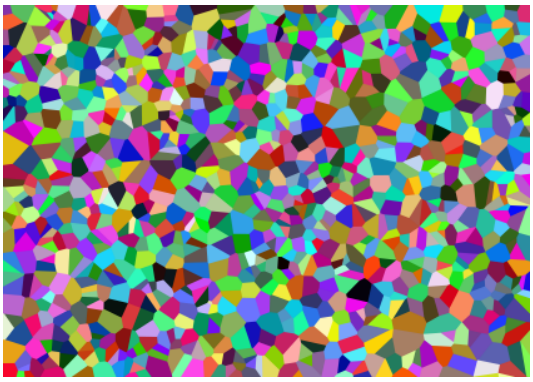
To cover the external surfaces of the boxes, the Voronoi diagram is printed on A0 posters with 300 DPI and then cropped to match the size of the boxes. The average size of the polygons in the diagram is about 3 cm × 3 cm. Since patterns are randomly generated, the diagram is different in each poster. Each polygon in the Voronoi diagram is filled with a random colour. Voronoi diagrams to cover the boxes are presented in Figure 3-1.

Inspired by the Tetris game, several different layouts can be created with a pair of boxes. There are seven one-sided tetrominoes in Teris game, including two enantiomeric pairs [107]. These shapes are not superimposable in 2D space and can be translated, rotated but not reflected. By excluding one of the shapes from each enantiomeric pair, five free tetrominoes can be obtained [35]. As shown in Figure 3-2, the shapes of five free tetrominoes are I, O, T, L and S. Based on these free tetrominoes, five layouts of two boxes can be created.

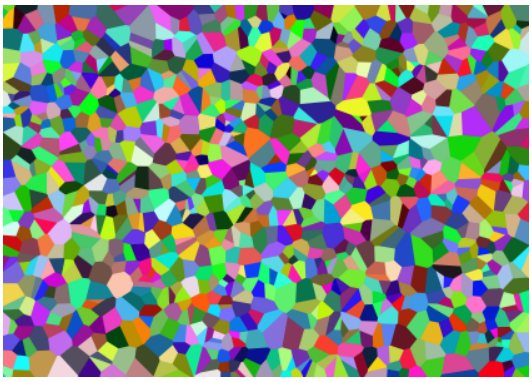


(a) (b) (c) (d)

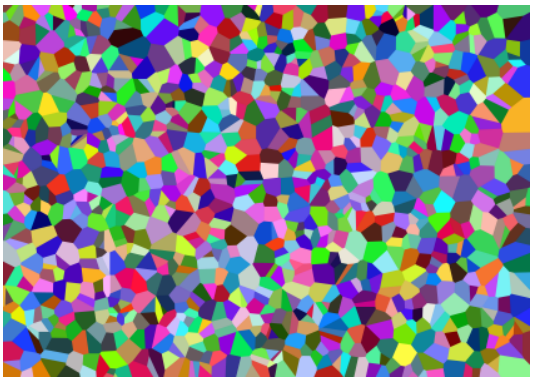
Figure 3-1: Voronoi diagrams. (a–d) Randomly generated patterns.



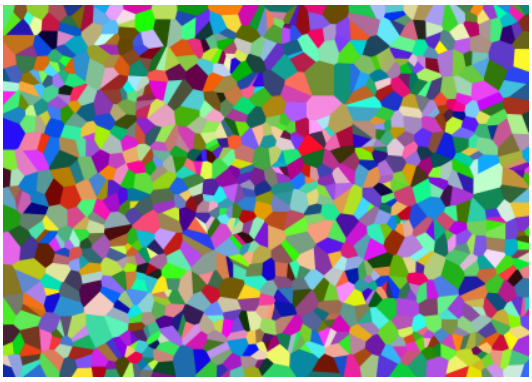
(e)



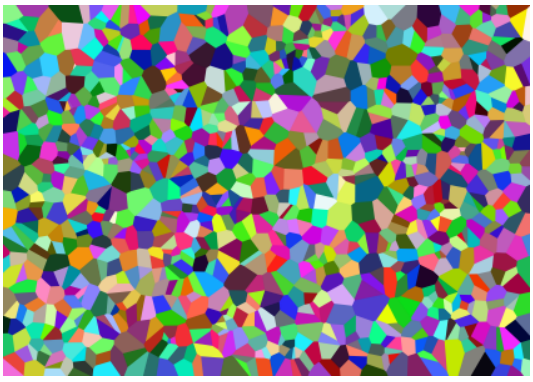
(f)



(g)



(h)



(i)

(j)

Figure 3-1 (continued): Voronoi diagrams. (e–j) Randomly generated patterns.

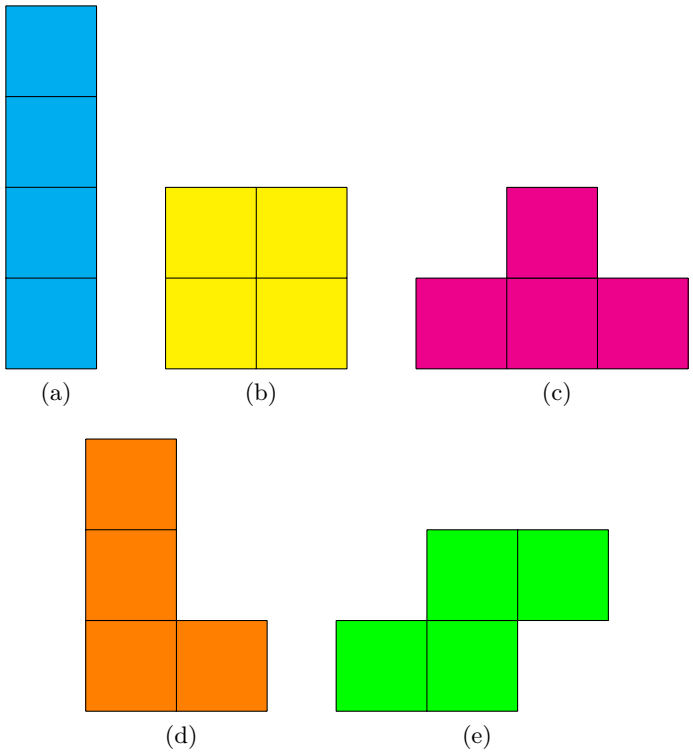


Figure 3-2: Free tetrominoes. (a) I tetromino. (b) O tetromino. (c) T tetromino. (d) L tetromino. (e) S tetromino.

3.1.3 Data Collection

Considering two environments, two textures and five layouts, 20 data sets will be collected. A circle of approximate radius 2.96 m is drawn on the ground. Then the boxes are put in the centre of the circle and arranged in a layout as one of the free tetrominoes. The centre of the circle is also the centroid of the bottom surface of each layout.

A ZED stereo camera (Stereo Labs, USA) is placed in front of the boxes and the initial relative position between the camera and boxes is measured. Then the camera moves along the circle orbiting around the objects twice to record videos at HD resolution (2560×720 pixels for a stereo camera). Figure 3-3 shows boxes of different layouts and textures in two environments, and corresponding camera trajectories are presented in Figure 3-4. The camera moves in a anti-clock direction in the experiments. For each data set, ORB-SLAM introduced in Sections 2.2.2 is implemented to produce camera poses. Then the trajectory of the camera can be plotted with the pose of each frame in the data set. The origin point in each coordinate system is the initial position of the camera.

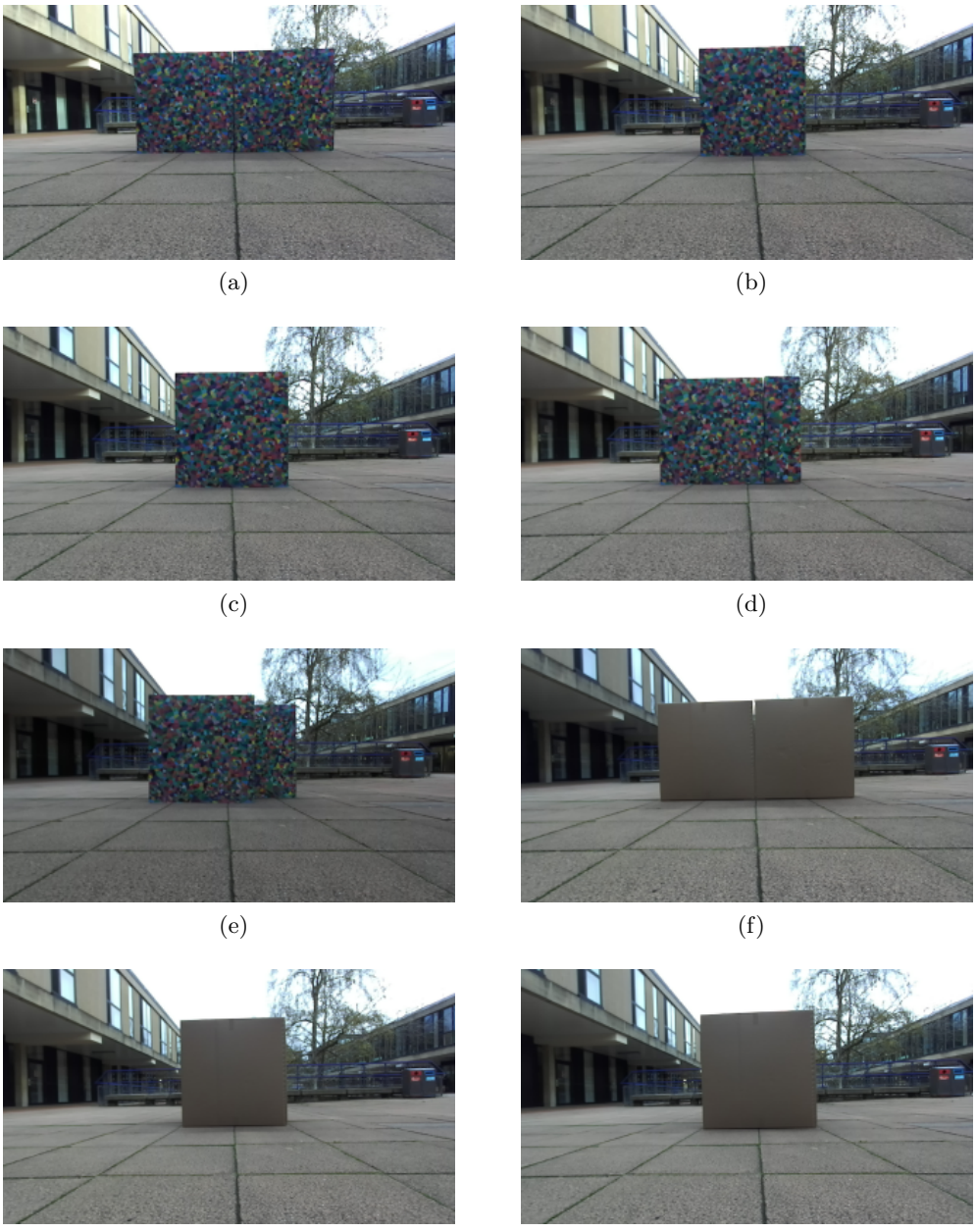


Figure 3-3: Boxes of five layouts and two textures in two environments. (a–e) I, O, T, L and S layout boxes with Voronoi diagrams in front of buildings. (f–h) I, O and T layout plain boxes in front of buildings.

(i)

(j)

(k)

(l)

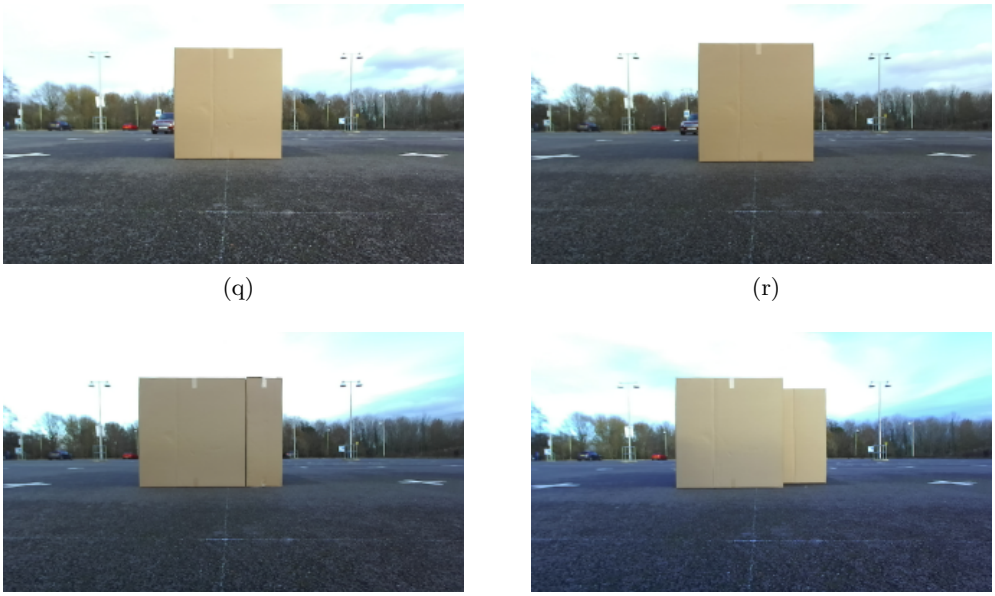
(m)

(n)

(o)

(p)

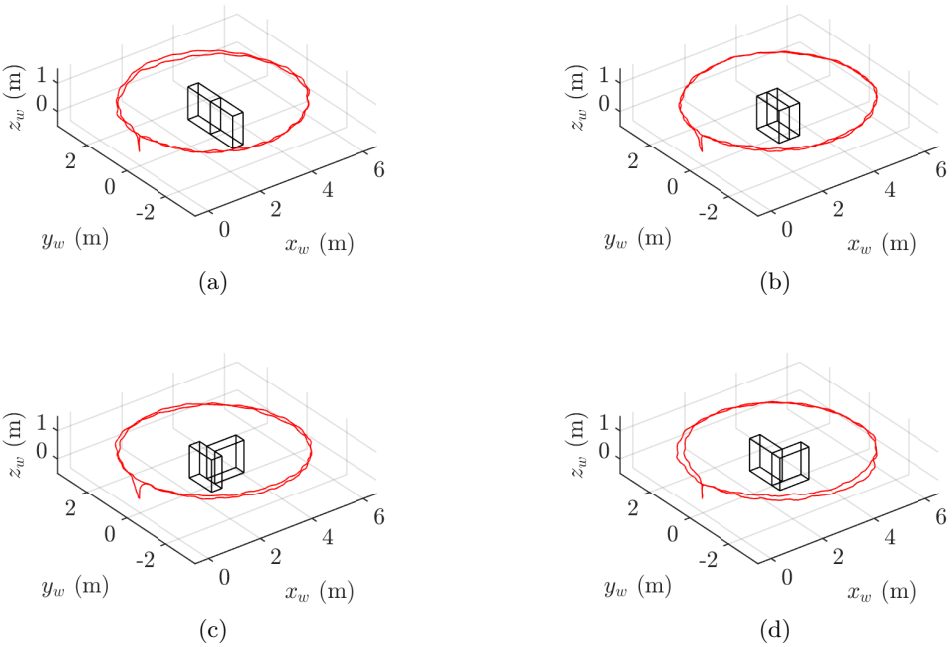
Figure 3-3 (continued): Boxes of five layouts and two textures in two environments. (i–j) L and S layout plain boxes in front of buildings. (k–o) I, O, T, L and S layout boxes with Voronoi diagrams in the parking lot. (p) I layout plain boxes in the parking lot.



(q) (r)

(s) (t)

Figure 3-3 (continued): Boxes of five layouts and two textures in two environments. (q–t) O, T, L and S layout plain boxes in the parking lot.



(a) (b)

(c) (d)

Figure 3-4: Camera trajectories of boxes of five layouts and two textures in two environments. (a–d) I, O, T and L layout boxes with Voronoi diagrams in front of buildings.

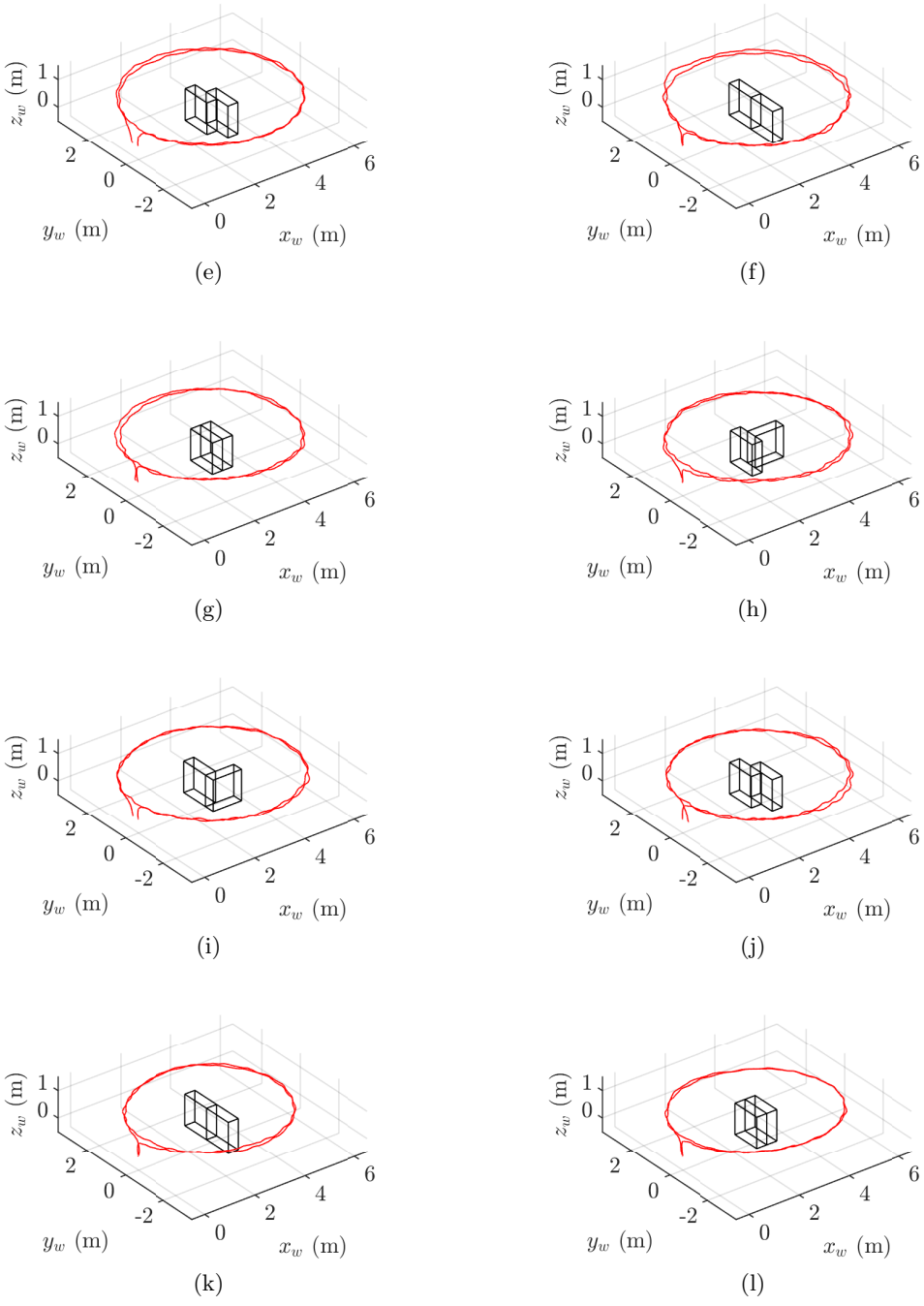


Figure 3-4 (continued): Camera trajectories of boxes of five layouts and two textures in two environments. (e) S layout boxes with Voronoi diagrams in front of buildings. (f–j) I, O, T, L and S layout plain boxes in front of buildings. (k–l) I and O layout boxes with Voronoi diagrams in the parking lot.

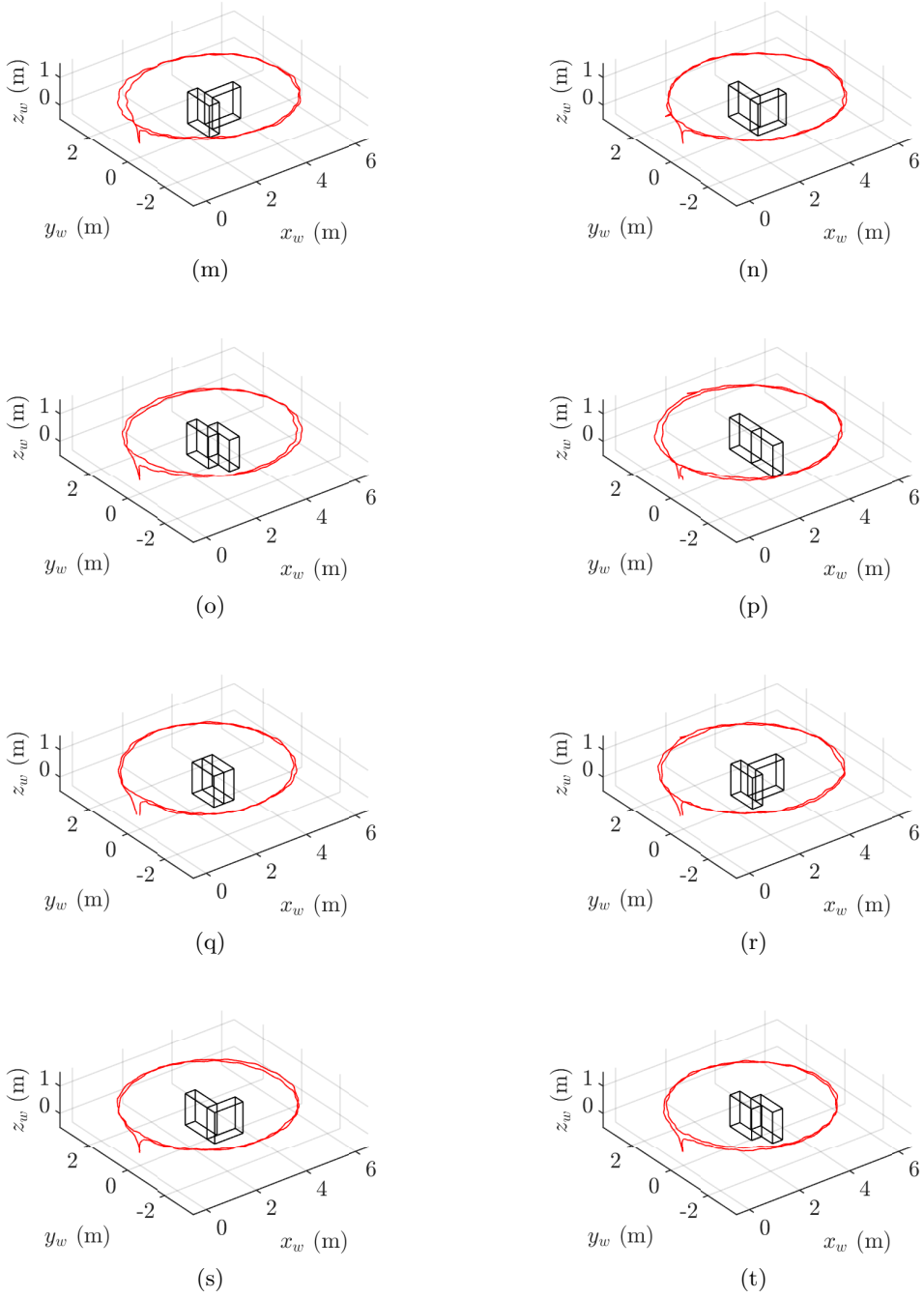


Figure 3-4 (continued): Camera trajectories of boxes of five layouts and two textures in two environments. (m–o) T, L and S layout boxes with Voronoi diagrams in the parking lot. (p–t) I, O, T, L and S layout plain boxes in the parking lot.

3.1.4 Ground Truth

A ground truth is needed as a reference to evaluate the performance of mapping algorithms. The dimensions and locations of the targets in 3D space are required. The dimensions of each box can be easily measured and the locations can be derived by the relative positions between the camera and boxes. Ground truths are generated with following steps.

1. The camera is put into the real environment and its initial position can be derived. The world coordinate system is created on the left camera. The origin point of the world frame is the optical centre of the left camera and the $y_w z_w$ plane is perpendicular to the principal axis of the left camera.

2. In the principal axis of the left camera, the point whose distance to the optical centre is 2.96 m can be located and its projection on the ground is the centroid of the bottom surfaces of two boxes. Boxes are arranged in one of five layouts in Figure 3-2 and their external surfaces towards the camera are parallel to $y_w z_w$ plane. Then the locations of each surface of the boxes can be calculated since the coordinates of the bottom centroid are known.

3. The resolution of the ground truth is set to 0.1 m, which will also be introduced in Section 4.7.1. The nodes containing the external surfaces of boxes can be located in the ground truth and are marked as occupied while free nodes are marked accordingly. Normally, occupied nodes form a shell and the space inside the shell is unknown since the inside is not observable.

If the quality of the data collected by a senor is good enough, the corresponding occupancy map derived by a mapping algorithm should match well with this ground truth. The ground truth settings for each data set is presented in Figure 3-5.

