UNIVERSITY OF GLOUCESTERSHIRE

The Park, Cheltenham GL50 2RH

Innovative Methods for 3D Point Cloud Processing of Large Data Sets and Its Practical Implementations

Rishika Singh

PhD

31 August 2022

‘A thesis submitted to the University of Gloucestershire in accordance with the requirements for the degree of Doctor of Philosophy in the School of Computing and Engineering.’

Abstract

Various point cloud processing applications demand fast and accurate results for extracting feature information from the data. Given that point clouds are implemented in various fields, this thesis focuses on the point clouds used by surveyors and civil engineers for geographic information systems. Examples of point clouds used within the industry include urban sites, city blocks, terrains for road development, construction sites, quarries, mines, etc.

The technological advancements (evolution of laser scanners) that allow the data to be captured in millions created their own recurring problems. The algorithms developed in this thesis are targeted at large point clouds containing various features and shapes. However, while capturing the point clouds, several outliers and noise are captured with the regular data due to the reflection of surfaces like glass and mirrors or weather conditions. Therefore, the detection and deletion of these outliers and noise are required to address and simplify the feature detection process. Hence, point cloud filtration is the first step of point cloud processing. After filtration, the data is relatively smaller and free from outliers and noise. The next step is to detect and extract the features from the point clouds and perform segmentation and other points analysis techniques. **This thesis proposes, designs, develops and implements the methods and algorithms for robust and efficient point cloud processing. The processing includes filtrations followed by detecting and extracting primitive shapes such as planes, edges and cylinders**.

Contributions: The first contribution of this thesis is the method of removing noise and outliers using the designed tools. The second contribution is a novel PCA-based algorithm for detecting edges and edge streams in point clouds. Finally, a voxel-based algorithm to detect trunks and pole-like objects. These proposed methods and algorithms directly benefit the processing of the point clouds with properties like filtration, extraction, segmentation, clusterisation and accuracy. **The results of the proposed methods and algorithms are implemented on commercial software used by UK and worldwide users.**

Declaration

I declare that the work in this thesis was carried out in accordance with the regulations of the University of Gloucestershire and is original except where indicated by specific reference in the text. No part of the thesis has been submitted as part of any other academic award. The thesis has not been presented to any other education institution in the United Kingdom or overseas.

All views and opinions presented in this PhD thesis are solely mine and not, in any way, those of the University of Gloucestershire.

Signed ……Rishika Singh…….. Date……31/8/2022…….

doi: 10.46289/9K7C33MM

Acknowledgements

First, I want to thank my supervisor Prof. Shujun Zhang, whose spirited enthusiasm for the field and his dedication powered me with motivation. Without his proper guidance, inspiration, and lots of scolding taught me a great deal and made this work possible.

I want to thank and express my gratitude to my former colleague and pops, Nigel Lorriman who helped me a lot with my thesis proofreading and motivated me to keep going and never give up.

I want to thank my papa Dr Shivesh Singh who encouraged me to start my research journey, and my mummy Dr Rashmi Singh, who supported, inspired, and helped me by proofreading my thesis with her expertise. Having professors in the family and growing up seeing them with research papers, books and thesis have motivated me to achieve my goals.

My sister Devika Singh tried to inspire and encourage me as much as possible through all those WhatsApp calls. It pushed me to action and, at the same time, filled me with positive energy.

Last but not least, I want to thank my husband, Harshit Srivastava, wholeheartedly for his patience, motivation, support, love, and yet to improve cooking skills. I cannot thank you enough for taking care of me while I was busy between my research and full-time job, appreciating this work and listening to all my problems (although I know I put you to sleep several times). This work would not have been possible without you.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

2D	Two Dimensional
3D	Three Dimensional
AEC	Architecture, Engineering and Construction
ANN	Artificial Neural Network
ALS	Aerial/Airborne Laser Scanning
BIM	Building Information Modelling
BEV	Bird's Eye View
CAD	Computer-Aided Design
CNN	Convolution Neural Network
DEM	Digital Elevation Model
DL	Deep Learning
DTM	Digital Terrain Model
DSM	Digital Surface Model
EDM	Electronic Distance Measurement
FN	False Negatives
FP	False Positives
FD	Feature Detection
GCN	Graph Convolution Network
GIS	Geoinformation System

GM	Gaussian Model
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
HD	High Definition
HT	Hough Transform
IMU	Inertial Measurement Unit
KNN	K-Nearest Neighbour
LiDar	Light Detection and Ranging
LSS	Land Survey System
MLS	Mobile Laser Scanning
NN	Neural Network
IRLS	Iteratively Reweighted Least Squares
PC	Principal Component
PCS	Point Cloud Segmentation
PCL	Point Cloud Library
PCD	Point Cloud Data
RAM	Random Access Memory
RANSAC	Random Sample Consensus
RGB	Red, Green, Blue
SAN	Spatial Aggregation Network
SDLC	Software Development Life Cycle

SOR	Statistical Outlier Removal
SVM	Support Vector Machine
TN	True Negative
TP	True Positive
TLS	Terrestrial Laser Scanning
UAV	Unmanned Aerial Vehicle

Glossary

Following are the technical terminologies used in this thesis.

- Point cloud – A set of data points in space is called a point cloud (*What Are Point Clouds ?, 2018*). Generally, point clouds are a 3D system where each point has its cartesian coordinates (X, Y, Z) and represent the outer surface of objects.

- Laser Scanners – "Scanning is a popular land surveying method to accurately measure and collect data from objects, surfaces, buildings and landscapes". Laser scanners collect information as point clouds using laser beams (*What Is Laser Scanning and How Can It Be Used ?, 2020*). Then, the point cloud data is analysed to extract valid information to create 3D models.

- Principal component analysis - Principal component analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components

- Point cloud processing – The process of extracting information from large point clouds to convert it into models is point cloud processing. Typical point cloud processing operations are classification, noise filtering (to clean or eliminate outliers), segmentation, edge and boundary identification, feature recognition and modelling.

- Feature detection – In computer vision, feature detection refers to methods for computing abstraction and valid information in relation to the objects. These objects are the shapes and features extracted into models. Examples of shapes and features include building footprints, pipes, kerbs, breaks of slope, building edges, rough surfaces, trees, dip and volume of the terrain, utility holes, road furniture and road marking.

- Geographical Information System (GIS) - A geographical information system is a software application for capturing, storing, manipulating, analysing, managing and presenting all spatial or geographical data types.

- Voxel – In 3D computer graphics, a voxel is a grid in 3D space. These fixed-sized grids have the same length, width and depth. In this thesis, voxels are used for space partitioning and clustering.

- Digitising – The process of extracting points or links from the point clouds to models is called digitising.

- DTM - In 3D computer graphics, a digital terrain model is a mathematical representation of the terrain or topographical surface of the earth to capture the unique elevation (in the form of a grid in which a unique elevation value is assigned to each pixel (Mallet and David, 2016).

- PointNet – It is a deep neural network that consumes the 3D point cloud and provides a unified approach for tasks such as classification and segmentation (Qi *et al*., 2017)

Chapter 1 Introduction

1.1 Background

In recent years, as computer systems have become more powerful, their role continues to have a profound impact in almost every field of human endeavour. Accordingly, the specialisms of surveying and geoinformation systems have similarly evolved in line with those technological advancements. Modern technologies, such as Global Positioning System (GPS), Geographic Information Systems (GIS) and high-speed laser scanner systems, have resulted in more accurate, higher resolution and faster surveying methods.

Surveying (Geomatics) is the branch of science that deals with collecting, analysing, and interpreting data relating to the earth's surface. The ancient Egyptians were the earliest proponents of this practice as early as 1400 BC. They created perfectly aligned pyramids using simple tools and basic geometry. They used ropes to measure distances by tying knots at various intervals (Thank the Egyptians; The History of Surveying & Mapping, 2019). In 1576 Joshua Habermel invented a land surveying tool called the Theodolite, comprising a compass and tripod (Avram *et al.*, 2016). The first example of using laser scanners for measuring buildings, together with basic tools for data analysis, appeared in 1977 (Thrun, Burgard and Fox, 1998).

Laser scanners are used as a surveying tool in a variety of fields. The advancement of laser scanners enabled surveyors to capture high-precision data in various environmental conditions (Fröhlich and Mettenleiter, 2004). The introduction of airborne and terrestrial laser scanning technology has enabled the collection of large 3D scanned data of urban scenes and landscapes. These data, by their very nature, are very large. Therefore, computer software innovations emerged to handle these large data.

The scanner-produced data is point clouds which provide the real-world context for recreating and extracting valid information about objects. Point clouds are like human vision as laser scanners typically have a limited field of view and partial perception of distant objects. The

scanned point density depends on the distance between the scanner and the object, the angle of incidence, and the environment or weather (Gargoum and El-Basyouny, 2019).

Due to the complexity and geometric details captured by the point cloud, it becomes essential to extract important geometrical and non-geometrical information and identify or classify the features of these large data sets, which requires efficient and accurate point cloud processing. Therefore, **point cloud processing** is vital and uses various computation algorithms/methods for extracting features.

With the increasing equipment complexity and powerful applications, the advancement of calculations and mathematical models has been developed. Mathematical advances are correlated to technology development. This thesis is a result of several years of research investigating solutions to the challenges that emerged while processing point clouds, primarily in the surveying industry.

Figure 1.1 Point Cloud of Celtic Manor Resort, Newport Wales

Point Clouds:

Point clouds are the real-world representation, as shown in Fig 1.1. A point cloud is a huge collection of individual points plotted in 3D space. These points are captured by 3D scanners; for example, a road is scanned, where each virtual point of the road point cloud represents a real-world point (Point Clouds for Beginners: Your Questions Answered, 2022). The scanner combines vertical and horizontal angles by the laser beam to calculate each point's 3D X, Y, and Z coordinates. These points generally contain the colour (RGB) and intensity values. Colour is usually captured by a separate camera-mounted inline with the scanner and added to each point post-capture. Intensity is recorded as the return strength of the laser beam. The lower intensity values indicate low reflectivity, while a high number indicates high reflectivity (Gregorius B, 2019).

These details are converted into a digital 3D model that provides an accurate, detailed picture of the scanned object (Point Clouds for Beginners Your questions answered, 2022). A dense point cloud can capture every minute detail compared to a low-density point cloud. An example of the details shown in Fig 1.1, such as the wall's texture and small clock features, can be seen if the point cloud data is zoomed in.

The point clouds are 3D formats that support the rendering and navigation in 3D to virtually view the data on the computer screen from several angles and perspectives. The point cloud data set can be anything from manufactured parts of cars for quality inspection to large geographical areas or forests for inventories and detailed analysis. In geographic information systems, point clouds are captured to analyse terrain, non-terrain and elevation (buildings) data.

History of Point Clouds:

Since the 1960s, point clouds and laser scanners have been used in various industries. In 1971, the first publicly known LiDar surface mapping was achieved during the Apollo 15 lunar mission, where the Moon's surface was mapped to create height maps (Kaula *et al.*, 1973).

By the 1980s, military and space agencies used them to scan the terrain. It was also used by aircraft to accurately plot their position and map in great detail (Rooms Filip, 2019). By the 1990s, with the evolution of technology and computers, point clouds were introduced to various

industries such as architecture, engineering and archaeology. Everything in the real world can be scanned and transformed into point clouds, from landscapes to buildings to tiny archaeological artefacts (Senior, 2021). As a result, point clouds gained popularity quickly.

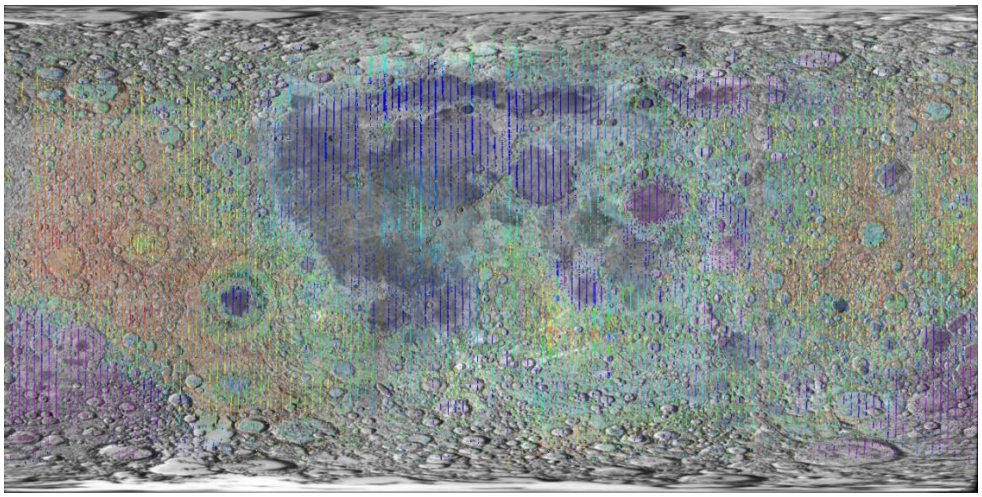


Figure 1.2 LiDar map of the moon surface during the Clementine mission launched by NASA on January 25, 1994 (Source: Hamilton, 1995)

The point cloud applications and usages are as follows 1) Architects and construction professionals use AEC (Architecture, Engineering and Construction) software applications to map and extract real-world data 2) Archaeologists use point clouds to analyse the terrain surfaces or capture artefacts in detail 3) Medical professionals use point clouds for reconstructive treatments, and 4) Entertainment companies use point clouds for games and visualisation (Senior, 2021).

Point Clouds Generation:

There are two ways to capture point clouds 1) Laser Scanners and 2) Photogrammetry. As the name suggests, laser scanners collect data using rapid laser pulses to gather up to millions of accurate measurements per second, and the photogrammetry method uses overlapping photographs to construct it into 3D Space (Higgins, 2021).

In photogrammetry, the objects are scanned once or multiple times, depending on the viewports of the scanned object. On the other hand, laser scanners require multiple scans because the laser beam only captures the data points of the 3D surface in a direct scanner's line of sight. As a result, laser scanners generally have higher accuracy than photogrammetry scanners (Higgins, 2021). These scans of different viewports are then merged or stitched to form a point cloud. The merging is also known as 'Registration'. The laser scanners that capture the point clouds are terrestrial, airborne, mobile, or handheld.

- Airborne laser scanners are usually used to capture the earth's surface,
- Mobile scanners are non-invasive and quickly capture the data ideal for asset management, utilities, planning, disaster management, tunnel, airport design,
- The terrestrial scanner emits constant waves of varying lengths and is reflected back to the scanner. They are ideal for architectural construction, surveying, engineering, planning and forensics (O'Day, 2013)

The point cloud data is stored in various formats. Some popular formats are ASCII, E57, PTS, FLS, LAS, LAZ, and TXT. In this thesis, the point clouds captured by laser scanners are used for the case study and testing of proposed algorithms.

Point Cloud in Various Industries:

It is a non-intrusive, accurate and the fastest way to capture lots of data for feature detection and analysis in 3D. Point cloud data is useful to a variety of industries. For example, museums and stadiums do not need to shut down to be measured, or a policeman captures a vehicle collision site to analyse the cause of the crash. Several industries use point clouds, such as

- Land Surveying
- Construction
- Architecture
- Environmental monitoring
- City planning and Civil engineering
- Manufacturing
- Digital designs

- Archaeology
- Visualisation

Point cloud has become the primary way to collect data among construction and architecture industries, saving their time and on-site costs. Manufacturing industries adopt point clouds to design and model the parts for visualising and quality assessment. (Point Clouds for Beginners Your questions answered, 2022). In the construction industry, laser scanners are used to capture minute details of buildings, plan extensions, and renovate or document the progress of building projects (Nicole, 2021). 3D construction models and city development sites are helpful for planning and quality assurance at each stage of construction. The BIM (Building Information Model) has already become the standard for plotting and planning buildings (Thomson, 2019).

1.2 Research Motivation

Point clouds acquired from the real world have several challenges. The most significant challenge is the identification of objects, shapes or features of the real world in point clouds. The challenge does not depend on the acquisition process but on the point cloud's natural property and lack of information connectivity (Schnabel, Klein and Gumhold, 2010). The complex geometrical shapes such as planes, cones, cylinders, and spheres are comparatively easier to detect as compared to other non-geometrical shapes such as vegetation, vehicles, ghosts (people walking by), and other man-made objects. However, identifying these objects and structures in point clouds is very complicated, especially when the objects on top hide the underlying surface information. In addition, real-world scans are infused with massive noise and outliers. The huge challenge is to differentiate between these and true points. The presence of inevitable noise and outliers is due to

- reflective man-made surfaces such as mirrors, glasses on windows
- dust
- environmental conditions such as rain, lightning and wind
- moving objects such as people on the road, birds, passing vehicles

The algorithms processing point clouds should provide the functions to tackle these noise and outliers. The scanners also generate range-dependent noise during the data collection as the scanner sensors are based on the time of flight, optical triangulation, and multiple frequency phase shifts. Noise level varies as it is mainly affected by the light source on the scanning surface (Unnikrishnan, 2008). The divergence of the laser beam, either by reflection or if the light source causes point location uncertainty, can generate possible outliers or additional random errors across the point cloud. In addition, mixed pixel discrepancy is generated if more than one scanned surface is placed according to the line of sight. These mixed pixels are caused due to non-point spot size of the beam. Therefore, removing outliers or filtration of the point cloud is essential for fast and accurate geometrical object detection in point cloud processing (Tuley, Vandapel and Hebert, 2005).

The traditional methods used by the existing geographical information system typically have issues such as (1) systems are manual, time-consuming and have low accuracy while processing point clouds, (2) lack accuracy in identifying and extracting the geometrical features from point clouds, (3) lack a robust method for solving the problems of noise and outliers, (4) lack graphical presentation functions, (5) have few functions for generating high-quality meshes, grids etc. from a point cloud (6) have limited functions for efficiently and effectively processing huge point clouds and (7) have issues associated with RGB and intensity processing (Remondino, 2004; Devore *et al.*, 2013).

Point clouds are generally very large, containing millions or billions of points, depending on the real-world scanned area. The density of point clouds can be selected in the scanner before scanning. The density varies with the distance. The area near the scanner will be denser than the area far from the scanner. Identifying and categorising the geometrical and non-geometrical shapes in high-density points is easier and more efficient than in low-density points, which is challenging due to missing points and gaps. The research continues in the field of surveying and engineering to produce efficient and robust methods.

Therefore, it is essential and reasonable to investigate, design, and develop a robust algorithm/s that can handle the removal of noise and outliers, identify the objects in dense data sets, and categorise them accordingly.

1.3 Scope of Research

Many researchers have developed various methods for extracting information and important features. Since scanning technologies have evolved in recent years, researchers have commonly used scanner data (point clouds) for the accurate and detailed extraction of 3D spatial information. Due to the nature of point clouds and their various applications in different industries, this thesis has to have a more focused scope. As mentioned above, the focus is on point cloud processing in order to extract features and information. As the market for point clouds expanded, so did the solution for processing the information in point clouds. However, the problem remains the same with the variety of software and free tools options: not efficiently extracting features and information from large data. This thesis presents and proposes algorithms/methods in later chapters to address and solve the problems.

The thesis primarily focuses on point cloud processing from the surveyor's and civil engineer's view, concentrating more on the features that a surveyor and civil engineer would want to derive. The surveying and civil engineer users focus on feature extraction from point clouds into a survey (DTM), which can be modelled. This extraction is achieved through software by implementing various methods where the software controls the quality and appearance of the model. As surveying and civil industries are involved in planning and accessing the area, they expect the solution to be versatile and accurate (even by 1 mm). Examples of popular features are roads, kerbs, building footprints, trees, and poles rather than inside buildings or vehicles.

For the collection of point clouds, two methods have prevailed. These methods are laser scanning and photogrammetry. Although both technologies are used for data acquisition, laser scanning is most popular for collecting urban and non-urban data sets.

Terrestrial and aerial laser scanning has become the popular land surveying method (*What Is Laser Scanning and How Can It Be Used?, 2020*). Photogrammetry is unsuitable for the surveying world as photogrammetry is relatively inaccurate, more expensive, has poor textures, is slower than traditional mapping and requires many photos to generate the 3D model (Puzzo, 2021). Therefore photogrammetry is not considered in the scope of this thesis.

As a note of clarification, wherever the term surveyor has been used, it exclusively means a land surveyor. The thesis focuses on the technology aspect of point cloud processing rather than the managerial aspect. Once the point clouds are collected, the results are large data sets.

> *"The process of collecting, measuring and using these point clouds to design models of the target object or surface is called Point Cloud Processing*" (*What Is Point Cloud Processing and Why Is It Important?, 2019*).

The thesis presents proposed solutions for processing these point clouds (filtration, edge detection, and feature extraction) that could be adaptable in different scenes of point clouds (urban, forest, land, building and fields) with high accuracy, flexibility, efficiency and with less user intervention.

1.4 Aims & Research Objectives

The research motivation generates these research objectives. These objectives are formulated to guide the research:

- RO1 – Identify, analyse, and evaluate the problems associated with existing methods and algorithms for 3D point cloud processing, including filtration, classification, edge detection and segmentation.

- RO2 - Research, design, and develop a new method for filtration to remove bad points (outliers and noise) and conserve good points.

- RO3 – Identify, analyse and evaluate the problems associated with existing methods and algorithms for feature extraction (edge detection and tree trunks and pole-like object detection).

- RO4 – Research, design, develop and test a new PCA-based algorithm for 3D point cloud edge detection to improve the efficiency, accuracy and enhance user experiences.

- RO5 – Research, design, develop and test a new Voxel-based algorithm for 3D point cloud feature extraction of tree trunks and pole-like objects to improve the efficiency, accuracy and enhance user experiences.

These research objectives are most useful to the surveyors and civil engineers in the UK using geographic information systems (GIS) that help them to generate models such as DTM (digital terrain models), 3D surface models, quarry and earthworks design and BIM (building information modelling). Also, the users who generate the models use point clouds for quarry and mining, waste management, planning, collision investigation, dredging, and coastal defence.

This thesis aims to propose, design, and develop new algorithms to efficiently and accurately perform point cloud processing to solve the research objectives.

1.5 Thesis Contributions to New Knowledge Generation

In this thesis, the key steps to point cloud processing has been shown and presented by the proposed methods and algorithms. The point cloud processing steps include the filtration methods, feature detection and extraction of edges. During the design and development of these methods, tools and techniques are used to analyse, evaluate and explain the real-world scenarios within point clouds.

The principal contributions of the thesis are as follows:

- An introduction, design, and development of methods to handle outliers and noise that are defined using the point's characteristics in point clouds. This results in designing fast and easy tools for removing types of noise and outliers from various kinds of point cloud data. The proposed methods are "NR-S" (Noise Removal using Sphere), "NR-B" (Noise Removal using Box) and "OF-OB" (Outlier Filtration using Octree Boxes), which are applied by tools called search sphere, Octbox and 3D boxes.

- After removing the noise and outliers, the research and study of the feature extractions emerged. The first and topmost important feature that is highly demanded in the field of surveyors and civil engineers is edges. Hence, an efficient and robust "PCA-based algorithm for real-time Edge Detection" in the large point cloud data is proposed, designed, developed, and implemented.

- The edge detection works in real-time in the 3D point clouds. However, the process is manual. To address this, the proposed algorithm for detecting edges is further extended to a method called "Edge Stream". The edge stream is an automated version of finding the edge with user-controlled parameters.

- Based on the study, the other important features are tree trunks and pole-like objects. To address that, a feature recognition algorithm for detecting cylindrical objects, trees and pole structures from urban point cloud data. This study led to the design, development, and implementation of the proposed Voxel-based algorithm to detect cylindrical objects in 3D point clouds.

- While implementing the proposed algorithm for cylindrical feature detection, classification is required between ground and non-ground points as the important features to be detected are above the ground. Therefore, the sampling and extraction of ground points mean that the algorithm is only implemented on non-ground points, which saves a lot of computation time. Hence, a gridding system called "Terrain Extraction" has been introduced to address the issue and extract the ground point from 3D point clouds.

- The Voxel-based algorithm for detecting cylindrical features is implemented on large 3D point clouds, which could take hours. Therefore, to address this issue and automate the process, efficient segmentation and clustering techniques are introduced to save computation time.

The proposed methods/algorithms discussed and presented in this thesis are firstly implemented in LSS - 3DVision commercial software. The implementation solves the practical problems encountered by the software's users (surveyors, civil engineers,

archaeologists etc.). The proposed algorithms and methods of this research are already used by 1000-1200 customers across the UK and Worldwide.

1.6 Thesis Structure

The introduction of point clouds and the challenges related to point cloud processing on real-world data are presented in Chapter 1. The algorithms must be developed to provide the solution and address the problems associated with existing methods. This thesis focuses on point cloud filtration, point cloud segmentation, edge detection, and feature extraction methods.

Chapter 2 presents and discusses existing novel methods to process point clouds. The algorithms proposed are based on:

- point cloud registration,
- existing point cloud processing methods,
- existing point clouds filtration to remove outliers,
- existing point cloud edge detection and
- feature detection in point clouds.

The literature illustrates the essential point cloud processing techniques/methods and the acceptance and use of point clouds to extract information to construct a 3D model. Therefore, it is vital to gain an understanding and knowledge of the existing method, especially from a land surveyor's perspective.

Chapter 3 covers the history of point clouds, the importance of point cloud processing and the current issues of processing point clouds, followed by the research method used in this thesis. Chapter 3 also presents the methodology of proposed algorithms, data collection and validation methods. As shown in Chapter 3, this thesis categorises point cloud processing into three stages. The research investigations and questions lead to discuss in the following chapters:

- Filtration - Chapter 4 presents the research of the existing methods and performance in the surveying industry. Following this, it presents the filtration method of the outliers and noise.

- Edge detection - Chapter 5 presents the research and investigates the current edge detection methods. Following this, it introduces a new PCA-based method for detecting edges and edge streams.

- Feature Extraction - Chapter 6 presents the research and investigates the current methods of cylindrical object recognition. Following this, it introduces a new voxel-based method for tree trunks and pole-like object detection and classification.

This is followed by a proposed algorithm implementation on the commercial software and a case study to demonstrate the point cloud processing and modelling in Chapter 7. Finally, the thesis is rounded off in Chapter 8 with the conclusion of the work, how it relates to research questions posed at the beginning and the key contributions of the thesis.

Chapter 2 Literature Review

2.1 Introduction

Different techniques are available in remote sensing technology and image processing applications to obtain 3D models. Some of the popularly used methods include the technologies of LiDar, MLS, and others. This chapter reviews existing literature on the nuances of 3D feature extraction and processing on point cloud datasets obtained from aerial, terrestrial or mobile methods.

The scope of this chapter's literature domain is architecture, engineering, geomatics, construction and computer science. The chapter aims to investigate and distil the literature found into the research objectives that fill the gaps in knowledge by working through this thesis.

Firstly, the chapter explains the technology used to capture point clouds. It is important to understand the technology to understand its issues clearly. The point cloud processing challenges are largely dependent on its capturing technologies.

Next, the chapter reviews the 3D point cloud processing methods that face challenges in achieving accuracy on a large point cloud. Followed by the methods used by other researchers and algorithms found in the literature are reviewed to understand existing gaps and limitations in this area.

The key steps involving point cloud processing are noise removal and feature detection. Therefore, the literature review assesses the approaches of the existing methods and algorithms. This is followed by the gap analysis of existing methods and the reason why there is a need for a new algorithm to capture and extract features from point clouds.

2.1.1 Brief Introduction of Point Cloud Capturing Technology

Point clouds are datasets that represent objects in a coordinate system. A single point on an underlying sample surface or object can be represented by x, y and z geometric coordinates. Point clouds collate numerous spatial measurements into one dataset to represent them as a whole. In other words, point clouds provide 3D points to bring to life the object or underlying sample surface represented by x, y and z coordinates (Ma *et al*., 2018). Generation of point cloud datasets is possible through 3D-laser and LiDar (Light Detection and Ranging) scanners that are both terrestrial and aerial. Each has a different range and accuracy. Boehler and Marbs (2004, p. 292) defined scanning as

> *"3D scanning (often called laser scanning) is a surface-based three-dimensional measurement technique. One scan result in a large quantity of points in a systematic pattern – also called point cloud. Final results after processing of the raw data can be line drawings, CAD models, 3D surface models (with artificial or photorealistic textures) or video animations."*

Every point denotes one unique laser scan measurement (Achlioptas *et al*., 2018). Then, all the captured points are stitched together to create a complete scene through a process known as registration. Understanding real-world data has been enabled through technology using computational resources such as GPUs, 3D data processing using depth sensors and machine learning (ML) or deep learning (DL) methods (Guo *et al*., 2020).

For this thesis's evaluation and case study purposes, the data sets from both terrestrial and LiDar laser scanners are used.

2.2 Processing Point Clouds

Common problems with point cloud data are occlusions, in-depth discontinuities, shadows, poor texture, poor image quality and man-made objects. Aerial imagery space-born-based high-resolution satellite imagery, terrestrial or aerial scanners, and handheld scanners are used to extract objects (buildings, trees, other features) in an area of investigation. In complex data such as city environments with dense or high-rise buildings, the major challenges are

occlusions (DeVore *et al.*, 2013), reflections (Gao *et al*. 2022), passing vehicles and people (Balado *et al.*, 2019; Scheiner *et al.*, 2021), point density discontinuities (Rousell, 2014; Petras *et al.*, 2023), shadows (Guislain *et al.*, 2016), unfinished structures, and standard manufactured objects on the road (Shi *et al.*, 2021; Zhang *et al.*, 2019).

The LiDar technology could support overcoming problems in 3D point cloud data and automatic registration for the extraction of buildings and roads in an urban environment. However, Hui *et al*. (2019) described the low filtering accuracy in airborne LiDar point cloud data from environments with complex terrain as challenging. Besides, aerial images do not have geodetic coordinates and have different types of geometric distortions. Due to this phenomenon, the point cloud data from LiDar is difficult to co-register in aerial images.

Che, Jung and Olsen (2019) explained the general challenges in data processing for recognising objects and classifying point clouds. Some challenges include the effectiveness of data acquisition parameters, type of objects captured, problems in recognising more objects, achieving effectiveness and accuracy and issues with computation time in a large volume of data. Also, the data gathered by different sensors and platforms have different resolutions that can provide various types of information with redundancy. Hence, advanced algorithms have a more accurate capability and robust data processing modelling and analysis.

MATLAB uses various methods for point cloud processing. MATLAB is an abbreviation for "matrix laboratory" which is a programming and numeric computing platform used by engineers and scientists to analyse data, develop algorithms and create models (*MATLAB - MathWorks, Accessed: 12 February 2023)*. MATLAB provides the functionality to read, write, store, visualize, basic processing of point clouds and create geometric models. Gigli and Casagli (2011) used MATLAB to propose a new method based on the definition of least square fitting planes on the cluster of large point clouds extracted by moving a sampling cube. Similarly, Carrea *et al.*, (2021) used MATLAB environment to manage large point cloud datasets for landslide and rockfall investigation analysis. MATLAB is used mainly for visualization and graphical presentation (histograms etc.) by Wang *et al*. (2011), Pepe and Prezioso (2015) and Catalucci *et al.* (2018). Erdélyi, Kopacik and Kyrinovic (2017) used MATLAB for the graphical user interface to process the data efficiently.

2.2.1 Outlier and Noise Presence

Point cloud data gathered with 3D scanners, the technique that uses image capturing for reconstruction, is often corrupted with substantial outliers and noise. Ruchay, Dorofeev and Kalschikov (2019) stated that common noise reduction and outlier filtering approaches tend to reduce noise; however, the problem of removing the outliers contained in a raw point cloud is not handled efficiently. Therefore, Ruchay, Dorofeev and Kalschikov (2019) proposed multiple de-ionising methods to evaluate reconstruction accuracy, though noise and outlier filtering accuracy will depend on the point cloud's density and quality. Outliers must be removed as they are persistent in every point cloud data set with varying point densities. Further, filtering outliers in point cloud data can support applications that use topographical maps in decision-making, managing natural hazards, analysis, and interpretation.

According to Rodríguez *et al*. (2018), point clouds represented a large amount of data, and many coordinates are redundant. Hence, it is a requirement to filter the clouds before processing them. Importantly, the work is intensive when a large area is measured and may need multiple scanning to obtain an accurate 3D point cloud. Che, Jung and Olsen (2019) stated that a line-of-sight measurement is required, and laser scanners have a limited range in obtaining accuracy. Liu *et al.* (2018) explained that multiple scans at different angles are needed to map an area to capture objects. Outlier-filtered data facilitates analysing and interpreting the gaps effectively. In large-area scans, the problem of accuracy must be addressed. In addition, typical large data has the problems of sparse, irregular and unordered data structure in point clouds (Wu, Qi and Fuxin, 2019). Besides, applying convolution to point clouds is difficult (Wang and Solomon, 2019).

Further, scanning an area containing objects such as trees, shrubs, buildings, etc., can be difficult in outdoor environments. Consequently, several algorithms have been developed to achieve accuracy in overcoming problems with stationary objects. This is also known as targetless scanning, which requires a significant amount of overlapped points to register accurately (Xiang, Qi and Li, 2019).

2.2.2 Semantic Segmentation

Numerous algorithms to address problems of accurately registering point cloud data are proposed and demonstrated. Point cloud semantic segmentation techniques are used for learning methods, such as supervised machine learning and state-of-the-art deep learning to generate semantic information for each point (Xie, Tian and Zhu, 2020). Several algorithms have been proposed for semantic segmentation to produce a network to compute individual point features, such as PointNet, PointNet++, PointSIFT, etc. The convolution network, called Spatial Aggregation Network (SAN) by Cai *et al.* (2019), can operate on the local spatial structure information to accomplish efficiency and accuracy. Wu, Qi and Fuxin (2019) presented a novel convolution operation using a dynamic filter named PointConv. PointConv deals with the convolution kernels as non-linear functions. PointConv is implemented to create deep connected networks and experimented with ModelNet40, ShapeNet and ScanNet repositories to achieve semantic segmentation on 3D point clouds. Xin *et al*. (2019) proposed the Fermat Paths theory to capture light from the given visible scenario and unknown objects that are not in the camera's line of sight (LOS). The Fermat-Flow determines the object shape in non-LOS and generates sparse scene reconstruction. Another algorithm (Young *et al*., 2020) provided accuracy in recovering shapes that range from diffuse to specular and are hidden around corners or behind a wall. The algorithm is developed to achieve micron-scale reconstruction and demonstrate mm-scale shape recovery.

Wang and Solomon (2019) addressed the point cloud registration problem by finding rigid transformations by aligning one point cloud with another. Understanding global and domain features to facilitate rigid registration is provided using learned models. Roynard, Deschaud and Goulette (2018) described a Convolution Neural Network (CNN or ConvNet) based method to classify point clouds in urban or indoor scenes. The network is distinguished to determine point classification through the location of points in a multi-scale neighbourhood. Further, the method is reduced to the semantic 3D benchmark compared with other point cloud classification methods to show better performance; however, the method did not use the regularisation step.

Javed, Meraz and Chakraborty (2020) reviewed the contributions related to classifying, segmenting, and traversing 3D-point cloud data. Two methods handle the important point of classification. The first method is based on projection, where the point cloud data is processed

to get an image (2D or 3D). Further represented and followed by applying deep learning (DL) techniques. The second method is to directly process the 3D point cloud data to obtain desired outcomes. Region-based methods or single shots must solve classification problems. In addition, the segmentation is categorised as semantic segmentation, instance segmentation and part segmentation. Algorithms are widely used to resolve the problems identified under classification, segmentation and tracking. For instance, a convolution or graph-based network algorithm in DL is used to address classification. To overcome the object detection problem, classification and bounding box regression algorithms are applied (Shi, Wang and Li, 2019). The segmentation problem is grouped into semantic (point-based), instance (proposal-based) and part segmentation (fully convolution network (SFCN) algorithm). For example, a framework named SqueezeSegV3 for adaptive convolution of efficient point cloud segmentation is available (Xu *et al*., 2020). The challenges with 3D shapes are that they can look different from different perspectives. However, Wang *et al*. (2018) presented a 3D shape segmentation using DL methods. These are problems in object generalisation to all its parts using the algorithms discussed.

2.2.3 Feature Detection

3D representation of data as point clouds is widely explored and researched in computer graphics and computer vision. Numerous developments show object representation using point clouds to solve classical scene challenges in 3D classification and segmentation (Zaganidis *et al.*, 2018). Currently, 3D point clouds have improved in performance to achieve high levels of accuracy. This section explores and analyses the state-of-the-art techniques for feature detection in point clouds.

Uy *et al*. (2019) introduced a dataset generated in a real-world scenario by scanning an indoor environment. The dataset known as ScanObjectNN is developed to resolve the problems in object classification in scenarios where the framing of objects is done in real-world settings. The method presented is intended to solve the object classification challenges in the cluttered background and achieve state-of-the-art performance in identifying objects through a proposed point cloud classification neural network (NN). The proposed NN provides a new dataset for the object using the scanned real-world environment. This dataset is further used in

classification by training and testing the dataset. A benchmark is obtained for existing object classification techniques on real-world and synthetic point cloud data. Lastly, the proposed network could provide a classification of objects in a real-world environment using the combination of the classification method and segmentation method. The ModelNet40 dataset is used as synthetic data in the study, and the ScanObjectNN dataset is generated from real-world 3D scanning. However, from quantitative evaluations, real-world data object classification challenges must be resolved. The results of the benchmark provided up to 78.5% accuracy. The study revisits the state of art object classification methods. Although the results indicate that synthetic data provided the intended output regarding classification accuracy, on the other hand, it failed on real-world data, thus providing further research scope.

Li, Chen and Lee (2018) presented an architecture known as a permutation invariant network named Self Organizing Network (SO-Net). The SO-Net will create the model for the spatial distribution of the point cloud to create one self-organising map (SOM). The SOM performs hierarchical feature extraction using a point-to-node kNN (k-nearest neighbour) search and adjustable receptive field overlap. A proposed point cloud auto-encoder is used for pre-training and improves the network performance in processing. The architecture is examined by comparing it with other state-of-the-art approaches in point cloud processing to show improvements in recognition tasks in point cloud feature classification. The results demonstrated that the proposed architecture provided significantly faster training speed in point cloud reconstructions. However, the method must be tested with convolutional neural network state-of-art computer vision algorithms for image recognition, object classification, semantic segmentation, etc.

Many useful applications use point clouds, such as robot manipulation, geographical information systems (GIS), etc. Su *et al*. (2018) presented a new neural network architecture known as Sparse Lattice Network (SPLATNet) to improve point cloud processing. SPLATNet has many advantages, such as filtering neighbourhoods as in CNN architectures, handling sparsity in input point clouds by focusing on locations where data is available, computing the features based on the hierarchy and spatial input point clouds having sparse and efficient lattice filters. Moreover, the network architecture can map points in 2D into 3D space and vice-versa. SPLATNet architecture is experimented on two different benchmark datasets to compare with the state-of-the-art approaches in segmenting point cloud data. Nevertheless, the results

highlighted do not provide the potential for processing point cloud features such as texture, and classification accuracy is unavailable.

Another framework by You *et al*. (2018) called PointView Network (PVNet) focused on data of point clouds and data having multi-views for creating 3D shape representation. The framework uses ModelNet40 datasets to integrate point cloud and multi-view data to achieve better performance in point cloud features. The framework is a convolution network different from other 3D deep models and employs global features at a high level of multi-view data to support the feature extraction of point cloud data. Further, the framework provides a method known as an embedding fusion to embed global features of multi-view models and generate attention-aware features of point cloud models. This approach is considered more efficient in representing discriminative details of 3D data. In addition, the model is different from other models as it efficiently explores the complementary relation between data in the point cloud and multi-view to represent 3D shape representation. The results indicated a promising potential for point cloud feature extraction (different geometric properties). The evaluation and effectiveness of this framework are based on missing data with an example of shapes such as a chair, lamp and bottle, etc., without testing on real-world data.

Lu *et al*. (2019) introduced a framework that uses end-to-end learning to achieve registration accuracy for geometric methods in point clouds. The method is named DeepVCP, which implements different deep neural network structures to determine an end-to-end trainable network. The framework's effectiveness is evaluated using KITTI and the Apollo-South Bay datasets (the vehicle dataset contains LiDar point clouds, images, and IMU data). The point cloud registration accuracy is achieved by learning-based keypoint detection, novel corresponding point generation and loss function. In this method, local similarity and global geometric constraints enable better accuracy. The results showed better performance compared with geometry-based techniques. The behaviour and insights of the framework are illustrated by visualisation and ablation analysis. The overall results indicate low registration errors, making this framework attractive for applications involving point cloud registration tasks. The framework has the potential to be used in LiDar point cloud registration which is the foundation of a variety of applications. The drawback of the method is that the classification is based on the point weighing layer and probability distribution rather than based on geometric feature distribution.

2.2.4 Analysis and Semantic Interpretation

Many existing algorithms cannot automatically extract semantic information from the given scene. Also, shape analysis cannot be performed automatically on point cloud models. Williams and Ilies (2018) presented a method to analyse noisy point cloud data for real objects and objects that are incomplete. The method uses a heat diffusion kernel to build succinct shape signatures. Further, the method is designed to support a variety of clustering techniques that were earlier applied to mesh models. A Laplace-Beltrami convergent estimate operator for point clouds is implemented along with clustering techniques that work directly on point clouds to generate geometric features for various applications. One main advantage of this method is that it can operate directly on a point cloud model without surface reconstruction or meshing. Furthermore, the proposed technique is robust in handling incomplete point cloud data into semantically purposeful sub-shapes.

Grilli *et al*. (2019) used the ML and DL methods to analyse features geometrically and classify cultural heritage environments in the point cloud data. The classification is performed to understand the applications developed for modelling. Hence, the method mainly explores the cultural heritage sector because other studies have not explored this area in a geospatial field. The method's first covariance matrix is used to extract features. Then, the impact of features calculated in spherical neighbourhoods is analysed by deriving radii values from simple proportional and dimension rules used in constructing classical architectures. The features are not required to be extracted at different scales to obtain an accurate classification. Next, the RANDOM forest classifier is used. Finally, the confusion matrix is used to evaluate the label generated by the classifier vs the manual. The methods developed used an adaptive size strategy to retrieve the best results. The disadvantage of the method is that the test set does not explain the size of the set used for testing. The results are needed to examine the effectiveness of the developed method using more complex structures.

Wang, He and Ma (2019) proposed a model for semantic segmentation of point clouds to exploit local and global structures within the point cloud data. The model is based on contextual point representation in which each point is performed from one novel gated fusion of the point and its contextual points for enriching it. Qi *et al*. (2017) proposed a neural network called PointNet based on enriched representation. The PointNet module depends on a graph attention block to create and update each point representation in the local-point cloud structure. Finally,

the attention strategies to exploit global point cloud structure, spatial and channel-wise, are used to yield semantic labels for each point. The authors used the public cloud databases, S3DIS and ScanNet, to demonstrate the effectiveness of this model. This proposed model exploits the point cloud local and global structures using a PointNet module and attention structures (Wang, He and Ma, 2019). The superiority of this proposed model is demonstrated with existing datasets.

Models are available to demonstrate the feature learning abilities with regular data structures. However, there are still multiple challenges in the case of irregular data structures due to the limitations of methods that represent data (Wang, He and Ma, 2019). According to Graham, Engelcke and Maaten (2018), to yield semantic segmentation, existing approaches transform the point clouds into 3D voxel grids or a collection of images for input into traditional CNNs. Further, the existing approaches for 3D representation can be categorised as (1) 3D-voxel based, (2) set-based and (3) multi-view based. In the 3D-voxel-based method, the point clouds are transformed into regular 3D voxel grids, followed by 3D CNN directly to the image (Jaritz, Gu and Su, 2019). The primary objective of voxel-based methods is to store and process 3D data. Examples of voxel-based approaches are Oct-Net, Kd-Net and O-CNN. A set-based method is used to learn the descriptions directly from unstructured or unordered point cloud data (Ku *et al*., 2020). Examples of set-based methods are PointNet, PointNet++ and PointCNN. As the name suggests, a multi-view method renders multiple images at different view angles from point clouds. Subsequently, the image is processed by traditional 2D CNN methods (Yang *et al*., 2018). However, the multi-view method is not popular because of the problem of not knowing the number of angles required to capture the 3D space and using image cause information loss.

Balado *et al*. (2019) used point cloud data obtained from Mobile Laser Scanning (MLS) to segment the roadside components such as road surfaces, guardrails, fences, embankments, ditches, and borders. The PointNet model is used in the study for semantic segmentation of the road. The method is processed in two stages (1) the point cloud is segmented into sections along the trajectory of obtained samples by carefully distributing the road elements in each section (2) PointNet is applied to segment each sample further. The results provided indicate effective segmentation of a large number of objects. Furthermore, comparing ANN-based classification techniques and point-by-point extraction for segmenting road surfaces and fences provides accurate results.

Xie, Tian and Zhu (2020) stated that the DL techniques are useful in point clouds, 3D point cloud semantic segmentation (PCSS) and point cloud segmentation (PCS) and are popular areas for research in academia and other industry. There are many semantic segmentation techniques in the segmentation of point cloud data. The techniques are divided into four general categories. The categories are edge-based, region growing, model fitting and cluster. The edge-based semantic segmentation involves the principle of locating the points with quick changes in their intensity and is identical to 2D image segmentation. The algorithm for edge-based segmentation consists of two important stages. The first stage is to detect the edges from which extraction of boundaries is completed from different regions. The second stage is to group the points that generate the last segments by aggregating points in boundaries in the given region (Landrieu and Simonovsky, 2018).

In a Point Cloud Segmentation (PCS) review, Xie, Tian and Zhu (2020) defined PCS as grouping raw points into non-overlapping regions. For strong semantic knowledge of the points, the method divides the segmentation technique into four groups (1) edge-based, (2) region growing, (3) model fitting and (4) clustering-based. The edge-based approaches define objects' shapes as edges move from 2D images to 3D point clouds. Region growing is a classical method of PCS that combines features in two regions or between two points to determine similarity to merge pixels (Xie, Tian and Zhu, 2020). The merging is possible if the points and regions are close spatially and have identical surface properties. For example, the features of 2D pixels, 3D pixels and 3D voxels are merged in region growth (Guo *et al*., 2017). Model fitting is a shape detection and extraction method that matches point clouds to different primitive geometric shapes. The model fitting is built on two algorithms, mainly Hough Transform (HT) and Random Sample Consensus (RANSAC) (Poux and Billen, 2019). Lastly, clustering-based methods are a mixture of various methods that aim to group points/spectral features/spatial distribution belonging to similar geometric features in unsupervised PCS. Therefore, clustering-based is used for irregular points belonging to features like vegetation. Examples of clustering-based approaches are K-means, mean shift and fuzzy clustering.

Literature to explain the point cloud model as a structured graph for semantic segmentation is explored. Jiang *et al.* (2019) used semantic 3D scene labelling by investigating the relationship between each point and its neighbours through edges. A hierarchical graph framework is generated to include point features in the edge branch to generate and integrate the point and edge features. Landrieu and Simonovsky (2018) proposed a method SPGraph that deals with

large-scale point clouds. The semantic labels are predicted by first partitioning the points into geometrically homogeneous elements to develop a super-point graph and input to a graph convolution network (GCN) to predict semantic labels. Wang *et al*. (2018) explained that dynamic graph CNN (DGCNN) depends on edge-convolution operations to seize local shapes dynamically. Many methods are available in the literature, but it is important to note that most approaches use local relationships in the point cloud and not the global one.

2.2.5 Knowledge-based Data-driven Point Cloud System

Many applications consider point clouds as assets. However, inadequate semantic information within the point cloud ensembles raises technical limitations. Hence, connecting knowledge sources is time-consuming and a lengthy manual process, resulting in human errors. This problem requires a powerful domain-related data analysis method to develop coherent and structured information. Hence, point cloud processing can be used to create intelligent environments and knowledge discovery because object recognition and detection or classification of objects in datasets are important.

Poux and Billen (2019) stated that knowledge discovery in decision-making systems is possible by automating data processing in the point cloud. The method proposes feature engineering based on voxel to qualify point clusters and support both classifications, viz. strongly, supervised or unsupervised. The variations in generalised feature levels that permit frameworks to interoperate are discussed. The authors recommend a shape-based feature set (SF1) to leverage raw X, Y, and Z attributes or the point cloud. Further, the relationship and topology found in voxel entities are derived to obtain the 3D structural connectivity feature set (SF2).

Lastly, the knowledge-based decision tree is provided to allow classification based on infrastructure. Discussions are related to the synergy of SF1/SF2 on a new framework for semantic segmentation to constitute a higher representation semantically of point clouds in an associated cluster. The S3DIS dataset is used to benchmark this approach with novel and best-performing DL methods. The results discussed explain good performance, ease of integration and high scores for classes that are dominant by planar and comparable with deep learning methods.

The work by Ponciano, Trémeau and Boochs (2019) aimed to overcome the constraints in training datasets in ML by presenting a semantic-guided approach. Semantic plays a key role in analysing the objects in data sets for related information. This approach also modifies the processing according to object diversity and data characteristics as a learning stage. It uses web technology such as SPARQL queries to discuss semantic segmentation through an ontological model. Further, the model permits the selection and execution of algorithms implemented dynamically. SPARQL links knowledge found in an ontological model and algorithms-enabled processing.

Furthermore, the presented model can adapt a sequence of algorithms to an individual state in the process chain to make the solution more flexible and robust. The method accounts for data variation and objects representation to identify objects like walls, floors and ceilings successfully. The disadvantage of the method is that preset reasoning for points classification causes all points at the border of rooms to be not classified. In addition, it constantly has to update the knowledge-based for effective detection.

Petrova *et al*. (2019) presented research on discovering new knowledge and making informed decisions using evidence. The research is made using sustainable building designs. The tasks include outlining and determining diverse data sources and types, indicating the method for data analysis, demonstrating knowledge discovery can be implemented in the semantic integration layer and supporting in design. The outcome of this research is a performance-based decision support system and semantic data modelling for a different design.

Wengefeld *et al.* (2019) introduced an approach for the orientation of a person that is dependent on coloured point clouds. The classification approach is extended to the continuous domain to treat the real-time orientation estimation problem. The approach is compared with multi-class and regression problems. The results provided show promising data to compete with accuracies in the state-of-the-art and DL-based skeleton estimation approaches while the capability of standard CPU is maintained. Furthermore, this approach is verified for knowledge discovery of people orientation in human-robot interaction (HRI) tasks.

Wen *et al*. (2020) proposed DL 'Point2SpatialCapsule' network to aggregate features and spatial relationships. This capsule aims to learn a better representation of discriminate shapes by combining all features with spatial relations of local regions in point clouds. This network

is an experimental model, and the results obtained show that the capsule has the potential to outperform other methods in 3D point cloud data. However, the model must be tested with real-time data to understand its effectiveness.

2.3 Outlier/Noise Removal

Raw point clouds are often very noisy and have outliers. The challenge is to remove the points that makeup noise and outliers. Han *et al.* (2017) described filtering as an area of intensive research and the vital processing stage for a wide range of applications. Various authors categorise the methods into groups that have the same criteria or adoption. Examples of such categories are presented below.

Papadimitriou *et al.* (2003, p.315) divided the methods into five categories as follow:

- Distribution-based methods are found in statistics books which identify the outliers based on a distribution model such as normals (Hawkins, 1980; Barnett Vic and Lewis Toby, 1994)

- The depth-based approaches use computational geometry and convex hull on various layers to identify the outliers based on their position within the layers (Johnson, Kwok and Ng, 1998)

- The clustering algorithm's main purpose is to cluster the points; hence the outliers are just the by-product of clustering (Jain, Murty and Flynn, 2000)

- The distance-based approaches use the distance with parameters to identify objects further to that distance as outliers but can create problems if the data set is dense or sparse. It was first proposed by Knorr and Ng (Knorr and Ng, 1997; Knorr, Ng and Tucakov, 2000)

- The density-based approaches rely on the local outlier factor (LOF) of the object, which is the local density of its neighbourhood. Therefore, the objects with high LOF are

considered outliers. The density-based method was first proposed by Breunig *et al.* (2000).