3.2 Summary

Test scenes and targets are introduced in this chapter. Instead of using a large scene with irregularly shaped objects, a relatively small scene to be explored is created with two boxes, the external surfaces of which can be plain or covered with Voronoi diagrams. Five layouts of boxes are created with the free tetrominoes in the Tetris game. Data sets are collected in two environments with a controlled procedure. Through different textures and layouts of boxes, and different environments, different test scenes can be created, which allows the investigation of the impacts of textures and environments on mapping performance. The ground truth for each data set is generated based on

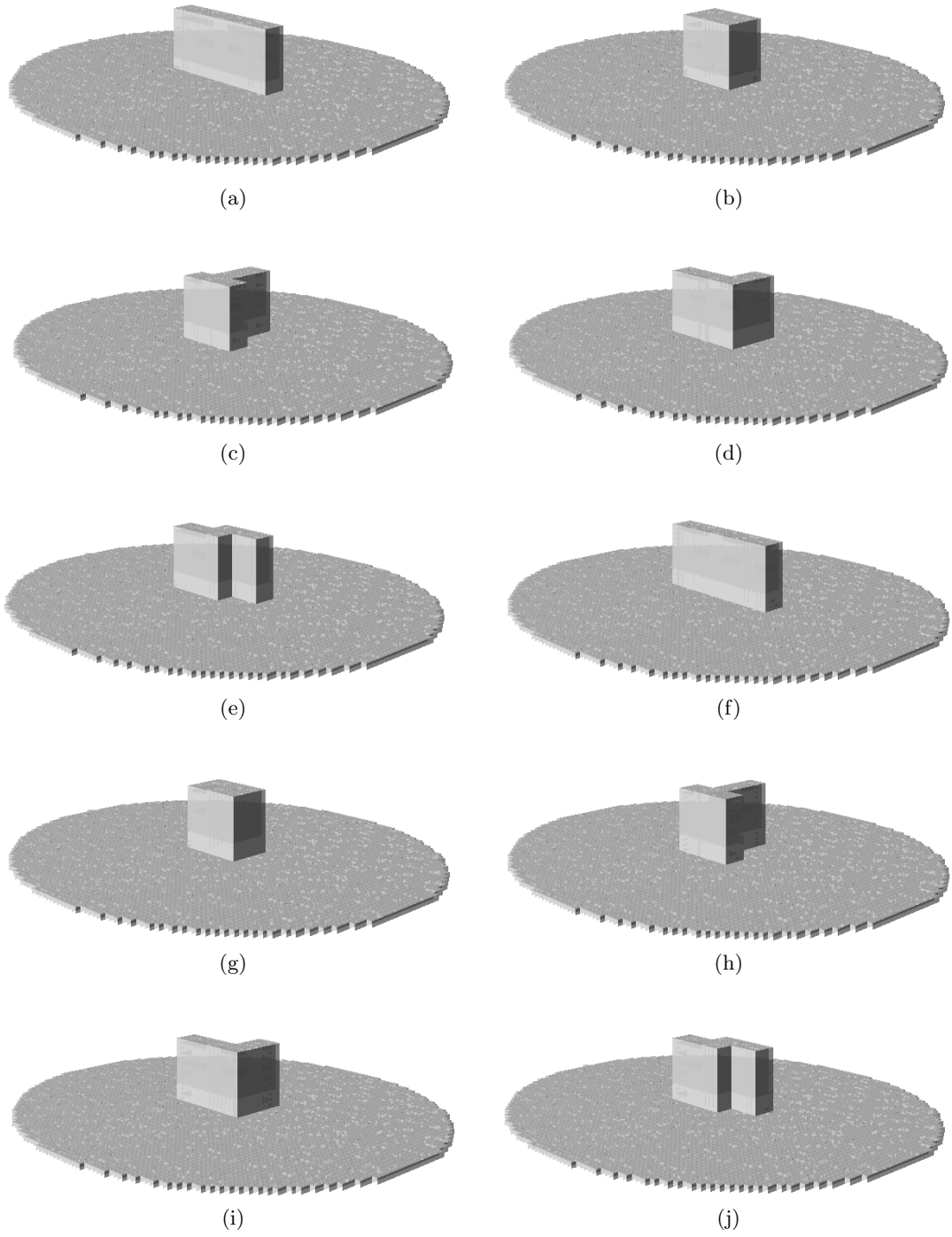


Figure 3-5: Ground truths of boxes of five layouts and two textures in two environments. Free nodes are not presented for clarity. (a–e) I, O, T, L and S layout boxes with Voronoi diagrams in front of buildings. (f–j) I, O, T, L and S layout plain boxes in front of buildings.

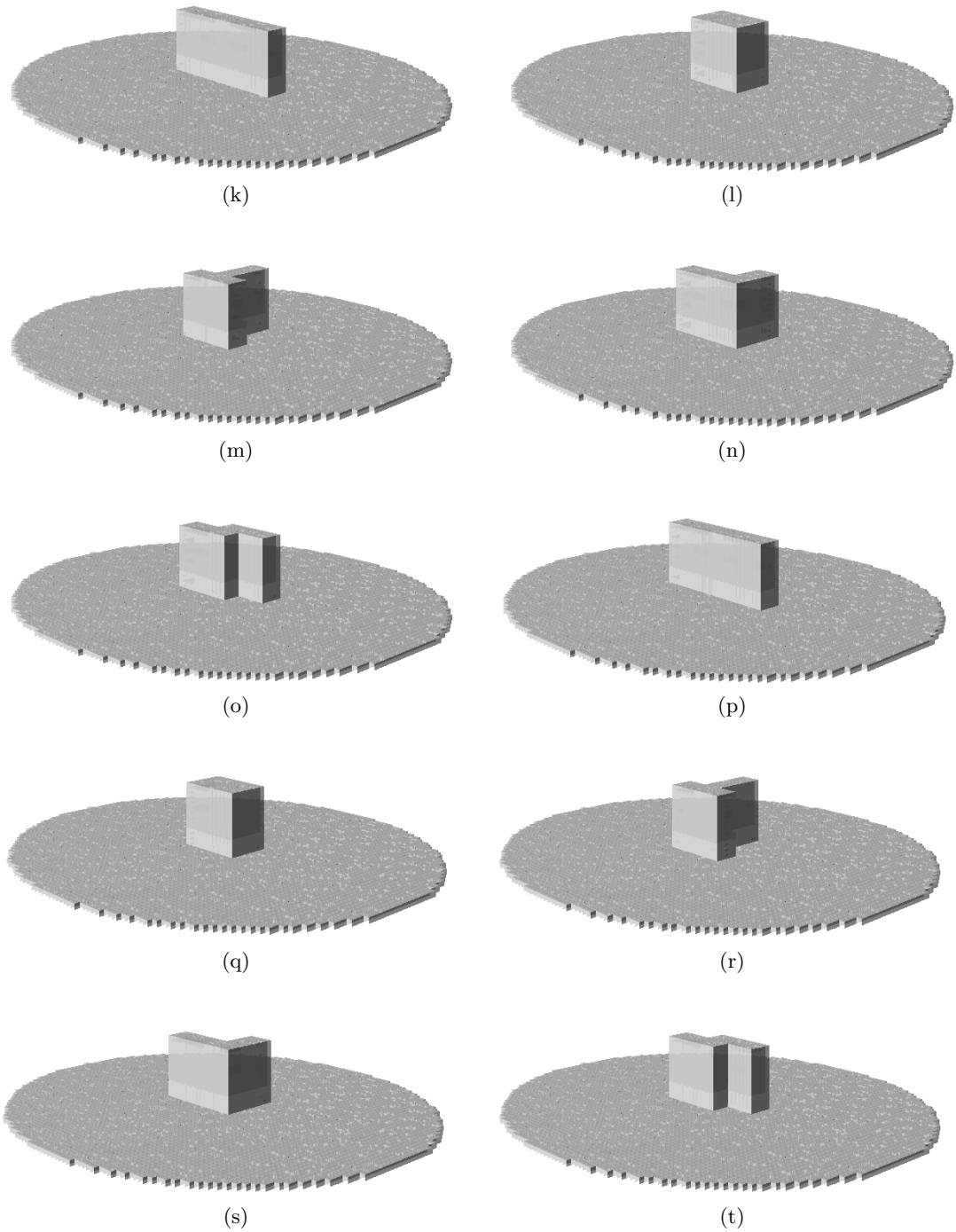


Figure 3-5 (continued): Ground truths of boxes of five layouts and two textures in two environments. (k–o) I, O, T, L and S layout boxes with Voronoi diagrams in the parking lot. (p–t) I, O, T, L and S layout plain boxes in the parking lot.

measurements in the real environment. The benefits of the data sets introduced in this chapter are as follows:

- Targets can be accurately measured and a precise baseline can be set for evaluation purpose in a low-cost way.
- Rich features introduced by Voronoi diagrams can provide reliable results when evaluating the performance of visual-based mapping algorithms.
- Data sets are collected in two different environments with two different textures, allowing to evaluate the effect of the scene to the process.

Based on the above data sets and ground truths, several experiments will be conducted to tune the performance of mapping algorithms in the next chapters. In addition, the performance of different mapping algorithms will also be compared.

Chapter 4

Parameter Reduction and Optimisation

This chapter is based on [101] and presents a systematic method to identify the parameter set for the octree based occupancy mapping to achieve better performance. The framework is using a two-step approach, by fisrt conducting NCA on all parameters and then performing grid search with ROC curves for the most significant ones. To offer a reliable comparison this study is conducted on 20 data sets collected in Chapter 3.

4.1 Parameter Space Considerations

The methodology will be tested on OctoMap. Point clouds are created by disparity maps generated using StereoSGBM [17] in the OpenCV library on the images from a stereo camera. The pose of each frame of the point clouds is produced by ORB-SLAM. Parameters of StereoSGBM, OctoMap and ORB-SLAM will be analysed.

4.1.1 Combinations of Parameters

The parameter space for the analysis will be generated in the following way. Each parameter will be generated by three values, i.e., minimum, maximum and step, as well as the algorithm-required relations with other parameters. The possible values of a parameter T can be denoted as:

$$\left\{ T \;\middle|\; T = T_{\min} + (i-1)T_s,\, i < \left\lfloor \frac{T_{\max} - T_{\min}}{T_s} \right\rfloor + 1,\, i \in \mathbb{N}^+ \right\}, \tag{4.1.1}$$

where $T_{\max}$ and $T_{\min}$ are upper and lower bounds on the parameter, and T_s is the step. A function is defined to denote the step of a parameter:

$$\psi(T) = T_s \,. \tag{4.1.2}$$

Another two functions are defined to describe the combinations of parameters. One function is:

$$g(T) = \left\lfloor \frac{\max(T) - \min(T)}{T_s} \right\rfloor + 1 \,. \tag{4.1.3}$$

The other one is:

$$h(T, T', n) = \min \left[\left\lfloor \frac{\min(T) + (n-1)T_s - \min(T')}{T'_s} \right\rfloor , \, g(T') - 1 \right] + 1 \,, \tag{4.1.4}$$

where T and T' are respective collections of possible values for two parameters, T_s and T'_s are corresponding steps, and n is an input integer.

With the above definitions, the number of all the combinations of the parameters for point cloud generation can be written as:

$$N_p = g(d_{\min})g(d_n)g(B_s)g(P_1)g(P_2)g(d_m)g(C)g(r_u)g(d_w)g(d_v)g(mode) \,. \tag{4.1.5}$$

In addition to the above grid of parameters, by considering upper and lower bounds on the probability, a reasonable set of OctoMap parameters should satisfy:

$$\begin{cases} 1 > p_{\max} \geq p_h \geq 0.5 > p_m \geq p_{\min} > 0 \\ p_{\max} \geq p_t \geq p_{\min} \end{cases} . \tag{4.1.6}$$

Here $p_{\max} = 1$ and $p_{\min} = 0$ are avoided in case the log-odds probability is not a number. From the above relationship, it can be inferred that:

$$\begin{cases} \min(p_{\max}) \geq \min(p_h) > \min(p_m) \geq \min(p_{\min}) \\ \min(p_{\max}) \geq \min(p_t) \geq \min(p_{\min}) \end{cases} . \tag{4.1.7}$$

Then the number of the combinations of OctoMap parameters is:

$$N_o = \sum_{i=1}^{g(p_{\max})} \sum_{j=1}^{g(p_m)} \sum_{q=1}^{h(p_m, p_{\min}, j)} g(p_t)h(p_{\max}, p_h, i) \,. \tag{4.1.8}$$

where the possible values of p_t correspond to $p_{\max}$ value: $\min(p_{\max}) + (i-1)\psi(p_{\max})$ and $p_{\min}$ value: $\min(p_{\min}) + (q-1)\psi(p_{\min})$.

The number of the combinations of ORB parameters is:

$$N_b = g(N_f)g(s_f)g(N_l)g(t_i)g(t_{\min})\,. \tag{4.1.9}$$

Parameters will be combined and tested on the different data sets introduced in Chapter 3. Let N_d denote the number of data sets, a random permutation of the indices of all the combinations will be generated and divided into N_d groups. The number of combinations for each data set is:

$$N_t = \frac{N_p N_o N_b}{N_d}\,. \tag{4.1.10}$$

4.2 Performance Metrics

Binary classification refers to tasks categorising elements into two classes and has a wide range of applications, including bankruptcy prediction [108], disease diagnosis [109], airlines delays prediction [110] and quality assurance [111]. Metrics used to evaluate the performance of a binary classifier depend on scenarios. For example, sensitivity and specificity normally have a wide range of usage in medicine, while recall, precision and f-factor are commonly used in machine learning [112, 113].

An occupancy map can be generated with a series of point clouds. For any node in the map, if the endpoints of any point cloud appear inside the node or the rays cast from the sensor to the endpoints of any point cloud traverse the node, the volumetric space inside the node is explored and the state of this node can either be occupied or free; otherwise, the node is not explored and its state is unknown since no observations from the sensor are available to determine the state of this node. The unknown state means the data derived by the sensor have no impacts on the node and the state of the node has never been determined by the mapping algorithm. So it is reasonable to exclude the unknown nodes when judging the quality of an occupancy map since for these nodes the mapping algorithm is never implemented to update their probabilities. Take an simple case for example, without any point clouds, a blank map can be created and the states of all the nodes in the map are unknown. It is not reasonable to compare the map with the ground truth to evaluate the performance of a mapping algorithm since the algorithm is never implemented on these nodes. By excluding the nodes of unknown states, OctoMap can be treated as a binary classifier since it classifies the

nodes in an occupancy map into occupied and free ones. Therefore, the performance measures for binary classifiers can be used to evaluate the mapping performance of the OctoMap algorithm.

4.2.1 Confusion Matrix

In machine learning, a confusion matrix, also called a contingency table, is a table layout to describe the performance of a classification algorithm [39, 114]. As shown in Table 4.1, a 2×2 confusion matrix summarises the number of true positives (TPs), false positives (FPs), true negatives (TNs) and false negatives (FNs). Positive or negative refers to the category a classifier assigns an instance to, while true or false refers to whether the classification is correct or not.

Table 4.1: Confusion matrix.

		True condition	
	Total population	Condition positive	Condition negative
Predicted condition	Predicted condition positive	True positive (TP)	False positive (FP)
	Predicted condition negative	False negative (FN)	True negative (TN)

Based on the outcomes in the table of confusion, a number of criteria can be defined to analyse the performance of a binary classifier. In this thesis, true positive rate (TPR), false positive rate (FPR) and false discovery rate (FDR) will be discussed. The definitions of these metrics are:

$$\mathrm{TPR} = \frac{N_{TP}}{N_{TP} + N_{FN}}\,, \tag{4.2.1}$$

$$\mathrm{FPR} = \frac{N_{FP}}{N_{FP} + N_{TN}}\,, \tag{4.2.2}$$

and

$$\mathrm{FDR} = \frac{N_{FP}}{N_{FP} + N_{TP}}\,, \tag{4.2.3}$$

where N_{TP}, N_{FP}, N_{TN} and N_{FN} are the numbers of TP, FP, TN and FN elements.

4.2.2 Receiver Operating Characteristic

A ROC curve, created by plotting TPR against FPR, is commonly used to demonstrate the diagnostic ability of a binary classifier when its threshold is varied [115]. The ROC curve is the summary of a confusion matrix at different thresholds. As shown in

Figure 4-1, defined by FPR and TPR as x and y axes, the ROC space illustrates the trade-off between TPR and FPR. As a metric for how good a binary classifier is, the area under the curve (AUC) refers to the area defined by an ROC curve, x axis and line $x = 1$. The perfect classification is on the top left corner. The diagonal line from the origin to the top right corner represents the random guess. The space above the diagonal line means the classification is better than random guess and the space below the line represents worse results than random guess. An ROC curve is normally above the line. The point produced by the classification will move on the ROC curve when the threshold is changed.

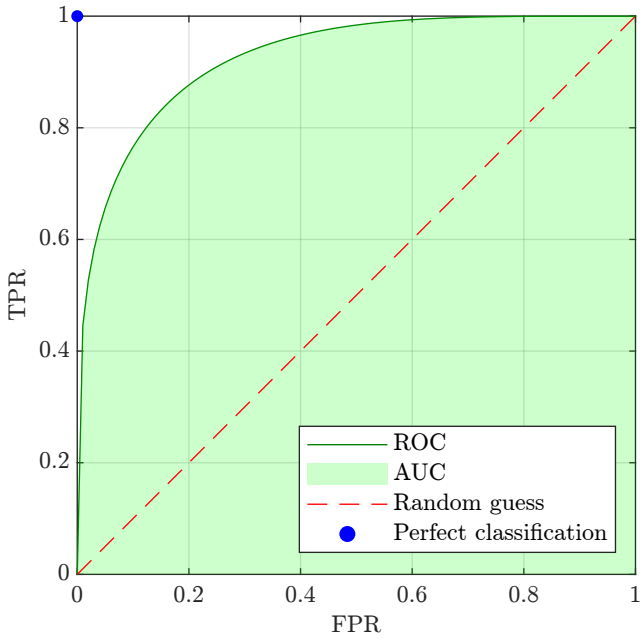


Figure 4-1: Receiver Operating Characteristic (ROC) space.

4.2.3 ROC Surface

The traditional AUC-ROC method is not effective in dealing with unbalanced data sets in which elements in one class are more than in others. For example in empty scenes, most elements in a map are classified as TNs, which will distort the ROC curve since only a small portion of the curve is relevant to the real test. To solve the class-skewed problem, researchers have come up with several approaches focusing on the sub-regions of the ROC curve, e.g., magnifying part of the curve [116] and computing part of AUC [117], or using another variant of ROC by replacing FPR with FDR [118, 119]. In [120], ROC surface (ROCS) is proposed for evaluating the classifier performance to address the issue in the AUC-ROC method when the data is class-skewed by TNs. It

combines the ROC method and its TPR-FDR variant. Three metrics, i.e., TPR, FPR and the true discovery rate (TDR), are used to generate a three-dimensional surface by projecting FPR-TPR-TDR curve to TPR-TDR plane. TDR can be derived by one minus FDR. Figure 4-2 shows the construction procedure of ROCS.

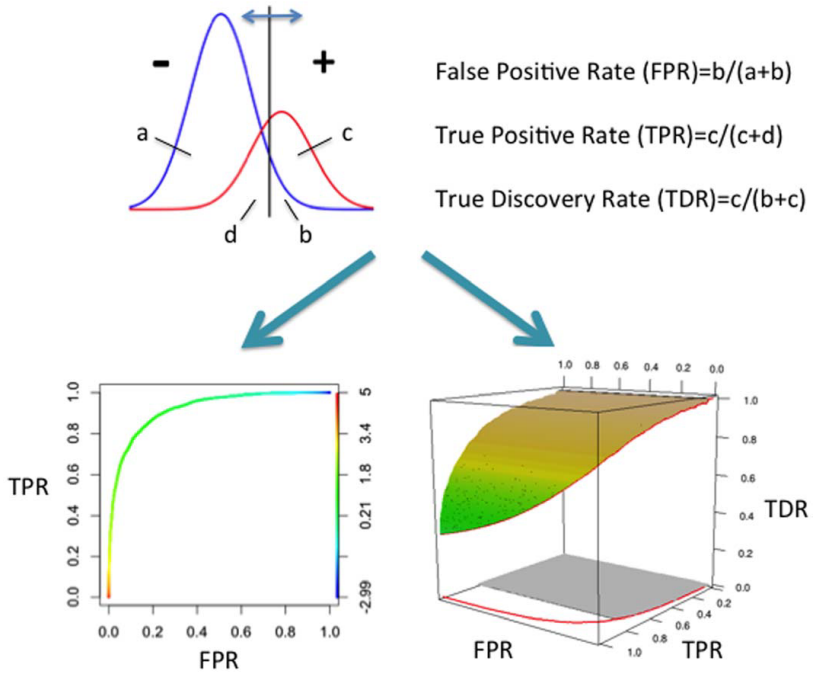


Figure 4-2: Construction of the Receiver Operating Characteristic (ROC) curve and the ROC surface (ROCS) [120].

In the ROCS method, AUC is replaced by the volume under the surface (VUS). VUS is defined as the volume between the ROCS and its projection on the FPR-TPR plane:

$$\mathrm{VUS} = \int_{-\infty}^{\infty} (1 - \mathrm{FPR}(\chi))\, \mathrm{TDR}(\chi)\; d\mathrm{TPR}(\chi)\;, \tag{4.2.4}$$

where χ is the threshold value. FDR-Controlled AUC (FCAUC) is defined the AUC under the original ROC curve limited by a pre-defined FDR range. When the cutoff of FDR is b, the corresponding minimum value of χ can be found by:

$$a = \arg\min_{\chi} (\mathrm{TPR}(\chi))\,,\; s.t.\, \mathrm{FDR}(\chi) \leq b\,. \tag{4.2.5}$$

Then FCAUC can be defined as:

$$\mathrm{FCAUC} = \int_{a}^{\infty} (1 - \mathrm{FPR}(\chi))\; d\mathrm{TPR}(\chi)\;. \tag{4.2.6}$$

When the TNs greatly outnumber TPs, the ROCS plot, and VUS and FCAUC values provide alternative methods to measure the performance of a classifier.

4.2.4 Choice of Performance Metrics

A map built from a date set collected in outdoor environments is heavily class-skewed by TNs since there is more free space than occupied space. In this case, the metrics in ROCS can be used for judging the performance of a mapping algorithm. However, the bottom surface is effectively defined by the traditional ROC curve, of which a small portion is relevant to real tests when TNs greatly outnumber other categories. With a reasonable set of point clouds and mapping parameters, the AUC of the curve in the bottom surface is approximately equal to 1 since the FPR of most points derived by real tests on the curve is approximately 1 and varies in a very small range. In addition, TDR can be derived by FDR. So FPR can be ignored and TDR can be replaced with FDR to reduce the metrics from three dimensions to two dimensions. In this thesis, the metrics in ROC variant, the TPR-FDR curve, will be used to evaluate mapping performance.

4.3 Node Classification

To compute performance metrics, an occupancy map will be compared with ground truth to classify the nodes in the map into four categories, i.e, TPs, FPs, TNs and FNs. This section introduces the procedure for the assessment against ground truth. The assumptions made for the evaluation are also introduced.

4.3.1 Assumptions for Evaluation

In this thesis, the evaluation of the quality of an occupancy map is based on the following assumptions:

- *The camera poses derived by ORB-SLAM are accurate.* As introduced in Section 2.2.2, BA is implemented to reduce projection errors in both local mapping and loop closing. In Section 3.1.3, data sets are collected by the camera orbiting around the targets twice. The cumulative error in the pose can be minimised when the loop closure is detected. It is assumed that ORB-SLAM can produce accurate camera poses and errors caused by ORB-SLAM can be ignored. The impact of ORB-SLAM parameters might be reduced since the optimisation by ORB-SLAM may minimise the possible difference in poses caused by different parameters.

- *The calibration for the camera and the re-projection model are ideal.* The ZED camera used in the experiments has been factory calibrated and the camera parameters are used for 3D reconstruction with a stereo camera model. The shift of a re-projected point on the ray cast from the camera can be ignored compared with the resolution of an occupancy map since the camera has been well calibrated and the images are rectified by camera parameters. With this assumption, the impacts of the camera parameters on the quality of an occupancy map are ignored.

- *The depth map derived by the camera is accurate.* The StereoSGBM algorithm in OpenCV is implemented for the derivation of the depth map of left and right images. Since OpenCV is a highly optimised library, the depth map obtained by the algorithm in it is considered to be reliable. The change in the parameters of the StereoSGBM algorithm might result in less significant difference in the depth map than other less optimised depth generation methods.

4.3.2 Pixel Connectivity

To deal with the fluctuation of points and identify real FPs, the concept of pixel connectivity [37] in image processing is introduced to the node classification procedure. In 3D space, for a specific pixel, 8-connected pixels are the neighbours that touch one of the surfaces of this pixel. In addition to 8-connected pixels, 18-connected pixels also touch one of the edges of the specified pixel. Besides 18-connected pixels, 26-connected pixels are the neighbours also touch one of the corners of the specified pixel. Figure 4-3 shows the example of neighbourhood pixels-association with 26 pixels.

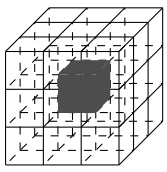


Figure 4-3: Association with 26 pixels.

For a node whose centre coordinates are (x, y, z), the corresponding coordinates of the connected pixels are presented in Table 4.2. s_l denotes the side length of a cube.

4.3.3 Classification Procedure

In an occupancy map, 3D space is evenly divided into cubic volumes of which the states can be occupied, free or unknown. An occupied node indicates that it has a high occupancy probability above the occupancy threshold, while a free node is with a low

Table 4.2: Coordinates of connected pixels.

Connectivity			Coordinates
26-connected	18-connected	8-connected	$(x \pm s_l, y, z)$
			$(x, y \pm s_l, z)$
			$(x, y, z \pm s_l)$
			$(x \pm s_l, y \pm s_l, z)$
			$(x \pm s_l, y \mp s_l, z)$
			$(x \pm s_l, y, z \pm s_l)$
			$(x \pm s_l, y, z \mp s_l)$
			$(x, y \pm s_l, z \pm s_l)$
			$(x, y \pm s_l, z \mp s_l)$
			$(x \pm s_l, y \pm s_l, z \pm s_l)$
			$(x \pm s_l, y \pm s_l, z \mp s_l)$
			$(x \pm s_l, y \mp s_l, z \pm s_l)$
			$(x \mp s_l, y \pm s_l, z \pm s_l)$

probability below the threshold. The unknown state of an node means the node has not been explored yet. If the unknown nodes in an occupancy map are not considered, the map itself can be treated as a binary test. In this test, "positive" refers to "occupied" and "negative" means "free". The following terms in the statistical method can be redefined to describe the nodes in a map:

- TP: Occupied nodes correctly identified as occupied.
- FP: Free nodes incorrectly identified as occupied.
- TN: Free nodes correctly identified as free.
- FN: Occupied nodes incorrectly identified as free.

For each node in an occupancy map, the coordinates of the node centroid are used to find the corresponding node in the ground truth. The minimum and maximum values of the bounding box of all occupied and free nodes in the map will be first computed. The order of querying nodes will affect the classification results; however, there is no obvious difference in the overall trend. The classification will be performed if the node exists. In real experiments, the points produced by the StereoSGBM algorithm will fluctuate near the external surfaces of objects and points may appear inside objects. If the state of a node in a map is known but the corresponding node in the ground truth is unknown, this node will be ignored in the classification procedure. Following this rule, in this work a node will be ignored if the corresponding node in the ground truth

is inside the boxes. The reasons for the ignorance of one such node are as follows:

- *The same observation may lead to completely different classification results.* Since the space inside the objects are invisible, the observation from the outside is always the same no matter the inside is full or empty. If the node is considered, it can be categorised to different classes when the observation is the same but the states inside the objects are different.
- *The nodes appearing inside the objects normally do not affect the use of an occupancy map.* If a node in the map appears inside the objects in the ground truth, it is usually behind the nodes containing the external surface of the objects. When a map is used for navigation or path planning, the inside nodes are not visible and thus has no impacts on the application.

TN and FN nodes For a free node in an map, the ground truth will be queried with the coordinates of the node centre. If the corresponding node in the ground truth is free, the node in the map will be marked as TN. Otherwise, it will be marked as FN. With the assumptions in Section 4.3.1, most points in point clouds should be in ideal positions. Since free nodes are determined by the rays cast from the sensor to end points, most free nodes will belong to TNs.

TP and FP nodes 26-connected pixels introduced in Section 4.3.2 will be used to identify TP and FP instances. For an occupied node in the occupancy map, the coordinates of the node centre and its 26-connected nodes can be obtained following Table 4.2. The ground truth will be queried by the node centre coordinates first and then the coordinates of the neighbourhood nodes. The order to query the neighbours can be found in algorithm 1. If the corresponding node in the ground truth is occupied, the occupied node in the generated map is successfully associated with this node and the query process stops. The node in the map will be marked as TP, while the node in the ground truth will be marked as associated and it cannot be associated with other occupied nodes in the generated map. On the contrary, an occupied node in the generated map will be marked as FP if the ground truth has been queried by corresponding 27 coordinates but none of the nodes can be associated. With the assumptions in Section 4.3.1, most points will appear on the external surfaces of the objects. Since occupied nodes are determined by the locations of endpoints, most occupied nodes will be categorised as TPs.