Schall, Belyaev and Seidel (2008) and Han *et al*. (2017) divided the existing methods into four categories. The first category is Statistical-based methods that utilise statistical concepts to filter outliers according to the point cloud used. Schall, Belyaev and Seidel (2005) used kernel-based clustering to filter the outliers by defining the global probability distribution function for noisy points. Pauly, Mitra and Guibas (2004) presented a framework for analysing the shape and variability, i.e., uncertainty, by introducing statistical representation that quantifies each point's likelihood of plane fitting through it.

Jenke *et al.* (2006) introduced Bayesian statistics to produce a smooth point cloud from a noisy point set. The probability distribution specifies the measurement and reconstruction model (density prior, smoothness prior and prior for sharp features) of data defined by the statistical concept of finite-dimensional representation to remove noise from the point cloud. Esmeide and Nallig Eduardo (2006) proposed a new variant of principal component analysis, which uses weighing factors inversely proportional to the euclidean distance to the mean. Next, a weighted covariance matrix is calculated. Finally, corresponding to the largest eigenvalue, a plane is fitted. Both normal and smallest eigenvalues are used to preserve the sharpness of each point's fitted plane to make it robust to outliers. Kalogerakis *et al.* (2009) delivered a statistical framework using Iteratively Reweighted Least Squares (IRLS) to estimate the curvature tensor and weight the assignment to each point neighbourhood. Both methods are used to correct the normals and help in outlier elimination and denoising the point clouds. Avron *et al.* (2010) introduced the L_1-sparsity paradigm to denoise point clouds. The point orientation is restored and then calculated using local planarity criteria point position. This is extended by Sun, Schaefer and Wang (2015) to provide the L_0 minimisation method, which applies the normal estimation and repositioning of points in order to denoise the point clouds.

The second category is Neighbourhood-based techniques that filter the point using similarity measures between the point and its neighbours. Tomasi and Manduchi (1998) first introduced the bilateral filter that combines the grey levels and colours based on geometrical closeness. Paris and Durand (2006) extended Tomasi's work and introduced a smoothing filter. Furthermore, the Paris and Durand method is extended to 3D mesh denoising by Fleishman, Drori and Cohen-Or (2003), Jones, Durand and Desbrun (2003) and Lee and Wang (2005).

Fleishman, Drori and Cohen-Or (2003) proposed an anisotropic mesh denoising algorithm, and Jones, Durand and Desbrun (2003) proposed a robust statistics approach based on local estimates of a surface. However, the methods by Jones *et al*. (2003), Fleishman *et al*. (2003) and Lee and Wang (2005) involved mesh generation, which is very noisy. Whereas Shi and Hernandez (Shi, Liang and Liu, 2011; Hernandez, Choi and Medioni, 2015) applied a bilateral filter on point clouds to overcome the problem.

The third category is Projection-based approaches to filter points by adjusting each point using projection strategies. The least-squares fitting recently became one of the most interesting research topics. Levin (1998) first proposed the moving least squares method, and Alexa *et al.* (2001, 2003) were the first to implement it in computer graphics. The noise points are handled by iteratively projecting them on a fitted plane. The problem with moving-least-squares is that for finding the fitted plane, the process accommodates the non-linear optimisation, which increases computation time. Later Alexa and Adamson (2004) proposed a similar projection method which calculates the weighted position of a point, and a normal is calculated using weighted input normal. Amenta and Kil (2004) introduced a new variant called the energy function in moving least squares to produce a point on the surface. Fleishman, Cohen-Or and Silva (2005) proposed a robust moving-least-squares method based on a forward search paradigm to filter noise and outliers. Dey and Sun (2005) proposed a new variant of moving least squares, an adaptive moving least squares operator that uses local feature size. The purpose is to analyse the non-uniform density to provide the reconstruction of the surface within the point set. Adamson, Alexa and Berlin (2006) adopted the decomposition of objects into cell complexes in order to preserve the shape features. A method fitting a quadratic patch on each neighbourhood point was presented by Fua and Sander (1992). The method deals with outliers to preserve the curvature by measuring if two points are on the same local surface. Wang *et al.* (2013) extended the method by Fua and used clustering and adaptive scaling to compute the planes to all points. By iterating the process, non-feature points are identified, and noisy points are removed.

The Fourth category is PDEs-based filtering techniques (Partial Differential Equations) which can be described as an extension of triangular meshes used for filtering in point clouds. Clarenz, Rumpf and Telea (2004) presented a framework for point cloud filtering using local finite matrices created from a single matrix.

To preserve non-linear features by using Anisotropic mean curvature is presented by Hildebrandt and Polthier (2004), which is further extended by Lange, Polthier and Berlin (2005). Taubin (1995) and Lange, Polthier and Berlin (2005) used directional and principal curvatures and a Weingarten map for outlier filtering. First, a Weingarten map is used to obtain the anisotropic geometric mean curvature flow. Then, the method uses the directional curvature to generate a Weingarten map to compute eigenvalues and eigenvectors corresponding to principal directions. Finally, anisotropic Laplacian is used to change the curvature information. This method is further extended to many methods. One of the examples of images in 3D point clouds by Lozes, Elmoataz and Lezoray (2014) represented point clouds using weighted arbitrary graphs considering the neighbouring point information. The Laplacian operator and PDEs operator are included in arbitrary graphs to filter the points in the point clouds (Ta, Elmoataz and Lézoray, 2011).

According to Ge and Feng (2021), Hodge and Austin (2004), Mansur *et al.* (2005), Kriegel, Kröger Peer and Zimek (2010), methods are divided according to a single criterion and combined methods as follows:

1. Single Criteria Methods identify outliers by using single criteria.
    - Distribution-based methods – points that deviate from a specific distribution are classified as outliers. The statistical outlier removal by Rusu *et al.* (2008) assumed normal distribution between a point and its neighbour, and the point that does not fit in a normal distribution is an outlier. Rousseeuw and Hubert (2011) fit a plane using the least trimmed squares estimator, and the points that have a large deviation from the plane are identified as outliers.

    - Proximity-based methods – points that are away from most of the other points are classified as outliers. Nurunnabi, West and Belton (2015) proposed two outlier detection methods that are successful in identifying and removing outliers 1) maximum consistency with minimum distance based on Z-score (MCMD_Z) and 2) maximum consistency with minimum distance based on Mahalanobis distance (MCMD_MD).

- Density-based methods – points are assigned with probability values based on the density of their local neighbourhood. Those points that have a high probability value are classified as outliers. Local outlier factor (LOF) (Breunig *et al*., 2000), local correlation integral (LOCI) (Papadimitriou *et al.*, 2003), and local outlier probability (LoOp) (Kriegel Hans Peter *et al.*, 2009) are three examples of density-based methods.

- Cluster-based methods – points in small clusters that are away from other clusters are classified as outliers. Cluster-based local outlier factor (FindCBLOF) (He, Xu and Deng, 2003) is an example of a cluster-based method.

- Depth-based methods – points outside the specified depth based on depth maps of geometric objects are classified as outliers. Wolff *et al.* (2016) proposed a depth map-based method for outlier detection.

- Learning-based methods – the model is trained, and then the same model is used to determine the points as normal or outliers. For example, Rakotosaona *et al.* (2020) proposed a method called PointCleanNet, and Stucker *et al.* (2018) used the random forest to classify the outliers.

- Graph-based methods – the relationship between two points is defined as an edge and each point as a node. Then, the graph is constructed, and the score is used to determine the outliers. For example, Hautamäki, Kärkkäinen and Fränti (2004) proposed a method that uses the k-nearest neighbour graph to identify the outliers.

2. Combined Methods use several criterion methods and classify outliers into different types. The aim is to remove different types of outliers based on their characteristics in point clouds. Sotoodeh (2007) proposed a hierarchical outlier removal method, and Ning *et al*. (2018) proposed methods that use local density and deviation from the local fitted plane to remove outliers.

The voxel grid and Quadtree are other methods for filtering noise and outlier points (Han *et al*., 2017). In voxel-based methods, for filtering outliers, a point is picked to calculate the

distance from all points inside each voxel. Whereas for quadtree, a data structure is used for neighbourhood search. Another method of filtering involves the RGB present in point clouds.

Ruchay, Dorofeev and Kalschikov (2019) used point cloud data gathered using the RGB-D sensor to analyse the accuracy of 3D object reconstruction. Point cloud algorithms are applied to the dataset to remove outliers and noise. The algorithms of statistical outlier removal filter (SOR), radius outlier removal (ROR) filter, Voxel grid (VG) filter and 3D Bilateral filter (3DBF) are compared for their de-ionising effectiveness and algorithms are evaluated by applying them for their effectiveness in 3D object reconstruction. The ROR filter algorithm is explained to provide better results compared to point cloud de-ionising algorithms.

Studies on outlier/ noise removal algorithms have been explored recently in industrial applications and reverse engineering Ning *et al.* (2018). Lan, Yew and Lee's (2019) work involved a probabilistic approach for outlier feature matching and loop-closure in front-end data. The association of outliers and loop closure can fail the back-end optimisation of point cloud 3D reconstruction. The approach involves a Bayesian network and the Expectation-Maximisation method. The outlier feature matches are suppressed on long-tail Cauchy distribution, and the outlier loop closure constraints are suppressed using a Cauchy-Uniform mixture model. The method is experimental and performs well on both indoor and outdoor datasets.

Zeybek and Şanlıoğlu (2019) presented the implementation of four algorithms using commercial and open-source software to filter outliers and noise from point cloud data. The method's input is UAV point cloud data on which these algorithms are applied in sequence:

(1) multi-scale curvature classification (MCC), (2) surface-based filtering algorithm (FUSION), (3) progressive TIN-based and (4) physical simulation processing using cloth simulation filtering (CSF) algorithms.

Finally, the results of the algorithms are validated with a reference dataset for accuracy. The work claims that these algorithms demonstrated similar results while extracting ground objects on distinct terrain features such as densely vegetated, flat/bare earth surfaces, and rough and complex landscapes. In addition, the CSF filtering method provided 93% classification on a flat surface.

Rakotosaona *et al*. (2019) used the DL (deep learning) architecture approach to estimate local shape properties in 3D point cloud data. Firstly, the outliers are discarded in the approach, and correction vectors that projected noisy points are estimated in the original clean surfaces. The approach is efficient in terms of variation in noise and outliers, and the DL can also handle large and dense point cloud data. The evaluation is performed using synthetic and real-time data, and the method can develop accurate surface reconstruction from a range of scans. The extremely noisy data and outliers are removed compared with other state-of-the-art methods. This method is simple and can be easily integrated with existing processing applications. The effectiveness of this approach must be tested on large datasets to evaluate its impact, as test samples were relatively small.

2.4 Edge Detection

In 3D point-based extraction, algorithms extract the edges of roads, buildings and boundaries (Shirowzhan *et al*., 2019). In MLS, point clouds are effectively applied with Gaussian function derivatives to extract edges (Yadav and Singh, 2018). Also, MLS data can extract road edges in urban settings and is possible by an active parametrically contoured snake model (Nguyen *et al*., 2019). Furthermore, Zai *et al*. (2018) proposed an algorithm that generates super voxels to automatically extract road boundaries and pavement surfaces using MLS point clouds. Other methods to detect objects in urban settings include differential and regression filters on data collected from MLS 3D point clouds (Zeybek, 2021a). In addition, studies have identified that focus on unique objects on the road, such as trees, roadside traffic lights, buildings, etc. The availability of mobile laser scanning (MLS) techniques provides the potential for advanced mapping potential for effective data collection in geospatial applications. MLS systems provide the flexibility and ability to gather dense point cloud data with time efficiency measurements and cost-effectiveness. MLS platforms can be mounted on vehicles, including LiDar and advanced digital cameras integrated with centralised computing facilities for synchronising data and management (Rastiveis *et al*., 2020). In geometric designs, extracting different objects from point cloud data is important. For instance, the extraction of road surfaces, buildings, driving lanes, etc., is needed (Ma *et al*., 2018).

Unlike LiDar, aerial scanners cannot view beneath a vegetation canopy, resulting in sparse points on the earth's surface. Yilmaz, Yilmaz and Güngör (2018) proposed a methodology involving image classification by a supervised method to filter 3D point clouds. The method involves overlapping a classified image with a point cloud to determine ground points for use in digital elevation model (DEM) generation. The method is evaluated qualitatively to show that filtering point cloud data has the potential to generate high-resolution DEM. The method overcomes aerial mapping application's disadvantages in generating dense 3D point clouds.

Chen *et al*. (2020) explained that photogrammetric techniques enable 3D meshes of aerial images. These photogrammetric point clouds do not provide interactions at the user or system level to distinguish between objects because they contain semantic information. However, these images are required for simulations and to develop a virtual environment. The essential requirement is to extract object information from segmenting generated point clouds and meshes. Therefore, to overcome these limitations, the method proposes a framework that can extract objects such as tree locations and related features and buildings. The framework will rank different point descriptors and evaluate supervised ML algorithms to segment photogrammetric point clouds. The framework must be verified with 3D point cloud data obtained in real-time and validated using data from the University of Southern California (USC) and the Muscatatuck Urban Training Center (MUTC).

Building polygons are used as input in urban applications, but extraction of building edges is difficult, time-consuming and labour-intensive. Widyaningrum, Gorte and Lindenbergh (2019) proposed an approach to display building edge points using an ordered points-aided Hough Transform (OHT) to extract building outlines from aerial LiDar point cloud data. First, the method constructs an accumulator matrix based on a voting scheme in parametric line space. Second, the dominant building direction is determined using the variance of angles in each column. Finally, the hierarchical filtering and clustering approaches are applied to get an accurate line from detected hotspots and ordered points. The ordered point list matrix having ordered building edge points renders line segment detection, resulting in effective quality building roof polygons. The method is tested with different benchmark datasets in Vaihingen, Germany and Makassar, Indonesia. The results provided high accuracy, up to 96.1%. The method is also demonstrated with other existing datasets; however, the method is dependent on edge points for allocating outlines and therefore failed to detect curved outlines.

Becker *et al.* (2018) presented a technique for classification to extract point-wise semantic class labels from aerial 3D point cloud data (PCD). The method incorporates colour information to increase semantic feature detection accuracy significantly. This classification method is tested with four real-world photogrammetric datasets from Pix4Dmapper with varying point densities. The ML techniques and new features could accurately train classifiers to generalise unseen data while processing point clouds with 10 million points within three minutes. This approach and model have the potential to generate digital terrain models accurately based on simple heuristics. However, the functionality given to users to add their dataset to training data results in an inaccurate classifier.

To research and discuss the creation of virtual environments using segmented data, Chen *et al*. 2020) introduced a model ensembling framework to segment 3D photogrammetry point cloud to top-level terrain elements. The elements include humans, ground-level objects and trees, etc. The data are pre-processed with designed methods to resolve data segmentation challenges that show photogrammetric data quality problems. A large UAV-based database is created from UAV-gathered images to validate the framework and methods. Comparing the framework with existing point cloud segmentation algorithms provided outputs to show that the proposed framework can outperform other algorithms. This method segments photogrammetric generated point clouds to create workable virtual environments for simulation purposes.

Besides, there is the challenge of change detection in the given environment. Tran, Ressl and Pfeifer (2018) suggested an approach to change detection (CD) of objects in a given environment. The method combines classification and CD as a single step and builds on the point cloud as an additional layer to obtain high-resolution geo-information from laser scans to match images. In this case, two-point clouds are made available as different epochs and ML is used as sample training data to identify if there is a change in the given location of the point as separate class information for each point. Based on supervised classification and applied to the entire area already generated as a point cloud. The approach provided good results to show changes in different classes such as a new tree, lost tree, lost building, new building, building and changed ground.

2.5 Tree Trunk, Lamp Post and Pole Detection in Point Clouds

Recently algorithms have been developed for urban and forest data sets to identify trees and pole-like objects, such as marker poles, signs and lamp posts. However, the literature review suggests that algorithms developed for a particular type of data set have difficulty achieving reliable results for another type. In existing studies, algorithms include point cloud segmentation to identify the prominent tree points by grouping all the tree points like the top of the tree (Carr and Slyder, 2018). Algorithms using the k-NN approach classify ground, stem and crown. In the given scene, the algorithm computes eigenvectors to define axis direction, and eigenvalues will provide the variance of points along the axes. The stem structure is identified as vertical shapes, and 3D cylinders are applied in modelling individual sections of the stem. Weighting is implemented for more accuracy (Tuominen *et al*., 2018).

The comparative shortest path algorithm is applied to TLS and MLS data to segment tree crowns based on ecology principles. The algorithm will detect tree trunks using density-based spatial clustering of applications with noise (DBASCAN) algorithm (Parkhan, 2019). The study by Chen *et al*. (2019) presented a point cloud classification algorithm that uses Mixed Kernel Function SVM to identify different ground objects. The algorithm effectively extracts objects such as trees compared to standard SVM methods.

Using geometric features, separating wood and leaf components in point cloud data is analysed. An algorithm that combines classification and segmentation methods is developed and applied to data gathered from LiDar. K-means and random sampling consistency (RANSAC) algorithms are used to classify the wood and leaf components in the tree. The method has the potential for extracting wood from point cloud datasets obtained from terrestrial LiDar technology (Su *et al*., 2019).

The use of the CNN technique is explained by Kumar *et al*. (2019) to develop three techniques, single CNN (SCN), multi-faceted CNN (MFC) and MFC with reproduction (MCFR), to classify MLS data automatically. These methods are applied to the KITTI dataset to accurately identify outdoor objects such as poles, trees, houses and lampposts (Kumar *et al*., 2019). Another method by Kang *et al*. (2018) proposed a voxel-based method for automatically

extracting 3D pole-type objects in urban data sets. A voxel-based shape recognition approach is used to generate pole-like object candidates. A circular model with an adaptive radius is used to detect and individualise pole-like objects. The method is applied to LiDar point cloud data. The proposed method can classify objects like lamp posts, utility poles, and tree trunks. Gupta *et al*. (2019) provided a method using voxel and connected component analysis to isolate and identify tree regions. Though this method provided partial accuracy, the method cannot separate all individual trees into a large clump.

2.6 Research Gap Analysis

Accurate reconstruction of 3D point clouds by extracting objects and surfaces is gaining popularity in architecture, surveying and algorithmic development (DL, ML), and AI techniques for automated detection and classification. However, to construct the models from the point clouds to capture 3D geometric features, the existing methods do not elaborate on the required needs and abstraction level suitable to form a model. This thesis discusses the requirements and desired outcomes of commercial clients in the domain of the surveying industry in Chapter 3.

The innovation in laser scanning technology provides faster acquisition systems for collecting and capturing 3D and 2D images. The reason is the improvements in capture rate. However, the increase in data collection means the point clouds end up very large. Furthermore, because of this reason, it remains complex data with good visual representations. They are just a set of points without a way to extract information from the point cloud. Subsequently, allowing for the development of efficiencies in the geometry extraction of features from various scenes, including indoors and outdoors. However, the extraction process takes longer due to the complexity and large number of these data sets. Therefore, the main requirement is for a fast and automatic method that is also cost-efficient.

The literature indicates that point cloud processing, including 3D classification, segmentation, and extraction, is automated up to a certain degree; however, a fully effective method is yet to be created. The extraction includes very simple geometry, and the methods are created especially for bespoke environments. The flexibility and effectiveness of the methods on

various kinds of point clouds are lacking. The thesis presents the algorithm/methods with flexibility, enabling users to use them in every environment.

The literature suggests that existing methods implement various approaches for point cloud processing for academic purposes, i.e., the testing and implementations are performed on synthetic data, or a smaller sample set or dataset that are not available publicly. However, there is no proof of practical implementation of the currently proposed methods. This thesis not only proves the proposed method academically but also presents a commercial software application.

The practicality and usage of the proposed algorithms/methods are designed by keeping the users in mind. The user's top priority is achieving the results as quickly as possible with little user intervention (user clicks on software). As the proposed algorithms are part of the software, the performance and accuracy are performed for quality checks of the methods. The quality checks are presented via visual reports on the screen to ensure quality control.

Many processes emerge from the literature for feature segmentation, representation of data, classification and feature detection methods. The challenge in segmentation is to group/cluster points that belong to the same geometry together. RANSAC is the popular choice for various methods of outliers and feature detection. Several researchers use deep learning and machine learning methods for semantic segmentation techniques. Along with the advancements in computers and technology for data storage and management, data processing is still a problem. As stated in the literature, they are handled by subsampling and partitioning due to the exceptional volume of point cloud data. These automatic processes are heavily data-driven methods which can result in huge computation times. Partitioning and subsampling are the measures used to reduce the number of points and, therefore, the processing time. Many approaches are reviewed, such as voxel-based methods that are used for subsampling. Therefore, a method to handle and sample unordered point clouds would be advantageous.

From literature reviews, it is observed that existing methods that develop building models focus on semantic registration approaches for irregular shapes. In addition, a literature review of current problems in point cloud data processing indicates a need to filter and remove outliers from the dataset before they are further classified and processed. The problem of noisy data and the presence of outliers can result in limitations in achieving accuracy. Findings on point cloud features, analysis and semantic representation, point cloud processing and noise/outlier

removal provided several methods and techniques using synthetic and real-world data. However, in most of the existing methods, it is noted that they are experimental and cannot achieve the level of accuracy desired in real-world large point cloud data. In addition, the techniques found are not robust in noise removal or sparseness, leading to difficulties in object detection.

Finally, the existing datasets test the methods and algorithms; most tests are experimental studies. Therefore, based on all these limitations of the current algorithms, a method must be developed to analyse point cloud data in the real world and resolve problems found in existing systems.

An important feature missing is the ability to perform a quality check on the extracted features. In order to achieve that, the existing methods should be tested on point clouds derived from various sources. However, the accuracy of the required results has yet to be defined to compare applications (Tang *et al.*, 2010). Furthermore, the results cannot be tested efficiently if a standard rule is missing. The accuracy for feature detection of geometrical shapes in point clouds faces challenges. The edge-based feature detection approaches are mostly oriented toward line tracing (Weber *et al.*, 2012) without inspecting the points. This creates a problem in performance as the geometric properties are not analysed to deliver the edges.

This thesis presents the algorithm which overcomes the issues of the current methods by completely examining the points that make the edges. The proposed method is flexible for users to take control, as discussed in Chapter 4. Other important features are trees, trunks, lamp posts and poles, considered cylindrical objects. Common challenges in detecting cylindrical objects are the ground/terrain slope and low point density. The ground slope affects the detection as their trunks are tilted. Additionally, as the laser scanner collects points from the ‘first return’ object, it can result in distorted and sparse low-density points. This thesis proposes an algorithm in Chapter 6 to solve the problem of low-density points and the ground slope of a given point cloud.

The chapter aims to investigate and distil the literature found into the research objectives that fill the gaps in knowledge by working through this thesis.

2.6.1 Purpose of the new algorithm

The proposed algorithms are designed and developed to solve the existing problems in point cloud processing methods. The new algorithm is a statistical procedure that uses multiple techniques to identify features. The primary contribution of the proposed approach is the connection of algorithms with processing. The process flow will allow the correction of scanned data by removing the outliers, obtaining edges and edge streams along the planar surfaces and finally detecting trunks and poles. In addition, the algorithm will include the classification of objects, analysis, filtering, segmentation, and model fitting on large real-world point cloud data sets, thus overcoming the limitations of existing algorithms.

2.7 Chapter Summary

The literature review section presents the common problems faced with identifying the accuracy of objects from scans in point clouds. Due to various factors such as sparseness, outliers, and distortion in the scanned point cloud, many techniques and methods are researched and proposed to overcome these limiting factors towards achieving accuracy. However, the evaluation highlights the common problems faced in point cloud data processing along with reviews related to current problems and issues identified from research. The algorithms reviewed are related to the semantic interpretation, outlier detection and removal, edge detection and identification of trees, lamp posts, and buildings.

Reviews of existing research are performed to understand other researchers' limitations in existing work. The outlier filtration methods are required to be easy to use, fast and accurately delete the noise and outlier points. The common challenges of edge detection algorithms are accuracy (detection of the edge point) and robustness to detect it in real-time and faster. After reviewing the feature detection algorithms, the common challenges are classification and isolating the trees and poles from other points with low density and gradient in the terrain. Finally, the existing gaps are summarised to justify the research, design, and development of a new algorithm to overcome existing limitations and achieve accuracy with efficiency.

Chapter 3 Research Methodology

3.1 Introduction

There are several traditional ways to capture data in geographical information systems, but these methods are tedious as they require manual processing of the geometrical points collected (Maguya *et al*., 2014). In geometry, a point is an exact location in space. A point has no size and is only a position defined in X, Y and Z coordinates. The recent development of laser scanner technologies quickly and accurately registered high-density scanned points to define the landscape, architectural and geographical information (Biosca & Lerma, 2008). This scanned data is called point clouds.

A point cloud is the collection of several points in three-dimensional coordinates representing the external surface of scanned objects. The information captured, other than points and their coordinates, is colour information, i.e., R (red), G (green) and B (blue) and intensity value (the optical power of the backscattered echo of the emitted signal (Pfeifer *et al*., 2007)).

Normally, point clouds are very large. The number of points in a point cloud may vary depending on the quality and resolution used while scanning (Borenstein, 2012). For example, two data sets captured for this thesis are:

- 4 GB file size with 257 million points for the Church dataset
- 14 GB file size with 581 million points for Fullwood Villa of the University of Gloucestershire data set.

In addition, the geometrical points captured are not organised in an orderly manner, as are those in mechanical reverse engineering. Therefore, it is very difficult to manage and efficiently process large data, including filtration, classification, edge detection, segmentation and geometrical feature extraction.

Many previous types of research (academic and commercial) have been undertaken to process large point cloud data sets. Users in the surveying and civil industry currently use some existing systems/software in the market for processing point clouds. However, various problems emerge when applying these systems for point cloud data processing (Levente & Editors, 2015; Li, 2014; Remondino, 2004). The research suggests that existing software and methods are mainly manual, time-consuming, and dependent on user expertise. Thus, processing point cloud data for a geographical information system is still challenging. This opens a field for research into new techniques and methods. The new system should overcome the current flaws of manual processing and improve the total time required for producing geometrical models and feature detection from point clouds with minimum manual involvement (Govorcin, Pribicevic and Đapo, 2014).

This thesis aims to research, design and develop new methods and algorithms to process point cloud data accurately and efficiently for geographical information systems. The main domains of the study are surveying, architecture, and civil engineering.

3.2 Processing

Point clouds are captured in detail as scanning technology becomes more sophisticated and portable. It is very common for point clouds to have more than a million points. Processing these large data sets is essential for highlighting, capturing and modelling real-world features. Technology advancement accommodates high-specification computers and data storage capacities are becoming more efficient and easier, allowing the point cloud data set to be very large, capturing every detail. Wang *et al*. (2021 p. 9581) described that

> "*Due to the massive data, disorder, irregularity, sparsity, high resolution, and lack of topological relations or texture information, the Point Cloud data processing is complex and challenging.*"

As the data sets can be very large, the processing methods/algorithms must be efficient and robust.

The traditional methods used by the existing geographical information systems typically have issues such as (Remondino, 2004)

1) systems are manual, time-consuming and have low accuracy while processing point clouds,
2) lack of accuracy in identifying the geometrical features from point clouds,
3) lack a robust method for solving the problems of noise and outliers,
4) lack of graphical presentation functions,
5) have few functions for generating high-quality meshes from a point cloud,
6) have limited functions for efficiently and effectively processing huge point clouds and
7) have issues associated with RGB and intensity processing.

Noise

Identifying geometrical objects or features from a point cloud is challenging because of the variable resolution of data, occlusions, missing data and noise (DeVore *et al.*, 2013). The noise can be defined as the points not at the scanned line (Landa, Prochazka and Štastny, 2013). The main factor for 3D geometric information processing of point clouds is the surface shape irrespective of its appearance due to outliers. Therefore, the 3D point cloud processing algorithms need to be invariant to the density of the given point cloud (Unnikrishnan, 2008).

Laser scanners also generate range-dependent noise during data collection as the scanner sensors are based on time of flight, optical triangulation, and multiple frequency phase shifts. Noise level varies as it is mainly affected by the light source on the scanning site (Unnikrishnan, 2008). The divergence of the laser beam is either by reflection or if the light source causes point location uncertainty, which can generate possible outliers or additional random errors across the point cloud.

In addition, mixed pixel discrepancy is generated if more than one scanned surface is placed according to the scanner's line of sight. These mixed pixels are caused due to the non-point spot size of the beam. Therefore, removing outliers or filtration of the point cloud is essential for fast and accurate geometrical object detection in point cloud processing (Tuley, Vandapel and Hebert, 2005).

Edges

This section concentrates on the challenges in edge detection in point clouds. Most existing methods separate the kerbs and incorrectly detect the edges using a height difference of 5 cm. However, the data between the kerb and the road could be missing, creating challenges to identification (Ibrahim and Lichti, 2012).

Edge detection is much easier to implement on images but cannot be applied to 3D point clouds. Edge detection needs automation and optimisation and should be easy to use so that processing them would not require expertise and experience (Dolapsaki and Georgopoulos, 2021). Several methods have been developed for edge detection using 3D geometric properties such as densities and elevation. However, challenges occur when data is missing (Soilán *et al.*, 2019).

Robust methods are identified and developed for point cloud processing: including feature detection and analysing gaps between data and outliers (Nurunnabi, West and Belton, 2015). PCA has been employed for various applications, from neuroscience to computer graphics, in all forms of analysis because of the method used for extracting relevant information from confusing data sets (Tipping & Bishop, 1999).

PCA is a statistical procedure that uses an orthogonal transformation to convert a group of observations of possibly correlated variables into values of linearly uncorrelated variables called principal components. There will be three components in the analysis for this thesis as the points are three-dimensional. The transformation using the eigenvectors of the covariance matrix is defined so that the first principle has the largest variance across the data set, then the second and the third. The first and the second variance produced is enough to identify a planar surface.

PCA is expected to be used for point classification, finding planes across the given data set, region growing methods where the data is missing and many more applications. However, PCA is sensitive to outliers and, where present, gives non-robust inaccurate results. In order to make the data outlier resistant, the principal components produced are used with robust methods (Nurunnabi, West and Belton, 2015).

Features

Different methods have been proposed for feature detection and are used by the current systems. Feature recognition algorithms are used to identify and extract features from point clouds. These features can be anything from breaks of slope to building footprints, from vehicles to tracks, and from edges to corners. This section concentrates on the challenges current methods face in recognising tree trunks and poles in urban point cloud data. Due to the fact that such point clouds can be extremely large, the applied methods should be automatic and time-efficient (Lehtomäki *et al.*, 2012). Another challenge is the variation in point density. The methods that implement the method by calculating points belonging to the same objects fail as a pole's point density can be different on the z-axis than on the other axis. Also, these methods need high computation for higher accuracy (Hůlková, Pavelka and Matoušková, 2018).

In conclusion, to solve the above-discussed problems, new methods and algorithms will be proposed, designed, and developed for point cloud processing, including the classification of objects, analysing, filtering, segmenting, edge extraction, and modelling. The proposed new methods in further chapters (Chapter 4, Chapter 5 and Chapter 6) are anticipated to overcome the problems associated with existing geographic information systems and terrain modelling in the land surveyor and civil engineer domains. The proposed algorithms are effective and efficient for point clouds' semi-automated/automated processing. The expected features and objects to be detected/identified inside huge point clouds are best-fit lines, best-fit planes, edges, edge points, boundaries, trunks, and pole structures in urban point clouds.

3.3 Research Methodology

The research paradigm is an important assumption about the way of viewing the research world, and it is the stance used to contextualise the research aim and present it logically (Grix, 2019). According to Guba (Guba and Lincoln, 1994), paradigms can be characterised through ontology, epistemology and methodology. This research will follow a pragmatic paradigm. Pragmatism rejects the idea that the function of thought is to describe, represent, or mirror

reality. It advocates the use of mixed research methods "*sidesteps the contentious issues of truth and reality*" by Feilzer (2010, p. 3)

and "*focuses instead on what works as the truth regarding the research questions under investigation*" (Tashakkori and Teddlie, 2010). Pragmatism places the research problem as the key concern and applies all the approaches to understand the problem.

This thesis is mainly exploratory but is accompanied by experimental validation work through self-designed and developed software environments. The software in relation to the proposed research is pursued at McCarthy Taylor Systems Ltd (commercial partner for this research). The point cloud software is '3D Vision'; the other application for producing digital terrain models (DTMs) is called 'LSS'. This element will adopt a quantitative approach as the main method of investigation to address the research objective with a "design, develop and test" technique. Both 3D Vision and LSS are used to implement the proposed algorithms and methods for point cloud processing (from filtration, feature detection, segmentation, edge detection and modelling).

3.4 Methodology

This section looks at approaches taken to investigate and answer the research questions mentioned in Chapter 1 through this thesis. The first research objective is to verify and investigate the existing algorithms and evaluate them. The objective is to understand what point cloud processing means, how point cloud processing has been applied to existing methods, and methods proposed to process the point cloud efficiently. A study has been conducted to identify the process, knowledge and technologies required to create models by processing point clouds. Researching this leads to answering the first research objectives. Literature on point clouds and how the information is extracted and processed helps to understand its development throughout the years.

The second research objective is an outcome of the prior research on the first question. The main purpose of point cloud processing is the feature detection process to detect the specific acquisition context of the point cloud data from a laser scanner. Of course, the acquisition context is affected by various factors that affect the detection of the real object, such as

occlusions and incompleteness occurring due to the presence of reflective and transparent surfaces. Therefore, the first challenge in point cloud processing and extracting meaningful information is eliminating noise and outliers. Hence, the research question must be addressed by proposing a method for removing and filtering the noise and outliers.

After filtering and removing the noise and outliers, the point clouds are left with good points and many features. This leads to the third research objective investigating the existing method and algorithms used to extract the features from these filtered point clouds. These features are dependent on the field of the user. For example,

- a road surveyor is interested in features of road marking, kerbs, and road furniture,
- a civil engineer is interested in extracting building footprints and a dip or slope,
- a policeman is interested in finding the reason for the collision; therefore, he wants to extract features around the site,
- A tree surveyor would want to extract trees so that they can preserve the mature trees.

As point clouds are real-world sites, they can be very generalised. This thesis focuses on urban surveyors and civil engineers. The investigation identifies important features that a surveyor and civil engineer want to extract regularly on-site, resulting in the fourth and fifth research objectives.

The knowledge gained from the third research objective led to the fourth and fifth research objectives. The fourth research objective is to identify the most user-required features extracted from any given point cloud. The answer to that question is Edges. The other important feature to extract is cylindrical objects in urban point clouds. These objects are tree trunks, traffic light poles, lamp posts and market poles.

To answer and identify the fourth and fifth research objectives, an algorithm to find edges and cylindrical objects in point clouds is proposed. The comparison and research of these algorithms are presented in various case studies. The commercial software presents the algorithm to extract the feature from point clouds and convert it into DTMs.

3.4.1 Point Cloud Processing Categorization

Point Cloud processing is the term used to process point clouds to extract meaningful information into models. These models can be line models or 3D surface models. In this thesis, point cloud processing is divided into the following categories:

1) Point Cloud Filtration, which includes noise reduction/removal and outlier filtration (proposed new method discussed in Chapter 4)

2) Edge Detection (PCA-based algorithm presented in Chapter 5)

3) Feature detection (Voxel-based algorithm for identifying cylinders presented in Chapter 6) and segmentation (spatial partitioning)

4) Modelling (DTM)

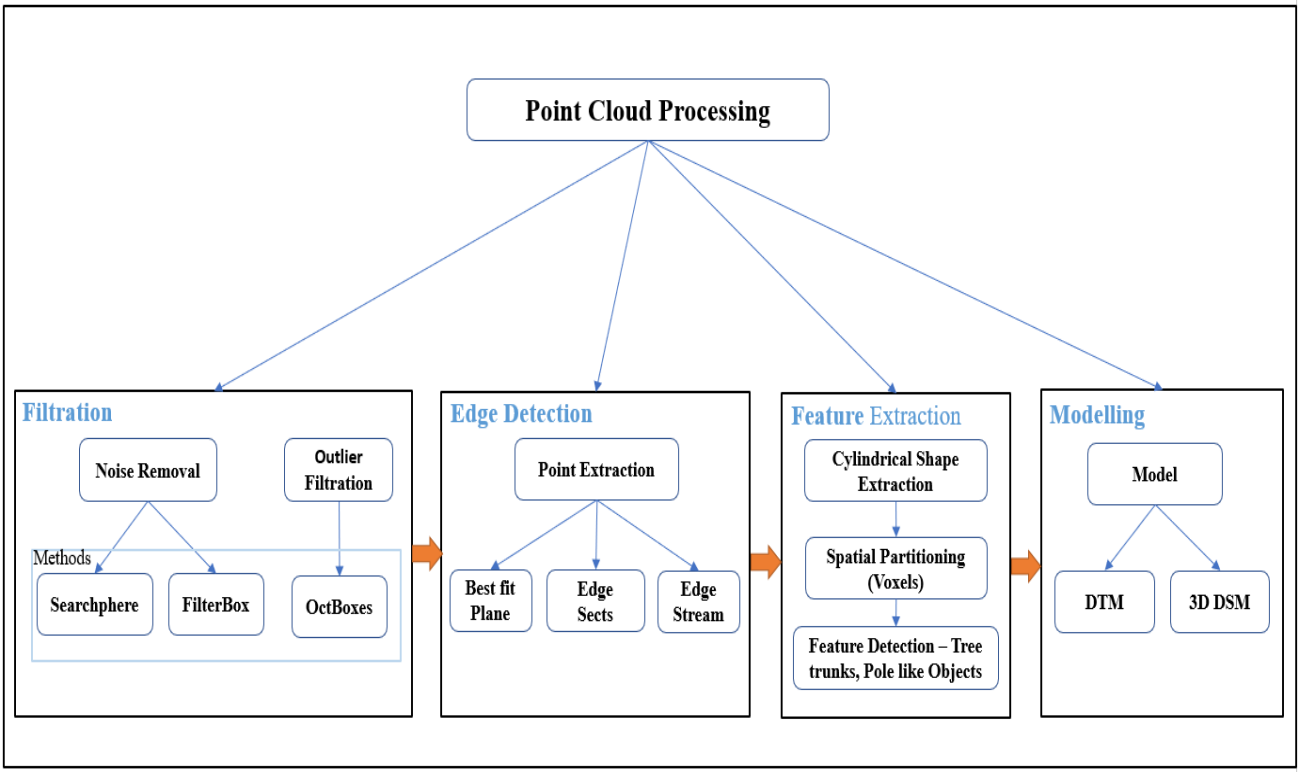


Figure 3.1 Point Cloud processing is categorised in this thesis into Filtration, Edge Detection, Feature Extraction and Modelling

Once laser scanners have acquired 3D raw point cloud data, they need processing to extract useful information. The definition and categorisation of point cloud processing vary across different fields of research and domains. According to Galantucci and Percocol (2005), point cloud processing is divided into pre-processing, segmentation and surface reconstruction. Pre-processing includes noise removal, data compression, smoothing, merging, and tessellation (Galantucci and Percocol, 2005). These processes are carried out before surface modelling and feature analysis in point cloud processing.

The Point Cloud Library (PCL) is a standalone, large-scale, open-source project for point cloud processing and 3D geometry processing. According to PCL, point cloud processing consists of filtering, segmentation, feature estimation, object recognition, surface reconstruction, 3D registration and model fitting (Rusu and Cousins, 2011). Many point cloud processing software systems transform raw point cloud data into 3D images or BIM.

In this thesis, point cloud processing is divided into four stages to transform the data into models, as shown in Figure 3.1:

1. Filtration – proposed methods let users filter outliers (using Oct boxes) and remove noise (using search sphere and filter box) from point cloud data sets.

2. Edge detection – proposed algorithm lets users find edges sects in realtime with the move of a cursor and automatically using edge stream.

3. Cylindrical feature detection – proposed algorithm lets users automatically recognise and extract features such as tree trunks, poles, marker poles, traffic poles, and lamp posts from urban point cloud data sets.

4. Models – let users extract information (points and links) from the point cloud by digitising to create DTMs and DSMs (3D surface models).

The purpose of point cloud processing of data from laser scanners is to transform the scanned data into a model. For that process and the acquisition of objects, various algorithms are implemented. First, filter and eliminate the noise and outliers that are not part of the features. Second, extract features such as edges and edge streams (lines) from the data, reducing the

complexity of the next stage. Third, to segment and identify complex structures and calculations. Finally, feed the extracted information into the model to present the data.

3.4.2 Proposed Algorithms

The research is based on 3D computational geometry. The research establishes various mathematical models to propose, design and develop new methods and algorithms. PCA will be used with other methods. Along with PCA, other methods such as (1) Nearest cluster method - is method that can be used to perform several types of agglomerative hierarchical clustering, in which a hierarchy of clusters is created by repeatedly merging pairs of smaller clusters to form larger clusters, (2) Voxelisation - is a spatial partitioning technique that transforms the points into voxels grids and estimates the geometries and attributes that are created by points inside the grid (Xu, Tong and Stilla, 2021), (3) Octree - is a tree data structure with exactly eight sub-nodes often used to partition a three-dimensional space by recursively subdividing it into eight octants. To achieve the objectives of the research, the above methods are used.

3.5 Methods for Data Collection

The research is based on extracting valid information from point clouds, so point cloud data plays a major role. Two types of point cloud data are used in this research.

(1) The primary data were collected using a laser scanner (a laser scanner manufactured by FARO Technologies UK Ltd) to validate new algorithms and compare the effectiveness and efficiency with the existing implemented methods.

(2) The secondary data was collected through a literature review of existing methods and algorithms for 3D point cloud processing and feature extraction (identification, analysis and modelling) of 3D objects.

3.6 Methods for Data Analysis

Different approaches are defined for analysing the data (points) in the point cloud, including techniques to show high dimensional structure through low dimensional representation and assemble discrete points into a global structure. Some traditional techniques are (1) projections or axis-based technique, (2) eye view, (3) high dimensional analysis, and (4) structural analysis using simplicial chains for point cloud processing. However, the traditional techniques lack issues as described above. To overcome those issues, the proposed data analysis method for the thesis will use Principal Component Analysis's eigenvalues and eigenvectors for the direction of the dominant data, which is further extendible for filtering, segmentation, and edge detection.

3.7 Methods for Algorithm Validation

The new methods are tested on several point cloud data sets to verify if they are more effective and efficient in identifying features such as planes, edges and cylinders. Also, the proposed new methods and algorithms are coded (using the computer language C#) and integrated into McCarthy Taylor Systems' software application called **3D Vision**. The methods are tested by the company's existing users/clients (civil engineers, geologists and land surveyors). The user's feedback is applied to improve the algorithms.

3.8 Research Ethics

The research follows the moral and ethical guidelines of the University of Gloucestershire. As the study involves point cloud software (licensed by McCarthy Taylor Systems Limited), the technology, process, and scanned data are strictly confidential, and information will only be used for academic purposes.

3.9 Chapter Summary

Point Cloud processing methods are frequently developing and evolving as point clouds become popular. However, improving the efficiency and accuracy of feature recognition and extraction has been long highlighted as an issue across the surveying industry. Land surveyors are the collectors, suppliers and processors of these data and need a new method for transforming the 3D data into meaningful models. In addition, laser scanning technology has been specified as the preferred collection tool as the scanners are fast, portable and comprehensive in capturing the surrounding environment.

Chapter 4 A Method for Noise Removal and Outliers Filtering

4.1 Introduction

Point clouds are often collected at a site that can be indoors or outdoors. To capture point clouds, scanners emit laser beams that record the objects' surface and are therefore affected by outliers and noise. The objects scanned are similar to the real world but with much distortion. The point cloud data set is affected by outliers as the laser scanners generate falsification points due to incorrect processing, scanner path reflection and unwanted objects. A typical point cloud scanned data usually is noisy, sparse and temporarily incoherent (Ning *et al*., 2018). In geoinformation systems, accuracy is an important aspect of designing and maintaining, which is highly affected by noise. System accuracy is measured in terms of absolute, relative and precision (Lewis, 2021). Therefore, removing noise and outliers is significant for accurate and efficient results. Rakotosaona *et al.* (2020) argued a point cloud data set clean-up method should balance between denoising and feature preservation. An outlier in point clouds fits a description by Hawkins M D (1980, p. 1)

"*An outlier is an observation which deviates so much from the other observations as to arouse suspicions that a different mechanism generated it*".

In this thesis, Outliers and Noise are categorised as follows:

1) Outliers are defined as isolated point/s, as shown in Fig 4.1.
2) Noise is defined as a non-isolated points cluster that is not part of any relevant features or objects, as shown in Fig 4.2.

The scanner generates millions of points, which requires high computation time to process all the points. Hence, removing these outliers and noise is necessary to analyse and process the point clouds efficiently. A proposed method uses applications to remove the outliers and noise

from the point clouds before processing the data to extract meaningful geoinformation. The method is tested on different data types. Each application has a unique implementation to remove the points that are considered outliers and noise. The analysis of related work is discussed and evaluated in Section 4.2. The proposed methods are presented in Section 4.3. Section 4.3.2 presents stage 1: the outlier/noise categories and Section 4.3.3 introduces stage 2: the application's distinctive suggested usage of the type of outlier and noise. Section 4.4 presents the result analysis by demonstrating the proposed methods for the commercial software. Finally, Section 4.5 presents the conclusion of the methods.

This chapter proposes three methods for noise removal and filtration of outliers.

1) Noise Removal using Sphere (NR-S)
2) Noise Removal using 3D Box (NR-B)
3) Outlier Filtration using Octree Boxes (OF-OB)

The noise removal and outlier filtration method from Chapter 4 is implemented on the data sets prior to implementing the proposed algorithms presented in Chapter 5 for edge detection and Chapter 6 for tree trunks and pole-like objects detection.

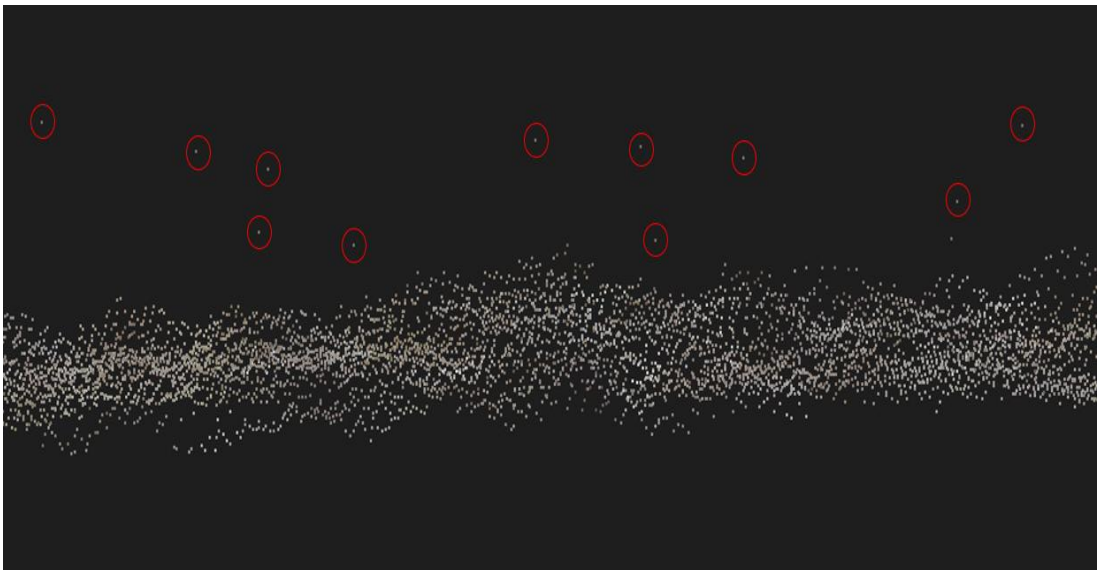


Figure 4.1 Example of outliers (in red circles) in point cloud data

Figure 4.2 Example of noise, such as moving people captured in point cloud data

4.2 Analysis and Evaluation of Existing Methods

The filtration of outliers and noise is a continuing research problem in many fields of study, such as computational statistics, computer graphics etc. (Griffioen, 2018). In geoinformation systems, outliers/noise removal is the fundamental data processing task to ensure the quality of scanned point cloud data (Ning *et al.*, 2018). Therefore, removing these points is essential. However, the points are classified and identified before removing them to separate the good points from the bad points (outliers/noise). This section presents and evaluates the existing methods of removing outliers, followed by a summary underlining the existing method's drawbacks and problems.

4.2.1 Existing Methods

The existing methods are presented by dividing them into five categories. (1) Density-based methods by (Ning *et al.*, 2018; Zhang Bibo *et al.*, 2017; Li & Wei, 2021; Sotoodeh, 2006), (2)

Surface fitting methods by (Arvanitis *et al*., 2018; Jia *et al*., 2018; Nurunnabi *et al*., 2015a; Shao, Ijiri and Hattori, 2015; Y. Wang & Feng, 2015), (3) The type-based method by (Ge and Feng, 2021), (4) Statistical Outlier removal (SOR) by (Balta *et al*., 2018; Pirotti *et al*., 2018; Rousseeuw & Hubert, 2018; Rousseeuw & Leroy, 1987; Yin, Wan and Liu, 2013; Zeybek, 2021b), (5) K-d tree-based by (Luo and Liao, 2010; Shen *et al.*, 2011).

1) Density-Based Methods

Breunig *et al*. (2000) introduced the density-based Outlier detection algorithm based on knowledge discovery in database (KDD) applications. Sotoodeh (2006) proposed the outliers detection method based on the Breunig KDD algorithm. First, the algorithm calculates the local outlier factor for neighbourhood points within the defined distance. Then, the *k*-distance of a point, the *k*-distance neighbourhood of a point, the reachability distance between two points and the local reachability density of a point are calculated. The local outlier factor is calculated as the local reachability density of a point is inverse of the average reachability distance based on the nearest neighbour. The outlier factor of the point decides if the point is an outlier or not. Finally, the clusters of outliers are detected based on nearest neighbours. The advantage of the algorithm is that it uses the local behaviour of points that work for different densities. The disadvantage of the algorithm is that it detects the part of outliers, not all the outliers, due to point density similarities.

The method by Li and Wei (2021) presented outlier removal from UAV point clouds. DBSCAN (density-based spatial clustering of applications with noise) is used to denoise the pylons and enhance the DBSCAN algorithm to denoise electricity transmission lines. The method first roughly classifies pylons and powerlines from vegetation and ground. Then, the pylons are extracted by projecting the point cloud on the XOY horizontal plane, followed by k-d tree implementation to search and group points. The search radius for the k-d tree is dependent on the size of pylons in different scenarios. The extraction of powerlines is accomplished based on height criteria. Next, the denoising algorithms are implemented. A k-d tree accelerated DBSCAN algorithm is used to cluster and denoise pylon points, followed by a statistical outlier removal (SOR) filtering method used to denoise the ground points. Principal Component Analysis (PCA) is then used to denoise powerlines. Separating powerline points

from noisy points becomes difficult because of the lean structure of powerlines. Therefore, separating and denoising are achieved by finding the principal axis between two pylons. The accuracy of the proposed method is 98%. However, the method lacks to provide the details of point-based rough classification. Also, the method is tested on a single point cloud dataset; hence, there is no adequate proof that the method will work on different point clouds.

Zhang, Xiang and Zhang (2017) proposed a density-based approach to remove large-scale noise from point clouds. The method uses DBSCAN and is divided into three approaches 1) local consistency factor, 2) parameter estimation and 3) distance measure. The method starts with calculating the local consistency factor (LCF) to indicate the local density similarity of points. The dense points are closer to each other than sparse points; the LCF is distance-based on the mutual reachability distance of point sets with variable densities. Next is parameter estimation, which is assessed using reliable points from an inlier and trust points are found using these estimations. Next, a colour-based distance measured approach is implemented to classify inliers and outliers. The spatial and colour distances are combined to measure the difference between the two points. Then, the Gaussian kernel and spatial distance by LCF are compared to the group to reject the points. The final step is the density-based clustering method. The clusters are selected as inliers if they satisfy the density consistency. The advantage of this approach is that it automatically estimates the parameters to reduce user interactions, and screening reduces complexity. The disadvantage of the approach is that it is based on the assumption that objects will have the same point density. The point distance calculated between any two points has to be continuously picked at different points, which will take a lot of time, making the method slow and time-consuming.