Figure 4-4 is a 2D example of node classification. The classification can be extended to 3D space by considering the third axis. On the left side is the occupancy map to

be evaluated and on the right side is the ground truth. The nodes in the map will be classified into different categories one by one by querying the states of the nodes in the ground truth. The node marked with dashed square in the map is the current one to be categorised. With the coordinates of this node, the corresponding node and its connected neighbours can be found in the ground truth and are surrounded by the dashed rectangular-shaped area. Then the explored node in the map can be classified into one of four categories following the query order in pixel connectivity and the classification procedure, which has been previously introduced. More details on the classification operation are presented in algorithm 1.

With the above classification methods in this section, algorithm 1 shows the process of classification of nodes in an occupancy map in this thesis. m and m_r denote the map and the ground truth, respectively. s_l is the resolution of the occupancy map or the side length of the cube in the octree, and m_r' is a copy of m_r. Lines 3 to 4 get the bounds of the bounding box of a map. Lines 11 through 15 identify TP and FP nodes with the aforesaid approach. Once a node in m_r' is associated using pixel connectivity, line 12 will delete it. Lines 17 to 20 categorise free nodes into TN and FN ones.

4.4 Framework for Parameter Reduction and Optimisation

The method is achieved by the reduction of the parameter space defined in Section 4.1 using NCA and then a grid search of the OctoMap parameter space. The respective steps are presented in this section.

4.4.1 Parameter Reduction

Point cloud parameters, OctoMap parameters and ORB parameters are needed to generate occupancy maps from the original images. Normally, there are a number of parameters and these parameters affect to different degrees the performance of a mapping algorithm. The analysis becomes complicated when a number of parameters are taken into account. Therefore, the parameters of lower impacts on the performance metrics can be neglected to reduce the complexity of analysis.

Feature selection is implemented to exclude parameters which are less relevant to the performance metrics. Feature selection, also known as variable selection, will select a set of relevant features from given features to construct a regression model [121]. In machine learning or pattern recognition, a feature is defined as an observation of

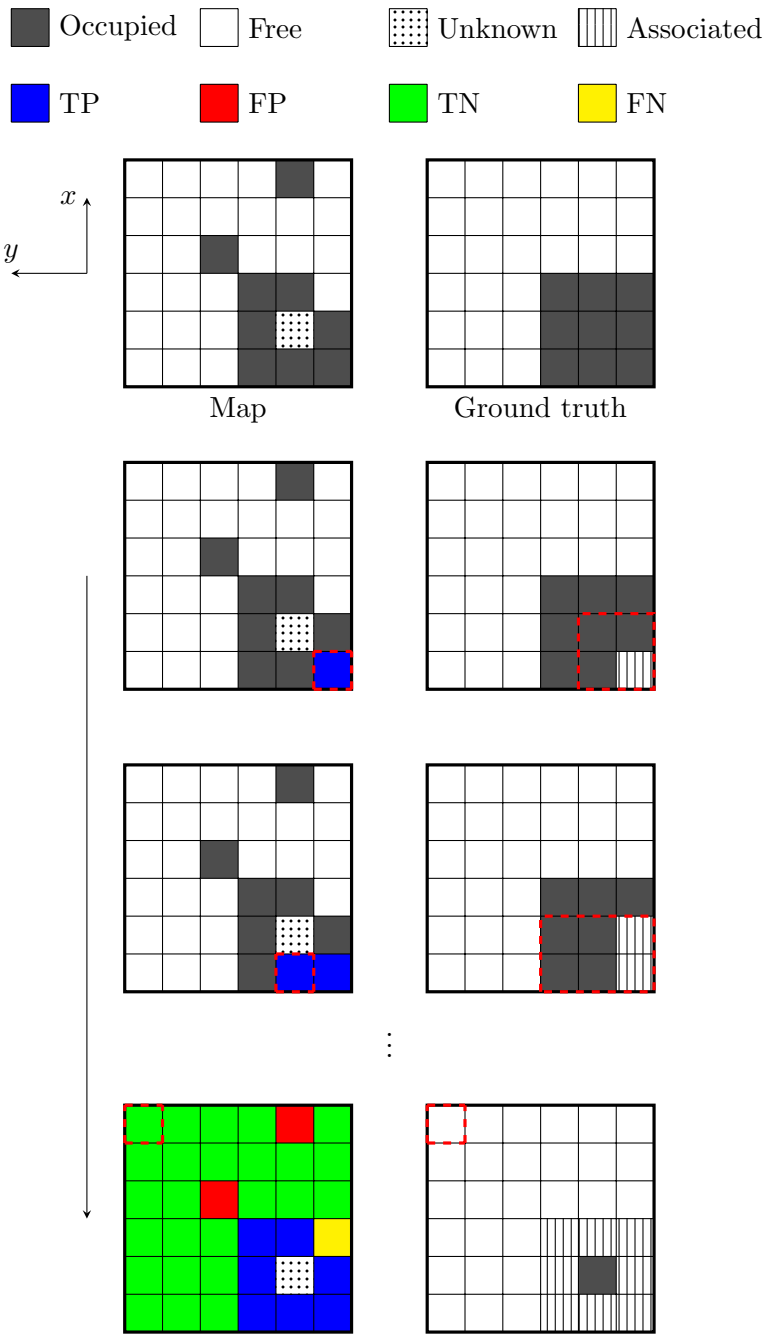


Figure 4-4: 2D example of node classification.

Algorithm 1: Classification of nodes

Input: m, m_r, s_l
Output: N_{TP},N_{FP}, N_{TN}, N_{FN}

N_{TP}, N_{FP}, N_{TN}, $N_{FN} \leftarrow 0$
$m_r' \leftarrow CopyGroundTruth(m_r)$
$x_{\min}$, $y_{\min}$, $z_{\min} \leftarrow GetMetricMin(m)$
$x_{\max}$, $y_{\max}$, $z_{\max} \leftarrow GetMetricMax(m)$
for $z \leftarrow z_{\max} - \frac{s_l}{2}$ **to** $z_{\min} - \frac{s_l}{2}$ **step** $-s_l$ **do**
  for $x \leftarrow x_{\max} - \frac{s_l}{2}$ **to** $x_{\min} - \frac{s_l}{2}$ **step** $-s_l$ **do**
    for $y \leftarrow y_{\max} - \frac{s_l}{2}$ **to** $y_{\min} - \frac{s_l}{2}$ **step** $-s_l$ **do**
      $m_i \leftarrow m.SearchNode(x, y, z)$
      $m_j \leftarrow m_r.SearchNode(x, y, z)$
      if $m_i \neq \varnothing$ **and** $m_j \neq \varnothing$ **then**
        if $m.IsNodeOccupied(m_i)$ **then**
          if $IsNodeAssociated(m_r', x, y, z)$ **then**
            $N_{TP} \leftarrow N_{TP} + 1$
          else
            $N_{FP} \leftarrow N_{FP} + 1$
        else
          if $m_r.IsNodeOccupied(m_j)$ **then**
            $N_{FN} \leftarrow N_{FN} + 1$
          else
            $N_{TN} \leftarrow N_{TN} + 1$

individual measurable property or characteristic of a phenomenon [122]. In this chapter, each parameter can be treated as a feature or variable.

With each combination of point cloud, OctoMap and ORB parameters, an occupancy map can be generated. The corresponding TPR and FDR are computed by the number of nodes in the four categories, i.e., TPs, FPs, TNs and FNs. The weight of each parameter under two metrics can be used for judging which parameters are more important. The derivation of parameter weights from parameters and metrics is a regression problem. NCA feature selection introduced in [36] is a non-parametric method for selecting features with the goal of maximising prediction accuracy of regression or classification, and thus can be used to learn parameter weights with parameters as predictors and performance metrics as responses. The results are then normalised as per [36] to compare the results derived by different data sets.

4.4.2 OctoMap Parameter Optimisation

To evaluate the mapping approach performance, a consistent evaluation of the quality of the point cloud used is needed. Given different combinations of point cloud parameters will give different quality of point clouds, a naive mapping policy is implemented here as a proxy to point cloud quality.

The naive mapping policy is a non-parametric approach. A node will be marked as occupied if it contains points, while a node will be marked as free if it contains no points and is traversed by rays cast from the sensor to endpoints. Once a node is marked as occupied, it cannot be converted to a free node. This is a simple and naive way to generate an occupancy map, which is not affected by parameters. In this update policy, the states of all potentially occupied nodes are guaranteed to be occupied. As a result, the FDR of the generated map is the proxy for the cleanness of the point cloud set. The higher the FDR is, the more nodes have been incorrectly identified as occupied due to noise points being in empty space.

Point cloud sets produced by different parameters from each data set are ranked by the FDR derived by the naive mapping approach and then five of those point cloud sets are selected. A series of combinations of OctoMap parameters will be generated and implemented on the selected point cloud sets. The AUC of TPR-FDR variant is used to optimise parameters. The best AUC derived by grid search will be compared with that derived by default parameters as suggested in [11]. In this work, the data sets introduced in Chapter 3 will be randomly divided into two groups for training and test purposes. The parameter set derived by training through searching grid parameter space will be validated on test data sets.

4.5 Relationship Between OctoMap Parameters and Performance Metrics

Two methods, i.e., linear regression and neural networks, are used to investigate the relations between OctoMap parameters and performance metrics. Since in Section 4.8 the relationship can be well fitted by a single-layer neural network, a linear relationship might exist between parameters and metrics. Therefore, a linear model is also tested here.

4.5.1 Fitting Relationship with Multiple Linear Regression

A linear relationship might exist between OctoMap parameters and performance metrics. A multiple linear regression (MLR) [123] is implemented to fit the relation between parameters and metrics. For any outcome y, the linear model of predictor variables can be written as:

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n + \epsilon, \tag{4.5.1}$$

where β_0, β_1, ..., β_n are coefficient estimates, x_1, x_2, ..., x_n are input variables and n equals to the number of OctoMap parameters, and ϵ is the error term.

4.5.2 Learning Relationship with Neural Networks

The shallow neural network (SNN) [124] is also implemented to fit the relationship between input parameters and output performance metrics. Figure 4-5 shows the structure of neural networks. The neural network has three layers, i.e., the input layer, the hidden layer and the output layer. Each layer consists of several neurons. The input layer is the initial input data and the number of neurons is the same as that of inputs. The network takes the inputs, does the computation and produces outputs.

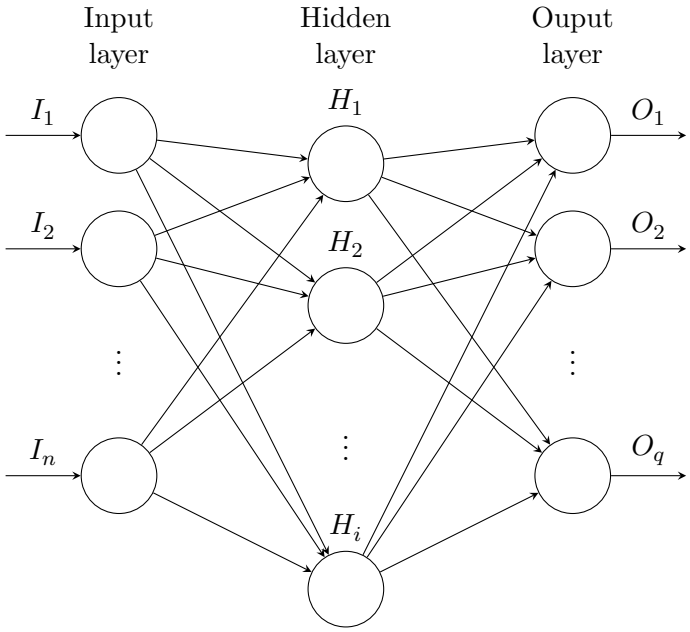


Figure 4-5: Structure of neural networks.

Figure 4-6 shows how a neuron does the computation. The neurons of the hidden and output layers assign significance to their corresponding inputs by combining weights and bias with input data. The weights will determine how important each input element is.

The sum of these products are put into the activation function for further calculation.

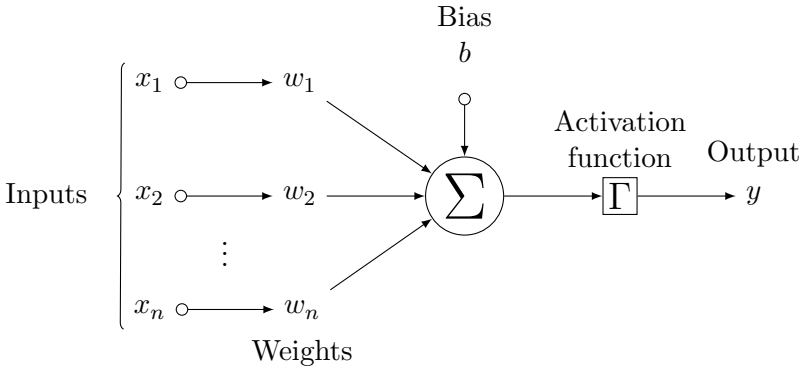


Figure 4-6: Structure of a neuron.

For each neuron, the input can be denoted as:

$$\mathbf{X} = \begin{bmatrix} x_1 & x_2 & \dots & x_n \end{bmatrix}^\top, \tag{4.5.2}$$

where n equals to the number of input elements. The weights are:

$$\mathbf{W} = \begin{bmatrix} w_1 & w_2 & \dots & w_n \end{bmatrix}. \tag{4.5.3}$$

With an activation function $\Gamma(x)$, the output follows:

$$y = \Gamma(\mathbf{W}\mathbf{X} + b). \tag{4.5.4}$$

The activation functions for hidden and output layers are sigmoid and identity functions, respectively. The activation function can be denoted as:

$$\Gamma(x) = \begin{cases} \dfrac{1}{1+\exp(-x)} & \text{hiddern layer} \\ x & \text{output layer} \end{cases}. \tag{4.5.5}$$

The aforesaid procedure is called forward propagation which moves forward through the neural network from inputs until the activation of a neuron is derived. Apart from this, the backpropagation is also required to calculate the error attributable to each neuron and tweak the weights and biases. Bayesian regularisation [125] is used for the bakpropagation to train the neural network.

4.5.3 Pearson Correlation Coefficient and Mean Squared Error

Pearson Correlation Coefficient (PCC) [126] and Mean Squared Error (MSE) [127] are used to evaluate the fitting results derived by MLR and SNN.

Correlation statistics depicts the strength of the relationship between two variables [128]. The range of a correlation coefficient normally varies between -1 and 1. The sign of a correlation coefficient represents whether the correlation is positive or negative. The absolute value is a measurement of the strength of a relationship. For example, correlations of 1 and -1 mean perfect positive and negative correlations, respectively. While 0 means there is no linear relationship between two variables. For two variables x and y, PCC is given as follows:

$$R = \frac{\text{cov}(x, y)}{\sigma_x \sigma_y}, \tag{4.5.6}$$

where $\text{cov}(x, y)$ is the covariance, and σ_x and σ_y are the standard deviations of x and y, respectively. The covariance is defined as:

$$\text{cov}(x, y) = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - E(x))(y_i - E(y)), \tag{4.5.7}$$

where n is the number of elements and $E(x)$ is the expectation of x.

MSE measures the average of the squares of errors and is defined as:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2, \tag{4.5.8}$$

where y and $\hat{y}$ are observed values and predicted values.

4.6 Time Estimation Model

For OctoMap mapping, the time elapsed for a single run is proportional to the processed point number, so the run time for mapping can be defined as:

$$\Phi_o(N) = aN + b, \tag{4.6.1}$$

where a and b are coefficients, and N is the number of points in the point cloud set used for generating an occupancy map.

4.7 Experimental Method

4.7.1 Design of Experiments

Parameter weights will be analysed to reduce the parameters of lower impacts and the residual most important parameters will be optimised to improve the mapping performance. An exhaustive test on point cloud and OctoMap parameters will be first performed to study the impact of each parameter. In this case, ORB parameters are of default values. Then point cloud parameters will be fixed, and mapping parameters and ORB parameters will be combined to conduct experiments on different data sets. Figure 4-7 shows the workflow for experiments.

To generate an occupancy map, point clouds and corresponding poses are required. In the experiments, ORB-SLAM [33] is implemented to generate the poses of the camera. A series of keyframes is produced by ORB-SLAM and with the time stamps of those keyframes the images can match with their respective keyframe pose. The StereoSGBM algorithm is implemented on the images of keyframes to produce disparity maps, from which point clouds can be reconstructed. Points will be preserved if their distances are within 8 m from the camera in the principal axis.

PCL has been developed for processing point clouds [8]. PCL provides functional modules and has a wide range of applications, including point cloud registration [129], object recognition and pose estimation [130], segmentation [131], and mapping [132]. In this section, point clouds are down sampled by the VoxelGrid filter in PCL at resolution 0.1 m before they are processed by OctoMap, to reduce computational time. 3D space will be first divided by 3D voxel grids or cubes at a given resolution. Then the points in a single voxel grid will be approximated by their centroid.

The leaf size in OctoMap is also set to 0.1 m. The resolution is chosen based on the scale of test scenes and the size of targets. If the leaf size is too big, the details on the external surface of an object will be lost. The fluctuation of the points has also been considered. Since point clouds derived by stereo images are not perfect, points will fluctuate near their real positions. This leaf size can tolerate the fluctuation. Moreover, the resolution is constant here since maps of different leaf sizes are not comparable with the total number of nodes being different, even derived by the same point cloud set. Maximum range for how long individual beams are inserted is set to 4 m since the fluctuation of points is not serious within this range.

Occupancy maps derived by different combinations of parameters will be compared with ground truths to classify the nodes into TPs, FPs, TNs and FNs using the method in-

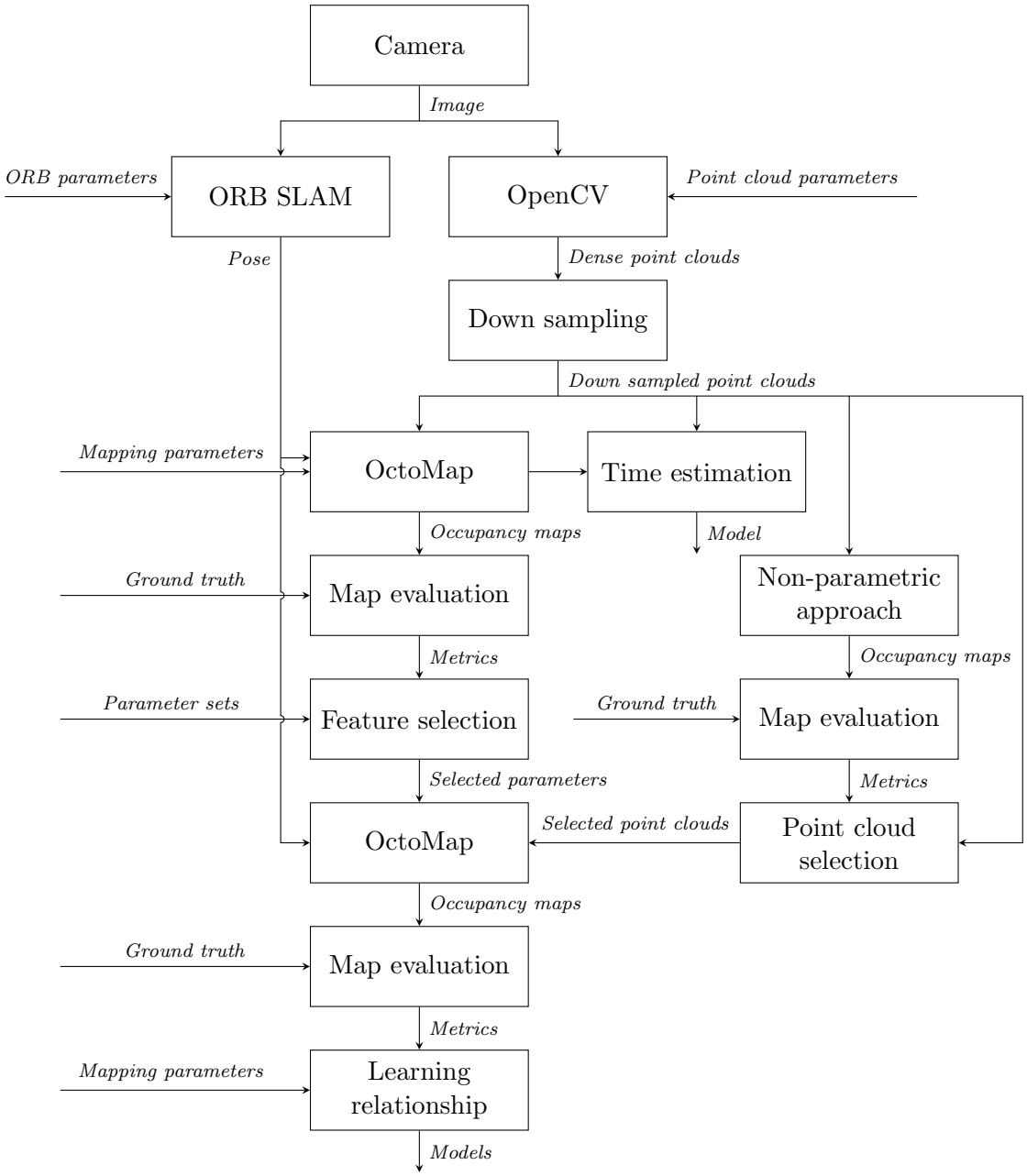


Figure 4-7: Experimental method for parameter reduction and optimisation.

troduced in Section 4.3. The performance metrics derived by the number of the nodes in each category along with different parameter sets will be analysed by NCA feature selection to determine which parameters should be neglected or optimised. The most important ones will be optimised on point cloud sets selected by the non-parametric naive approach explained in Section 4.4. The relationship between mapping parameters and performance metrics will be learned by the neural network as introduced in Section 4.5. The time estimation model proposed in Section 4.6 will be verified as well.

4.7.2 Parameter Space for Analysis

In this section, the minimum, maximum and step values of the parameter space are given for point cloud generation, and occupancy map generation, and pose generation.

The configuration of point cloud parameters is shown in Table 4.3. The steps for specified parameters are not required since they are either of constant values or determined by other parameters. Specifically, $d_{\min}$, d_n and *mode* are constants. $d_{\min} = 0$ since the camera is well calibrated at the factory. d_n controls visible depth and has no impacts on the quality of the disparity map and is set to constant to make point clouds comparable when the other parameters are varied. *mode* is set as true to improve the quality of disparity maps. P_1 and P_2 are determined by image channel number N_c and parameter B_s. d_w is up to 1000 following recommended setting for relatively large object targets. As a result of those minimum, maximum and steps the number of combinations of point cloud parameters is 1600.

Table 4.3: Configuration of point cloud parameters.

Parameter	Minimum	Maximum	Step
$d_{\min}$	0	0	N/A
d_n	80	80	N/A
B_s	3	15	4
P_1	$8N_cB_s^2$	$8N_cB_s^2$	N/A
P_2	$32N_cB_s^2$	$32N_cB_s^2$	N/A
d_m	0	1	1
C	10	50	10
r_u	5	35	10
d_w	200	1000	200
d_v	1	2	1
mode	true	true	N/A

Table 4.4 shows the configuration for the OctoMap parameter space. For parameter p_t from $p_{\min}$ to $p_{\max}$ two steps are investigated, a step of 0.12 for the reduction analysis

and a step of $(p_{\max} - p_{\min})/8$ for optimisation, with 9 points for generating the ROC curve. If the metric of a point is not a number, that point will be excluded. The step size of 0.12 and 9 points were decided following a limited testing of cases using step size 0.06 and 17 points. No obvious difference was found in results, but there was a significant computational time penalty. As a result, for the experiments reported here, the steps given in Table 4.4 will be used with OctoMap parameter combinations for reduction and optimisation being 1000 and 1350, respectively.

Table 4.4: Configuration of OctoMap parameters.

Parameter	Minimum	Maximum	Step
$p_{\max}$	0.5	0.98	0.12
p_h	0.5	0.98	0.12
p_m	0.02	0.38	0.12
$p_{\min}$	0.02	0.38	0.12
p_t [a]	$p_{\min}$	$p_{\max}$	0.12
p_t [b]	$p_{\min}$	$p_{\max}$	$(p_{\max} - p_{\min})/8$

[a] Configuration for parameter reduction.
[b] Configuration for parameter optimisation.

ORB parameters are varied in reasonable ranges to study their impacts. The configuration of ORB parameters is presented in Table 4.5. The considerations for parameter settings are as follows. The number of features in each image should not be too small to make sure the features in two images can be matched. Considering the number of pyramid levels, 1.3 as maximum for s_f is relatively large and appropriate since the image size will be divided by $s_f^{N_l}$ at N_l-th level. N_l and t_i are varied based on the default values, and $t_{\min}$ is further decreased. The number of the combinations of ORB parameters is 243.

Table 4.5: Configuration of ORB parameters.

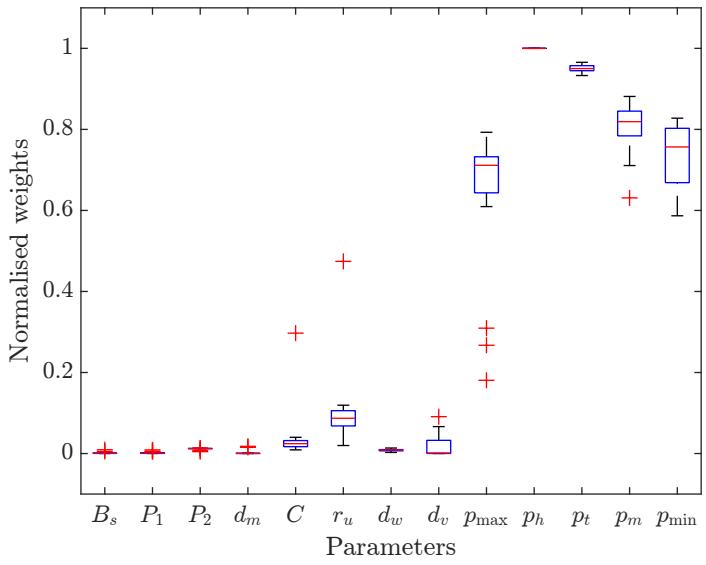
Parameter	Minimum	Maximum	Step
N_f	1500	2000	500
s_f	1.1	1.3	0.1
N_l	7	9	1
t_i	11	13	1
$t_{\min}$	5	7	1

4.8 Results and Analysis

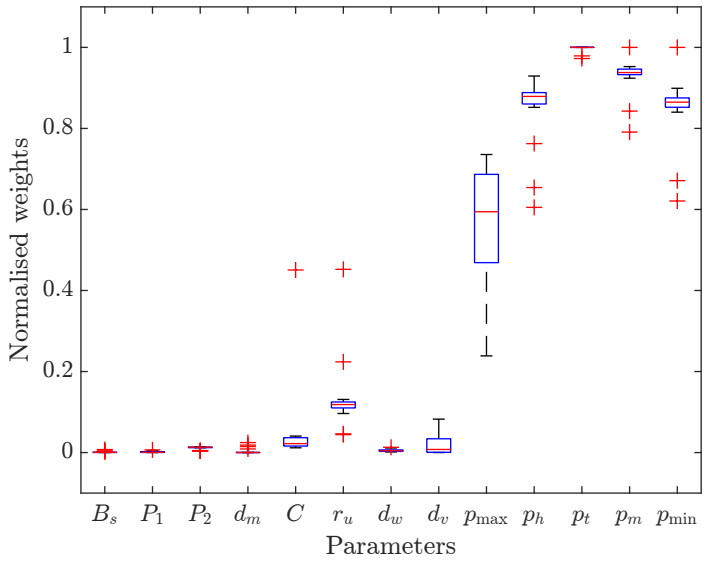
The first set of results is the weight of each of the point cloud and OctoMap parameters on performance metrics TPR and FDR. Here ORB parameters remain default values. With the configurations in Tables 4.3 and 4.4, there are 80 000 combinations of point cloud parameters and OctoMap parameters for each data set. Parameters of constant values are excluded from this analysis. Parameter weights are calculated by implementing the NCA method on node classification results derived by the 20 data sets. Figure 4-8 shows the normalised weights of different parameters. The last five ones are OctoMap parameters, and they show higher weights under both performance metrics. For TPR, the majority of OctoMap parameter weights are over 0.6 while most point cloud parameter weights are under 0.2. FDR is similar to the case TPR with the weights of OctoMap parameters and point cloud parameters mostly being above 0.5 and under 0.2, respectively.