Ning *et al*. (2018) presented a method to remove outliers based on geometrical characteristics. The two geometrical characteristics are local density and deviation from the local fitting plane. The outliers are divided into three types: sparse, isolated and non-isolated. Firstly, the local density of points is analysed by calculating the local covariance matrix of each point in the *k*-nearest neighbourhood. Then, the probability of a point belonging to an outlier is calculated. The local density only works on isolated and sparse outliers. Second, for non-isolated outliers, the local plane fitting method is applied as they are closer to objects. Finally, a local plane is fitted on the target outlier points by applying PCA. The fitted plane depends on the *k*-nearest neighbour value; if the value is bigger, it deforms the model, and if it is small, it does not fit the model. The limitation of the method is that it does not work on high-density data. Therefore,

the method gives good results for sparse and isolated outliers but not non-isolated outliers. Moreover, all the points in the model are analysed for the *k*-nearest neighbour, which costs more computation time. Also, the method does not present the test results on large point cloud data but is tested on sampled objects.

2) Surface Fitting Methods

Jia *et al.* (2018) proposed a method to filter the point cloud based on the surface variation. The method first estimates the normal vector using the weighted principal component analysis. Then, the weight of points in the neighbourhood is allocated based on the distance between points and the neighbourhood's mean. The weighted analysis helps determine a point's position in the neighbourhood and the distribution of outliers. Then, the surface variation factor of points is derived. Second, the point cloud model is divided into flat and mutant regions by comparing these derived surface variation factors of sample points with the average surface variant factor of the sampled *k*-nearest neighbouring points. The planar surfaces with small surface variations are flat regions, and large surface variations and more noise are mutant regions. Finally, an improved median filtering algorithm is applied to flat regions, and an improved bilateral filtering algorithm is applied to mutant regions to denoise. In the median filtering algorithm, all points in the neighbourhood are projected to the normal vector, and projected values are calculated and sorted. The position of a point is derived from the median point. In a bilateral algorithm, the sampling point is moved along the normal vector. The angle change between the neighbourhood point and sample point considers whether outliers influence the surface or not. The advantage of the method is that it considers two different types of surfaces for outlier removal. The disadvantage of the method is that it was tested on the sampled point cloud. Also, the method lacks 1) real point cloud data testing with different shape and size features and 2) no estimation of the time consumption for the applied method.

Nurunnabi, West and Belton (2015a) proposed a method for outlier detection and surface normal-curvature estimation. The outliers are detected locally in their neighbourhood and removed for more accurate local saliency features. First, PCA is implemented to derive the local region by searching its local neighbourhood. Then, to search neighbourhood, *k*-Nearest Neighbourhood is applied to determine the *k* points with the least distance. Next, the maximum

consistency set is extracted by implementing the plane and calculating the orthogonal distance of all points to the plane. Next, the *z*-score, median, and absolute deviation produce a robust *z*-score. In addition, another algorithm, Robust Mahalanobis Distance (RMD), is obtained by mean and covariance matrix. Finally, the two methods are used for outlier detection and result in points marked as inliers and outliers. The advantage of the method is that it is used for both outlier detection and saliency feature extraction by fitting a plane. However, the disadvantage is that the method lacks the ability to define the types of outliers detected.

Another method by Wang and Feng (2015) used surface curvatures for outlier detection using majority voting. The outliers are divided into sparse, isolated or non-isolated. The method is focused on non-isolated outliers that are more challenging to detect. The voting scheme is used to differentiate between non-isolated surfaces and scanned surfaces. A surface is fitted on points with small variations stated as regular points. The regular points are identified by implementing PCA in each point's *k*-nearest neighbourhood to record the surface variation. A minimal ellipsoid and fitted plane smoothness determine the local surface variation. The neighbourhood point's curvature and noise information determine whether it is a sharp feature, smooth region, high noise level or high curvature. Next, a histogram is generated based on point population and surface variation. The histogram is used to classify regular and irregular points by Bi-means clustering.

The method results in regular points with low variation and irregular points as outlier connection regions. These irregular points are analysed by a majority voting scheme to categorise each point as a good point or an outlier. For each irregular point, only neighbouring regular points are used as voters. Each voter evaluates if the irregular point fits into the local geometry. The voting average is considered, which results in isolated and sparse outliers. Finally, these outliers are removed using a clustering algorithm and boundary criteria to keep the good points. The advantage of the method is that 1) it can differentiate between non-isolated outliers and other types of outliers, and 2) it uses boundary criteria instead of cluster size. The disadvantage of the method is that using the boundary criteria could cut some good points. Also, the method compares each point's neighbourhood twice, first to record the variation in the surface and second to each point's neighbourhood during the voting scheme to check whether the point is an outlier or a good point. Hence, the method is not time efficient, and the test data are also limited.

Shao, Ijiri and Hattori (2015) used ellipse fitting for outlier removal. A point set is fitted with an ellipse to minimise the sum of inlier points to the ellipse curve projection distance. Firstly, the method starts with algebraic ellipse fitting based on Taubin's method. Secondly, the Maximum-Likelihood-based (ML) method is used to refine the noise measurements. Thirdly, edge points are collected around the estimated centre point. Fourthly, three stages of the method are implemented.

1) Clustering the data points based on proximity to derive the inliers.
2) Searching the subsets to minimise the algebraic fitting distance.
3) Refining the algebraic solution by geometric fitting.

The clustering is accomplished based on the Euclidean distance from its neighbour. Breadth-first searching is used to reduce the energy function and maximise the number of inliers. Finally, all the inlier points are refined based on geometrical parameters. The advantage of the method is grouping of inlier points by ellipse fitting. The disadvantage of the method is that it is not efficient to process large datasets. Furthermore, the method only works on the grouped outliers, not the isolated or closer object's surface.

Arvanitis *et al.* (2018) presented another method to detect outliers in urban environments. The detection is achieved by:

1) Exploiting the spatial consistency of an object's geometry
2) The sparsity of outliers in the spatial domain

Small clusters share the same geometrical information based on the analysis and observation in point clouds. Therefore, a spatial coherence matrix is calculated. Then, a cluster consisting of *k*-nearest neighbours for each point is created. Robust PCA (RPCA) is applied, resulting in low-rank and sparse component matrices. The low-rank matrix contains coordinates of scanned points, and the sparse matrix contains offsets of outliers and low-rank positions. Next, an Augmented Lagrange Multiplier (ALM) scheme is applied to optimise the convex matrix and optimise the iterations of the variables using an alternative direction. Next, various operators are calculated, such as 1) the shrinkage operator for element-wise application and 2) the singular value thresholding operator for enhancing runtime. Finally, the outliers are identified using a sparse matrix and are removed based on the selected threshold. The advantage of the method is that it can remove both large-scale and small-scale outliers. The disadvantage of the

method is that it uses a *k*-nearest neighbour search, which means that the computation time will be significant for huge point clouds as each point is clustered based on its nearest neighbour.

3) Type-Based

Ge and Feng (2021) proposed a type-based outlier removal framework (TBORF) method. The method is divided into two stages 1) Determine the input point cloud type with three metrics, and 2) Deal with outliers for each type. In the first stage, the three metrics are used to divide the input point clouds based on characteristics. The three metrics are point cloud thickness, uniformity and ambiguity. In addition, three metrics are independent of translation, rotation and scale.

1) The geometric shape determines the thickness.
2) The point distribution determines the uniformity.
3) The discrimination index between outliers and regular points determines the ambiguity.

The geometric shape to get the thickness is evaluated by support points. The support points are defined as the points that obey the planar distribution. The point distribution to get the uniformity of point clouds is achieved by calculating the distance between objects and the scanner. Fuzzy points determine the ambiguity of point clouds. In the second stage, according to quantitative results, the point cloud is classified into type 1, type 2, type 3 and type 4. Common characteristics of each type of point cloud are evaluated. Finally, outlier removal methods are implemented based on *k*-dist, local neighbour and Tukey's fences by 1) the single-criteria method and 2) some concepts in the type-based combined method. The single criteria methods to remove the outliers are:

a) outlier detection via local neighbours' gradient distance based on gradient distance which is the distance between a 3D point and its neighbours,
b) outlier detection via local neighbours' farthest distance which is the largest distance between outlier and regular points, and
c) outlier detection via central limit theory, which computes the average gradient distance.

The advantage of the method is that the outliers are removed based on point cloud analysis, their distribution, the geometric shape of objects and distances of points to their neighbourhood and scanning system. The disadvantage is that the method can not automatically remove outliers depending on geometric shapes. Furthermore, the method is only focused on isolated outlier points.

4) Statistical Outlier Removal

The SOR methods remove outliers based on their distance in the point cloud. SOR also includes distance-based methods. Rousseeuw and Leroy (1987) first introduced the statistical outlier removal methods (Rousseeuw and Hubert, 2018).

Zeybek (2021b) proposed a method that detects outliers based on the patterns of points. The method focuses on isolated outliers and isolated and clustered outliers. Firstly, statistical outlier removal filtering is implemented. Then, each point is analysed based on the distance from its neighbouring points to determine the distribution of points. Next, the points are clustered based on the Euclidean distance for isolated and clustered outliers. Then, Euclidean clustering, followed by the distance tolerance value, is applied. The SOR filtering is used with eigenvalue computation to identify the relationship between neighbouring points. Eigenvalues determine the shape characteristics. Secondly, the machine learning system reclassifies the filtered outliers. Finally, both a Random Forest and Support Vector Machine are used to classify the points as inliers and outliers. The advantage of the method is that it uses the geometrical shapes of the objects to classify them as inliers and outliers. The disadvantage of the method is that it cannot be used for random outliers close to objects as the method is not surface-based.

Balta *et al.* (2018) introduced a method called Fast Cluster Statistical Outlier Removal (FCSOR), an extension of the statistical outlier removal method that uses voxels to subsample the data. The method is divided into two stages: data handling and filtering.

1) Data handling - Voxels are used to downsample the point density. The sampling helps as density varies with range measurement. The point cloud is divided into 3D grids using the same size or the same number of voxels in each direction. Next, the centroid of each voxel is calculated and output.

2) Fast cluster statistical outlier removal – The 3D space is divided into clusters. Each point in the cluster calculates the euclidean distance to its *k*-nearest neighbour. Then, the points are added to the clusters based on the calculated distance. Next, the clusters are filtered by calculating the mean of clusters; if the mean is higher than average, that cluster is rejected.

The method claims that it is faster than existing outlier removal methods; however, it implements the Euclidean distance that has to be calculated for each point to decide which cluster it belongs to. This process is very time-consuming; as the data set gets big, it will take longer to process as the method checks each point.

Yin, Wan and Liu (2013) proposed another method of removing outliers by statistical analysis. The first step is pre-processing the point cloud by determining each point's *k*-neighbours. Then, the Euclidean distance is estimated to determine the approximation of points. The second step is statistical analysis 1) Mean distance is calculated for each point to its neighbourhood points 2) Estimating distribution based on mean distance. The third step is filtering. Finally, the points with a mean distance similar to their neighbours remained. The outliers are deleted based on the mean distance filtered by distance expectation and standard deviation. The advantage of the method is the implementation of squares calculus to save the implementation time as each point neighbour search is done. The disadvantage of the method is that not many data sets have been tested. Furthermore, the filtering is based on standard deviation, which is different for point cloud data sets. The method could not be implemented for large point cloud data because of the long computation time.

Another method by Pirotti *et al.* (2018) presented outlier removal using two methods Statistical Outlier Removal (SOR) and Local Outlier Factor (LOF). First, four predictors are calculated two for each method – SOR and LOF of each point, and an absolute difference in their median SOR and LOF values. Median values reflect the correct distribution of points. Points with SOR and LOF values further away from the median are considered not outliers. SOR is a distance-based method to access each point for an outlier. Then, the local density is calculated using the distance between a point and its nearest neighbour. Next, LOF is a method that assigns a score to each point based on its local density deviation among the neighbouring points. So, the outlier points have a lower density than the inlier points. Next, the number of neighbours is chosen based on the point cloud data set, which can vary. *K*-distance is calculated as every distance

between a point and the *k*-nearest point. Then, *R*-dist is the reachability distance of each point, and its *k*-neighbour is calculated. Next, the local reachability density (LRD) is calculated, which is the inverse of the average reachability distances of each point. Finally, the LOF value is calculated using LRD values to classify points as outliers. The method's advantage is that the SOR and LOF combination is implemented to analyse the points based on their density and distribution. The disadvantage is that the method used to calculate the distance of each point and its nearest neighbour is correlated to the time consumption. If the point cloud data set is large, time consumption will be greater. Further, selecting the nearest neighbour number depends on the user's knowledge and experience.

5) K-d tree based

Luo and Liao (2010) presented an algorithm to detect outliers using slicing, local distance-based outlier factor and a k-d tree. First, the algorithm slices the point cloud based on the reference Z-axis and a group of planes perpendicular to the selected Z-axis. Then, the points whose distance to the slice plane is less than half of the slice interval are grouped. The k-d tree is built for those projected points to accelerate the neighbour query efficiency. Next, the local distance-based outlier factor (LDOF) is used for the outlier detection criteria. Then, the local distance of a point to its *k*-nearest neighbour is calculated and sorted. Next, the points with higher LDOF values are removed. Finally, all the slices are processed until a clean point cloud set is left. The advantage of the method is that the slicing helps to analyse the points. The disadvantage of the method is that it selects slicing intervals. Moreover, the algorithm lacks the evaluation of different point cloud data sets. Also, the algorithm requires iterations to clear the point cloud; these iterations are not specified clearly, such as stopping criteria are not defined, and the iterations are not tested for efficiency, i.e., time consumption.

Shen *et al.* (2011) presented a k-d tree method to remove outliers. The k-d tree is a binary tree where each node is a *k*-dimension. Based on visual observation and analysis, the distance between the outlier and *k*-nearest points is larger than the distance between normal points and its *k*-nearest neighbour. Therefore, for outliers, *k* values are small. The outliers are defined using the following criteria:

1) According to the elevation histogram and threshold elimination, low and high outlier points,
2) a k-d tree is constructed of the remaining points,
3) The average distance between a point and its *k*-nearest neighbour is computed,
4) The threshold is used to separate the normal points and outliers.

The threshold is based on two values: the average distance histogram and the distance threshold to its *k*-nearest neighbour. The advantage of the method is using a k-d tree and elevation histograms to eliminate the outlier points. The disadvantage is that each point is checked for low or high outliers. Moreover, the *k* value assigned for the *k*-nearest neighbour is not defined and is implemented by trial and error.

4.2.2 Summary

This section presents the summary of the existing methods presented and reviewed in Section 4.2.1. The method's authors, their approach, the size of the data set used to present the method and other information like the type of scanner used to collect the data or the type of models are shown.

Table 4.1 Summary of existing methods

Authors	Outlier Removal Methods Approach	Data Set Size	Other
Zhang, Xiang and Zhang, 2017	Density-based and Clustering	Artefact 125k pts, Plant 379k pts	Synthetic and real data models
Li and Wei, 2021	Statistical based, Density-based, k-d tree-based	Real-world data set – 63 M pts	UAV

Sotoodeh, 2006	Density-based	ALS – 11 M pts	ALS and TLS
Ning *et al*., 2018	Density-based and Plane fitting	15k pts and large 58 M pts	3D models
Jia *et al.*, 2018	Surface estimation	Small data set	Stanford Bunny
Nurunnabi, West and Belton, 2015a	Distance-based and Surface plane fitting	13k pts	MLS
Wang and Feng, 2015	Plane fitting and Clustering	Gear – 280k pts, Room – 968k pts	Digital camera LDI
Shao, Ijiri and Hattori, 2015	Ellipse fitting	Small data set	Synthetic
Arvanitis *et al.*, 2018	Plane Fitting	Urban data set	LiDar
Pirotti *et al.*, 2018	Statistical Outlier Removal (SOR) and Local outlier factor	99 M pts	SFM and FCD
Balta *et al.*, 2018	Statistical Outlier Removal (SOR), Voxel-based	8-9 M pts	UGV
Yin, Wan and Liu, 2013	Statistical Analysis	45 M pts	-
Zeybek, 2021b	Statistical Outlier Removal (SOR)	41 M pts, 26 M pts, and 10 M pts	ALS
Ge and Feng, 2021	Type-based	110-140k pts	Shapes
Luo and Liao, 2010	k-d tree based	33 M pts	Leica (TLS)
Shen *et al.*, 2011	k-d tree based	Rural & Urban	LiDar

Abbreviations used in Table 4.1 are as follows:

- Pts- Points, TLS- Terrestrial Laser Scanner,
- MLS- Mobile Laser Scanner,
- ALS - Aerial Laser Scanner,
- LiDar – Light Detection and Ranging (Aerial),
- UGV - Unmanned Ground Vehicle (Terrestrial scanner),
- UAV- Unmanned Aerial Vehicle (LiDar),
- LDI – Laser Direct Imaging (Surveyor laser scanning system),
- SFM- Structure From Motion (Photogrammetric),
- FCD – Floating Car Data (Captures Road Networks)

4.2.3 Analysis of the Problems

In Section 4.2, the existing methods are analysed and discussed. Table 4.1 summarises the existing methods, approaches and point cloud data sets used for testing. The common problems of density-based, surface fitting, type-based, statistical-based and k-d tree-based methods have been identified. The methods lack a clear definition of outliers that have been identified and removed. In addition, several existing methods use *k*-nearest neighbour search for each point, which is very costly and time-consuming. Even with modern technology and high-specification computers, the point-based search within a radius has computation time problems. The larger the point cloud, the longer it will take to process each point, which is inefficient.

Furthermore, the test data sets used by existing methods are not real-world point cloud data. The outliers of the real-world data set have different problems compared to synthetic data sets.

To overcome the abovementioned problems, a method is proposed that is completely adaptable and flexible enough to detect all types of outliers. Moreover, the proposed method is adaptable to individual needs of removing outliers. For example, the outliers and noise can be buildings and vehicles for road surveyors, whereas, for BIM designers, the outliers can be passing people and trees.

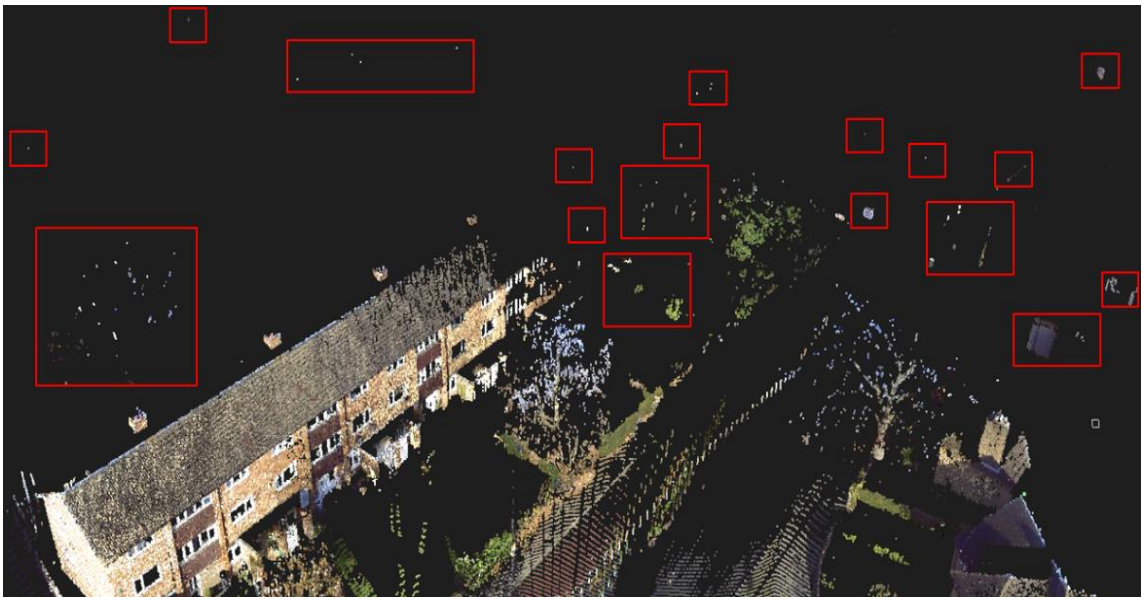
4.3 A Method to Remove Noise and Filter Outliers

4.3.1 Overview

In Section 4.3, the proposed method is presented and discussed. The proposed method has been presented in two stages (1) Defining the outlier types based on three criteria, and (2) Based on the outlier type and density, three applications are implemented. The unwanted points in the data set outliers/noise are defined as follows in this thesis

1) **Outliers** are global, isolated and scatter points in 3D with low point density, as shown in 4.3 (a). These sparse points could be close to objects or in space.
2) **Noises** are local, non-isolated, present in point clusters in 3D and are not a meaningful feature for the 3D modelling. Noise is also dependent on the type of user using the point clouds. For example, noise can be building and vehicles for a tree surveyor, whereas a tree can be noise for an architect or civil engineer.

This chapter presents a method for the outlier filtration and noise removal in two stages: (1) Stage 1 identifies the type of outlier or noise demonstrated in Section 4.3.2, and (2) Stage 2 presents the applications based on the identified type used for noise removal and outlier filtration demonstrated in Section 4.3.3.



(a)

(b)

Figure 4.3 Examples of (a) outliers and (b) noise are shown in red boxes and circles

4.3.2 Tools

This section presents the tools used in this thesis. The proposed algorithms and methods are academically discussed and presented; however, making user-friendly and easy-to-use systems for commercial applications is essential. The tools in this section are the bridge between the proposed algorithms and the commercial software. These tools are used to handle large point cloud data by sampling the points and providing faster calculation results on the points. For this chapter, three tools are presented for point cloud filtration.

1) Search Sphere

The search sphere is one of the tools invented for the first time in this thesis. The search sphere is versatile and has many applications. A search sphere is a method of sampling the point cloud

using a real-time 3D sphere. It is constantly searching for the points inside the sphere. The search sphere is rendered at the movement of the cursor in 3D space. The main reason for choosing the sphere for searching is because the sphere is the quickest to calculate inclusion compared to other geometric shapes such as cylinders, cubes and triangles. The principle of inclusion in 3D is checking that a point $p(x, y, z)$ is located inside or outside the boundaries of a geometrical shape, as shown in Fig 4.4.

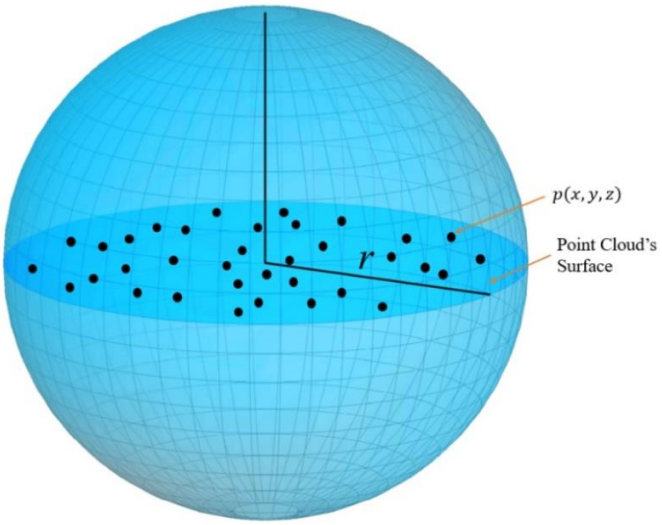


Figure 4.4 Representation of points inside the search sphere

The proposed algorithm is part of commercial software and is a user-oriented solution. Hence, the other reason for choosing a sphere for searching and sampling, unlike other geometrical solid shapes in three dimensions with length, width and height, a sphere can be defined just by a diameter. Therefore, it is suitable and easy to manage if the user wants to change the size, so they have to adjust one value (diameter) instead of three values. Moreover, as the point clouds are real-world data acquisitions, each point cloud is different from the others; thus, it is crucial to provide the flexibility to the user to change the size when required. The search sphere's size is user changeable. The recommended size depends on the user's applications and the size of the feature to be extracted. All the points inside the search sphere are highlighted as the chosen points are apparent.

2) 3D Box

A box has the properties of six faces, eight vertices and twelve edges in 3D. The 3D box is used in this chapter to remove noise points shown in Fig 4.5. The reason that a 3D box is fit for the task is 1) it can cover a larger area as compared to the search sphere, 2) The six phases of the box give the user flexibility to manipulate the phases to cover the points of a particular object or area in the point cloud. An example of the 3D box is shown in Fig 4.5 and is presented in detail in Section 4.3.4.

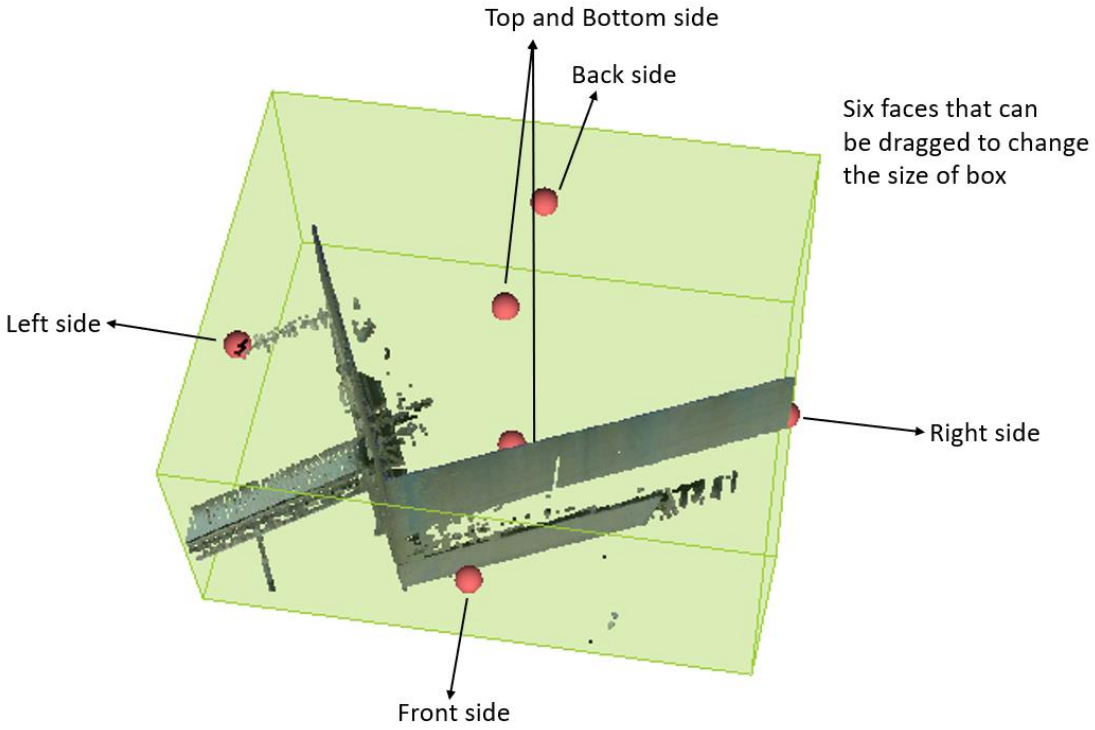


Figure 4.5 A 3D box (green) with six faces and a point cloud inside it

3) OctBox

The third tool used in this chapter is OctBox. This chapter presents the use of OctBox for the filtration of points for the first time. The literature suggests that this innovation is a new contribution and has never been used. OctBox is the 3D box constructed from the tree data structure of Octrees. The Octree structure is used to spatially divide the point cloud's points for storing points and hierarchical information. The points stored in the trees are presented in a box which is then used to filter the points, as shown in Fig 4.6.