Since OctoMap parameters have higher impacts on performance metrics, point cloud parameters are then fixed to study the weights of OctoMap parameters and ORB parameters. Point cloud parameters correspond to the 800th point cloud set in Voronoi box of I layout in front of buildings when the ORB parameters are of default values. Keyframes produced by ORB-SLAM are normally different when ORB-SLAM is run multiple times, even with the same data set and same parameters. To make results comparable, point clouds of the keyframes generated by default ORB parameters from each data set are used for map generation. When ORB parameters are different, the poses corresponding to these keysframes can be derived by camera trajectories through matching keyframe time steps. With the settings of OctoMap parameters in Table 4.4 and ORB parameters in Table 4.5, the number of combinations of parameters for each data set is 12 150. The weight of each parameter is shown in Figure 4-9. OctoMap parameters show higher weights than ORB parameters in both performance measures with mostly being above 0.5. While most ORB parameter weights are under 0.1, and s_f shows the highest weight among these parameters, but its weights are mostly under 0.4.

Then 20 data sets are split into training and test groups, with 70% randomly selected for optimisation and the other 30% for validation. Since OctoMap parameters have higher weights than both point cloud and ORB parameters, optimisation will only be performed on OctoMap parameters with the training data sets. Here the ORB parameters are of default values. For each data set, 1600 sets of point clouds can be generated with the parameter configuration in Table 4.3. Point cloud sets are

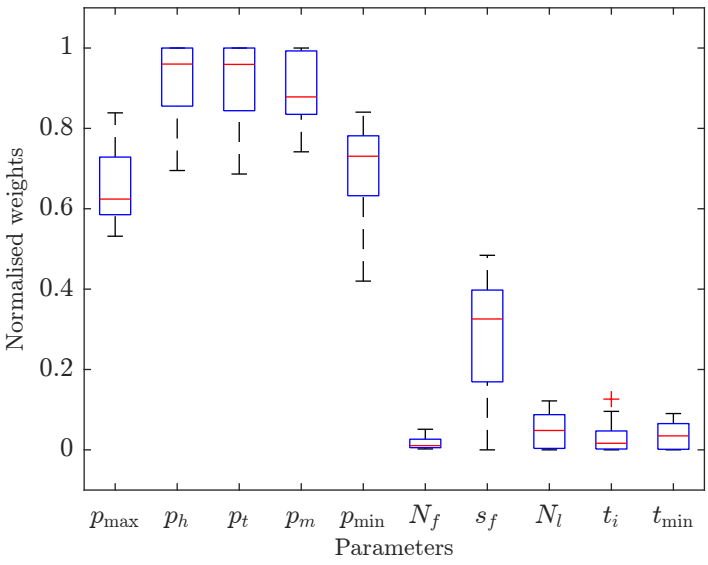


(a) True positive rate (TPR).

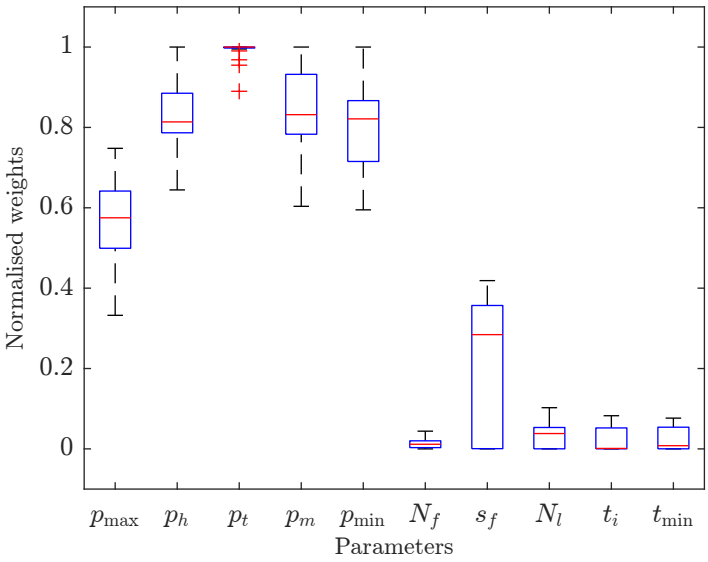


(b) False discovery rate (FDR).

Figure 4-8: Normalised weights of point cloud and OctoMap parameters for performance metrics. (a) True positive rate (TPR). (b) False discovery rate (FDR).



(a) True positive rate (TPR).



(b) False discovery rate (FDR).

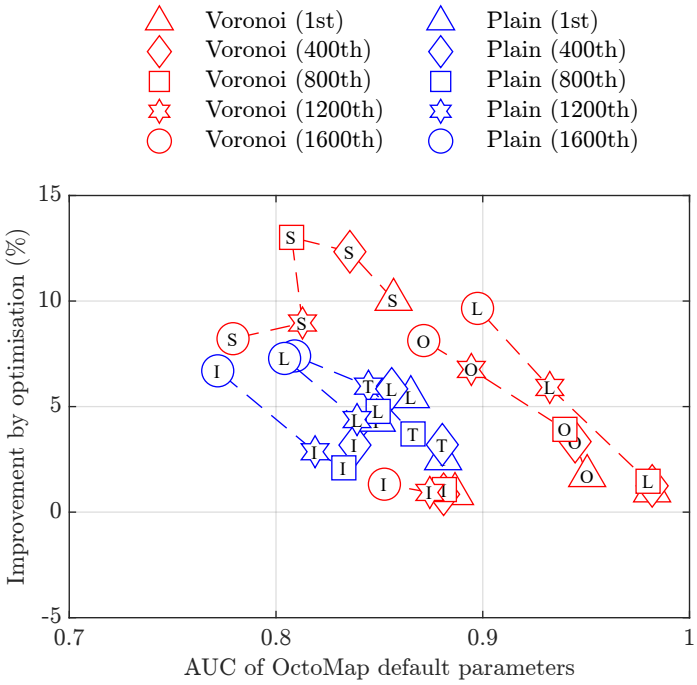
Figure 4-9: Normalised weights of OctoMap and ORB parameters for performance metrics. (a) True positive rate (TPR). (b) False discovery rate (FDR).

ranked by the simple non-parametric mapping approach from Section 4.4.2. The 1st, 400th, 800th, 1200th and 1600th (a lower number indicates better quality, i.e., cleaner point clouds) ranked point cloud sets from each data set are selected to perform the optimisation of OctoMap parameters, of which the grid parameter space is generated by Table 4.4 and the relations in Section 4.7.2. The AUC of TPR-FDR curve specified in Section 4.4.2 is used as the performance measure for optimisation. The results of optimisation against OctoMap default parameters [11] on the training data sets are presented in Figure 4-10. By optimisation, improvements can be achieved on all cases in the two different environments, with highest improvement over default parameters of up to 15%. Overall, the AUC derived by default parameters using the building data set (Figure 4-10a) is better than that using the parking lot data set (Figure 4-10b). The quantitative results corresponding to Figure 4-10 are presented in Table 4.6.

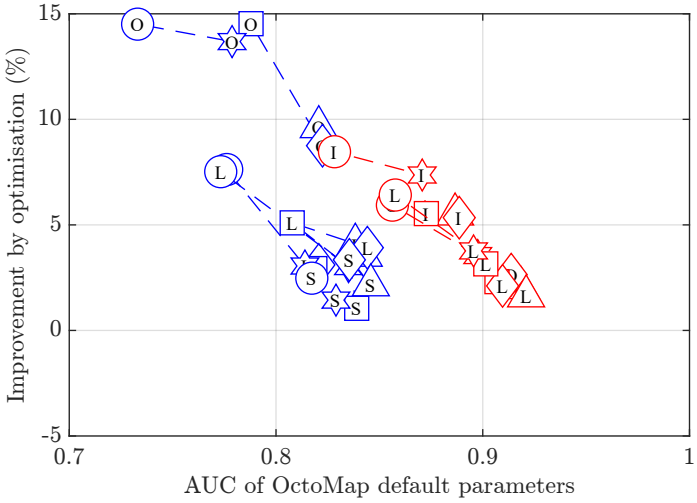
Overall, the improvement by optimisation over default parameters increases when the AUC of default parameters decreases. The mapping performance in the environment with buildings is better than that in the parking lot since there are more objects to provide image features. The performance also benefits from the rich features introduced by Voronoi diagrams. The baseline AUC generated by OctoMap default parameters in two environments is normally better when boxes are covered with Voronoi diagrams. There is not an obvious trend in the five tetromino layouts of boxes. However, a higher improvement can normally be achieved when the quality of the point clouds degrades in each data set.

The frequency of the parameter values which achieve best performance in the training data sets through searching grid parameter space has also been analysed. The parameters derived by gird search are divided into five groups according to the point cloud set ranking. The frequency of the values of each parameter derived by grid search is presented in Figure 4-11. There is not an obvious change in $p_{\max}$ and p_h with the ranking (quality) of point cloud sets. Overall, these two parameters are dominated by 0.98 and 0.62, respectively. For p_m and $p_{\min}$, the most frequent parameters tend to be smaller as the quality of point clouds becomes worse. The frequency of smaller values is higher than larger ones, especially when the point cloud quality degrades.

The results are cross-validated using the test data sets and the findings from the parameter frequency analysis in Figure 4-11. Figure 4-12 shows the validation on testing data sets with the most frequent values of $p_{\max}$, p_h, p_m and $p_{\min}$ derived by training at 0.98, 0.62, 0.14 and 0.02, respectively. The improvement increases to 9% as AUC derived by default parameters decreases, but can be negative when AUC is already relatively large. The details corresponding to Figure 4-12 are presented in Table 4-7.



(a) Building.



(b) Parking lot.

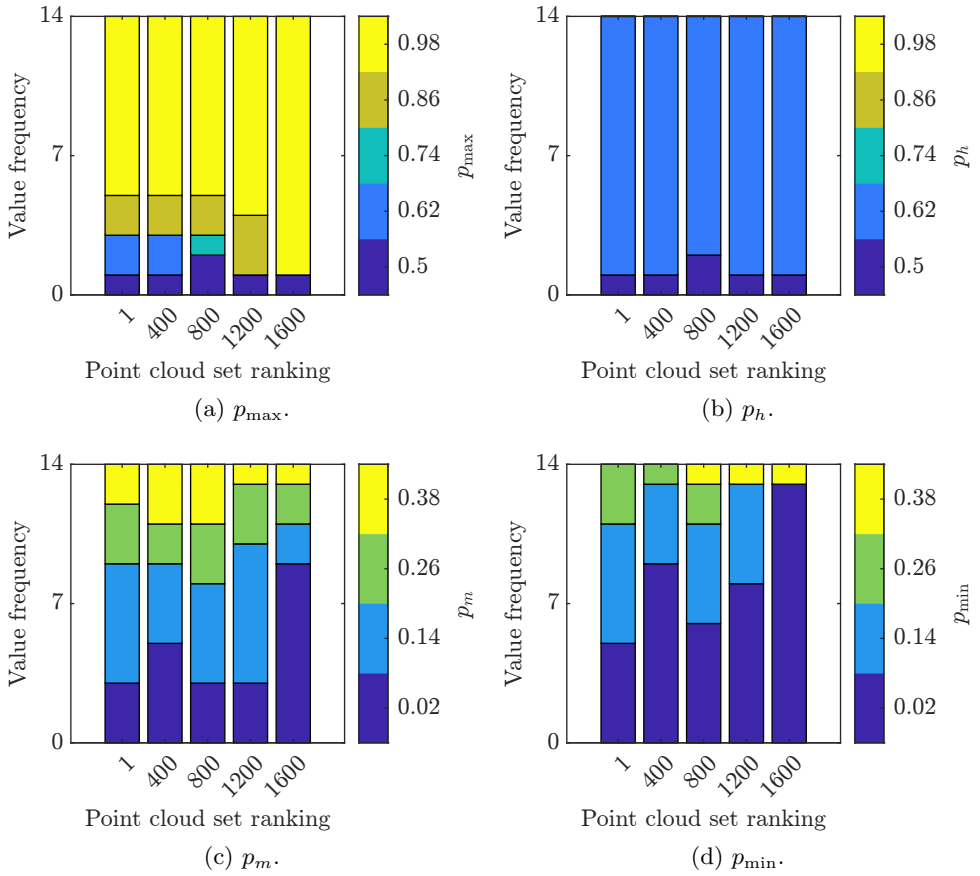
Figure 4-10: Improvement by grid search over the area under the curve (AUC) of OctoMap default parameters on training data sets. (a) Building. (b) Parking lot.

Table 4.6: Improvement over OctoMap default parameters on training data sets.

Environment	Texture	Layout	Point Cloud Ranking	AUC		Improvement	
				Default parameters	Grid search	AUC	%
Building	Plain	I	1	0.8487	0.8856	0.0368	4.34
Building	Plain	I	400	0.8382	0.8648	0.0266	3.18
Building	Plain	I	800	0.8328	0.8503	0.0175	2.10
Building	Plain	I	1200	0.8189	0.8423	0.0234	2.86
Building	Plain	I	1600	0.7718	0.8235	0.0517	6.69
Building	Plain	T	1	0.8808	0.9030	0.0222	2.51
Building	Plain	T	400	0.8804	0.9086	0.0281	3.19
Building	Plain	T	800	0.8663	0.8984	0.0321	3.70
Building	Plain	T	1200	0.8447	0.8951	0.0504	5.97
Building	Plain	T	1600	0.8090	0.8689	0.0599	7.40
Building	Plain	L	1	0.8652	0.9124	0.0472	5.46
Building	Plain	L	400	0.8560	0.9059	0.0500	5.84
Building	Plain	L	800	0.8494	0.8900	0.0405	4.77
Building	Plain	L	1200	0.8392	0.8760	0.0368	4.38
Building	Plain	L	1600	0.8042	0.8627	0.0585	7.28
Building	Voronoi	I	1	0.8866	0.8942	0.0077	0.86
Building	Voronoi	I	400	0.8811	0.8885	0.0075	0.85
Building	Voronoi	I	800	0.8814	0.8909	0.0094	1.07
Building	Voronoi	I	1200	0.8744	0.8825	0.0082	0.93
Building	Voronoi	I	1600	0.8524	0.8637	0.0113	1.32
Building	Voronoi	O	1	0.9505	0.9668	0.0163	1.72
Building	Voronoi	O	400	0.9447	0.9762	0.0316	3.34
Building	Voronoi	O	800	0.9397	0.9766	0.0369	3.92
Building	Voronoi	O	1200	0.8945	0.9549	0.0605	6.76
Building	Voronoi	O	1600	0.8715	0.9423	0.0708	8.13
Building	Voronoi	L	1	0.9819	0.9917	0.0097	0.99
Building	Voronoi	L	400	0.9819	0.9942	0.0122	1.25
Building	Voronoi	L	800	0.9800	0.9943	0.0143	1.46
Building	Voronoi	L	1200	0.9324	0.9875	0.0551	5.91
Building	Voronoi	L	1600	0.8975	0.9841	0.0866	9.65
Building	Voronoi	S	1	0.8569	0.9431	0.0863	10.07
Building	Voronoi	S	400	0.8357	0.9388	0.1031	12.33
Building	Voronoi	S	800	0.8075	0.9127	0.1051	13.02
Building	Voronoi	S	1200	0.8128	0.8856	0.0728	8.96
Building	Voronoi	S	1600	0.7792	0.8432	0.0640	8.22

Table 4-6 (continued): Improvement over OctoMap default parameters on training data sets.

Environment	Texture	Layout	Point Cloud Ranking	AUC		Improvement	
				Default parameters	Grid search	AUC	%
Parking lot	Plain	I	1	0.8384	0.8725	0.0341	4.07
Parking lot	Plain	I	400	0.8207	0.8462	0.0255	3.11
Parking lot	Plain	I	800	0.8188	0.8428	0.0240	2.93
Parking lot	Plain	I	1200	0.8140	0.8388	0.0248	3.05
Parking lot	Plain	I	1600	0.7761	0.8353	0.0592	7.62
Parking lot	Plain	O	1	0.8207	0.8998	0.0791	9.64
Parking lot	Plain	O	400	0.8225	0.8945	0.0720	8.75
Parking lot	Plain	O	800	0.7879	0.9024	0.1145	14.53
Parking lot	Plain	O	1200	0.7788	0.8853	0.1066	13.68
Parking lot	Plain	O	1600	0.7331	0.8394	0.1064	14.51
Parking lot	Plain	L	1	0.8424	0.8723	0.0299	3.55
Parking lot	Plain	L	400	0.8442	0.8773	0.0331	3.92
Parking lot	Plain	L	800	0.8078	0.8490	0.0412	5.10
Parking lot	Plain	L	1200	0.8351	0.8603	0.0253	3.02
Parking lot	Plain	L	1600	0.7733	0.8314	0.0581	7.52
Parking lot	Plain	S	1	0.8459	0.8640	0.0182	2.15
Parking lot	Plain	S	400	0.8355	0.8634	0.0279	3.34
Parking lot	Plain	S	800	0.8390	0.8478	0.0088	1.05
Parking lot	Plain	S	1200	0.8290	0.8409	0.0119	1.44
Parking lot	Plain	S	1600	0.8173	0.8375	0.0202	2.47
Parking lot	Voronoi	I	1	0.8866	0.9354	0.0488	5.50
Parking lot	Voronoi	I	400	0.8886	0.9361	0.0475	5.34
Parking lot	Voronoi	I	800	0.8727	0.9210	0.0483	5.54
Parking lot	Voronoi	I	1200	0.8707	0.9348	0.0641	7.36
Parking lot	Voronoi	I	1600	0.8283	0.8984	0.0701	8.46
Parking lot	Voronoi	O	1	0.9185	0.9370	0.0185	2.01
Parking lot	Voronoi	O	400	0.9138	0.9379	0.0242	2.64
Parking lot	Voronoi	O	800	0.9069	0.9278	0.0210	2.31
Parking lot	Voronoi	O	1200	0.8976	0.9294	0.0318	3.55
Parking lot	Voronoi	O	1600	0.8563	0.9071	0.0508	5.94
Parking lot	Voronoi	L	1	0.9210	0.9359	0.0149	1.62
Parking lot	Voronoi	L	400	0.9096	0.9288	0.0192	2.11
Parking lot	Voronoi	L	800	0.9015	0.9299	0.0284	3.15
Parking lot	Voronoi	L	1200	0.8955	0.9293	0.0337	3.77
Parking lot	Voronoi	L	1600	0.8577	0.9127	0.0550	6.42



(a) $p_{\max}$. (b) p_h. (c) p_m. (d) $p_{\min}$.

Figure 4-11: Frequency of the values of OctoMap parameters through grid search against point cloud set ranking on training data sets. (a) $p_{\max}$. (b) p_h. (c) p_m. (d) $p_{\min}$.

Table 4-7: Improvement for cross-validation.

Environment	Texture	Layout	Point Cloud Ranking	AUC		Improvement	
				Default parameters	Grid search	AUC	%
Building	Plain	O	1	0.9017	0.9094	0.0077	0.86
Building	Plain	O	400	0.8849	0.9080	0.0231	2.61
Building	Plain	O	800	0.8910	0.9292	0.0382	4.29
Building	Plain	O	1200	0.8825	0.9273	0.0448	5.08
Building	Plain	O	1600	0.8326	0.9007	0.0682	8.19
Building	Plain	S	1	0.9228	0.8972	-0.0256	-2.77
Building	Plain	S	400	0.9087	0.9082	-0.0005	-0.05
Building	Plain	S	800	0.9068	0.9139	0.0071	0.78
Building	Plain	S	1200	0.8895	0.9292	0.0397	4.46
Building	Plain	S	1600	0.8600	0.9214	0.0613	7.13
Building	Voronoi	T	1	0.9780	0.9532	-0.0248	-2.53
Building	Voronoi	T	400	0.9759	0.9652	-0.0107	-1.10
Building	Voronoi	T	800	0.9683	0.9779	0.0097	1.00
Building	Voronoi	T	1200	0.9313	0.9802	0.0489	5.25
Building	Voronoi	T	1600	0.9059	0.9630	0.0571	6.30
Parking lot	Plain	T	1	0.8261	0.8689	0.0429	5.19
Parking lot	Plain	T	400	0.8139	0.8483	0.0344	4.22
Parking lot	Plain	T	800	0.7833	0.8133	0.0300	3.83
Parking lot	Plain	T	1200	0.7978	0.8426	0.0449	5.62
Parking lot	Plain	T	1600	0.7438	0.8069	0.0630	8.47
Parking lot	Voronoi	T	1	0.9327	0.9345	0.0017	0.19
Parking lot	Voronoi	T	400	0.9252	0.9340	0.0088	0.96
Parking lot	Voronoi	T	800	0.9129	0.9356	0.0226	2.48
Parking lot	Voronoi	T	1200	0.8998	0.9264	0.0266	2.95
Parking lot	Voronoi	T	1600	0.8561	0.9209	0.0648	7.56
Parking lot	Voronoi	S	1	0.9210	0.9064	-0.0146	-1.59
Parking lot	Voronoi	S	400	0.9290	0.9258	-0.0032	-0.34
Parking lot	Voronoi	S	800	0.9145	0.9204	0.0059	0.65
Parking lot	Voronoi	S	1200	0.8757	0.9087	0.0331	3.78
Parking lot	Voronoi	S	1600	0.8676	0.9117	0.0441	5.08

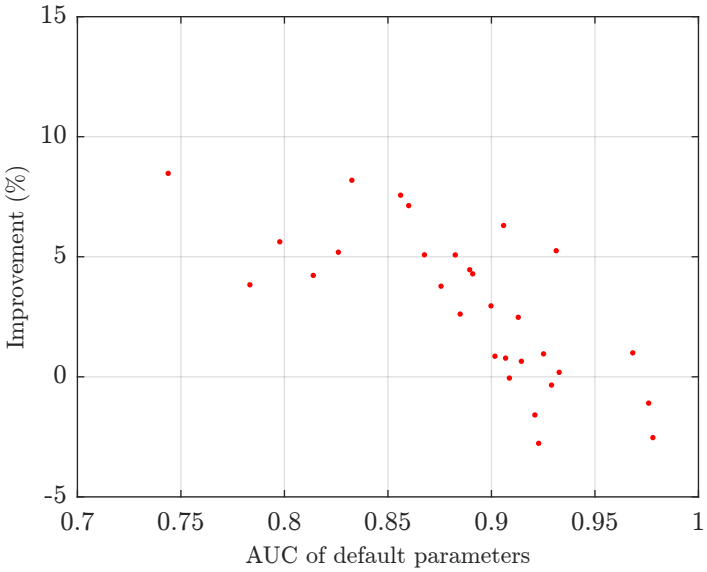


Figure 4-12: Cross-validation on test data sets.

The values of $p_{\max}$ and $p_{\min}$ achieving best performance among all the parameter combinations are similar for different point cloud sets, which reinforces the finding that parameters for point cloud generation are less important. Despite of the high consistency in these two parameters, one parameter set cannot achieve the best improvement in all cases. With a higher baseline AUC, more occupied nodes can be preserved with bigger p_m and $p_{\min}$. These occupied nodes mostly belong to the TP category since point clouds corresponding to higher AUC are usually of less noise, and thus the AUC can be improved. Loss in improvement can be observed when the values of p_m and $p_{\min}$ increase since FPs are likely to be introduced, resulting in a worse FDR. On the contrary, smaller p_m and $p_{\min}$ benefit mapping performance when point clouds are of low quality. On one hand, the probability of a node drops faster with a smaller p_m. On the other hand, the probability can decrease further with a lower $p_{\min}$. Therefore, it would be more difficult for the probability to go beyond the threshold, benefiting FDR.

Then the 800th point cloud set derived by I layout Voronoi boxes in front of buildings when ORB parameters are of default values is used to investigate the relationships between OctoMap parameters and performance metrics. The relations are fitted by MLR and SNN. As specified in Section 4.5.3, PCC and MSE are implemented to evaluate the performance of two regression methods. Figure 4-13 compares the results derived by MLR and SNN. The results show the benefits of SNN, with better PCC and smaller MSE than MLR.

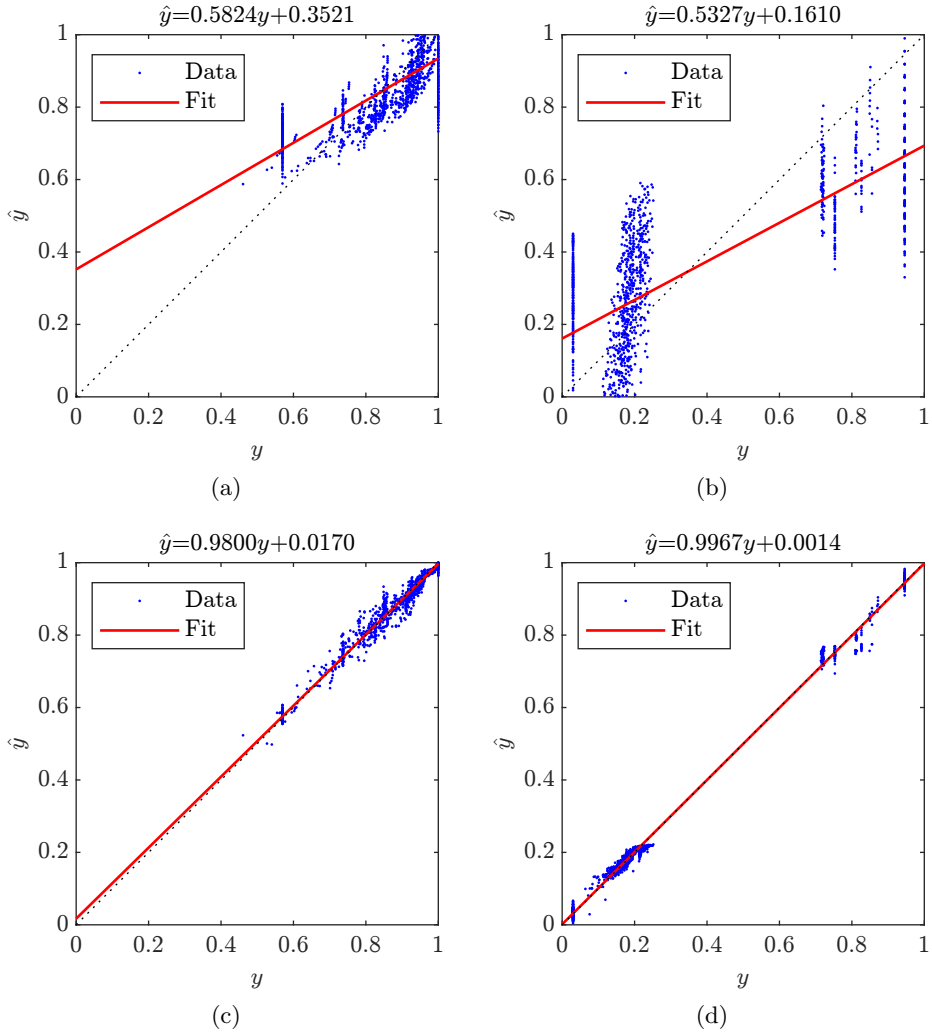


Figure 4-13: Comparison of multiple linear regression (MLR) and shallow neural network (SNN) fittings of true positive rate (TPR) and false discovery rate (FDR). y and $\hat{y}$ denote target values and predicted values, respectively. (a) TPR, MLR, $R = 7.6315 \times 10^{-1}$ and MSE=8.5420×10^{-3}. (b) FDR, MLR, $R = 7.2987 \times 10^{-1}$ and MSE=4.8161×10^{-2}. (c) TPR, SNN, $R = 9.9021 \times 10^{-1}$ and MSE=3.9846×10^{-4}. (d) FDR, SNN, $R = 9.9844 \times 10^{-1}$ and MSE=3.2099×10^{-4}.

The two models are also cross-validated. The settings of data sets have been introduced before, 70% for training and 30% for validation. For each data set, 3% of the results derived by the 1st, 400th, 800th, 1200th and 1600th ranked point cloud sets are randomly selected for cross-validation. Figure 4-14 shows the cross-validation results for MLR and SNN. The results derived by SNN are still better than those derived by MLR, with smaller MSE and better PCC. However, the performance of SNN for cross-validation is not as good as that in Figure 4-13.

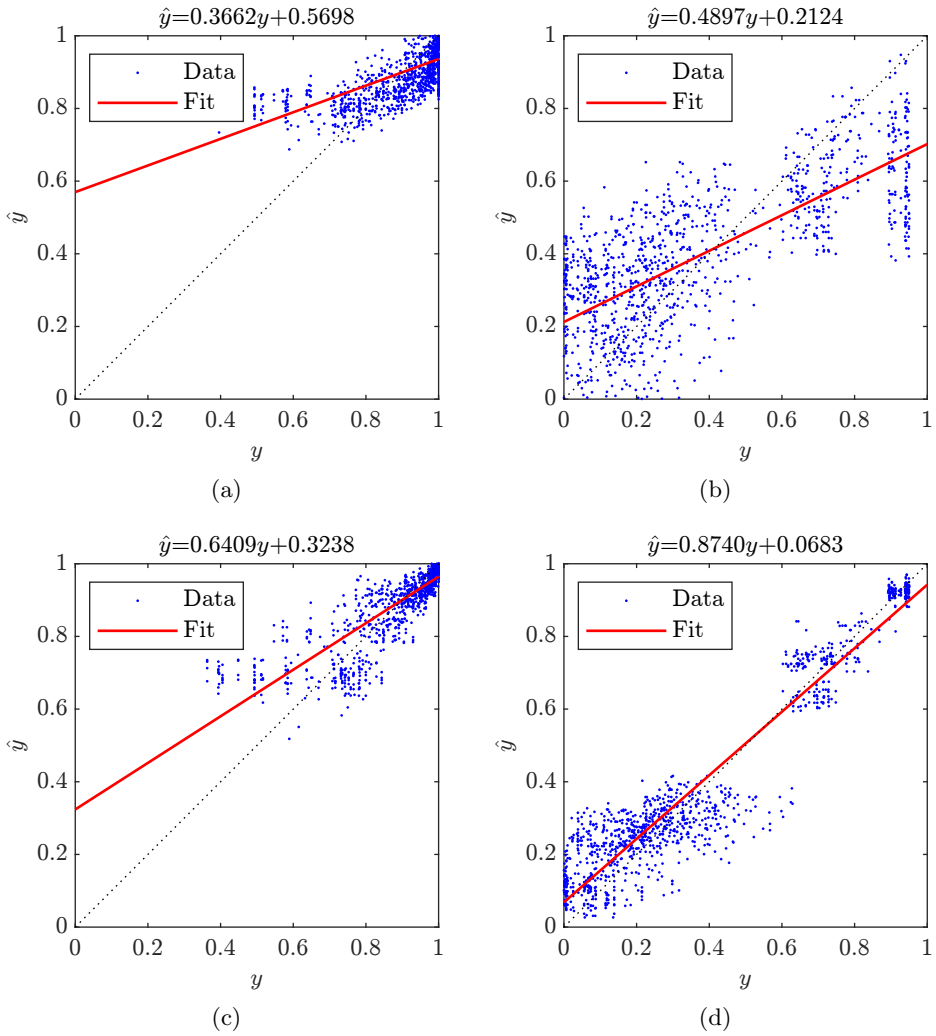


Figure 4-14: Cross-validation for multiple linear regression (MLR) and shallow neural network (SNN) fittings of true positive rate (TPR) and false discovery rate (FDR). y and $\hat{y}$ denote target values and predicted values, respectively. (a) TPR, MLR, $R = 6.3655 \times 10^{-1}$ and MSE=1.2207×10^{-2}. (b) FDR, MLR, $R = 7.2245 \times 10^{-1}$ and MSE=4.3655×10^{-2}. (c) TPR, SNN, $R = 8.4319 \times 10^{-1}$ and MSE=6.0337×10^{-3}. (d) FDR, SNN, $R = 9.5055 \times 10^{-1}$ and MSE=9.2167×10^{-3}.

Finally, the time model in Section 4.6 is verified. Figure 4-15 presents the linear fit for the run time of OctoMap. Corresponding coefficients in (4.6.1) are $a = 2.2711 \times 10^{-5}$ and $b = 1.0567$. 5% of the parameter reduction results of each data set are randomly selected to plot points in Figure 4-15. The result shows that run time is proportional to the number of points processed by the OctoMap algorithm.

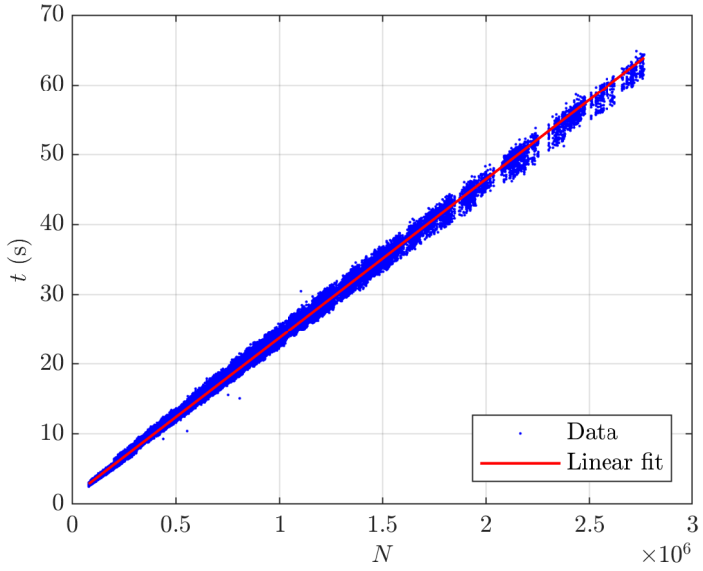


Figure 4-15: Linear regression for the run time of OctoMap.

4.9 Summary

This chapter presents a framework of parameter reduction and optimisation for point cloud generation and occupancy mapping algorithms. The ROC curve variant (TPR-FDR) is used as a performance metric to deal with skewed data in a confusion matrix due to mainly free space in outdoor environments. Pixel connectivity concept in image processing is implemented for node classification to deal with the fluctuation of points. Through NCA, the number of parameters can be reduced and the residual most important parameters can be optimised by investigating the grid parameter space. The proposed method is verified by the implementation of the StereoSGBM algorithm in OpenCV and OctoMap, strengthening the potential of the methodology to be applicable with a variety of systems and sensors. Results show that the proposed approach is effective in reducing parameters and robust in improving mapping performance. In addition, the relations between mapping parameters and performance metrics are investigated by MLR and SNN. The proposed time estimation model is verified as well.

The key findings of this chapter are:

- Compared with point cloud parameters and ORB parameters, mapping parameters have a higher impact on performance metrics TPR and FDR.
- Through grid search optimisation, the performance of OctoMap can be improved over default parameters.

Chapter 5

K-Nearest Neighbours Based Occupancy Mapping

This chapter is based on [133] and [134], and presents a map update policy using the context of neighbouring points. The node in an occupancy map is updated with its inside points, of which the probabilities are determined by corresponding average distances to their k-NN. The parameters of the proposed k-NN method are investigated and optimised, aiming to achieve better performance. OctoMap will be the baseline and compared with the k-NN method.

5.1 K-NN Based Inverse Sensor Model

5.1.1 Inverse Sensor Model

In a point cloud, a point is likely to be noise if it is isolated from the points nearby. Based on this assumption, the average distance from a point to its k-NN can be used to represent the occupancy information of this point. A point should be assigned with a higher probability if it has a smaller average distance, and vice versa. A relationship between the average distance and the change in probability has been defined in the previous work [133]. Here minor changes are made and the relationship is redefined in [134]. Let $F(x)$ denote the distribution of the average distance. Then the probability

representing the occupancy information of a point can be denoted as:

$$p(s) = p_u - \frac{\int_{-\infty}^{s} F(x)dx}{\int_{-\infty}^{\infty} F(x)dx}(p_u - p_l) \coloneqq p_u - G(s)(p_u - p_l)\,, \tag{5.1.1}$$

where s is the average distance from a point to its k-NN, p_u and p_l are the upper and lower bounds on the probability, and $G(s)$ is the CDF of the average distance and $G(s) = \int_{-\infty}^{s} F(x)dx$.

The inverse sensor model based on (5.1.1) can be defined as:

$$l(m_i \mid z_t) = \begin{cases} \sum_j l\left[p(s_j)\right] & \text{if the } j\text{th point in node } m_i \\ l_{in} & \text{if traversed by rays and points within range } s_c \\ l_{out} & \text{if traversed by rays and points outside range } s_c \end{cases}, \tag{5.1.2}$$

where s_c is the maximum range for how long individual beams are inserted, and l_{in} and l_{out} are the respective log-odds values assigned to the nodes traversed by the rays cast from the sensor to the points whose distances to the sensor are within and outside the range s_c. If a node satisfies both the requirements of l_{in} and l_{out}, it will be updated with l_{in} only. Then an occupancy map can be updated with (2.2.33) and (2.2.35).

5.1.2 Distribution of Average Distances

The average distance of a point is computed by searching its k-NN in the corresponding point cloud among points whose distances to the sensor are within range s_c. Different distributions, i.e., Generalised Extreme Value (GEV) distribution, Log-logistic distribution, Rayleigh distribution, Kernel Density Estimation (KDE) and Normal distribution, are used to fit the average distances of all the points within range s_c in a point cloud set generated from one data set. Results are shown in Section 5.4. The CDF of the average distance is nonsensitive to types of distributions. Although KDE can fit the average distance better than other distributions, it would be difficult to change the k-NN model if KDE is applied due to its non-parametric property. Since there is no obvious change in CDF when the distribution is different, the average distance is assumed to be subject to the Normal distribution:

$$F(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right], \tag{5.1.3}$$

where μ is the mean and σ is the standard deviation.

To avoid brute force calculation and improve accuracy, the method for calculating corrected sums of squares and products noted by [135] is implemented, which can reduce rounding errors in computer implementation. As a result, a series of values of mean and standard deviation will be generated as the number of points grows. Let n denote the total number of elements to calculate the mean and the standard deviation. For $i = 1, 2, \ldots, n$, the following process will be performed:

$$\begin{cases} \mu_i = \dfrac{i-1}{i}\mu_{i-1} + \dfrac{1}{i}s_i \\ Q_i = Q_{i-1} + \dfrac{i-1}{i}(s_i - \mu_{i-1})^2 \\ \quad = Q_{i-1} + (s_i - \mu_{i-1})(s_i - \mu_i) \end{cases}, \tag{5.1.4}$$

where s_i is the average distance from the ith point to its k-NN in the corresponding point cloud, μ_{i-1} and μ_i are the mean values for $i-1$ and i points, Q_{i-1} and Q_i are the sums of the squares of the deviations for $i-1$ and i points, and $\mu_0 = 0$ and $Q_0 = 0$. Then the mean and the standard deviation of the Normal distribution can be derived by:

$$\begin{cases} \mu = \mu_n \\ \sigma = \sqrt{\dfrac{Q_n}{n}} \end{cases}. \tag{5.1.5}$$

5.1.3 K-NN Parameter Space Considerations

With the inverse sensor model defined in Section 5.1.1, the parameters of k-NN mapping are as follows:

- $p_{\max}$ is the upper clamping threshold, which is the upper bound on the probability.
- p_t is the threshold. A node will be marked as occupied when the threshold is reached.
- p_m is the probability of a "miss". A node will be updated with p_m if it is traversed by rays and corresponding endpoints are within range s_c.
- p'_m is the probability of a "miss". A node will be updated with p'_m if it is traversed by rays and corresponding endpoints are outside range s_c.
- $p_{\min}$ is the lower clamping threshold, which is the lower bound on the probability.
- p_u is the upper bound on the probability derived by the average distance from a

point to its k-NN.

- p_l is the lower bound on the probability derived by the average distance from a point to its k-NN.
- k is the number of nearest neighbouring points.

The algorithm-required relations with other parameters should also be considered when generating parameters. $p'_m \geq p_m$ since p'_m corresponds to points further to the sensor than p_m. A reasonable set of the parameters of the k-NN method should satisfy:

$$\begin{cases} 1 > p_{\max} \geq 0.5 > p'_m \geq p_m \geq p_{\min} > 0 \\ p_{\max} \geq p_t \geq p_{\min} \\ p_u \geq p_l \end{cases} . \tag{5.1.6}$$

Considering the relations in (5.1.6), the number of the combinations of the parameters in k-NN model is:

$$N_k = g(k) \left(\sum_{i=1}^{g(p_u)} h(p_u, p_l, i) \right) \left(\sum_{i=1}^{g(p_{\max})} \sum_{j=1}^{g(p'_m)} \sum_{q=1}^{h(p'_m, p_m, j)} \sum_{w=1}^{h(p_m, p_{\min}, q)} g(p_t) \right) , \tag{5.1.7}$$

where the possible values of p_t correspond to $p_{\max}$ value: $\min(p_{\max}) + (i-1)\psi(p_{\max})$ and $p_{\min}$ value: $\min(p_{\min}) + (w-1)\psi(p_{\min})$.

The investigation of parameters is presented based on different data sets, the overview of which is given in Section 3.1. Let N_d denote the number of data sets. A random permutation of the indices of all the possible combinations of k-NN parameters is first generated and then divided into N_d groups according to their indices. With the number of combinations of point cloud and ORB parameters, the number of combinations for each data set is:

$$N_t = \frac{N_p N_k N_b}{N_d} . \tag{5.1.8}$$

5.1.4 Analysis of the K-NN Method

When the ray-casting operation is performed, nodes with points inside may be traversed by the rays from the sensor to endpoints. To make the analysis clear, here the points whose corresponding nodes are traversed by any rays are called traversed points. OctoMap updates a node containing traversed points with l_{free} as if there are no points inside. However, such a node in the k-NN method will be updated with both the probability derived by the average distance and the probability assigned to

free nodes. Normally, in an occupancy map, the nodes containing points not traversed by rays normally greatly outnumber the nodes containing traversed points. Therefore, the impact of traversed points is ignored in this section. The update policies of the nodes containing endpoints in OctoMap and the k-NN method will be compared mathematically.

In the k-NN method, when $p_u = p_l$, the probability of a point is constant:

$$p(s) = p_u \,. \tag{5.1.9}$$

A node m_i containing points will be updated with (5.1.2):

$$l(m_i \mid z_t) = N_i \ln\left(\frac{p_u}{1-p_u}\right), \tag{5.1.10}$$

where N_i is the number of points in the node m_i. While OctoMap will update nodes containing endpoints with (2.2.34):

$$l(m_i \mid z_t) = \ln\left(\frac{p_h}{1-p_h}\right). \tag{5.1.11}$$

In OctoMap, it is required that parameter $p_h \geq 0.5$. Let $p_u = p_h$ and then:

$$N_i \ln\left(\frac{p_u}{1-p_u}\right) \geq \ln\left(\frac{p_h}{1-p_h}\right) \geq 0\,. \tag{5.1.12}$$

In this case, the probabilities of the nodes containing points in k-NN model are higher than the probabilities of the nodes containing endpoints in OctoMap.

Let:

$$p_u = \frac{\left(\frac{p_h}{1-p_h}\right)^{-\max(N_i)}}{1+\left(\frac{p_h}{1-p_h}\right)^{-\max(N_i)}}. \tag{5.1.13}$$

Then:

$$N_i \ln\left(\frac{p_u}{1-p_u}\right) \leq \max(N_i) \ln\left(\frac{p_u}{1-p_u}\right) = \ln\left(\frac{p_h}{1-p_h}\right). \tag{5.1.14}$$

In this case, the probabilities of the nodes containing points in k-NN model are lower than the probabilities of the nodes containing endpoints in OctoMap.

When the values of threshold p_t in both mapping algorithms are identical, p_u might

exist:

$$\frac{\left(\frac{p_h}{1-p_h}\right)^{-\max(N_i)}}{1+\left(\frac{p_h}{1-p_h}\right)^{-\max(N_i)}} \leq p_u \leq p_h\,, \tag{5.1.15}$$

such that the numbers of occupied nodes in two mapping approaches are equal if the map is continuous in space. However, an occupancy map is discrete in space since it is built with a given resolution. So the number of occupied nodes in OctoMap might never be equal to that in k-NN based approach. But at least the k-NN method has the potential ability to compete with OctoMap.

When $p_u > p_l$ and p_u satisfies (5.1.13), for the nodes containing points in the k-NN method:

$$l(m_i \mid z_t) = \sum_{j=1}^{N_i} \ln\left[\frac{p(s_j)}{1-p(s_j)}\right] \leq N_i \ln\left(\frac{p_u}{1-p_u}\right), \tag{5.1.16}$$

which means under a specific value of p_u, $l(m_i \mid z_t)$ can be further decreased. This shows the ability of the k-NN method to potentially decrease the probability of a node despite points being present.

5.1.5 Map Update

An example is given in Figure 5-1 to show how the proposed map update policy works. As explained in Section 5.1.1, if the points are closer to each other, the corresponding node will get a higher probability, and vice versa. The nodes traversed by rays will be updated with l_{in} and l_{out} depending on the endpoint is within range s_c or not. Figures 5-1a and 5-1b present a point cloud and the corresponding occupancy map, respectively.

Algorithm 2 shows the mapping process with the proposed inverse sensor model. z_t is the measurement, i.e., the point cloud at time t. The corresponding position of the vision sensor is denoted as x_t. m is the occupancy map. Lines 3 through 10 update the nodes containing points. Lines 11 to 20 update the nodes traversed by the rays cast from the sensor to endpoints. The traversed nodes are updated with log-odds values l_{in} and l_{out}, corresponding to the probabilities of p_m and p'_m.

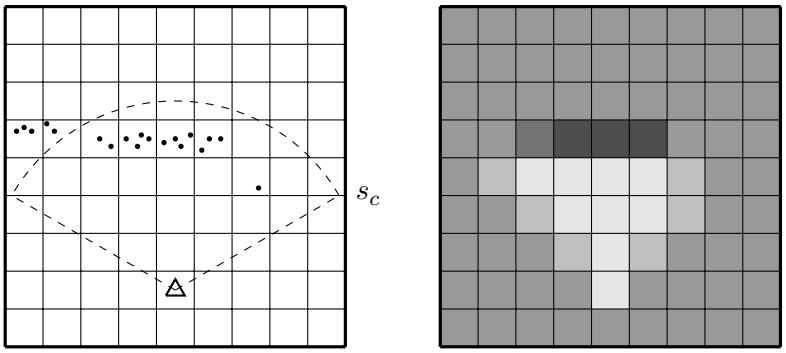


Figure 5-1: Example of map update. (a) Point cloud. (b) Occupancy map.

Algorithm 2: Map Update

Input: z_t, x_t, k, m, μ, σ
Output: m

$z_{in} \leftarrow GetInnerPoints(z_t, x_t, s_c)$
$z_{out} \leftarrow GetOuterPoints(z_t, x_t, s_c)$
$m' \leftarrow \varnothing$
for $P_i \in z_{in}$ **do**
    $s_i \leftarrow GetAverageDistance(z_{in}, P_i, k)$
    $m'.UpdateNode\left(P_i, l\left(p(s_i)\right)\right)$
for $m_i \in m'$ **do**
    $P_i \leftarrow m_i.GetNodeCentreCoordinates()$
    $l_i \leftarrow m_i.GetLogOddsProbability()$
    $m.UpdateNode(P_i, l_i)$
$m_{in} \leftarrow m.GetTraversedNodes(z_{in}, x_t)$
for $m_i \in m_{in}$ **do**
    $m.UpdateNode(m_i, l_{in})$
$m_{out} \leftarrow m.GetTraversedNodes(z_{out}, x_t)$
$m'_{out} \leftarrow \varnothing$
for $m_i \in m_{out}$ **do**
    if $m_i \notin m_{in}$ **then**
        $m'_{out}.insert(m_i)$
for $m_i \in m'_{out}$ **do**
    $m.UpdateNode(m_i, l_{out})$

5.1.6 Time Estimation Model

The run time of k-NN mapping is proportional to parameter k and the number of points N. So the run time can be derived by:

$$\Phi_k(k, N) = (ak + b)N + c\,, \tag{5.1.17}$$

where a, b and c are coefficients.

5.2 Random Down Sampling Based on the K-NN Method

This section presents a point cloud down sampling approach using the k-NN method. To find out proper down sampling level of point clouds, a simulation is used to analyse the approximate relationship between the occupancy probability and the number of points in a node.

5.2.1 Simulation Design

Figure 5-2 shows the relationship of multiple coordinate systems when the camera observes a node. $O_b - x_b y_b z_b$ and $O_c - x_c y_c z_c$ are the cube frame and the camera frame, respectively. Plane κ_1 contains O_b and is perpendicular to axis z_c. Plane κ_2 contains O_c and is parallel to plane κ_1. The camera can move in plane κ_2 and rotate about axis z_c. By controlling distance s_d between two planes and the corresponding pixel coordinates of the intersection point of axis z_c and plane κ_1, the movement of the camera in plane κ_2 can be controlled. A sphere centred on the cube centre is created. Since the cube can be observed by the camera from any direction, z_c axis may be parallel to any radius of the sphere. So the direction of the radius of the sphere can be used to control the direction of axis z_c. Due to the symmetrical characteristics of the cube, only one eighth of the sphere needs to be considered.

In the cube frame, the direction of the radius of the one eighth sphere surface is:

$$\mathbf{n} = (\sin\theta\cos\varphi, \sin\theta\sin\varphi, \cos\theta)\,, \tag{5.2.1}$$

where φ is the rotation about z_b axis and θ is the rotation about y_b axis. The respective ranges of these two angles are $0 \leq \theta \leq \pi/2$ and $0 \leq \varphi \leq \pi/2$. Then plane κ_1 and plane κ_2 can be denoted as:

$$ax_b + by_b + cz_b = 0 \tag{5.2.2}$$

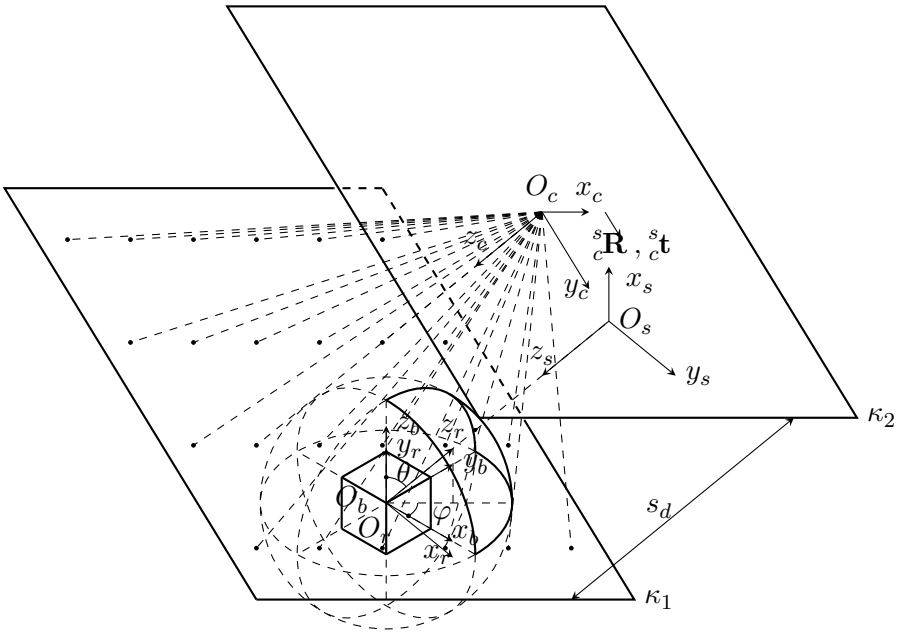


Figure 5-2: Points in a cube.

and

$$ax_b + by_b + cz_b - d = 0, \tag{5.2.3}$$

where $a = \sin\theta\cos\varphi$, $b = \sin\theta\sin\varphi$, $c = \cos\theta$ and $d = -s_d\sqrt{a^2 + b^2 + c^2}$.