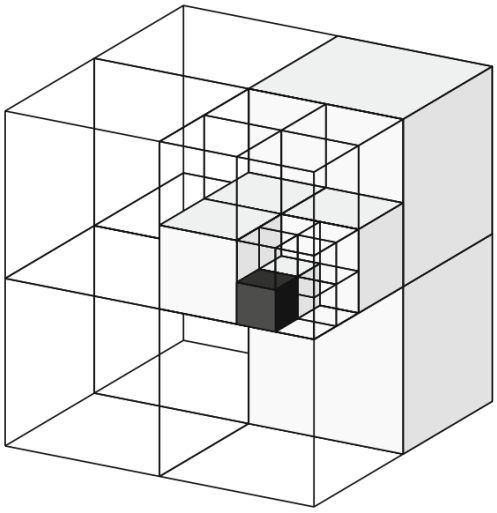


Figure 4.6 Highlighted box in Octree structure

4.3.3 Stage 1: Outliers and Noise Types in Point Cloud

Finding and removing outliers and noise in the point cloud could be challenging due to their nature of being dispersed and non-correlated.

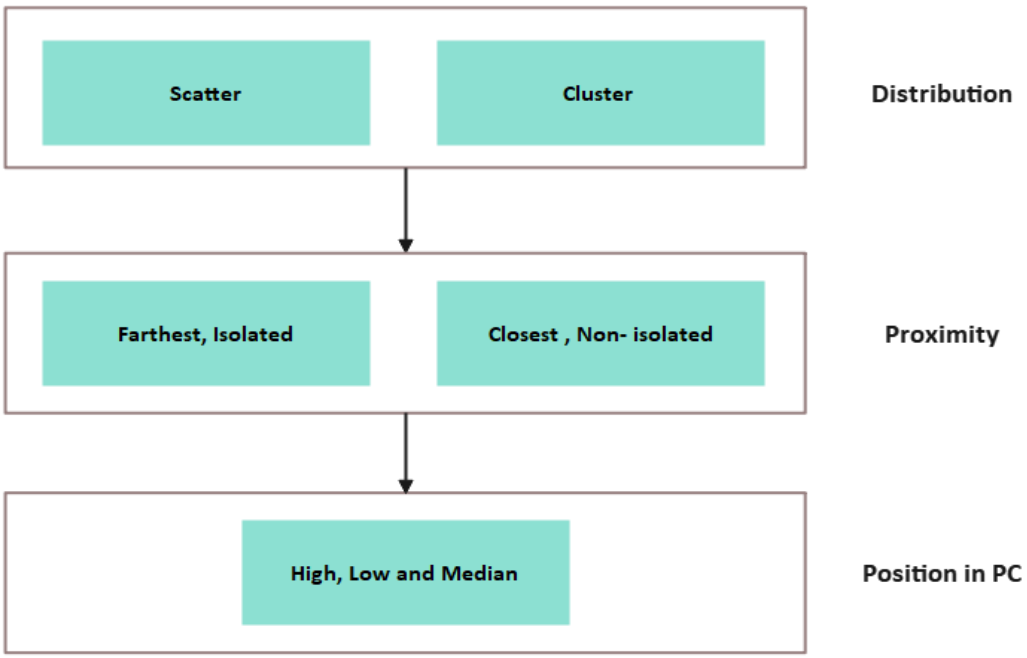


Figure 4.7 Outlier/Noise classification according to their characteristics

Therefore, the proposed method categorises the outliers and noise to understand well in order to use the applications presented in Section 4.3.3 accordingly. However, as discussed in Section 4.2.3, the existing methods lack clear criteria for defining the outlier and noise within the same point data sets, leading to problems in their methods. Outliers are defined by three characteristics in this thesis to address these issues. The categories allow efficient organising of the outliers and noise to understand further to deal with them. These categories are:

a) Points distribution
b) Points proximity to other objects
c) Points position

4.3.3.1 Point's Distribution

The first categorisation is *Distribution*. The laser scanners collecting point cloud data sets use laser beams. Terrestrial laser scanners capture objects as their laser beams fall on the object's surface and return the points. As a result, objects closer to the scanner have full coverage, i.e., high-density points, and the farthest object surface has less coverage, i.e., low-density points. Similarly, mobile and handheld laser scanners capture variable density points due to changes in the movement and geometry of data acquisition sensors or vehicles (Nurunnabi, West and Belton, 2015a). Therefore, point cloud points have variable density and distribution of points.

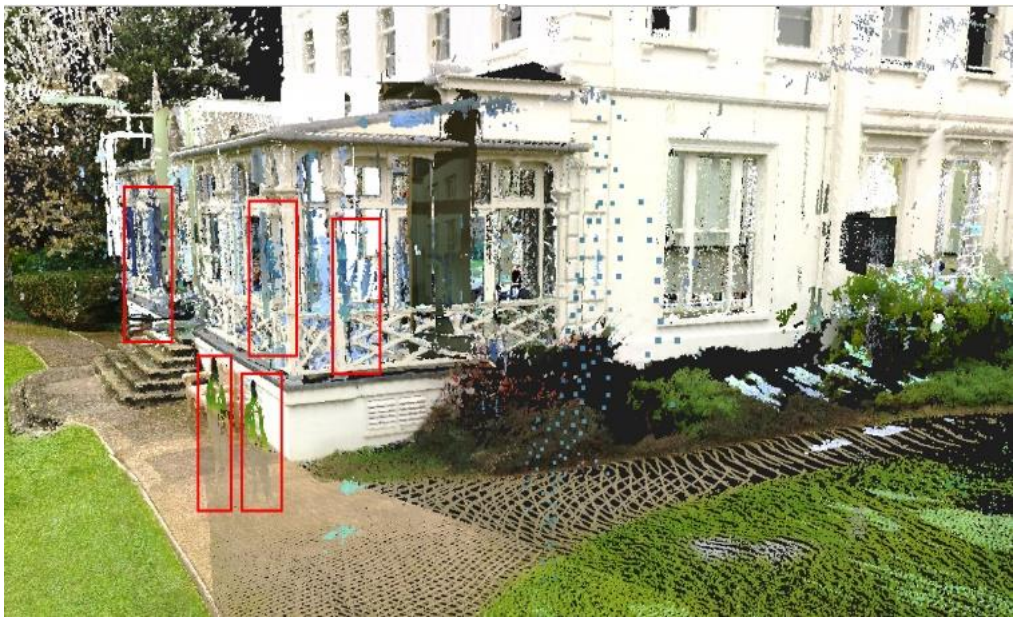
The density and distribution of points are denoted by a characteristic called 'scatter' or 'cluster'. If the points are scattered, i.e., low-density points, they are classified as outliers. Furthermore, if the points are in the cluster, i.e., high density, they are classified as noise. An example of low and high-density points is shown in Fig 4.8, where the objects away from the scanner have low-density points, and objects close have high-density points.

4.3.3.2 Point's Proximity

The next point categorisation is *Proximity*. Laser scanners often scan and capture a lot of data around the important features in geoinformation systems. These features are buildings, road features, trees, kerbs and other meaningful objects in the point cloud data. Therefore, it is important to differentiate the points that are closer to these features and objects of interest. The

proximity of the points is denoted by a characteristic called ‘isolated’ or ‘non-isolated’. If the points are closer to the objects of interest, they are classified as non-isolated points. Whereas if the points are far away from these objects of interest, they are classified as isolated points. A good example of point proximity is shown in Fig 4.8. All isolated points are considered outliers, and non-isolated points as noise.

(a)



(b)

Figure 4.8 Examples of (a) low-density and isolated points and (b) non-isolated high-density points

4.3.3.3 Point's Position

The final category for point classification is *Position.* The points have information on whether the distribution is high or low and isolated or non-isolated. The other important information about a point is the position of points related to other points in the point cloud. It is essential to determine the position to decide if the point is an outlier or a regular point. Therefore, the position of points is denoted by low, high and median.

A median is calculated to find all the points' centre of gravity. If the point is closer to the median, it also means that they will be non-isolated. Therefore, these points are considered noise. Whereas if the points are away from the median, low or high points are considered outliers. An example is shown in Fig 4.9.

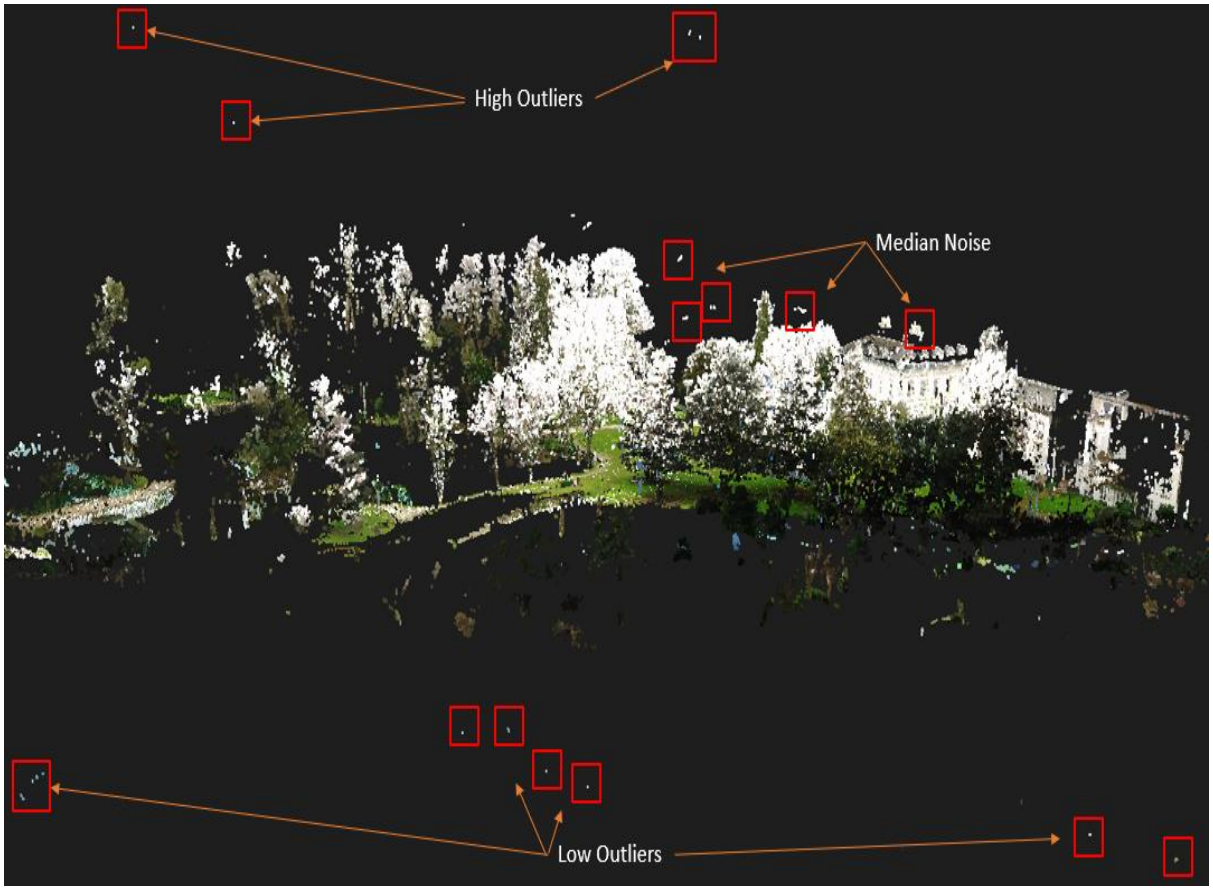


Figure 4.9 Points are categorised based on their position in the point cloud

4.3.4 Stage 2: Applications to Denoise and Remove Outliers

After identifying the outliers and noise, the next stage is removing them. The outliers and noise removal will help fast processing and more accurate results for meaningful feature detection. The human eye and observation skills are far beyond compared to any algorithms. Therefore, for removing the noise and filtering the outliers, three methods are proposed that remove them in real-time. The user manually handles the application to navigate the outliers and noise in 3D point clouds.

For filtering these outliers, Outlier Filtration using Octree Boxes (OF-OB) is used as it can capture the isolated and scattered points with their high or low position with respect to the whole point cloud. On the other hand, Noise is the group of points that are non-isolated, clustered and present closer to the median of the point cloud. Therefore, Noise Removal using a Sphere (NR-S) and Noise Removal using a 3D Box (NR-B) are used to remove the noise. NR-S and NR-B are user-controlled and can be efficiently used for non-isolated noise removal. The classification of the outliers and noise is shown in Fig 4.10 and the respective methods to remove them.

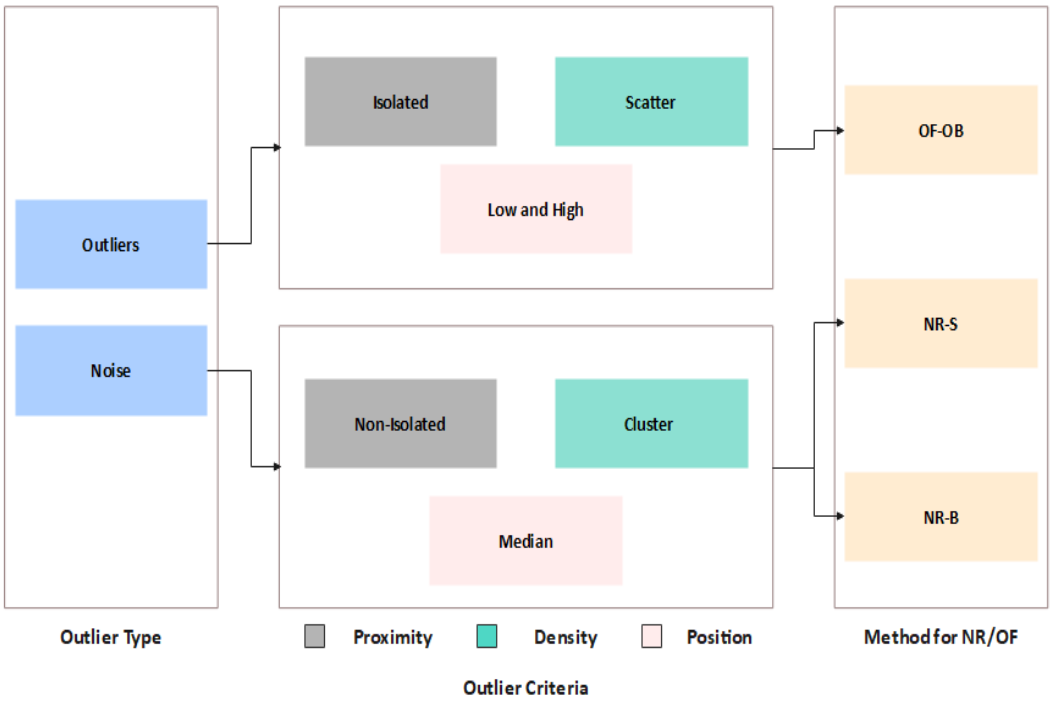
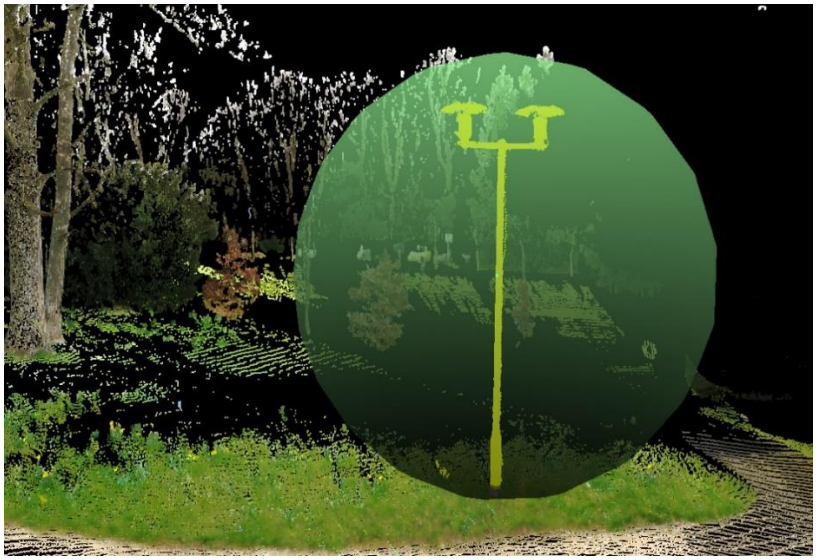


Figure 4.10 Overview of outliers/noise types and methods to remove them

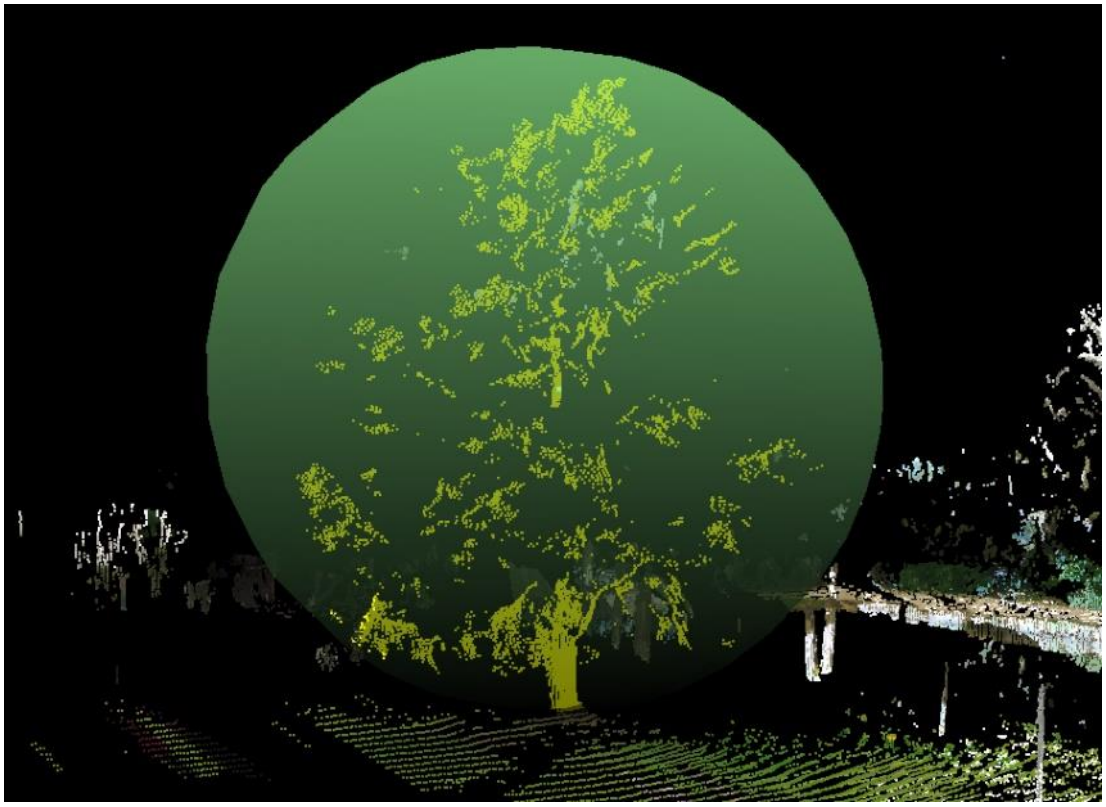
4.3.4.1 Noise Removal using Sphere (NR-S)

The sphere in the 3D point cloud is used in real-time to navigate and delete the points in the point cloud. The sphere can be set manually using the mouse cursor to any position. The points in the sphere are highlighted to display the points selected, as shown in Fig 4.11 (a). Therefore, removing the points that are closer to other objects is easier to grab using NR-S, an example shown in Fig 4.11 (b) and (c).

(a)



(b)



(c)

Figure 4.11 (a) A sphere representation in a point cloud (b) Example of lamp post captured by sphere to delete and (c) Example of a tree captured by sphere to delete

4.3.4.2 Noise Removal using 3D Box (NR-B)

The 3D Box is used to filter out the non-isolated clustered points. The box has six phases and can be manually set to the position of the points to be deleted. An example is shown in Fig 4.12 of a non-isolated car, a noise and non-feature that is not required to process the point cloud.

Conversely, the ground underneath the car is an important feature and can be used during the extraction of the terrain. Therefore, the NR-B is used to delete the cars/vehicles and save the ground points underneath them.

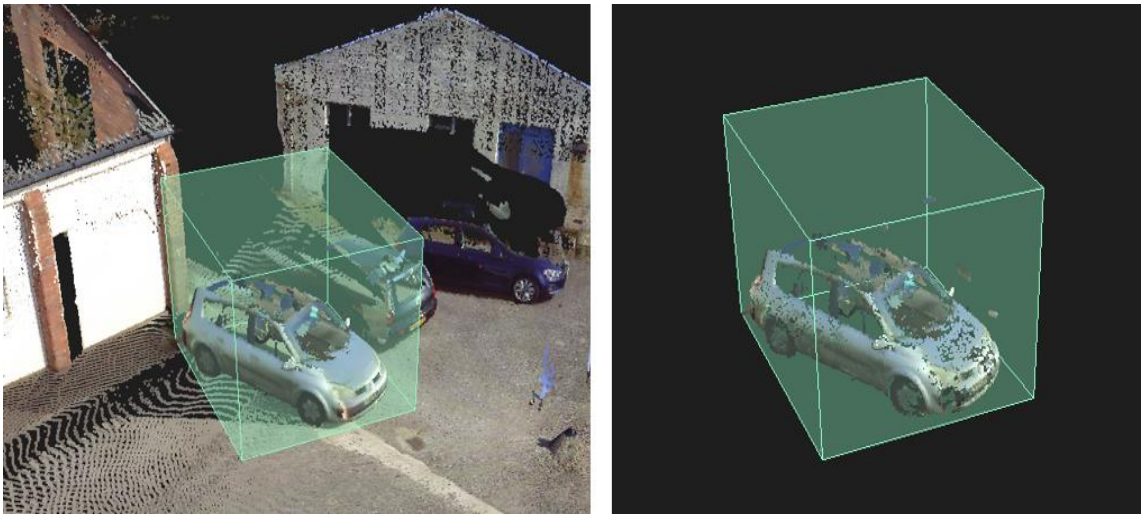


Figure 4.12 Noise removal by using a 3D box

4.3.4.3 Outlier Filtration using Octree Boxes (OF-OB)

The next application is OF-OB, which is used on outliers that are isolated, scattered and positioned high or low with respect to the whole point cloud. For outlier filtration, an octree is applied. An Octree is a tree/hierarchical structure. The hierarchical tree results in a recursive decomposition of a cubic region into eight equally sized octants, which are cubic regions (Boulic and Renault, 1991). These cubic regions are denoted as nodes. Each node has eight children except the end nodes, and the root of the octree refers to the entire volume. Octrees are generally used to partition a three-dimensional space by recursively dividing the nodes into eight child nodes and each of those eight nodes into eight other child nodes (Eder J, 1992).

An example is shown in Fig 4.13, where nodes are divided into child nodes until an empty node or maximum resolution node is denoted as a leaf node. The condition where the nodes do not further divide are 1) empty nodes shown by a cross and 2) full nodes shown by a circle on the nodes. Octrees are the 3D analogue of quadtrees (space is recursively subdivided into four sub-regions). The most common approach to managing octrees is pointer-based. The pointer-based octrees save the node's position by a pointer for each child node. Another approach is the linear octree which traverses and saves all the tree's nodes. The problem is remembering the pointer to access the right node, which could be tricky. Therefore, a naming convention is the better choice to be called as ZYX-convention. The binary number 1-8 describes the nodes (Eder J, 1992).

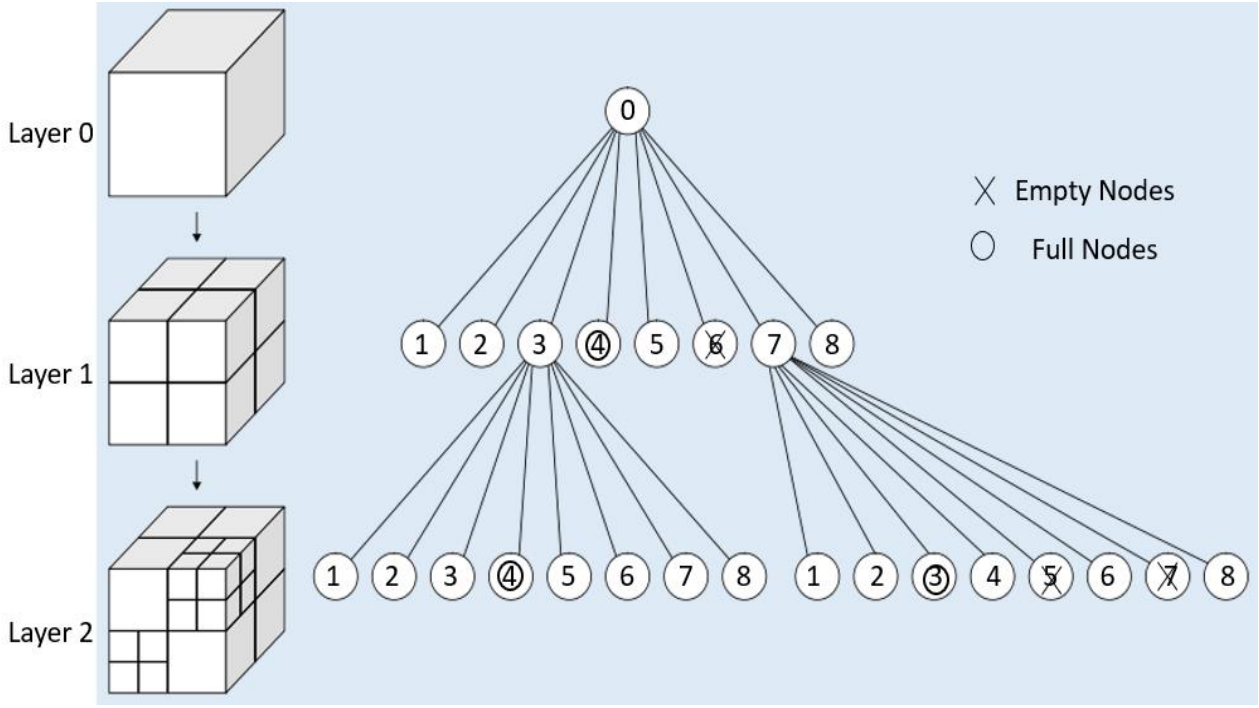


Figure 4.13 Octree structure with the eight octants at each layer

Many research studies included Octrees for various reasons in different fields. Examples include:

- representing spatial relationships of geometrical objects,
- 3D graphics for texturing objects (Lefebvre, Hornus and Neyret Fabrice, 2002),
- isosurface generation by passing information between octree neighbours (Wilhelms Jane and Gelder Allen, 2000),
- spatial indexing for accelerating isosurface extraction (Shi and Jaja, 2006; Schön *et al.*, 2013),
- compressing 3D data to generate volumetric 3D environment models keeping them compact (Hornung *et al.*, 2013),
- creating 3D mesh for 3D data generating polyhedral Delaunay meshes (Contreras and Hitschfeld-Kahler, 2014),
- the generation of Hexahedral element meshes (Schneiders, Schindler and Weiler, 1996; Zhang, Liang and Xu, 2013; Turner, Moxey and Peiró, 2015),
- nearest neighbour search methods for efficient searching within the radius (Sankaranarayanan, Samet and Varshney, 2007; Behley, Steinhage and Cremers, 2015),

- region-growing methods and segmentation in point clouds (Vo *et al.*, 2015),
- octree data structure where each node is a voxel and is used for ray tracing (Laine and Karras, 2010),
- collision detection for virtual surgery systems (Hu *et al.*, 2020),
- downsampling the point cloud for plane-fitting (El-Sayed *et al.*, 2018).

In this thesis, the outliers are filtered using the octree nodes. An example of outlier filtration is shown in Fig 4.14. The different coloured boxes represent different layers of the octree nodes. The bigger boxes are initial layers that are not further divided as the number of points in the boxes is less than the minimum threshold for octree octant nodes. The smaller boxes were further divided into layers of octant nodes.

The nodes' colour on each layer is based on rainbow colours, i.e., red, orange, yellow, green, blue and purple for layer $l = \{l_0, l_1, l_2, \dots l_n\ \}$. The user may control the number of outliers to delete them. For example, the scattered points present at low and high positions can be deleted using the approximate number to remove them.

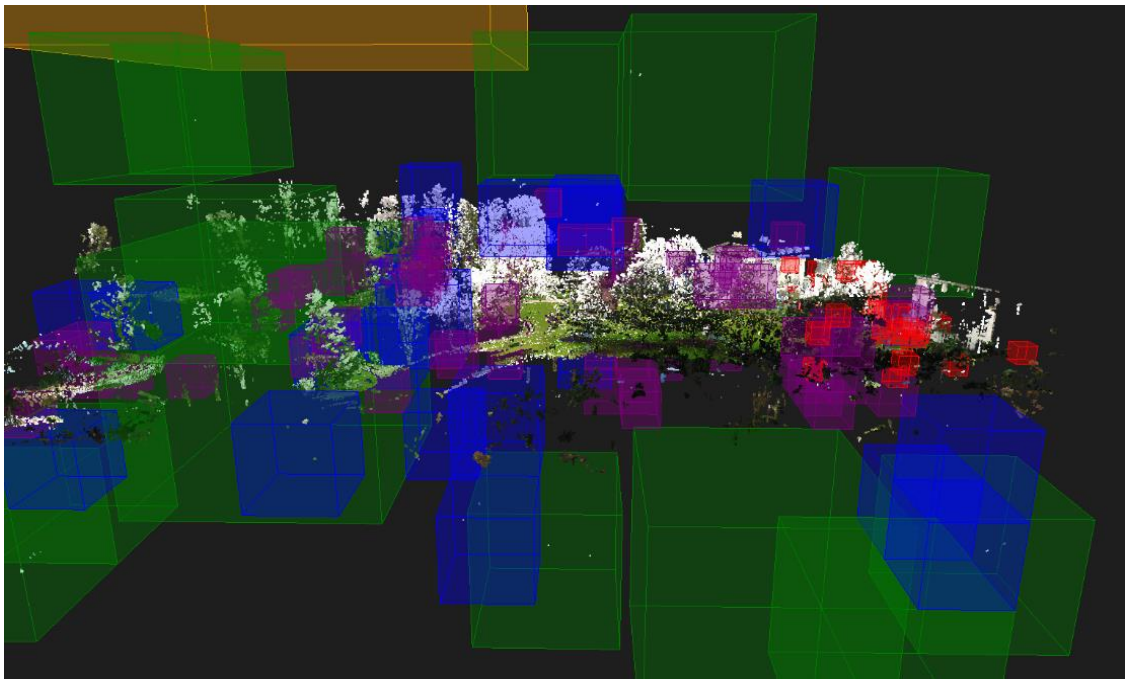


Figure 4.14 Octree boxes visually presented in a point cloud

4.4 Evaluation and Applications of Method on Commercial Software

This section evaluates and demonstrates the usage and functionality of the outlier filtration and noise removal methods. The methods are evaluated in terms of speed, accuracy and flexibility to be efficient on data sets collected by aerial, terrestrial and mobile laser scanners. The examples of datasets used are presented in Section 4.4.1. This section tests the three proposed methods using commercial software called 'LSS-3D Vision'. LSS is the commercial partner of this thesis, and all the proposed algorithms are implemented in it. The parameter and settings used in the software are presented in Section 4.4.2. Section 4.4.3 presents the comparative analysis of the proposed methods with existing methods. Section 4.4.4 discusses the results of the proposed methods.

(a) (b)

Figure 4.15 Datasets (a) Dorchester and (b) University of Gloucestershire

4.4.1 Datasets

The datasets used for the evaluation and validation are generated using terrestrial laser scanners or LiDar scanners. The FARO terrestrial laser scanner captures the point clouds; the model is FOCUS 350. The focus scanner has a range of up to 350 metres for long-range measurements, and the measurement speed is up to 976,000 points/second. This model has an integrated GPS

and GLONASS receiver, allowing for the absolute positioning of the scan location (Focus - FARO® Knowledge Base, 2016). The resolution of the scanner can also be changed.

Leica RTC360 3D is a LiDar laser scanner that captures point cloud data for up to 130 metres with 2 million points/second measurement speed. This model has multi-sensors GPS, compass, height sensor and dual-axis compensator (Leica RTC360 3D Laser Scanner | Leica Geosystems, 2018).

Data sets used for evaluation are (a) Dorchester data set and (b) University data set. The FARO laser scanner captured the University data set, and the Leica laser scanner captured the Dorchester data set. The scanned datasets consist of points in 3D (x, y, z) along with each point's R, G, and B and an intensity value.

The point cloud data set description is as follows:

i. **Dorchester data set** – is a scanned urban site captured by Leica. It has 161 million points.

ii. **University data set** – is scanned University of Gloucestershire's building in the park campus and has 580.94 million points.

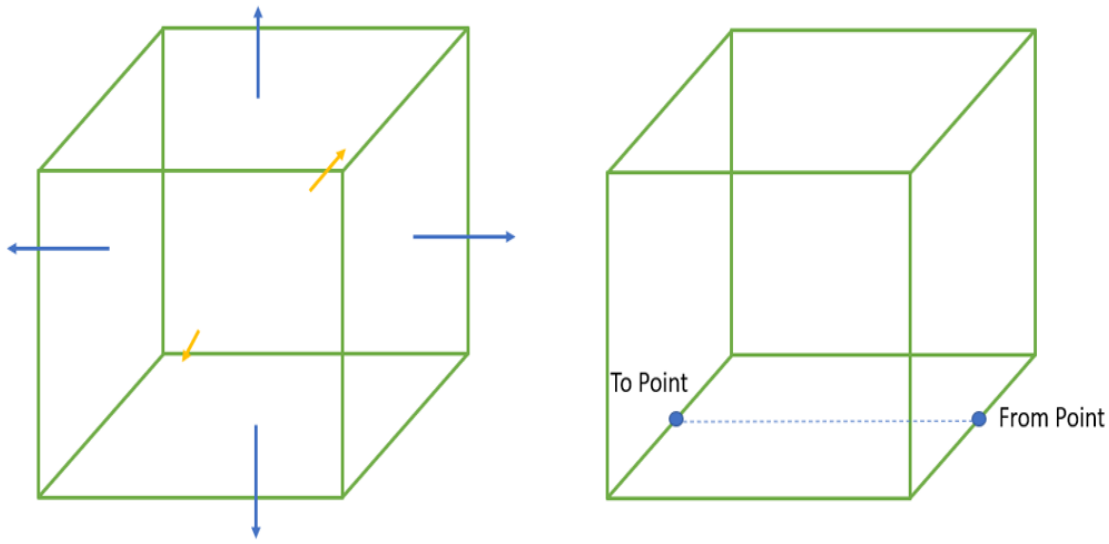
4.4.2 Parameter and Settings

This section demonstrates the parameter that the user can set to control the performance of the proposed filtering methods. All three methods have different default values and settings in the commercial environment.

These settings offer the flexibility of filtering the noise and outliers for the desired results in various point cloud data sets.

1. NR-S – The default setting for the sphere size is set to 1 metre. For most cases, the 1m sphere covers the maximum area for removing non-isolated noise and isolated outliers.

2. NR-B – For the control of the box, there are two options. The first option provides flexibility for the user to manipulate the size of the box by grabbing the box phase, as shown in Fig 4.16 (a). All six phases are resizable. For example, if the noise objects are on the ground's surface, such as a vehicle, the box bottom phase can be set just above the ground, saving the ground from deletion and getting rid of a vehicle. The second option is to set the box baseline points to form a box with a height and width. The height and width are always the same creating the selection box a cube by default. However, the box is not always a cube as it allows users to select two baseline points, i.e., the box can be cuboid as well. The 'To' and 'From' points can be set using X, Y, and Z coordinates to form a box, as shown in Fig 4.16 (b).



(a) (b)

Figure 4.16 In the point cloud, a box can be obtained using (a) its six phases or (b) selecting baseline points

3. OF-OB – The parameter for outlier filtration is controlled by searching the total number of points in the Octree box. The limitation on point numbers provides the user to either pick the outliers with even one point in a box or find isolated point clusters. The points that are clustered in the Octree box are listed with the number of points inside for quality check before the deletion of points, as shown in Fig 4.17.

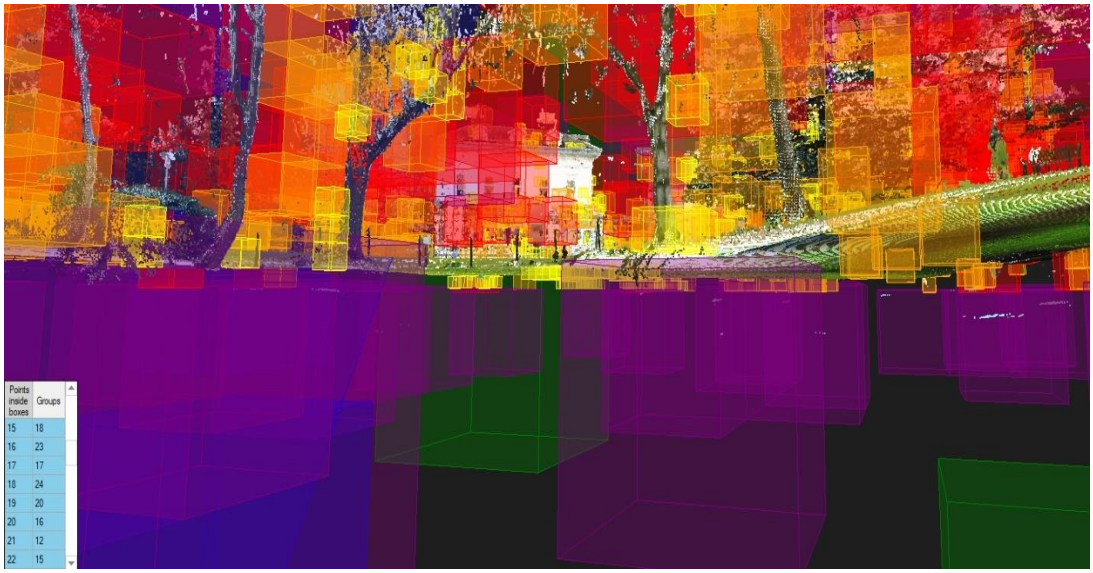


Figure 4.17 List of point groups and the number of points reported by Octree Box

4.4.3. Comparative Analysis

Many approaches and methods are available. Some of the issues with the currently existing methods are discussed in Section 4.2. A comparison of the proposed method to other methods is made in this section. In comparing methods, the common pattern observed is that the existing methods do not apply to all kinds of point cloud data. For example, the method by Li and Wei (2021) only considered UAV point clouds, Sotoodeh (2006), Zeybek (2021b) and Griffioen (2018) considered aerial laser scanner data, Nurunnabi, West and Belton (2015a) only considered mobile laser scanned data and Arvanitis *et al*. (2018) and Shen *et al.* (2011) considered LiDar point cloud data. On the other hand, the proposed method in this thesis can be implemented on any point cloud data, regardless of how they are captured, including UAV, LiDar, MLS, ALS and TLS.

The comparative analysis highlights that the existing methods also lack the definition of outlier or noise. Without defining them, providing solutions is not very helpful. The definition not only explains the context but also helps the user to implement the tools accordingly. This gives the user more control and flexibility to use the outlier filtration and noise removal on the range of different point cloud data.

The testing and evaluation of all three proposed methods (NR-S, NR-B and OF-OB) are performed on real clients' data which are typical of that collected by surveyors and engineers. The existing methods use synthetic data for verification, which does not prove whether the method would work on large or real-world data.

The advantage of the proposed method is that the tools used for outlier filtration and noise removal are adaptable according to the size of the object. For example, the sphere (NR-S) size can be changed to remove objects from moving people to trees, the box (NR-B) size can be used to remove from buildings to large areas, and for Octbox (OF-OB), the number of points inside the boxes can be a user-defined value to filter those points.

4.4.4. Results Analysis

The implementation to remove the noise and filter outliers by three proposed methods is tested. The methods are reliable for deleting outliers and noise points, isolated and non-isolated, and low or high-density points. NR-S method works best for the removal of both isolated and non-isolated outliers. The isolated noise with low or high point density near the object's surface that results from a reflection can be easily removed by NR-S. Furthermore, NR-S is also very handy for removing non-isolated noise. For example, a reflected surface that is very close to the ground. The intention is to remove the noise without removing the ground points, as shown in Fig 4.18.

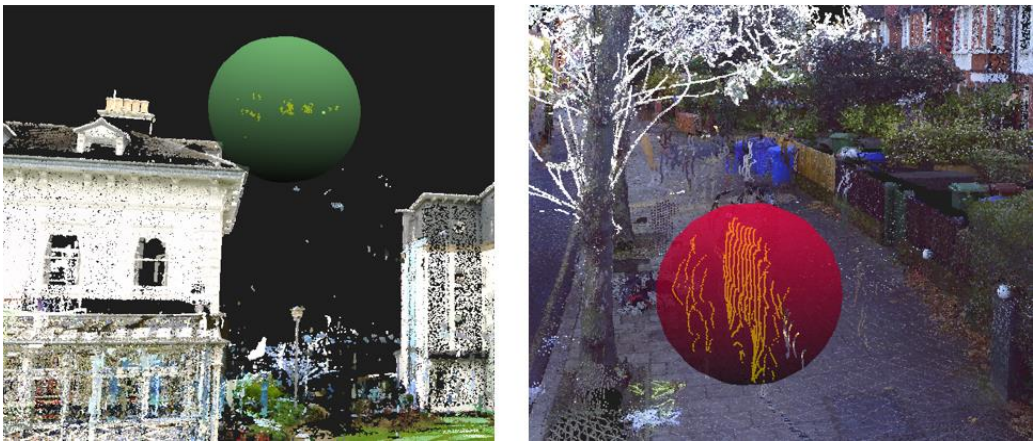


Figure 4.18 Isolated and non-isolated outliers and noise captured by NR-S to remove

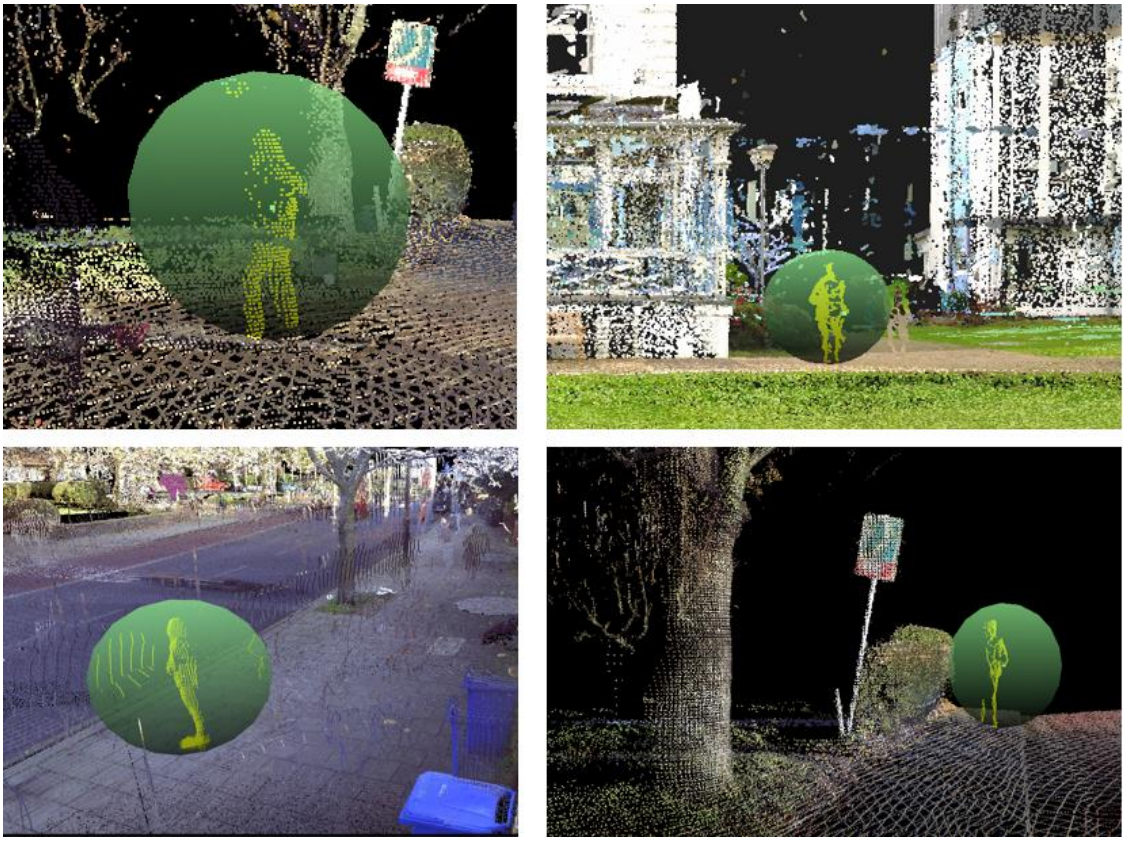


Figure 4.19 Examples of non-isolated noise (ghosts) successfully captured by NR-S to remove

The NR-S method is ideal to remove noise that is isolated such as buildings, trees etc. and non-isolated noise such as ghosts. The ghost is the term used in geoinformation systems for the moving or passing-by people that are captured by laser scanners. These ghosts are non-isolated noise close to important features like trees or buildings. Therefore, removing ghosts is essential. The example of ghosts is shown in Fig 4.19.

The NR-B method works best for the larger patch of isolated and non-isolated noise and outliers. The noise and outliers captured from an urban scanned point cloud are usually large compared to the non-urban sites. The reason is that when the data is captured, it does not exclude non-stationary objects. This noise and outliers can be the reflection from the window or mirrored surface, moving vehicles, ghosts, and weather conditions, which can capture the raindrops as noise.

The NR-B method is perfect for removing large noise and outliers. A step-by-step process of removing is demonstrated in Fig 4.20. Fig 4.20 (a) shows a patch of Dorchester data set inside

the box that displays the noise caused by moving people and vehicles. The next Fig 4.20 (b) shows the selection by the box above the ground that is noise. Finally, Fig 4.20 (c) shows that the noise is removed. Fig 4.20 shows that the NR-B isolates the non-isolated noise and removes it without deleting the ground points. Another example of such noise is shown in Fig 4.21, where a scanner captures the people waiting at the bus stop. In this case, the people/ghosts are the noise removed using the NR-B method.

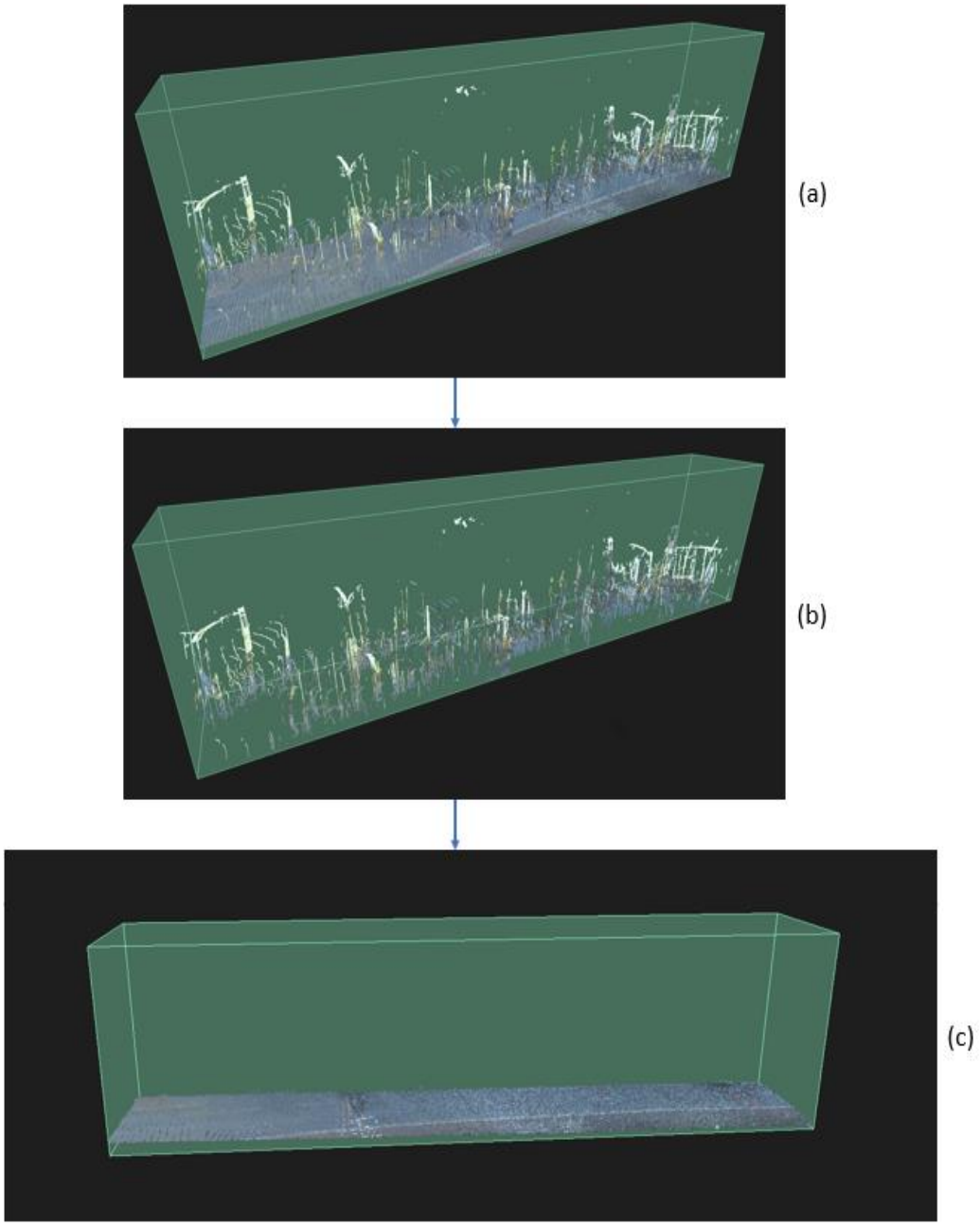


Figure 4.20 Noise removal by selection box on the busy street point cloud data

Figure 4.21 Noise example on the footpath as people at the bus stop are captured

The OF-OB method works best for isolated outliers, as shown in Fig 4.22. The example shows reflected points from the scanner positioned below the point cloud. OF-OB captures these scattered, isolated points efficiently. For flexibility, the user can define the number of points inside each cluster. The example in Fig 4.22 has the total number of points set to 50, i.e., each node in Octree that has points less than equal to 50 is detected and removed.