In Figure 5-2, axis z_r and the radius of the sphere are co-linear, then the transformation matrix from $O_r - x_r y_r z_r$ to $O_b - x_b y_b z_b$ is:

$${}^b_r\mathbf{T} = \begin{bmatrix} \mathbf{R}_z(\varphi)\mathbf{R}_y(\theta) & 0 \\ 0 & 1 \end{bmatrix}, \tag{5.2.4}$$

where

$$\mathbf{R}_y(x) = \begin{bmatrix} \cos x & 0 & \sin x \\ 0 & 1 & 0 \\ -\sin x & 0 & \cos x \end{bmatrix} \tag{5.2.5}$$

and

$$\mathbf{R}_z(x) = \begin{bmatrix} \cos x & -\sin x & 0 \\ \sin x & \cos x & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{5.2.6}$$

The origin of the frame $O_s - x_s y_s z_s$ is in the plane κ_2 and axis z_s is co-linear with z_r. Axes x_s and y_s are parallel to y_r and x_r, respectively. Then the transformation from

$O_s - x_s y_s z_s$ to $O_r - x_r y_r z_r$ can be denoted as:

$$ {}^{r}_{s}\mathbf{T} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & s_d \\ 0 & 0 & 0 & 1 \end{bmatrix} . \tag{5.2.7} $$

The camera frame moves in plane κ_2 and the transformation from frame $O_c - x_c y_c z_c$ to $O_s - x_s y_s z_s$ is:

$$ {}^{s}_{c}\mathbf{T} = \begin{bmatrix} {}^{s}_{c}\mathbf{R} & {}^{s}_{c}\mathbf{t} \\ 0 & 1 \end{bmatrix} , \tag{5.2.8} $$

where ${}^{s}_{c}\mathbf{R} = \mathbf{R}_z(\alpha)$, α is the rotation angle about axis z_c, ${}^{s}_{c}\mathbf{t}$ is the translation matrix, and ${}^{s}_{c}\mathbf{t} = \begin{bmatrix} s_x & s_y & 0 \end{bmatrix}^\top$.

Since a rectangle has 2 lines of symmetry, only a quarter of the rotation needs to be considered, i.e., $0 \leq \alpha \leq \pi/2$. For any point in the camera frame, its coordinates in the cube frame can be denoted as:

$$ {}^{b}\bar{\mathbf{P}} = {}^{b}_{r}\mathbf{T}\,{}^{r}_{s}\mathbf{T}\,{}^{s}_{c}\mathbf{T}\,{}^{c}\bar{\mathbf{P}} \coloneqq {}^{b}_{c}\mathbf{T}\,{}^{c}\bar{\mathbf{P}} . \tag{5.2.9} $$

With the pinhole camera model [88], the homogeneous coordinates of the intersection point of plane κ_1 and the ray cast from pixel (u, v) in the image can be denoted as:

$$ {}^{c}\bar{\mathbf{P}} = \begin{bmatrix} \frac{s_d}{f_p}(u - c_x) & \frac{s_d}{f_p}(v - c_y) & s_d & 1 \end{bmatrix}^\top , \tag{5.2.10} $$

where $0 \leq u \leq u_{\max}$, $0 \leq v \leq v_{\max}$ and $u_{\max}$ and $u_{\max}$ are maximum values in pixel coordinates. All the intersection points in plane κ_1 are within a rectangle. The vertices of the bounding rectangle in the camera frame are:

$$ \begin{cases} {}^{c}\bar{\mathbf{P}}_1 = \begin{bmatrix} -\frac{s_d c_x}{f_p} & -\frac{s_d c_y}{f_p} & s_d & 1 \end{bmatrix}^\top \\ {}^{c}\bar{\mathbf{P}}_2 = \begin{bmatrix} \frac{s_d}{f_p}(u_{max} - c_x) & -\frac{s_d c_y}{f_p} & s_d & 1 \end{bmatrix}^\top \\ {}^{c}\bar{\mathbf{P}}_3 = \begin{bmatrix} \frac{s_d}{f_p}(u_{max} - c_x) & \frac{s_d}{f_p}(v_{max} - c_y) & s_d & 1 \end{bmatrix}^\top \\ {}^{c}\bar{\mathbf{P}}_4 = \begin{bmatrix} -\frac{s_d c_x}{f_p} & \frac{s_d}{f_p}(v_{max} - c_y) & s_d & 1 \end{bmatrix}^\top \end{cases} . \tag{5.2.11} $$

The centre of the cube should be guaranteed to be inside the bounding rectangle. With (5.2.9) and (5.2.11), the variables s_x, s_y and α should satisfy:

$$\begin{cases} -\dfrac{s_d}{f_p}(u_{max}\cos\alpha - v_{max}\sin\alpha) \le s_x \le \dfrac{s_d}{f_p}(u_{max}\cos\alpha - v_{max}\sin\alpha) \\ 0 \le \alpha \le \dfrac{\pi}{2} \\ -\dfrac{s_d c_x}{f_p} \le -s_x\cos\alpha - s_y\sin\alpha \le \dfrac{s_d}{f_p}(u_{max} - c_x) \\ -\dfrac{s_d c_y}{f_p} \le s_x\sin\alpha - s_y\cos\alpha \le \dfrac{s_d}{f_p}(v_{max} - c_y) \end{cases} . \tag{5.2.12}$$

When the camera observes the cube, point O_b can appear in any position of an image. By controlling the corresponding coordinates, i.e., u and v, of O_b in the pixel frame and the distance, i.e., s_d, between two planes, the relative position of O_b and O_c can be controlled. In addition, the relative pose of the cube and the camera frame can be changed through three angles, i.e. θ, φ and α. For any given set of u, v, s_d, θ, φ and α, the translation matrix ${}^{s}_{c}\mathbf{t}$ can be derived by (5.2.4) (5.2.7) (5.2.8) and (5.2.9). Then the coordinates of O_c in the cube frame can be derived. Assume the coordinates of O_b and O_c in the cube frame are (x_0, y_0, z_0) and (x_1, y_1, z_1), respectively. Then the line through O_b and O_c can be denoted with the following parametric representation:

$$\Omega(\xi) = \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} + \begin{bmatrix} x_1 - x_0 \\ y_1 - y_0 \\ z_1 - z_0 \end{bmatrix} \xi \,, \tag{5.2.13}$$

where ξ is the parameter.

The surface of the object intersects the cube is simulated by a plane. A plane which contains O_b is used to cut the cube. The plane can be denoted with (5.2.1) and (5.2.2). However, the ranges of two angles are $0 \le \theta \le \pi$ and $0 \le \varphi < 2\pi$, respectively. Together with (5.2.13), the intersection points of the randomly generated plane and the lines projected from the camera centre can be obtained.

The cube is bounded by three pairs of parallel planes:

$$\begin{cases} x_b \pm \dfrac{s_l}{2} = 0 \\ y_b \pm \dfrac{s_l}{2} = 0 \\ z_b \pm \dfrac{s_l}{2} = 0 \end{cases} . \tag{5.2.14}$$

To make sure the points in the cube have enough neighbourhood points, another bounding box is implemented:

$$\begin{cases} x_b \pm \frac{3}{2} s_l = 0 \\ y_b \pm \frac{3}{2} s_l = 0 \\ z_b \pm \frac{3}{2} s_l = 0 \end{cases} . \tag{5.2.15}$$

The intersection points within this bounding box will be preserved for the calculation of the normal distribution. While the points within the cube will be used to analyse the relation of the number of points to occupancy probability.

Since the noise in measurements can cause the fluctuation of points, a noise model should be implemented when generating points. In [136], the noise model of the ZED camera has been investigated and defined as:

$$\Psi(z_c) = a \exp(b z_c) , \tag{5.2.16}$$

where a, b are two coefficients and z_c is the depth. In [136], the coefficients corresponding to the resolution of HD720 are $a = 0.0184$ and $b = 0.2986$.

5.2.2 Simulation Results

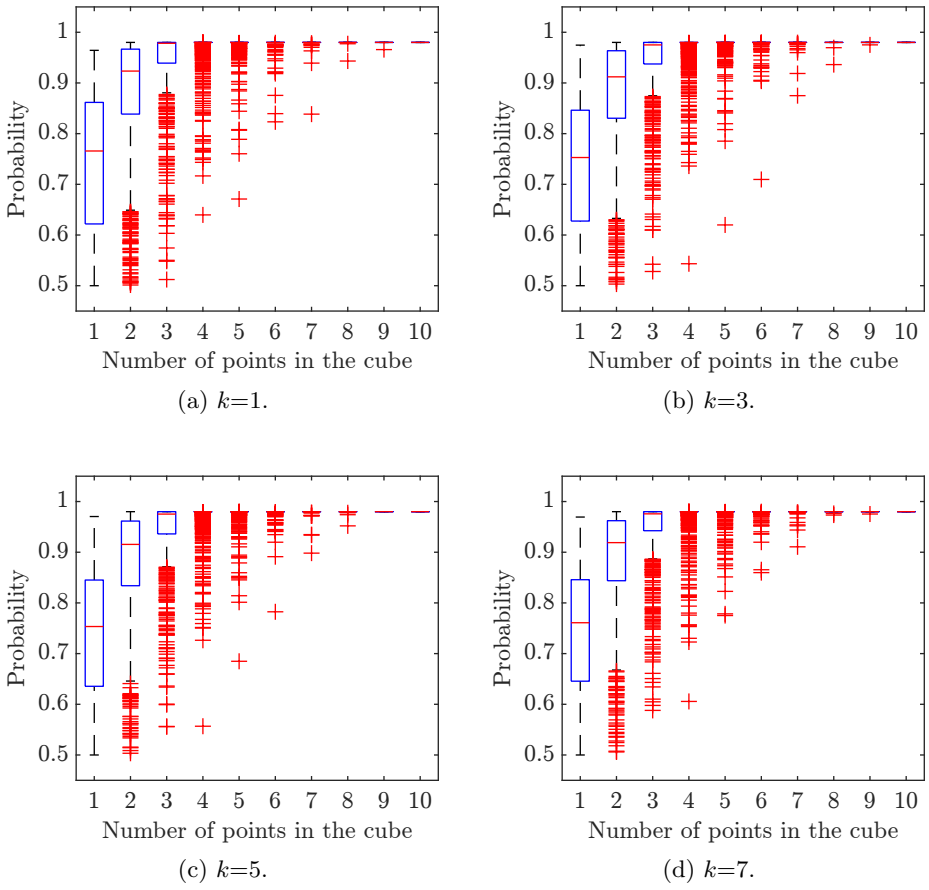
As introduced in Section 5.2.1, the relative position of the cube and the camera are controlled by u, v, s_d, θ, φ and α. The configuration of the pixel coordinates and three angles are shown in Table 5.1.

Table 5.1: Configuration of simulation.

	Minimum	Maximum	Step
u	100	1180	270
v	100	620	130
θ	0	$\frac{2}{3}\pi$	$\frac{\pi}{6}$
φ	0	$\frac{2}{3}\pi$	$\frac{\pi}{6}$
α	0	$\frac{\pi}{2}$	$\frac{\pi}{6}$
s_d	4	4	N/A

To determine how many points should be preserved in a cube, the k-NN method is implemented with $p_{\max}$, $p_{\min} = 0.02$, $p_u = 0.98$ and $p_l = 0.5$. Figure 5-3 presents

the simulated relationship of the occupancy probability and the number of points in the cube using the configuration in Table 5.1. The simulation result shows that the occupancy probability will increase with the number of points. The upper limit will be reached with three or four points. If the down sampling of point clouds is uniform, the relationship will still apply to the down sampled point cloud. It is unnecessary to preserve hundreds of points in a cube. To make sure the probability of the cube can be changed quickly, it is better to choose the number whose corresponding probability is not close to the maximum clamping threshold.



(a) k=1. (b) k=3. (c) k=5. (d) k=7.

Figure 5-3: Probability against the number of points in the cube. (a) k=1. (b) k=3. (c) k=5. (d) k=7.

In Figure 5-3, results are similar when parameter k varies, which is consistent with the results in Section 5.4. If a point is noise, it is normally far away from its neighbouring points in the point cloud. Since a noise point is isolated from other points, the change in parameter k does not make a big difference in the average distance from the point

to its neighbours. If a point is on the external surfaces of an object, it is much closer to its neighbours than a noise point. The points on an objects often form clusters. In this case, parameter k will not cause significant change in the average distance. In summary, no matter a point is noise or on the surface of objects, the average distance from a point to its k-NN does not change much when parameter k is different. Since the occupancy probability of a point is defined by its corresponding average distance, the probability in Figure 5-3 shows similar trends when k varies.

5.2.3 Down Sampling Level

The following function is defined to decide the down sampling level of point clouds:

$$N_s = \frac{s_d^2 N_e N_0}{f_p^2 s_l^2}\,, \tag{5.2.17}$$

where N_0 and N_s are the respective numbers of points in a point cloud before and after down sampling, N_e is the expected point number in the node of resolution s_l at a distance of s_d. A proper N_e is chosen based on the simulation.

Each point in the point cloud has its own index. For a finite sequence, Fisher-Yates shuffle is an effective way to generate a random permutation [137]. The modern form algorithm proposed in [138] reduces the time complexity and is popular in computer use. Based on the Fisher-Yates shuffle, the point cloud can be down sampled. However, in this implementation, rather than get the whole permutation, the sequence only needs to be shuffled for N_s times if N_s points are picked from a point cloud of N_0 points ($N_s < N_0$). N_e is set to 1.5 based on the simulation results in Section 5.2.2.

5.3 Experimental Method

In this section, the best performance of OctoMap and the k-NN method through grid search will be compared to illustrate the effectiveness of the k-NN method.

5.3.1 Design of Experiments

Point cloud parameters, mapping parameters and ORB parameters will affect the mapping performance. However, less important parameters can be reduced to accelerate computation. As introduced in Section 4.8, mapping parameters have a higher impact on the mapping performance other parameters. This section only focuses on mapping parameters. The k-NN parameters are first reduced and then the mapping performance achieved by optimising the residual most significant parameters will be compared with

the best OctoMap performance derived by searching all the possible combinations of parameters. K-NN parameter reduction follows the procedure introduced in Chapter 4. The workflow to compare two mapping algorithms is shown in Figure 5-4.

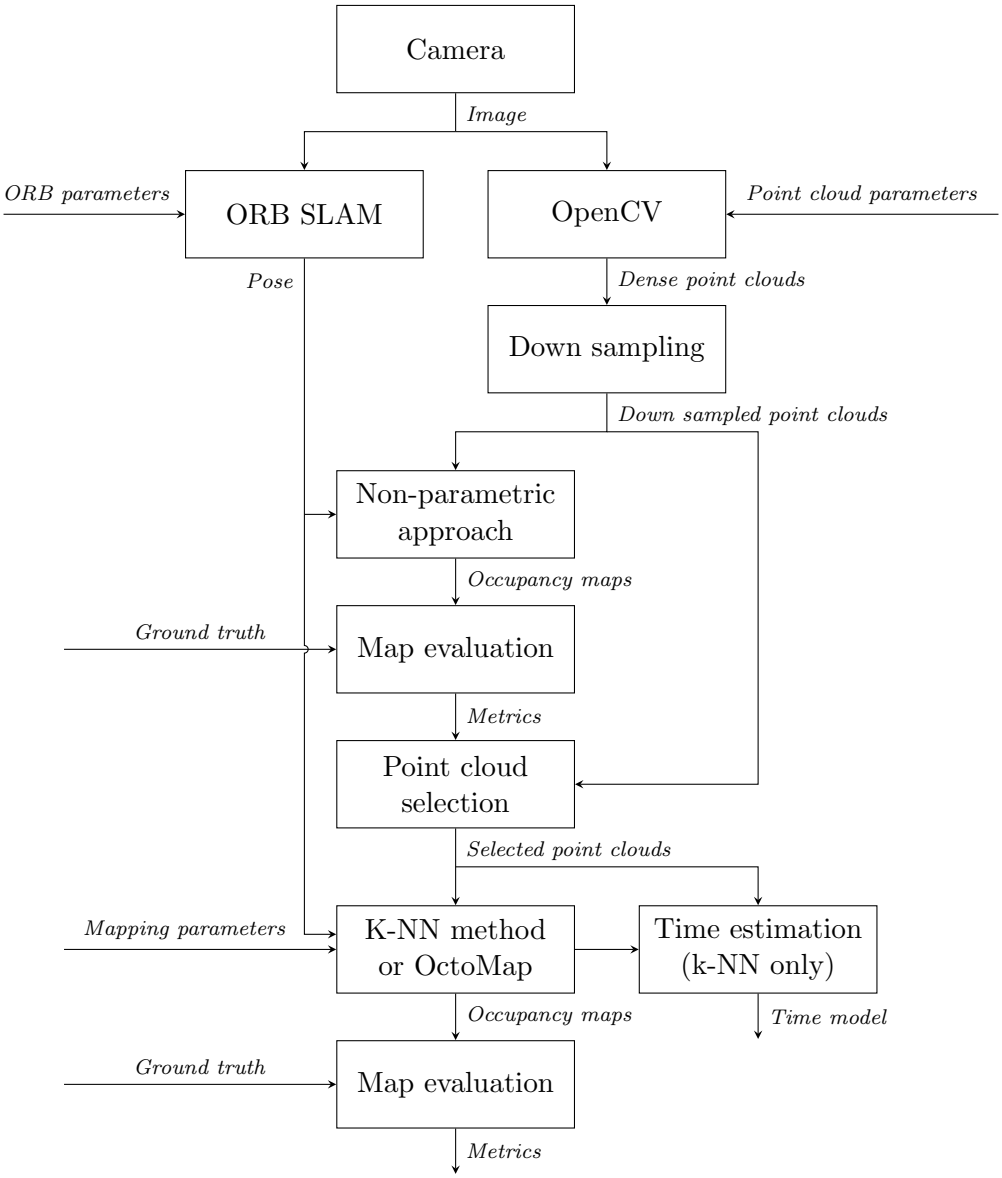


Figure 5-4: Experimental method for comparing the k-Nearest Neighbours (k-NN) method and OctoMap.

The method introduced in Section 4.7.1 will be used to generate point clouds. Dense point clouds are generated by the StereoSGBM algorithm in OpenCV and down sampled by the VoxelGrid filter in PCL as per Section 4.7.1. In the k-NN method, the octree module in PCL will be used to search the nearest neighbours of a specific point

in a point cloud.

For k-NN parameter reduction, the combination of point cloud parameters remains same on different data sets. Point cloud sets are chosen with FDR derived by the non-parametric naive mapping approach proposed in Section 4.4.2 based on their ranks. The choice of the point cloud set or the combination of point cloud parameters for k-NN parameter reduction will be given in Section 5.3.2. For each combination of k-NN parameters, an occupancy map can be generated. By comparing the map with the ground truth, the nodes in the map can be classified into four categories using the method in Section 4.3. Metrics TPR and FDR are computed from the number of nodes in each category. NCA feature selection proposed in [36] will be applied to analyse parameter weights, which has been introduced in Section 4.4.1. For the ease of comparison, the weights derived by different data sets are normalised as per [36].

Parameters can be optimised by grid search in the parameter space defined by the method in Section 5.1.3. Based on the above k-NN parameter reduction results and the optimisation results of OctoMap parameters in Section 4.8, parameters to be optimised can be determined and will be introduced in Section 5.3.2. The AUC of TPR-FDR variant is used as the performance metric. Three point cloud sets in each data set will be selected using the non-parametric approach to search the parameters which can achieve the best performance among all the combinations. The best AUC of the k-NN method will be compared with that of OctoMap.

The time estimation model for the k-NN method will be verified. The down sampling approach based on the k-NN method will be compared with the VoxelGrid filter in PCL in terms of the number of the nodes in occupancy maps as well.

5.3.2 Parameter Space for Analysis

The default ORB parameters are used to generate poses. In Section 4.7.2, there are 1600 combinations of point cloud parameters, which means 1600 point cloud sets can be generated for each data set. As specified in Section 5.3.1, for the reduction of k-NN parameters, the combination of point cloud parameters can be fixed since they are less important than mapping parameters in performance. The combination of parameters corresponds to the 800th ranked point cloud set of the data set collected with I layout Voronoi boxes in front of buildings. For the optimisation of mapping parameters, the 1st, 800th, and 1600th (lower number indicates better quality, i.e., cleaner point clouds) ranked point cloud sets are chosen from each data set to compare the performance of the k-NN method and OctoMap.

The configuration of mapping parameters is shown in Table 5.2. $p_{\max}$, p_m, p_t and $p_{\min}$ are shared by two mapping approaches. The choice of the step of OctoMap parameters has been discussed in Section 4.7.2. 0.12 is a reasonable step and will not affect results. To investigate k-NN parameter weights, $p_{\max}$, $p_{\min}$ and k are varied with corresponding steps, and p_t changes with $p_{\max}$ and $p_{\min}$. With (5.1.7), the total number of the combinations is 112 500. To optimise k-NN parameters, based on the results in Section 5.4, k is constant since it has a lower impact on the performance metrics. $p_{\max}$ and $p_{\min}$ are set to 0.98 and 0.02 since these values show highest frequencies in Section 4.7.2. Also, $p_{\max}$ and $p_{\min}$ are the upper and lower bounds on the probability but not the parameters for inverse senor models, and shared by both the k-NN method and OctoMap. Therefore, locking these two parameters does not benefit any approach but can decrease the number of the combinations of parameters to reduce the computational time. Besides $p_{\max}$ and $p_{\min}$, the setup of other OctoMap parameters is depending on the configuration in Section 4.7.2. To compare the k-NN method and OctoMap, the combinations of parameters for two mapping algorithms are 4050 and 180, respectively.

Table 5.2: Configuration of mapping parameters.

Parameter	Minimum	Maximum	Step	Method
$p_{\max}$ [a]	0.5	0.98	0.12	K-NN
$p_{\max}$ [b]	0.98	0.98	N/A	Both
p_h	0.5	0.98	0.12	OctoMap
p_m	0.02	0.38	0.12	Both
p_m'	0.02	0.38	0.12	K-NN
$p_{\min}$ [a]	0.02	0.38	0.12	K-NN
$p_{\min}$ [b]	0.02	0.02	N/A	Both
p_t	$p_{\min}$	$p_{\max}$	0.12	Both
p_u	0.02	0.98	0.12	K-NN
p_l	0.02	0.98	0.12	K-NN
k [a]	1	7	2	K-NN
k [b]	1	1	N/A	K-NN

[a] Configuration for parameter reduction.
[b] Configuration for parameter optimisation.

5.4 Results and Analysis

Firstly, the results of the average distance fitted by different distributions, i.e., GEV distribution, Log-logistic distribution, Rayleigh distribution, KDE and Normal distribution, are presented. An example is given by the average distance derived by 800th

point cloud set of I layout Voronoi boxes in front of buildings. Corresponding CDF is presented in Figure 5-5. Results show that the CDF of the average distance is nonsensitive to types of distribution. Given the CDF is implemented to define the relationship between the average distance and the change in the occupancy probability, a parametric distribution can be used for the ease of adjusting k-NN model.

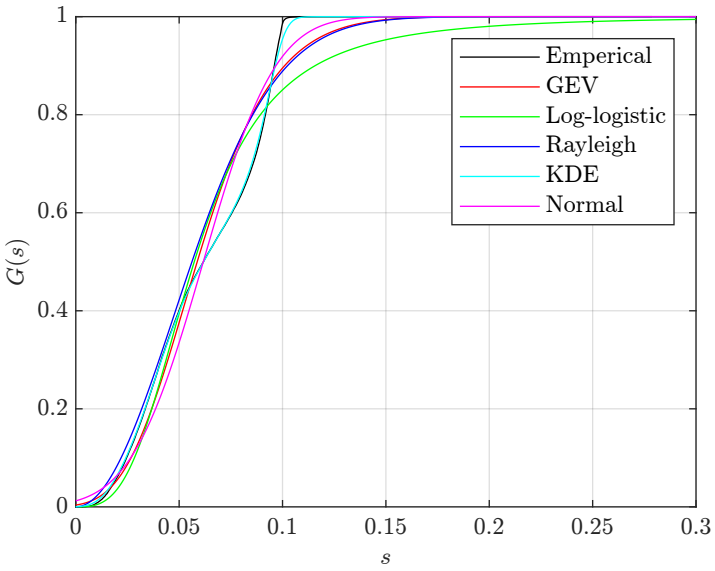
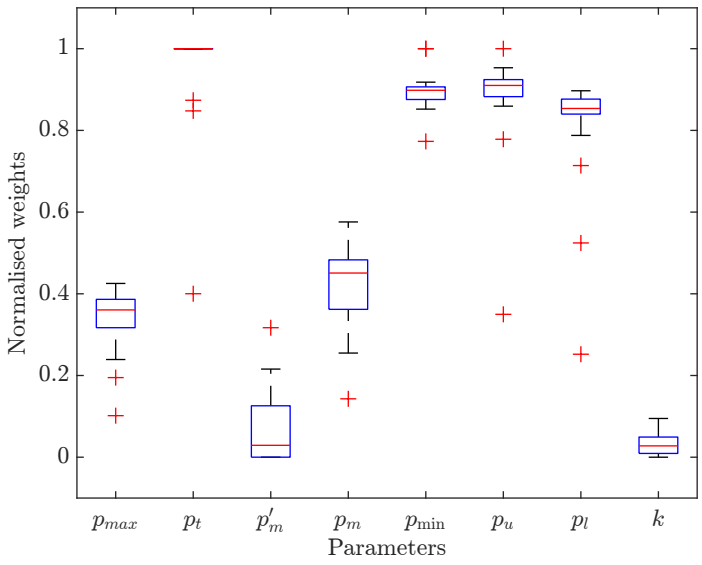


Figure 5-5: Cumulative Density Function (CDF) of the average distance fitted by different distributions.

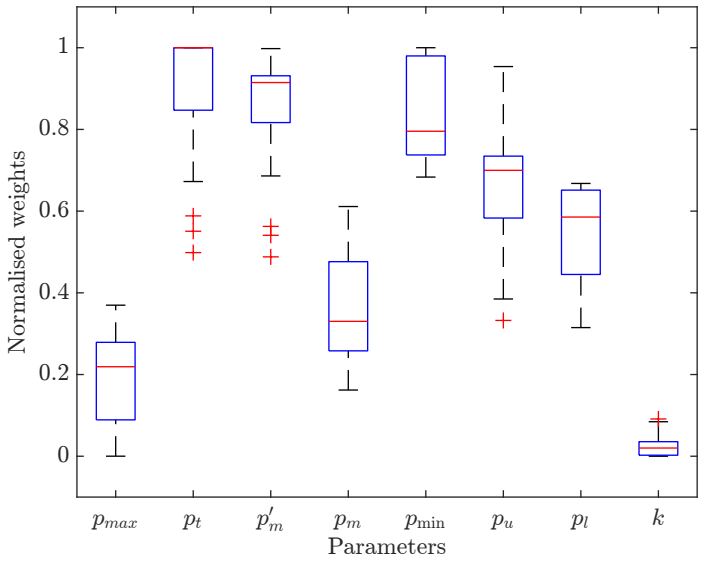
Then the results of the weights of k-NN parameters under performance metrics TPR and FDR are presented. As specified in Section 5.3.2, point cloud parameters are consistent in 20 data sets. With the configuration of mapping parameters in Table 5.2, the weights of k-NN parameters are computed by implementing NCA feature selection on TPR and FDR derived by node classification results. With (5.1.7) and (5.1.8), the number of parameter combinations for each data set is 5625. The normalised weight of each k-NN parameter is shown in Figure 5-6. Overall, among all the k-NN parameters, parameter k has a lower impact on the performance metrics and thus can be fixed to reduce computational time. In addition, based on the optimisation of OctoMap parameters in Section 4.8, $p_{\max}$ and $p_{\min}$ can be set as constants to further reduce the computational complexity.

The collected data sets are used to simulate different conditions in the real world to offer reliable evaluation results when comparing two mapping algorithms. Data sets are adequate due to the following considerations and settings:

- In real applications, objects can be of rich features or lack of features. This is



(a) True positive rate (TPR).



(b) False discovery rate (FDR).

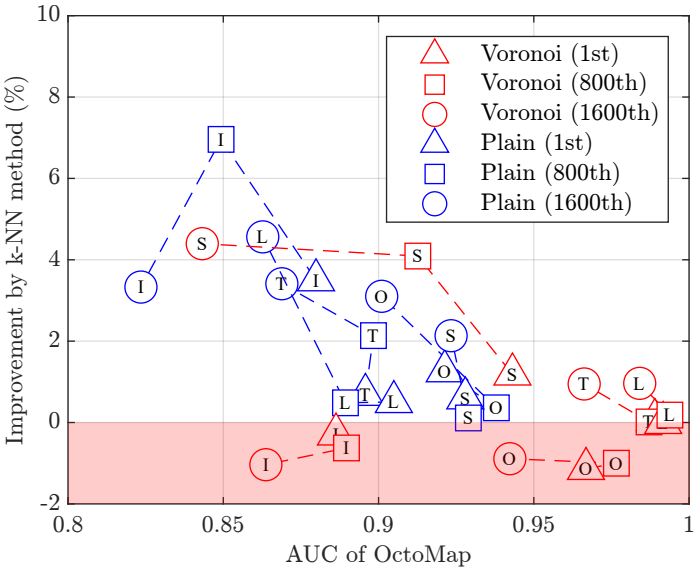
Figure 5-6: Normalised weights of k-Nearest Neighbours (k-NN) parameters on performance metrics. (a) True positive rate (TPR). (b) False discovery rate (FDR).

simulated by different textures, i.e., Voronoi diagrams and the plain box texture.

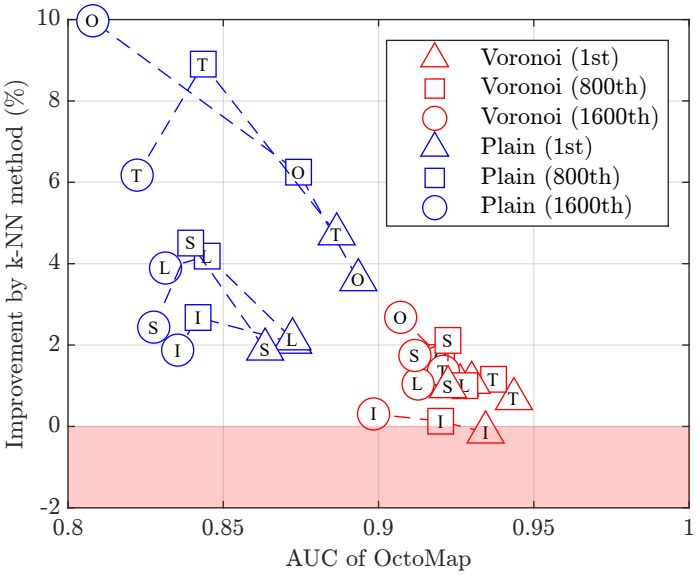
- Since a scene in the real world can be full or empty, data sets are collected in two environments, i.e., in front of buildings and in a parking lot.
- As the obstacles in the real world come in different shapes and sizes, the boxes are arranged in different layouts to create different test scenes.
- The quality of the point clouds produced by different sensors can be different. In the experiments, 1st, 800th and 1600th ranked point cloud sets generated from each data set are selected to simulate this.

With the above data sets, the optimisation results of two mapping algorithms will be compared. The configuration of parameters has been explained in Section 5.3.2. All the parameters except $p_{\max}$, $p_{\min}$, p_t and k will be optimised by searching the best AUC of TPR-FDR variant using the grid parameter space defined with Table 5.2. p_t is varied to generate points on TPR-FDR curve. For any combination of parameters, 9 points will be produced. Figure 5-7 shows the improvement achieved by the k-NN method over the AUC derived by OctoMap through grid search on 20 data sets. The improvement increases as the best AUC achieved by searching OctoMap parameter space decreases, but can be negative. With the parameter configuration in Table 5.2, the best improvements are normally achieved by $p_u > p_l$ when the best AUC of OctoMap is lower, while by $p_u = p_l$ when the AUC of OctoMap is higher. By implementing the k-NN method, an improvement up to 10% can be achieved. Overall, the mapping performance of Voronoi boxes is better than that of plain boxes. The details of the results in Figure 5-7 are shown in Table 5.3.

Figures 5-8 and 5-9 show two examples of occupancy maps derived by OctoMap and the k-NN method. Figure 5-8 corresponds to improvement of the 1st point cloud set of O layout Voronoi boxes in the parking lot in Figure 5-7. Figure 5-9 corresponds to the 800th point cloud set of O layout plain boxes in the parking lot. For each example, two points of similar FDR on the curves derived by different mapping approaches are selected, corresponding maps are presented. In Figure 5-7, improvements by the k-NN method against OctoMap can be observed on most data sets in terms of AUC. However, when the k-NN method achieves highest improvement, TPR-FDR curves derived by two mapping algorithms may intersect, i.e., the points on the curve derived by the k-NN method is always better than those on the curve derived by OctoMap when FDR is smaller than that of the intersection point, while worse when FDR is larger than that of the intersection point. But normally a combination of k-NN parameters can be found whose improvement against the AUC derived by searching OctoMap parameter



(a) Building.



(b) Parking lot.

Figure 5-7: Improvement through grid search over the k-Nearest Neighbours (k-NN) method over the area under the curve (AUC) of OctoMap. (a) Building. (b) Parking lot.

Table 5.3: Improvement by the k-Nearest Neighbours (k-NN) method over OctoMap.

Environment	Texture	Layout	Point Cloud Ranking	AUC		Improvement	
				OctoMap	K-NN method	AUC	%
Building	Plain	I	1	0.8799	0.9107	0.0308	3.50
Building	Plain	I	800	0.8493	0.9084	0.0591	6.96
Building	Plain	I	1600	0.8235	0.8509	0.0274	3.33
Building	Plain	O	1	0.9213	0.9330	0.0116	1.26
Building	Plain	O	800	0.9379	0.9413	0.0034	0.36
Building	Plain	O	1600	0.9010	0.9290	0.0280	3.10
Building	Plain	T	1	0.8958	0.9019	0.0061	0.68
Building	Plain	T	800	0.8984	0.9176	0.0192	2.14
Building	Plain	T	1600	0.8689	0.8985	0.0296	3.41
Building	Plain	L	1	0.9049	0.9095	0.0046	0.51
Building	Plain	L	800	0.8893	0.8936	0.0043	0.48
Building	Plain	L	1600	0.8627	0.9021	0.0394	4.56
Building	Plain	S	1	0.9279	0.9335	0.0055	0.60
Building	Plain	S	800	0.9289	0.9300	0.0010	0.11
Building	Plain	S	1600	0.9234	0.9431	0.0197	2.13
Building	Voronoi	I	1	0.8863	0.8837	-0.0026	-0.29
Building	Voronoi	I	800	0.8897	0.8841	-0.0056	-0.63
Building	Voronoi	I	1600	0.8637	0.8548	-0.0090	-1.04
Building	Voronoi	O	1	0.9668	0.9558	-0.0110	-1.14
Building	Voronoi	O	800	0.9766	0.9668	-0.0098	-1.00
Building	Voronoi	O	1600	0.9423	0.9339	-0.0084	-0.89
Building	Voronoi	T	1	0.9890	0.9897	0.0007	0.07
Building	Voronoi	T	800	0.9871	0.9873	0.0001	0.01
Building	Voronoi	T	1600	0.9662	0.9754	0.0092	0.95
Building	Voronoi	L	1	0.9917	0.9916	-0.0001	-0.01
Building	Voronoi	L	800	0.9938	0.9956	0.0018	0.18
Building	Voronoi	L	1600	0.9841	0.9936	0.0095	0.96
Building	Voronoi	S	1	0.9431	0.9543	0.0112	1.18
Building	Voronoi	S	800	0.9127	0.9500	0.0373	4.09
Building	Voronoi	S	1600	0.8432	0.8803	0.0371	4.40

Table 5-3 (continued): Improvement by the k-Nearest Neighbours (k-NN) method over OctoMap.

Environment	Texture	Layout	Point Cloud Ranking	AUC		Improvement	
				OctoMap	K-NN method	AUC	%
Parking lot	Plain	I	1	0.8725	0.8905	0.0180	2.06
Parking lot	Plain	I	800	0.8419	0.8644	0.0225	2.68
Parking lot	Plain	I	1600	0.8353	0.8510	0.0157	1.88
Parking lot	Plain	O	1	0.8935	0.9256	0.0322	3.60
Parking lot	Plain	O	800	0.8743	0.9289	0.0546	6.25
Parking lot	Plain	O	1600	0.8080	0.8885	0.0806	9.97
Parking lot	Plain	T	1	0.8865	0.9284	0.0419	4.73
Parking lot	Plain	T	800	0.8436	0.9187	0.0751	8.90
Parking lot	Plain	T	1600	0.8223	0.8730	0.0508	6.18
Parking lot	Plain	L	1	0.8723	0.8910	0.0187	2.15
Parking lot	Plain	L	800	0.8448	0.8802	0.0354	4.19
Parking lot	Plain	L	1600	0.8314	0.8638	0.0324	3.89
Parking lot	Plain	S	1	0.8635	0.8798	0.0164	1.89
Parking lot	Plain	S	800	0.8395	0.8773	0.0378	4.51
Parking lot	Plain	S	1600	0.8276	0.8478	0.0202	2.44
Parking lot	Voronoi	I	1	0.9345	0.9331	-0.0014	-0.14
Parking lot	Voronoi	I	800	0.9200	0.9212	0.0012	0.13
Parking lot	Voronoi	I	1600	0.8984	0.9012	0.0027	0.31
Parking lot	Voronoi	O	1	0.9263	0.9357	0.0094	1.02
Parking lot	Voronoi	O	800	0.9203	0.9359	0.0156	1.69
Parking lot	Voronoi	O	1600	0.9071	0.9314	0.0243	2.68
Parking lot	Voronoi	T	1	0.9436	0.9500	0.0064	0.68
Parking lot	Voronoi	T	800	0.9370	0.9479	0.0109	1.16
Parking lot	Voronoi	T	1600	0.9209	0.9334	0.0125	1.36
Parking lot	Voronoi	L	1	0.9300	0.9400	0.0100	1.08
Parking lot	Voronoi	L	800	0.9278	0.9371	0.0093	1.00
Parking lot	Voronoi	L	1600	0.9127	0.9222	0.0095	1.04
Parking lot	Voronoi	S	1	0.9224	0.9313	0.0089	0.97
Parking lot	Voronoi	S	800	0.9223	0.9418	0.0195	2.11
Parking lot	Voronoi	S	1600	0.9117	0.9276	0.0159	1.74

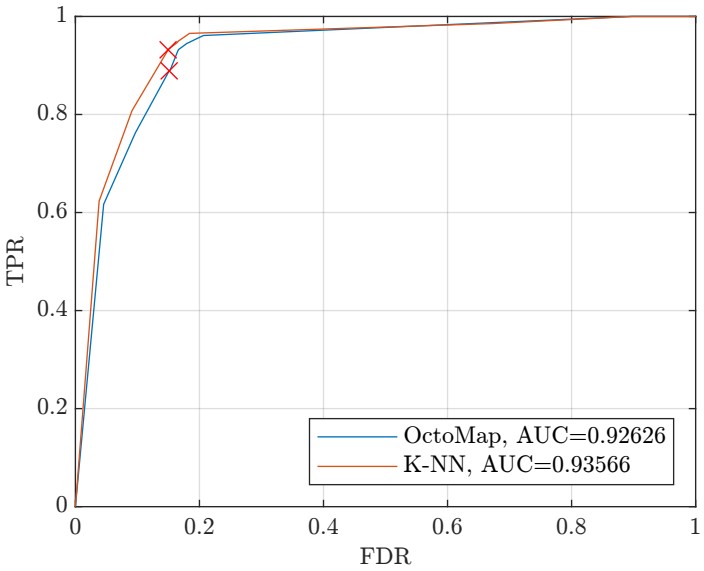
space is less significant than that achieves in Figure 5-7 such that for each point on the TPR-FDR curve of OctoMap a point of better performance can be found on the curve derived by the k-NN method.

In Figures 5-8 and 5-9, the improvements are mostly observed in the ground areas. For the point clouds used for generating occupancy maps, a mapping algorithm processes the points on the ground and the points on the boxes in the same way, no matter the points are produced by the ground or the targets. In other words, for the mapping algorithm, the ground and the boxes are both obstacles. Since the area of the ground is larger than that of the external surfaces of the boxes, more points in the point clouds belong to the ground areas. When an occupancy map is generated using such point clouds, the number of the nodes in the ground areas is greater than that of the nodes on the boxes. Therefore, the improved nodes mainly show in the ground areas.

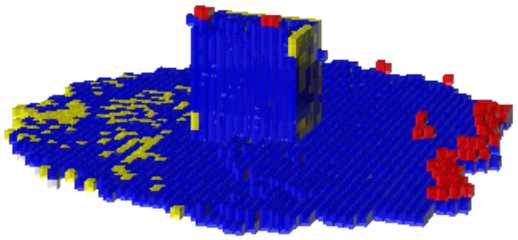
The AUC of k-NN shows an improvement up to 10% over that of OctoMap using grid search. The improvement achieved by the k-NN method increases as the AUC of OctoMap decreases. Although for Voronoi boxes of I and O layouts in front of buildings, OctoMap outperforms the k-NN method up to 1%, this is unlikely to happen in the real world since the textures of objects are more like the pattern on plain boxes rather than the regular Voronoi pattern. In addition, the k-NN method still performs better in other Voronoi cases. Overall, mapping performance is better in the environment with buildings since there are more image features on the objects nearby and the quality of point clouds is better. In each environment, mapping performance is normally better when targets are covered with Voronoi diagrams due to the extra features introduced by the diagrams. There is no obvious trend among different layouts.

Figure 5-10 verifies the time model proposed in Section 5.1.6. In each data set, 5% of the k-NN parameter reduction results are randomly selected and used to estimate the time model. The coefficients in the model are $a = 7.3915 \times 10^{-7}$, $b = 2.7630 \times 10^{-5}$ and $c = 1.5373$. The result shows that run time is proportional to parameter k and the number of points used for k-NN mapping.

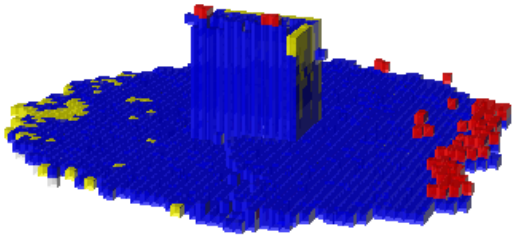
The last result compares the sampling approaches in terms of the number of the nodes in the map produced by down sampled point clouds. StereoSGBM parameters to generate disparity correspond to 800th ranked down sampled point cloud set generated by VoxelGrid filter. The OctoMap algorithm with default parameters is implemented to generate occupancy maps. The number of the nodes in the map generated by each sampling method with different data sets are shown in Figure 5-11. In each data set, the node number of VoxelGrid filter is greater than that of random sampling using



(a) True positive rate (TPR)-false positive rate (FDR).

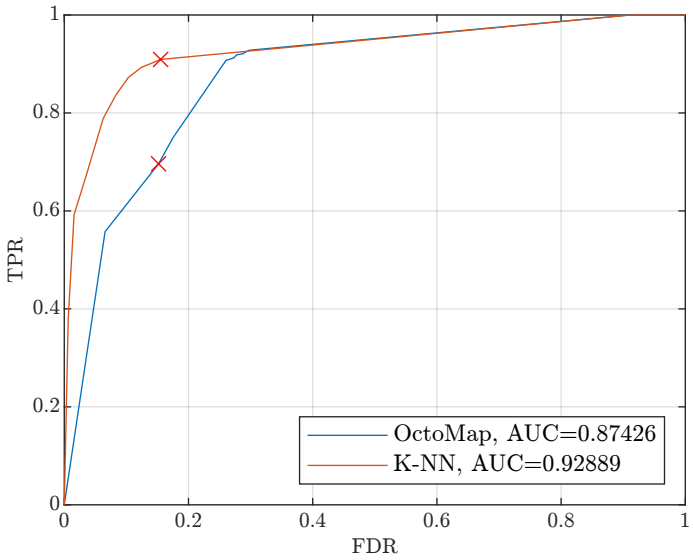


(b) OctoMap.

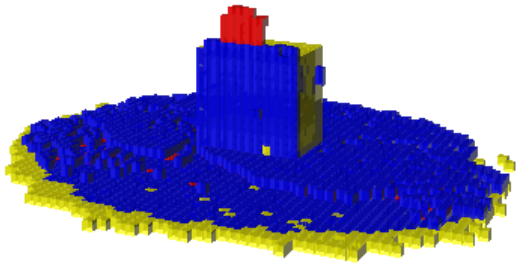


(c) K-Nearest Neighbours (k-NN) method.

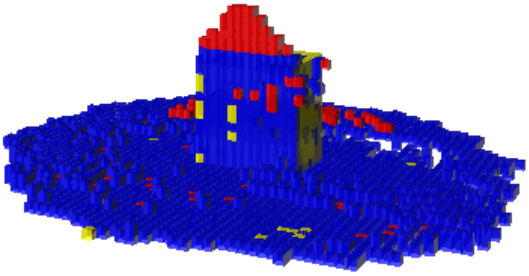
Figure 5-8: Occupancy maps derived by different algorithms using the data set of O layout Voronoi boxes in the parking lot. (a) Receiver Operating Characteristic (ROC) variant true positive rate (TPR)-false positive rate (FDR). (b) Occupancy map derived by OctoMap. Blue: TPs, red: FPs and yellow: FNs. TNs are not included for clarity. (c) Occupancy map derived by the k-Nearest Neighbours (k-NN) method.



(a) True positive rate (TPR)-false positive rate (FDR).



(b) OctoMap.



(c) K-Nearest Neighbours (k-NN) method.

Figure 5-9: Occupancy maps derived by different algorithms using the data set of O layout plain boxes in the parking lot. (a) Receiver Operating Characteristic (ROC) variant true positive rate (TPR)-false positive rate (FDR). (b) Occupancy map derived by OctoMap. Blue: TPs, red: FPs and yellow: FNs. TNs are not included for clarity. (c) Occupancy map derived by the k-Nearest Neighbours (k-NN) method.

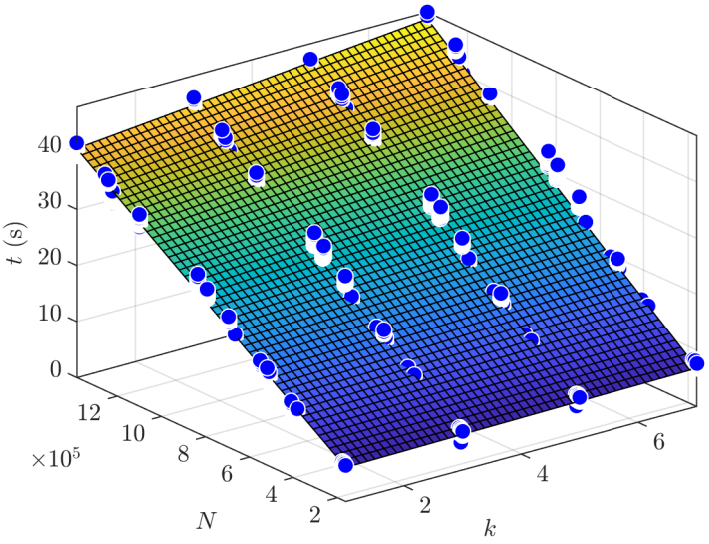


Figure 5-10: Polynomial regression for the run time of the k-Nearest Neighbours (k-NN) method.

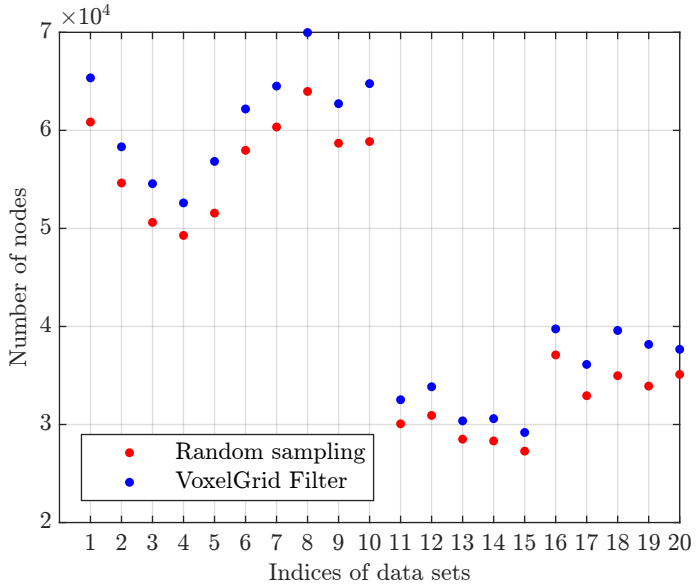


Figure 5-11: Node numbers of random sampling based on the k-Nearest Neighbours (k-NN) method and VoxelGrid filter. Indices 1 to 5: I, O, T, L and S layout boxes with Voronoi diagrams in front of buildings. Indices 6 to 10: I, O, T, L and S layout plain boxes in front of buildings. Indices 11 to 15: I, O, T, L and S layout boxes with Voronoi diagrams in the parking lot. Indices 16 to 20: I, O, T, L and S layout plain boxes in the parking lot.

the k-NN method, which means random sampling loses information. So the VoxelGrid filter is used for sampling point clouds in the previous chapters to preserve as much information as possible.

5.5 Summary

In this chapter, an inverse sensor model for occupancy mapping using the context of neighbouring points is presented. The occupancy information of a point is represented with the average distance to its k-NN. The relationship between the average distance and the occupancy probability is defined with the corresponding CDF. By implementing NCA, the parameter which has a lower impact on the mapping performance can be reduced. In addition, by considering the most frequent OctoMap parameters, the number of the parameters to be investigated can be further reduced. Through searching the grid parameter space, the residual most important parameters can be optimised to achieve better performance. The k-NN method is implemented on the point clouds derived by different data sets. Results show that the k-NN method is effective in improving performance over OctoMap. The time estimation model is verified as well. Random sampling based on the k-NN method and VoxelGrid filter in PCL are compared. Although random sampling may lose information, it can be potentially useful. Through analysis, the key findings are:

- The k-NN model is nonsensitive to different types of distributions.
- Parameter k is of lower impact than other k-NN parameters.
- Through grid search optimisation, the performance of OctoMap can be improved by the k-NN method.

Chapter 6

Conclusions and Future Work

The main contribution of the work in this thesis is the optimisation and extension of the octree base occupancy mapping. The novelty can be mainly decomposed into two parts, i.e., parameter reduction and optimal parameter searching, and a k-NN based mapping approach. This chapter summarises the thesis and makes suggestions on future work.

6.1 Conclusions

A principled methodology for reducing point cloud parameters and ORB parameters using NCA, and optimising OctoMap occupancy mapping parameters using grid search is proposed in Chapter 4. Experiments are conducted on 20 data sets introduced in Chapter 3, with specially designed targets providing precise ground truths, allowing to investigate the impacts of environments and the patterns on the external surfaces of the targets. The proposed methodology is verified on the point clouds sets generated by the StereoSGBM algorithm and selected by a non-parametric mapping approach. Pixel connectivity in image processing is implemented in the node classification procedure to tolerate the fluctuation of the points in point clouds. The ROC variant TPR-FDR is applied as performance metrics to deal with unbalanced data sets in which the elements in one class is more than in others due to empty space in the environments. By implementing NCA, parameters of lower impacts can be reduced. The residual most important parameters can be optimised by searching the grid parameter space defined by algorithm-required relations. The analysis on the results derived by 20 data sets shows that the proposed method is effective in improving the mapping performance of OctoMap. In Chapter 4, it has been found that mapping parameters are more

important than point cloud parameters and ORB parameters, and an improvement over default OctoMap parameters in mapping performance can be achieved through the two-step framework, i.e., first reducing parameters with NCA and then optimising mapping parameters with grid search.

Chapter 5 introduces a mapping approach using the context of neighbouring points to overcome the limitations in the update policy of OctoMap. The new method is built on the relationship between the probability and the average distance from a point to its k-NN. The probability of a point is effectively defined by the CDF of average distances. The parameter reduction method introduced in Chapter 4 is used to reduce the dimension of k-NN parameters and the less important parameter is identified. Along with the analysis of parameter weights in Chapter 4, parameters to be optimised to achieve better mapping performance are determined. Optimisation is performed on OctoMap and the k-NN method using the point cloud sets generated by 20 data sets to compare the performance of two mapping algorithms. Through the analysis in Chapter 5, it has been found that the k-NN method is nonsensitive to types of distributions and the k-NN method can outperform OctoMap in most cases. The proposed mapping approach is an effective extension to improve the performance of occupancy mapping.

Overall, the results in Chapters 4 and 5 have demonstrated the effectiveness and benefits of the parameter searching and the k-NN method as an extension of OctoMap. The key findings are:

- Mapping parameters are of higher impacts on the quality of the final map than point cloud parameters and pose generation parameters.
- A better performance of OctoMap can be achieved over default parameters through grid parameter searching.
- As the extension of OctoMap, the k-NN based mapping approach can outperform OctoMap.
- The k-NN model is nonsensitive to types of distributions and parameter k is less important than other parameters.

6.2 Future Work

Although the work presented in this thesis can improve the performance of the octree based occupancy mapping algorithm, current work can be improved in the future.

Firstly, the computation time can be optimised. One limitation of the proposed parameter searching method in Chapter 4 is that when the step is decreased, the number of combinations of parameters will increase exponentially. With certain steps, the approach is computationally prohibitive. However, it has been demonstrated in Section 4.7.2 that the results do not benefit from decreasing the step.

Secondly, the methodology in this work can be tested on point clouds generated with other methods to strengthen its usefulness. In this thesis, experiments are only conducted on the point clouds produced by the StereoSGBM algorithm in OpenCV. Other algorithms generating point clouds from stereo images can be considered. Also, other types of sensors such as LIDAR can also be used for point cloud generation to verify the methods in this thesis. Unlike StereoSGBM algorithm parameters, parameters of LIDAR, e.g., motor speed and sample rate, control the operation of hardware.

Finally, the performance metrics for optimisation can be further extended. In this thesis, the AUC of the ROC variant TPR-FDR is used as the performance metric. As specified in Chapter 5, when comparing the k-NN method with OctoMap, the k-NN method is better than OctoMap in terms of the absolute values of AUC; however, the ROC curves derived by two mapping approaches can intersect at a certain point, making the performance of the k-NN method better when the FDR is smaller than that of the intersection point while worse when FDR is larger. The parameter searching method can be potentially improved to produce more reliable results.

References

[1] J. P. Saarinen, H. Andreasson, T. Stoyanov, and A. J. Lilienthal, "3d normal distributions transform occupancy maps: An efficient representation for mapping in dynamic environments," *International Journal of Robotics Research*, vol. 32, no. 14, pp. 1627–1644, 2013.

[2] K. Katyal, K. Popek, C. Paxton, P. Burlina, and G. D. Hager, "Uncertainty-aware occupancy map prediction using generative networks for robot navigation," in *Proceedings of IEEE International Conference on Robotics and Automation*, 2019, pp. 5453–5459.

[3] T. Duong, N. Das, M. Yip, and N. Atanasov, "Autonomous navigation in unknown environments using sparse kernel-based occupancy mapping," in *Proceedings of IEEE International Conference on Robotics and Automation*, 2020, pp. 9666–9672.

[4] K. Lee and D. Kum, "Collision avoidance/mitigation system: Motion planning of autonomous vehicle via predictive occupancy map," *IEEE Access*, vol. 7, pp. 52 846–52 857, 2019.

[5] S. Hrabar, "3d path planning and stereo-based obstacle avoidance for rotorcraft uavs," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2008, pp. 807–814.

[6] S. Hoermann, M. Bach, and K. Dietmayer, "Dynamic occupancy grid prediction for urban autonomous driving: A deep learning approach with fully automatic labeling," in *Proceedings of IEEE International Conference on Robotics and Automation*, 2018, pp. 2056–2063.

[7] J. J. Im, A. Leonessa, and A. Kurdila, "A real-time data compression and occupancy grid map generation for ground-based 3d lidar data using wavelets," in

Proceedings of ASME Dynamic Systems and Control Conference, vol. 2, 2010, pp. 557–562.

[8] R. B. Rusu and S. Cousins, "3d is here: Point cloud library (pcl)," in *Proceedings of IEEE International Conference on Robotics and Automation*, 2011, pp. 1–4.

[9] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard, "An evaluation of the rgb-d slam system," in *Proceedings of IEEE International Conference on Robotics and Automation*, 2012, pp. 1691–1696.

[10] M. Harville and D. Li, "Fast, integrated person tracking and activity recognition with plan-view templates from a single stereo camera," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, 2004, pp. II–II.

[11] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilistic 3d mapping framework based on octrees," *Autonomous robots*, vol. 34, no. 3, pp. 189–206, 2013.

[12] D. J. R. Meagher, "Octree encoding: A new technique for the representation, manipulation and display of arbitrary 3-d objects by computer," Rensselaer Polytechnic Institute, New York, NY, Tech. Rep. IPL-TR-80-111, 1980.

[13] M. Hansard, S. Lee, O. Choi, and R. P. Horaud, *Time-of-flight cameras: Principles, methods and applications*. Springer, 2012.

[14] R. A. Hamzah and H. Ibrahim, "Literature survey on stereo vision disparity map algorithms," *Journal of Sensors*, vol. 2016, 2016.

[15] S. Foix, G. Alenya, and C. Torras, "Lock-in time-of-flight (tof) cameras: A survey," *IEEE Sensors Journal*, vol. 11, no. 9, pp. 1917–1926, 2011.

[16] W. Kazmi, S. Foix, G. Alenyà, and H. J. Andersen, "Indoor and outdoor depth imaging of leaves with time-of-flight and stereo vision sensors: Analysis and comparison," *ISPRS journal of Photogrammetry and Remote Sensing*, vol. 88, pp. 128–146, 2014.

[17] S. Brahmbhatt, *Practical OpenCV*, 1st ed. New York, NY: Apress, 2013.

[18] D. Scharstein and R. Szeliski, "High-accuracy stereo depth maps using structured light," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, 2003, pp. I–I.

[19] M. Herbert, C. Caillas, E. Krotkov, I. Kweon, and T. Kanade, "Terrain mapping for a roving planetary explorer," in *Proceedings of IEEE International Conference on Robotics and Automation*, 1989, pp. 997–1002.

[20] R. Triebel, P. Pfaff, and W. Burgard, "Multi-level surface maps for outdoor terrain mapping and loop closing," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2006, pp. 2276–2282.

[21] G. K. Kraetzschmar, G. P. Gassull, and K. Uhl, "Probabilistic quadtrees for variable-resolution mapping of large environments," *IFAC Proceedings Volumes*, vol. 37, no. 8, pp. 675–680, 2004.

[22] M. Yguel, O. Aycard, and C. Laugier, "Update policy of dense maps: Efficient algorithms and sparse representation," in *Proceedings of Conference on Field and Service Robotics*, 2008, pp. 23–33.

[23] N. Fairfield, G. Kantor, and D. Wettergreen, "Real-time slam with octree evidence grids for exploration in underwater tunnels," *Journal of Field Robotics*, vol. 24, no. 1-2, pp. 03–21, 2007.

[24] B. L. Floriani, N. Palomeras, L. Weihmann, H. Simas, and P. Ridão, "Model-based underwater inspection via viewpoint planning using octomap," in *Proceedings of OCEANS 2017-Anchorage*, 2017, pp. 1–8.

[25] B. O. Arnesen, S. S. Sandøy, I. Schjølberg, J. A. Alfredsen, and I. B. Utne, "Probabilistic localization and mapping of flexible underwater structures using octomap," in *Proceedings of IEEE European Control Conference*, 2018, pp. 268–275.

[26] Y. Morales, J. Even, N. Kallakuri, T. Ikeda, K. Shinozawa, T. Kondo, and N. Hagita, "Visibility analysis for autonomous vehicle comfortable navigation," in *Proceedings of IEEE International Conference on Robotics and Automation*, 2014, pp. 2197–2202.

[27] K. Tong, Z. Ajanovic, and G. Stettinger, "Overview of tools supporting planning for automated driving," in *Proceedings of IEEE International Conference on Intelligent Transportation Systems*, 2020, pp. 1–8.

[28] E. Galceran, R. Campos, N. Palomeras, D. Ribas, M. Carreras, and P. Ridao, "Coverage path planning with real-time replanning and surface reconstruction for inspection of three-dimensional underwater structures using autonomous underwater vehicles," *Journal of Field Robotics*, vol. 32, no. 7, pp. 952–983, 2015.

[29] L. Zacchini, M. Franchi, and A. Ridolfi, "Sensor-driven autonomous underwater inspections: A receding-horizon rrt-based view planning solution for auvs," *Journal of Field Robotics*, 2022.

[30] C. Wang, L. Meng, S. She, I. M. Mitchell, T. Li, F. Tung, W. Wan, M. Q.-H. Meng, and C. W. de Silva, "Autonomous mobile robot navigation in uneven and unstructured indoor environments," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2017, pp. 109–116.

[31] H. He and B. Upcroft, "Nonparametric semantic segmentation for 3d street scenes," in *Proceedings of IEEE/RSJ international conference on intelligent robots and systems*, 2013, pp. 3697–3703.

[32] X. Ruan, P. Guo, and J. Huang, "A semantic octomap mapping method based on cbam-pspnet," *Journal of Web Engineering*, pp. 879–910, 2022.

[33] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: A versatile and accurate monocular slam system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[34] R. Mur-Artal and J. D. Tardós, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[35] S. W. Golomb, "Polyominoes: Puzzles, patterns, problems, and packings," Princeton, NJ, 1994.

[36] W. Yang, K. Wang, and W. Zuo, "Neighborhood component feature selection for high-dimensional data," *Journal of Computers*, vol. 7, no. 1, pp. 161–168, 2012.

[37] C.-C. Cheng, G.-J. Peng, and W.-L. Hwang, "Subband weighting with pixel connectivity for 3-d wavelet coding," *IEEE Transactions on Image Processing*, vol. 18, no. 1, pp. 52–62, 2008.

[38] M. Smith, I. Baldwin, W. Churchill, R. Paul, and P. Newman, "The new college vision and laser data set," *The International Journal of Robotics Research*, vol. 28, no. 5, pp. 595–599, 2009.

[39] T. Fawcett, "An introduction to roc analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, 2006.

[40] "Awesome slam datasets," Available online: https://sites.google.com/view/awesome-slam-datasets, accessed on 1/6/2022.

[41] L. Meyer, M. Smíšek, A. Fontan Villacampa, L. Oliva Maza, D. Medina, M. J. Schuster, F. Steidle, M. Vayugundla, M. G. Müller, B. Rebele *et al.*, "The madmax data set for visual-inertial rover navigation on mars," *Journal of Field Robotics*, vol. 38, no. 6, pp. 833–853, 2021.

[42] I. Ali, A. Durmush, O. Suominen, J. Yli-Hietanen, S. Peltonen, J. Collin, and A. Gotchev, "Finnforest dataset: A forest landscape for visual slam," *Robotics and Autonomous Systems*, vol. 132, p. 103610, 2020.

[43] J. Delmerico, T. Cieslewski, H. Rebecq, M. Faessler, and D. Scaramuzza, "Are we ready for autonomous drone racing? the uzh-fpv drone racing dataset," in *Proceedings of IEEE International Conference on Robotics and Automation*, 2019, pp. 6713–6719.

[44] Z. Zhu, F. Xu, C. Yan, X. Hao, X. Ji, Y. Zhang, and Q. Dai, "Real-time indoor scene reconstruction with rgbd and inertial input," in *Proceedings of IEEE International Conference on Multimedia and Expo*, 2019, pp. 7–12.

[45] T. Pire, M. Mujica, J. Civera, and E. Kofman, "The rosario dataset: Multisensor data for localization and mapping in agricultural environments," *The International Journal of Robotics Research*, vol. 38, no. 6, pp. 633–641, 2019.

[46] M. Ferrera, V. Creuze, J. Moras, and P. Trouvé-Peloux, "Aqualoc: An underwater dataset for visual–inertial–pressure localization," *The International Journal of Robotics Research*, vol. 38, no. 14, pp. 1549–1559, 2019.

[47] S. Cortés, A. Solin, E. Rahtu, and J. Kannala, "Advio: An authentic dataset for visual-inertial odometry," in *Proceedings of the European Conference on Computer Vision*, 2018, pp. 419–434.

[48] C. Chen, P. Zhao, C. X. Lu, W. Wang, A. Markham, and N. Trigoni, "Oxiod: The dataset for deep inertial odometry," *arXiv preprint arXiv:1809.07491*, 2018.

[49] Y. Choi, N. Kim, S. Hwang, K. Park, J. S. Yoon, K. An, and I. S. Kweon, "Kaist multi-spectral day/night data set for autonomous and assisted driving," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 3, pp. 934–948, 2018.

[50] D. Schubert, T. Goll, N. Demmel, V. Usenko, J. Stückler, and D. Cremers, "The tum vi benchmark for evaluating visual-inertial odometry," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2018, pp. 1680–1687.

[51] A. Z. Zhu, D. Thakur, T. Özaslan, B. Pfrommer, V. Kumar, and K. Daniilidis, "The multivehicle stereo event camera dataset: An event camera dataset for 3d perception," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2032–2039, 2018.

[52] S. Golodetz, T. Cavallari, N. A. Lord, V. A. Prisacariu, D. W. Murray, and P. H. Torr, "Collaborative large-scale dense 3d reconstruction with online inter-agent pose optimisation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 11, pp. 2895–2905, 2018.

[53] M. Miller, S.-J. Chung, and S. Hutchinson, "The visual–inertial canoe dataset," *The International Journal of Robotics Research*, vol. 37, no. 1, pp. 13–20, 2018.

[54] E. Mueggler, H. Rebecq, G. Gallego, T. Delbruck, and D. Scaramuzza, "The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and slam," *The International Journal of Robotics Research*, vol. 36, no. 2, pp. 142–149, 2017.

[55] A. Mallios, E. Vidal, R. Campos, and M. Carreras, "Underwater caves sonar data set," *The International Journal of Robotics Research*, vol. 36, no. 12, pp. 1247–1251, 2017.

[56] A. L. Majdik, C. Till, and D. Scaramuzza, "The zurich urban micro aerial vehicle dataset," *The International Journal of Robotics Research*, vol. 36, no. 3, pp. 269–273, 2017.

[57] K. Leung, D. Lühr, H. Houshiar, F. Inostroza, D. Borrmann, M. Adams, A. Nüchter, and J. Ruiz del Solar, "Chilean underground mine dataset," *The International Journal of Robotics Research*, vol. 36, no. 1, pp. 16–23, 2017.

[58] J. McCormac, A. Handa, S. Leutenegger, and A. J. Davison, "Scenenet rgb-d: Can 5m synthetic images beat generic imagenet pre-training on indoor segmentation?" in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 2678–2687.

[59] S. Griffith, G. Chahine, and C. Pradalier, "Symphony lake dataset," *The International Journal of Robotics Research*, vol. 36, no. 11, pp. 1151–1158, 2017.

[60] N. Chebrolu, P. Lottes, A. Schaefer, W. Winterhalter, W. Burgard, and C. Stachniss, "Agricultural robot dataset for plant classification, localization and mapping on sugar beet fields," *The International Journal of Robotics Research*, vol. 36, no. 10, pp. 1045–1052, 2017.

[61] W. Maddern, G. Pascoe, C. Linegar, and P. Newman, "1 year, 1000 km: The oxford robotcar dataset," *The International Journal of Robotics Research*, vol. 36, no. 1, pp. 3–15, 2017.

[62] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The euroc micro aerial vehicle datasets," *The International Journal of Robotics Research*, vol. 35, no. 10, pp. 1157–1163, 2016.

[63] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2d lidar slam," in *Proceedings of IEEE International Conference on Robotics and Automation*. IEEE, 2016, pp. 1271–1278.

[64] J. Engel, V. Usenko, and D. Cremers, "A photometrically calibrated benchmark for monocular visual odometry," *arXiv preprint arXiv:1607.02555*, 2016.

[65] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 3213–3223.

[66] P. Oettershagen, T. Stastny, T. Mantel, A. Melzer, K. Rudin, P. Gohl, G. Agamennoni, K. Alexis, and R. Siegwart, "Long-endurance sensing and mapping using a hand-launchable solar-powered uav," in *Field and Service Robotics*. Springer, 2016, pp. 441–454.

[67] O. Wasenmüller, M. Meyer, and D. Stricker, "Corbs: Comprehensive rgb-d benchmark for slam using kinect v2," in *Proceedings of IEEE Winter Conference on Applications of Computer Vision*, 2016, pp. 1–7.

[68] N. Carlevaris-Bianco, A. K. Ushani, and R. M. Eustice, "University of michigan north campus long-term vision and lidar dataset," *The International Journal of Robotics Research*, vol. 35, no. 9, pp. 1023–1035, 2016.

[69] D. Caruso, J. Engel, and D. Cremers, "Large-scale direct slam for omnidirectional cameras," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2015, pp. 141–148.

[70] A. Handa, T. Whelan, J. McDonald, and A. J. Davison, "A benchmark for rgb-d visual odometry, 3d reconstruction and slam," in *Proceedings of IEEE International Conference on Robotics and Automation*, 2014, pp. 1524–1531.

[71] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti

dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[72] J. Shotton, B. Glocker, C. Zach, S. Izadi, A. Criminisi, and A. Fitzgibbon, "Scene coordinate regression forests for camera relocalization in rgb-d images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 2930–2937.

[73] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 573–580.

[74] P. Furgale, P. Carle, J. Enright, and T. D. Barfoot, "The devon island rover navigation dataset," *The International Journal of Robotics Research*, vol. 31, no. 6, pp. 707–713, 2012.

[75] G. Pandey, J. R. McBride, and R. M. Eustice, "Ford campus vision and lidar data set," *The International Journal of Robotics Research*, vol. 30, no. 13, pp. 1543–1552, 2011.

[76] T. Peynot, S. Scheding, and S. Terho, "The marulan data sets: Multi-sensor perception in a natural environment with challenging conditions," *The International Journal of Robotics Research*, vol. 29, no. 13, pp. 1602–1607, 2010.

[77] S. Grushko, A. Vysockỳ, V. K. Jha, R. Pastor, E. Prada, L. Miková, and Z. Bobovskỳ, "Tuning perception and motion planning parameters for moveit! framework," *MM Science Journal*, pp. 4154–4163, 2020.

[78] F. C. Ferreira, M. Zaera, P. T. Karfakis, and M. S. Couceiro, "Tailoring 3d mapping frameworks for field robotics," in *ICRA 2022 Workshop in Innovation in Forestry Robotics: Research and Industry Adoption*, 2022.

[79] M. Aernouts, B. Bellekens, and M. Weyn, "Mapfuse: complete and realistic 3d modelling," *Journal of Robotics*, vol. 2018, 2018.

[80] J. Zhang, S. Liu, B. Gao, and C. Zhong, "An improvement algorithm for octomap based on rgb-d slam," in *Proceedings of Chinese Control and Decision Conference*, 2018, pp. 5006–5011.

[81] M. G. Jadidi, L. Gan, S. A. Parkison, J. Li, and R. M. Eustice, "Gaussian processes semantic map representation," *arXiv preprint arXiv:1707.01532*, 2017.

[82] I. Armeni, S. Sax, A. R. Zamir, and S. Savarese, "Joint 2d-3d-semantic data for indoor scene understanding," *arXiv preprint arXiv:1702.01105*, 2017.

[83] L. Sun, Z. Yan, A. Zaganidis, C. Zhao, and T. Duckett, "Recurrent-octomap: Learning state-based map refinement for long-term semantic mapping with 3-d-lidar data," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3749–3756, 2018.

[84] K. Doherty, T. Shan, J. Wang, and B. Englot, "Learning-aided 3-d occupancy mapping with bayesian generalized kernel inference," *IEEE Transactions on Robotics*, vol. 35, no. 4, pp. 953–966, 2019.

[85] K. Doherty, J. Wang, and B. Englot, "Bayesian generalized kernel inference for occupancy map prediction," in *Proceedings of IEEE International Conference on Robotics and Automation*, 2017, pp. 3118–3124.

[86] J. Chen and S. Shen, "Improving octree-based occupancy maps using environment sparsity with application to aerial robot navigation," in *Proceedings of IEEE International Conference on Robotics and Automation*, 2017, pp. 3656–3663.

[87] L. Zhang, L. Wei, P. Shen, W. Wei, G. Zhu, and J. Song, "Semantic slam based on object detection and improved octomap," *IEEE Access*, vol. 6, pp. 75 545–75 559, 2018.

[88] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*, 2nd ed. Cambridge: Cambridge University Press, 2003.

[89] J. Heikkila and O. Silvén, "A four-step camera calibration procedure with implicit image correction," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1997, pp. 1106–1112.

[90] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *Proceedings of International Conference on Computer Vision*, 2011, pp. 2564–2571.

[91] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Proceedings of European Conference on Computer Vision*, 2006, pp. 430–443.

[92] C. G. Harris, M. Stephens *et al.*, "A combined corner and edge detector," in *Proceedings of Alvey Vision Conference*, vol. 15, no. 50, 1988, pp. 10–5244.

[93] E. H. Adelson, C. H. Anderson, J. R. Bergen, P. J. Burt, and J. M. Ogden,

"Pyramid methods in image processing," *RCA Engineer*, vol. 29, no. 6, pp. 33–41, 1984.

[94] P. L. Rosin, "Measuring corner properties," *Computer Vision and Image Understanding*, vol. 73, no. 2, pp. 291–307, 1999.

[95] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "Brief: Binary robust independent elementary features," in *Proceedings of European Conference on Computer Vision*, 2010, pp. 778–792.

[96] T. Taketomi, H. Uchiyama, and S. Ikeda, "Visual slam algorithms: a survey from 2010 to 2016," *IPSJ Transactions on Computer Vision and Applications*, vol. 9, no. 1, p. 16, 2017.

[97] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. Cambridge, MA: MIT Press, 2005.

[98] H. Moravec and A. Elfes, "High resolution maps from wide angle sonar," in *Proceedings of IEEE international Conference on Robotics and Automation*, vol. 2, 1985, pp. 116–121.

[99] J. V. Stone, *Bayes' rule: A tutorial introduction to Bayesian analysis*. Sebtel Press, 2013.

[100] A. Daskalaki, *Handbook of Research on Systems Biology Applications in Medicine*. Hershey, PA: IGI Global, 2008.

[101] Y. Miao, A. J. Hunter, and I. Georgilas, "Parameter reduction and optimisation for point cloud and occupancy mapping algorithms," *Sensors*, vol. 21, no. 21, 2021.

[102] K. Kise, A. Sato, and M. Iwata, "Segmentation of page images using the area voronoi diagram," *Computer Vision and Image Understanding*, vol. 70, no. 3, pp. 370–382, 1998.

[103] P. Bhattacharya and M. L. Gavrilova, "Voronoi diagram in optimal path planning," in *Proceedings of International Symposium on Voronoi Diagrams in Science and Engineering*, 2007, pp. 38–47.

[104] N. A. B. Ab Aziz, A. W. Mohemmed, and M. Y. Alias, "A wireless sensor network coverage optimization algorithm based on particle swarm optimization and voronoi diagram," in *Proceedings of IEEE International Conference on Networking, Sensing and Control*, 2009, pp. 602–607.

[105] F. P. Preparata and M. I. Shamos, *Computational geometry: An introduction*, 1st ed. New York, NY: Springer, 1985.

[106] M. Erwig, "The graph voronoi diagram with applications," *Networks: An International Journal*, vol. 36, no. 3, pp. 156–163, 2000.

[107] B. C. Barnes, D. W. Siderius, and L. D. Gelb, "Structure, thermodynamics, and solubility in tetromino fluids," *Langmuir*, vol. 25, no. 12, pp. 6702–6716, 2009.

[108] J. H. Min and C. Jeong, "A binary classification method for bankruptcy prediction," *Expert Systems with Applications*, vol. 36, no. 3, pp. 5256–5263, 2009.

[109] M. Khashei, S. Eftekhari, and J. Parvizian, "Diagnosing diabetes type ii using a soft intelligent binary classification model," *Review of Bioinformatics and Biometrics*, vol. 1, no. 1, pp. 9–23, 2012.

[110] S. Choi, Y. J. Kim, S. Briceno, and D. Mavris, "Prediction of weather-induced airline delays based on machine learning algorithms," in *Proceedings of IEEE Digital Avionics Systems Conference*, 2016, pp. 1–6.

[111] Y. Wang and B. Sridhar, "Convective weather forecast accuracy analysis at center and sector levels," in *Proceedings of IEEE Digital Avionics Systems Conference*, 2010, pp. 2.B.2–1–2.B.2–17.

[112] J. Yerushalmy, "Statistical problems in assessing methods of medical diagnosis, with special reference to x-ray techniques," *Public Health Reports (1896-1970)*, pp. 1432–1449, 1947.

[113] D. M. Powers, "Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation," *Journal of Machine Learning Technologies*, vol. 2, no. 1, pp. 37–63, 2011.

[114] S. V. Stehman, "Selecting and interpreting measures of thematic classification accuracy," *Remote sensing of Environment*, vol. 62, no. 1, pp. 77–89, 1997.

[115] A. P. Bradley, "The use of the area under the roc curve in the evaluation of machine learning algorithms," *Pattern Recognition*, vol. 30, no. 7, pp. 1145–1159, 1997.

[116] S. J. Swamidass, C.-A. Azencott, K. Daily, and P. Baldi, "A croc stronger than roc: Measuring, visualizing and optimizing early retrieval," *Bioinformatics*, vol. 26, no. 10, pp. 1348–1356, 2010.

[117] S. D. Walter, "The partial area under the summary roc curve," *Statistics in Medicine*, vol. 24, no. 13, pp. 2025–2040, 2005.

[118] K. Mori, T. Oura, H. Noma, and S. Matsui, "Cancer outlier analysis based on mixture modeling of gene expression data," *Computational and Mathematical Methods in Medicine*, vol. 2013, 2013.

[119] E. Turro, N. Bochkina, A.-M. K. Hein, and S. Richardson, "Bgx: A bioconductor package for the bayesian integrated analysis of affymetrix genechips," *BMC Bioinformatics*, vol. 8, no. 1, p. 439, 2007.

[120] T. Yu, "Rocs: Receiver operating characteristic surface for class-skewed high-throughput data," *PLoS ONE*, vol. 7, no. 7, p. e40598, 2012.

[121] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning*, 1st ed. New York, NY: Springer, 2013.

[122] C. M. Bishop, *Pattern recognition and machine learning*, 1st ed. New York, NY: Springer, 2006.

[123] W. Ambrosius, *Topics in Biostatistics*, ser. Methods in Molecular Biology. Totowa, NJ: Humana Press, 2007.

[124] C. C. Aggarwal, *Neural Networks and Deep Learning.* Cham: Springer, 2018.

[125] F. Burden and D. Winkler, "Bayesian regularization of neural networks," *Artificial Neural Networks*, pp. 23–42, 2008.

[126] J. Benesty, J. Chen, Y. Huang, and I. Cohen, *Noise reduction in speech processing.* Berlin, Heidelberg: Springer, 2009.

[127] Z. Wang and A. C. Bovik, "Mean squared error: Love it or leave it? A new look at signal fidelity measures," *IEEE Signal Processing Magazine*, vol. 26, no. 1, pp. 98–117, 2009.

[128] R. Taylor, "Interpretation of the correlation coefficient: A basic review," *Journal of Diagnostic Medical Sonography*, vol. 6, no. 1, pp. 35–39, 1990.

[129] D. Holz, A. E. Ichim, F. Tombari, R. B. Rusu, and S. Behnke, "Registration with the point cloud library: A modular framework for aligning in 3-d," *IEEE Robotics and Automation Magazine*, vol. 22, no. 4, pp. 110–124, 2015.

[130] A. Aldoma, Z.-C. Marton, F. Tombari, W. Wohlkinger, C. Potthast, B. Zeisl, R. B. Rusu, S. Gedikli, and M. Vincze, "Tutorial: Point cloud library: Three-

dimensional object recognition and 6 dof pose estimation," *IEEE Robotics and Automation Magazine*, vol. 19, no. 3, pp. 80–91, 2012.

[131] A. M. Ramiya, R. R. Nidamanuri, and R. Krishnan, "Segmentation based building detection approach from lidar point cloud," *The Egyptian Journal of Remote Sensing and Space Science*, vol. 20, no. 1, pp. 71–77, 2017.

[132] L. A. G. Velandia, R. D. Hernández, O. F. Avilés, J. M. Rosário *et al.*, "Mapping of indoor environments using point cloud library (pcl)," *International Journal of Applied Engineering Research*, vol. 11, no. 8, pp. 5704–5713, 2016.

[133] Y. Miao, I. Georgilas, and A. J. Hunter, "A k-nearest neighbours based inverse sensor model for occupancy mapping," in *Proceedings of Annual Conference Towards Autonomous Robotic Systems*, 2019, pp. 75–86.

[134] Y. Miao, A. J. Hunter, and I. Georgilas, "An occupancy mapping method based on k-nearest neighbours," *Sensors*, vol. 22, no. 1, 2022.

[135] B. P. Welford, "Note on a method for calculating corrected sums of squares and products," *Technometrics*, vol. 4, no. 3, pp. 419–420, 1962.

[136] L. E. Ortiz, E. V. Cabrera, and L. M. Gonçalves, "Depth data error modeling of the zed 3d vision sensor from stereolabs," *Electronic Letters on Computer Vision and Image Analysis*, vol. 17, no. 1, pp. 1–15, 2018.

[137] R. A. Fisher and F. Yates, *Statistical tables: For biological, agricultural and medical research*, 3rd ed. London: Oliver and Boyd, 1948.

[138] R. Durstenfeld, "Algorithm 235: Random permutation," *Communications of the ACM*, vol. 7, no. 7, p. 420, 1964.