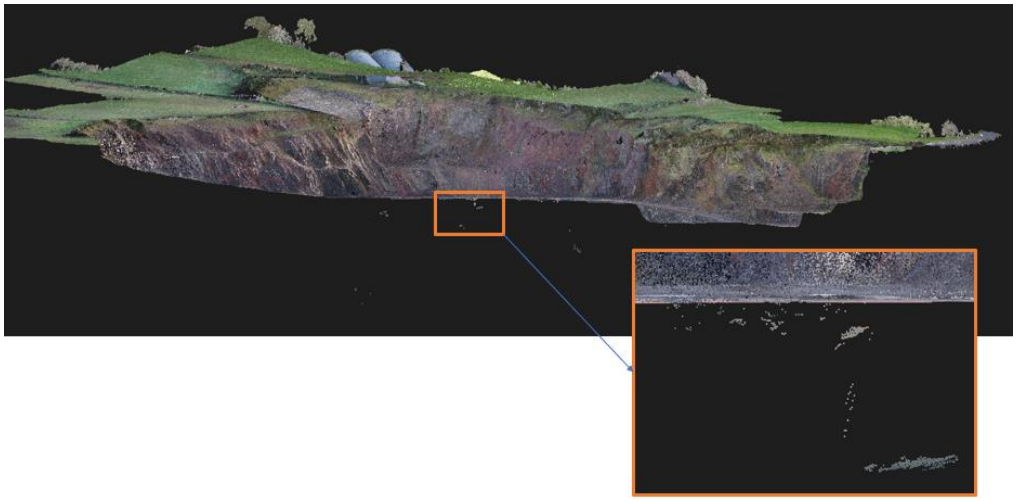


Figure 4.22 Isolated outliers example shown in an orange box

Before removing points, the octree boxes are represented to display nodes detected that have up to the equal number of points defined by the user, as shown in Fig 4.23. Once the user is happy with the visuals that demonstrate the capturing of the outliers, they are deleted.

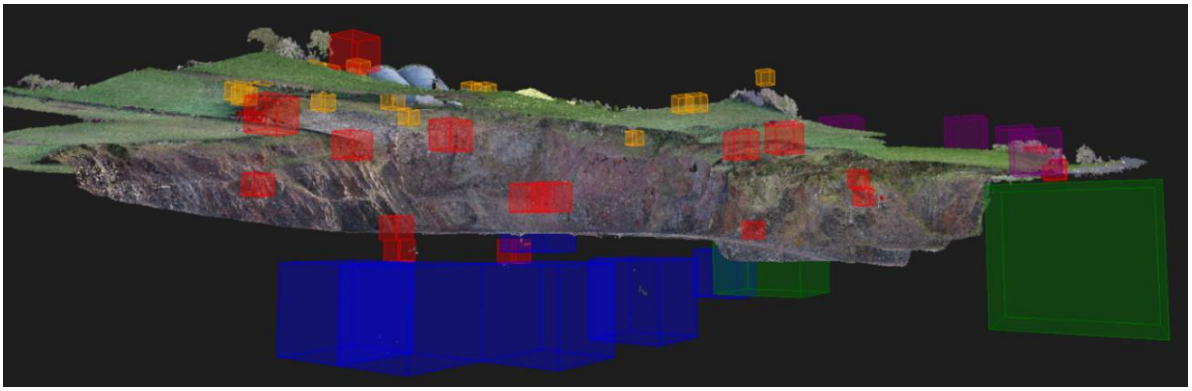


Figure 4.23 Captured and deleted outliers by OF-OB method

4.5 Chapter Summary

The comparison and analysis of existing methods to the proposed methods for point cloud filtration demonstrate that the proposed methods are capable of deleting all kinds of noise and outliers. Also, the proposed methods work on any point cloud without worrying about which laser scanner is used to collect it.

The main aim is to smooth the data by filtering and removing points to preserve the important feature details. The three different proposed methods, NR-S, NR-B and OF-OB, work efficiently for noise removal and outlier filtration of the point that is either isolated or non-isolated and has a high or low density. First, the outliers and noise are defined for easier classification and removal. Next, the points are classified based on point density, location and proximity. This provides the user with the clarity to understand the point's properties and use the appropriate method to remove them. The number of points deleted (using NR-S, NR-B and OF-OB) is dependent on the diameter of the sphere and the size of the boxes selected by the user.

There could be various possibilities to extend current methods that could not be implemented due to the time restriction to finish the thesis. However, the proposed methods are part of commercial software and will grow exponentially. For example, the search sphere can be automated to delete more points in one go. In addition, the Octbox to filter points can be implemented with PCA to fit the plane inside the boxes and analyse the points inside in a more sophisticated manner.

Chapter 5 A New PCA-Based Method for Edge and Edge Stream Detection

5.1 Introduction

The detection and modelling of the features from the urban point cloud are of great interest in the surveying and engineering industries. The important features in a typical urban point cloud data are walls, roofs, marker poles, lamp posts and kerbs, windows, buildings, trees, and vehicles. Apart from these features, edges and boundaries of the features are considered the basic requirements in urban scenes. For example, a surveyor will be interested in the edges of buildings, windows and doors, edges of the top and bottom of the kerb, edges of footpaths and many more. Bazazian and Parés (2021 p.1) defined edges as

> "*Edges in 3D point clouds are considered as remarkably meaningful features due to their capability of representing the topological shape of a set of points.*"

As 3D point clouds have become popular, edge and boundary detection has become an essential research topic. Edges and boundaries are important geometrical features in an urban scene, including artificial objects (Nie, 2016). Therefore, the prominent edge detection method is expected to accurately extract the edges with correct alignment. Moreover, the challenges continue as the edge detection methods are not properly evaluated to prove that they work on large point cloud data. In addition, limited studies and research are available on 3D edge extraction from point clouds as 3D data is more challenging. Most of the research is based on 2D image edge extraction, such as Robert edge detection, Sobel edge detection, Prewitt edge detection, Kirsh edge detection, Robinson edge detection, Marr-Hildreth edge detection, LoG edge detection and Canny edge detection (Mahmood, 2017).

To address the abovementioned problems, a new robust PCA-based 3D method is proposed to detect an edge that is then extended and automated to detect boundaries. First, the analysis and evaluation of the existing methods to extract the edge in remote sensing, photogrammetry, and geoinformation are presented and reviewed in Section 5.2. Then, a new proposed PCA-based

method is presented in Section 5.3. Next, the new method's evaluation in terms of low-cost computation, high accuracy, and efficiency is discussed in Section 5.4. The validation and application of the new method in a commercial software environment are presented in Section 5.5. Also, Section 5.5 states the problems like shadows, gaps and missing data using the proposed method on different point cloud data. Finally, the findings are concluded in Section 5.6.

5.2 Evaluation and Analysis of Existing Methods

5.2.1 Edge Detection

One of the problems in reverse engineering is that it is extremely difficult, if not impossible, to detect sub-regions robustly and automatically in the tessellated model with low computational efforts. Galantucci and Percoco (2005) proposed a multilevel automatic algorithm for edge detection in polygonised point clouds to overcome the problem. The aim is to obtain a Solid-to-Layer (STL) model. The algorithm uses the tessellated point cloud to attain the STL model by implementing a heuristic problem-solving technique to detect possible edge features in a given point cloud. Then, STL is divided into two models with different levels of detail (LOD). Points belonging to both models are ordered for recording the relation between points in the triangle connection. For each point around the low LOD, all the surrounding points are detected for both models, but there is no defined relationship between the original and subdivided models. After clustering the points, edge detection is achieved on a low LOD model. The contour evolves after finding the edge points at each iteration, and as the points are detected, they are aggregated into a growing contour. Finally, Gaussian curvature at the contour point is calculated. When the algorithm is finished, there are two sets of results, the first is the edge points, and the second is the connection between them (Galantucci and Percoco, 2005). The problem with the algorithm is that it assumes that each edge point has a neighbouring point that belongs to an edge. Hence the formulated contour cannot always produce accurate results.

Park and Jun (2002) proposed an artificial neural network to identify the tessellated point cloud features after an edge detection phase. The methodology first generates triangle meshes to identify boundaries. Then, this boundary meshes are connected in a loop segmented into sub-

regions. Finally, the sub-regions are merged into a single feature using an artificial neural network. The scanned points on feature-based reverse engineering systems reconstruct standard mechanical engineering products. Another approach uses the data from a coordinate measuring machine (CMM) to slice it in three orthogonal directions. A 2D NURBS spline is fitted on each slicing plane to calculate maximum curvature points. The points detected represent the object's edge (Chen and Liu, 1997). However, the results conflict with whether they belong to an edge or a boundary as there are no clear definitions of an edge or boundary presented. Also, a robust algorithm is required to check if edge points belong to an edge or not.

Existing research considers statistical and geometrical methods for estimating the typical edges to detect the edge's sharpness. The main challenge is to estimate the normal on the detected edge feature points as the points are highly dependent on the neighbourhood employed for edge detection. For example, the neighbourhood might be surrounded by points that belong to a different edge surface. Hence, giving incorrect points belonging to an edge (Bazazian, Casas and Ruiz-Hidalgo, 2015). In (Weber, Hahmann and Hagen, 2010b, 2010a; Weber *et al*. 2012), normal estimation is achieved using Delaunay triangulation, a technique for converting point clouds into 3D surfaces by building a network of triangles over existing vertices of the point cloud. The challenge of this technique is that it is susceptible to the points located around the edges. Additionally, this technique's computational process is costly and difficult to implement in real-world large point cloud data applications.

Among various methods for extracting an edge's sharpness, robust statistics are used by Fleischman, Cohen-Or and Silva (2005). The method applies a statistical method that segments neighbouring points into regions on the same surface. The neighbouring points are computed using moving least squares (MLS). As the method tries to fit a model in the data which may contain outliers, it uses a forward search method to identify the masked outliers. The masked outliers are those points that usually cannot be identified from the statistics of the model, which is fitted in the entire model set. These masked outliers can be a single point that affects the least squares calculation to unwanted results, therefore are very hard to detect. The forward search algorithm starts with an outlier-free small subset, and then to proceed through data, one sample is added in each iteration to refine the model. The initial model is computed using the Least Median of Squares (LMS) algorithm with a small K value where K is the randomly selected sample points. Forward search can monitor multiple parameters to differentiate influential points from outliers. Typically, forward search adds good samples first until all are exhausted

when outliers are added. The residual plot is used here to identify outliers. As the search proceeds, the good sample sets are added first, followed by the outliers. The residual plot is monitored to identify the residual threshold level. As soon as the outliers enter the sample set, this is visible in the residual plot, where the residual of the outliers decreases and the residual of good samples increases. Thus, as the residual reaches the threshold level, the samples are considered outliers after that level.

Iterative refitting is applied to the data set S. The next step is to remove the samples that are fitted $S = S/S_1$. This process is repeated until S is empty. The iterative refitting algorithm finally captures and identifies edges in the noisy data. The algorithm's limitation is that its forward search uses K value as the sample set of points (Fleischman, Cohen-Or and Silva, 2005). If the value of K is small, the algorithm works, and if the value of K is large, iterations are needed. However, for large data sets, the algorithm becomes sensitive to noise. As noisy data have ambiguity between the smooth region and sharp features or if the sample density signal-to-noise ratio is too low, the algorithm may classify the smooth region containing the sharp feature as an edge. Additionally, the position of the reconstructed edge will not be reliable if the two sides of the edge incline towards being collinear.

Extending the Fleischman *et al*. technique, Daniels *et al*. (2008) extracted the curves of the features on the reconstructed point sampled surface. A robust method that extracts shape edges in the model produces a set of connected and smoothed polylines identified as sharp features. The algorithm uses a robust moving least-squares (RMLS) framework to approximate the neighbouring surfaces. It also uses kernel regression to extend MLS further. An unorganised set of points computes S by moving least squares (MLS) defined by the projection operator. From these, points P near the potential features are extracted. This algorithm applies uniform noise by shifting a point at a random vector length. Robust MLS is used to fit multiple surfaces to neighbourhood points and project each point onto its adjacent intersection between two surfaces. The division of points produces a primary set of polylines that are then reconstructed to fill the feature gaps (Daniels *et al*., 2008). The technique's benefit is demonstrated in three applications. These applications include surface segmentation, surface reconstruction and shape compression. The surface segmentation process defines multiple surface regions, which are feature aligned by comparing Euclidean distance between the neighbours. Depending on a feature polyline's specified distance, the algorithm flags all points as boundaries or unvisited

points. The algorithm visits unvisited points and merges boundary points into the nearest point group. After segmentation, surface reconstruction is carried out by producing a high-quality mesh.

The algorithm uses MLS to produce models with a smooth sharp mesh. In addition, surface compression is carried out for fast transmission and efficient storage (Daniels *et al*., 2008). The limitations are that the algorithm uses MLS and RMLS; the calculation is considerably time-consuming. Like Fleishman *et al*. and Daniel *et al*. techniques, Oztireli, Guennebaud and Gross (2009) applied an algorithm based on robust MLS called novel MLS. Novel MLS includes built-in methods for handling outliers and high-frequency features, controlling the sharpness of the feature, spare sampling more frequently, and implementing novel MLS is easy and efficient as it is pure computation without any processing. Furthermore, the MLS surface is stated in local kernel regression, implemented by implicit moving least-squares (IMLS)—combining these two results in novel robust IMLS.

Surface Segmentation and in-line segmentation are also used to extract edges in point clouds. Demarsin *et al*. (2007) used segmentation to find sharp features using a graph approach with a minimum spanning tree. The method delivers a set of candidate points that represents feature lines. These feature lines are used to create a closed curve network. This algorithm applies the region growing method with normal estimation to cluster the points, reducing point cloud size. The method constructs a connected graph, where vertex and edges connecting 2 clusters are formed. Since the connected graph may contain smaller clusters with unwanted gaps, each cluster's size is considered. The method adds the edge of neighbouring small clusters, which results in an extended graph. Edges of small clusters could identify the sharp feature line's location; therefore, a minimum spanning tree is constructed. The weights of edges are calculated between small clusters. The weight results in a graph with reduced edges of a larger cluster. This graph is called the pruned graph. The graph may contain many short undesirable branches that must be removed since they do not correspond to actual features. The unwanted branches are removed by comparing with a threshold parameter resulting in a graph with exactly one incident edge. The algorithm, therefore, uses each endpoint with an appropriate point to reconstruct closed lines. As a final step, smoothing is performed to detect a smooth graph with sharp features (Demarsin *et al*., 2007).

Xu *et al*. (2015) proposed surface segmentation and edge detection on heritage fractured fragments by merging faces based on normal vectors. Xu *et al*. method primarily focuses on surface segmentation and edge detection from geometrical features (also fragmented surfaces), and then face and edge characters are merged into fractured surfaces. The method segments a surface to extract the edge feature lines on the triangular meshes. Firstly, the Laplace operator reduces noise by implementing a clustering algorithm based on vertex normal vector to find a rough surface segmentation. Secondly, an integral invariant is introduced to calculate surface roughness. The local bending energy function is defined by the ratio of vector differences and distance between the vertices, which demonstrate local roughness. The original and fractured surface points are differentiated using threshold values based on roughness. Then, an accurate surface segmentation is applied by merging faces as per face normal and roughness. The process is iterative and continues until the algorithm converges. Finally, edge feature lines are attained based on the segmentation of the surface (Xu *et al.*, 2015). The limitation of the method is that it only focuses on cultural relic fragments. The method depends on sampling normal vectors for clustering, assuming the point cloud data does not have any data gaps or overlapping geometry features, which will produce wrong clustering results. Also, there are no explanations of the set threshold values, which could be different for various point cloud data.

Lin *et al*. (2015) proposed a method that can be used to extract plane intersection line segments from unfiltered extensive point cloud data. This approach extracts a point set from a straight linear structure in 3D. Then, the 3D line support region and the line segment half-planes (LSHP) are merged, providing a geometrical shape, and making the line segment more correct and consistent. Next, the corresponding 2D line regions have been used to determine the projection direction of the 3D region; after that, V shape extraction from the projected point sets. The V shape consists of points that share a common endpoint. In order to extract the V shape, dynamic programming is performed. Once the V shape is determined, the point set is divided into two groups, and the 3D line segment is divided into two groups. Then, the two groups are fitted separately by two planes. Finally, the region growing method is applied to find the boundary of 3D line regions after constructing the LSHP structure. The method extracts the line segments from urban point clouds but cannot achieve high performance as this method fails to identify the boundary of 3D planes, the small region planes are difficult to detect, and if the data is complex with vegetation, it becomes difficult to fit planes (Lin *et al*. 2015).

5.2.2 Region Growing Method For Edge Detection

As mentioned in Section 5.2.1, the *region growing* algorithm is proposed by many authors. Region Growing is a method that segments point clouds into clusters and classifies regions with sharp edges by analysing the normals of the points. Gumhold, Wang and MacLeod (2001) used the Riemannian tree to build the information and then analyse the neighbourhood with PCA. A Riemannian graph contains data points to k nearest neighbours. The advantage of the Riemannian graph is that it can handle noisy data, and computation time is comparatively less.

The method directly extracts feature lines from the point cloud. The approach has two phases: (1) uses the points on the edge of the neighbour graph to assign penalty weights that are likely to be part of the feature. A feature pattern set is followed by extracting a subgraph that decreases the edge penalty weights. A Neighbour graph identifies points that are probably nearest to each point or close to the underlying surface and connects them to form a graph. The graph helps faster local computation and acts as a domain to identify feature patterns. After the neighbour graph is generated, k neighbours of each data point are analysed and classified into surface points, potential border points and potential crease points. Potential border points and potential crease points are assigned a penalty function, which helps measure the likelihood of data on a border or a crease.

(2) The next phase is to extract feature line patterns using the penalty function computed in the previous stage. Given penalty functions, the minimum spanning pattern on the neighbour graph subsets is calculated, resulting in cycles that contain more edges than a user-defined constant. Thus, this spanning pattern contains multiple short branches along with the edge. These short branches are removed from the minimum spanning pattern. In the feature recovery stage, the neighbourhood grouping is finished to recover the corner locations. Crease lines near the junctions are reconstructed according to the grouping near the junctions. Delaunay filtering is used to construct surface models.

The algorithm works with round edges, but the edges are undetected if implemented to singleton points (for example, peak points on a cow's horn) or sharp points as meshing is complicated with polygon faces (Gumhold, Wang and MacLeod, 2001).

Weber, Hahmann and Hagen (2010b) proposed a method to detect particularly sharp or point-sampled features in geometry. The method proposes to compute Gauss map clustering on all the neighbouring points to find which points lie on the sharp edge features. Further, jump edge filtering is used to reduce noise from the point cloud. The method starts with implementing the kd-tree for performance purposes by searching the k-nearest neighbour of the points. Afterwards, the triangles are formed by a point and its two nearest neighbours. Feature detection is performed by analysing the clustering of the normals on the Gauss Map. Since different features will result in different clustering of normals, noise in the point cloud can be easily identified as normals of the triangles formed with noise points. These noise points will not be part of another normal cluster and thus can be ignored. The cluster formation and direction denote whether the feature is sharp or curved. If there are two or more distinguishable clusters, it signifies a pointed feature and in case of no clustering and distribution of points on the Gaussian map signifies a curved or smooth feature. The method merges separate clusters into larger clusters by calculating the distance between the clusters, and the process will stop once the distance crosses a certain threshold. The result is the point cloud with sharp marked-up features. The advantage of the algorithm is that it implements adaptive local parameters for different regions of point clouds. The disadvantage is that only line types and corners are detected. Also, the test dataset was minimal to accurately estimate the time consumption on the large point cloud dataset.

Feng, Taguchi and Kamat (2014) proposed an algorithm to detect multiple planes in the scene in real-time. The algorithm starts by dividing the point cloud into non-overlapping point groups and forming a graph whose node and edge represent a group of points. Non-overlapping points give the algorithm the advantage of not detecting the boundaries of the planes. This graph is subjected to agglomerative hierarchical clustering (AHC) to merge nodes belonging to a similar plane until a point is reached when plane fitting mean squared error (MSE) is beyond a threshold. AHC extracts the coarse planes by removing nodes with a high mean square error, nodes with missing data, nodes containing depth discontinuities and nodes that lie at the boundary between two planes. The algorithm saves total computational time by not estimating the normal of each point like other algorithms. In AHC, initially, a node with minimum MSE is identified, and then the neighbouring node is searched, resulting in minimum plane fitting MSE. If the minimum merging MSE exceeds a threshold, the plane segment is identified and extracted from the graph. The extracted planes are further refined using pixel-wise region

growing. The extracted planes on point clouds are impressive. However, the algorithm is tested on limited data and node size selection is unclear.

5.2.3 Edge Detection in Other Fields

Since the scanners collect data in 3D, the processing is performed in 3D Datasets. However, Lin *et al*. (2017) stated that

> "*Sometimes to detect the corresponding points between two shapes, of an object efficiently the 3D shapes are converted into 2D images.*"

The transformation is fast and efficient. Unlike point clouds with 3D points, an image only has 2D coordinate points. Ando (2000) proposed an algorithm to extract the edges, corners, vertices and ridges in the 2D image, corresponding to 1D and 2D gaps in the intensity surface. In order to extract edges, the algorithm is divided into two nonparametric stages. The first stage is the image field categorisation based on the gradient distribution in order of its dimensionality. The image field categorisation is based on two operators: uni-directionally varying region (Univar) and Omni-directionally varying region (Omnivar).

As the Omnivar also includes uni-directionally, the second stage is dedicated to the detailed analysis of the informative axis of Univar and Omnivar to find edges, corners, vertex and ridges. The analysis includes the Gaussian curvature of the correlation function for texture analysis and the covariance matrix of the gradient vector, followed by the canonical correlation analysis. The analysis leads to an algorithm further classifying *Edge/Ridge* localisation by gradient projection and *Corner/Vertex* localisation by angular gradient projection. The algorithm assumes that the integration area is small, and the image is continuously differentiable. The presence of gaps in the image leads to imprecise results (Ando, 2000).

Lee, Koo and Jeong (2006) proposed another algorithm that separates the row and column edges from an edge image using primitive shapes. The aim is to extract straight lines from images. The edge images consist of various line segments such as curves, lines or arcs, categorised in rows and columns in digital coordinates. The detected edges in rows and columns are then labelled. Some edge pixels are labelled as both row and column. Therefore,

there are four types of labels: row, column, both, and single. The 8-neighbour connectivity is applied to extract the basic types of the straight line.

On each label, PCA is applied to extract eigenvectors and eigenvalues. The eigenvector helps determine the direction of the pixel distribution of a line, and the eigenvalue helps determine the distribution time. The lines extracted are then tested for their straightness and angle. To take into account noise effects, a threshold is determined based on the small eigenvalues of the lines. The absolute threshold results in a small number of accurate straight lines, and the relative threshold results in more lines with inaccuracy (Lee, Koo and Jeong, 2006). The pros of the algorithm are that it adjusts the threshold to reduce the noise effects, and the con is that it depends on the threshold value to find precise straight lines. However, the edge labelling and composition of the line take a lot of processing time. Also, the labelling could be ambiguous if the angle difference is negligible between two straight lines.

5.2.4 Principal Component Analysis and Extensions

5.2.4.1 PCA

Principal component analysis (PCA) is a mathematical, statistical method that converts a set of observations of a larger number of correlated variables (Kabacoff, 2019) into a smaller number of linearly uncorrelated variables using orthogonal transformations (Suryanarayan and Mistry, 2016).

In linear algebra, an orthogonal transformation preserves the length of vectors and angles between vectors. The set of uncorrelated variables is known as principal components. Principal components are always less than the original number of observations.

The first principal component is defined on the largest possible variance, and then the succeeding components with the largest variance under the constraint that it is orthogonal to the preceding components. The resultant vector direction is an uncorrelated orthogonal set.

Table 5. 1 PCA definition in various fields

Year	Name	PCA relation	Field
1901	Pearson, 1901	Pearson first introduced PCA as a fitting linear subspace method to multivariate data by minimising chi distance.	Mechanics
1933	Hotelling, 1933	Independently developed a method that explains the concept of uncorrelated linear combinations of original measurements by decreasing each variation	Mechanical Engineering
1983, **1993**	Golub and Van, 1983; Stewart, 1993	Singular Value Decomposition (SVD) is a factorisation of a matrix that generalises the eigen decomposition of the normal square matrix to any matrix	Mathematics
1996	Golub, Hoffman and Stewart, 1987	PCA is the Eigenvalue Decomposition (EVD) of X^TX	Linear Algebra
1997	Nievergelt, 1997	Schmidt–Mirsky theorem to identify the nearest matrix whose singular value	Psychometrics
2000	Chatterjee, 2000	PCA, known as Proper Orthogonal Decomposition (POD), is an elegant method of data analysis to obtain low-dimensional approximate descriptions of high-dimensional processes.	Mechanical Engineering
2009	Monahan *et al*., 2009	Empirical Orthogonal Functions analysis decomposes a data set in orthogonal basis functions.	Geophysics, Signal processing
2001	Dony, 2001	PCA, known as Karhunen-Loève Transform (KLT), is a linear combination of orthogonal functions. KLT is used for compressing and summarising information	Image Processing
2013	Gie Yong and Pearce, 2013	A related statistical method in machine learning and data mining is known as Factor Analysis	Machine learning and data mining
2014	Kadam, 2014	A method for face recognition using PCA and discrete cosine transform	Face Recognition

PCA is adaptable in various science disciplines for multivariate analysis, as shown in Table 5.1. It has been used in biology, chemistry, demography, agriculture, oceanography, psychology, quality control, genetics, geology, ecology and food research. PCA has also been used in economics and finance to study stock market changes (Suryanarayana and Mistry, 2016). In computer science, PCA is used as a tool for data analysis and for making predictive models. PCA can be calculated in two ways:

1) depending on semi-definite matrices, eigenvalue decomposition of the data covariance matrix and
2) depending on rectangular matrices, the Singular value decomposition of the data matrix by normalising or using Z scores of each data attribute (Abdi and Williams, 2010).

In terms of handling the variables while calculating PCA, it can be generalised as correspondence analysis (CA) in order to handle qualitative variables and as multiple factor analysis (MFA) in heterogeneous sets of variables (Abdi and Williams, 2010). As a result of the method, PCA produces component scores called factor scores (the transformed variable values corresponding to a particular data point) and loadings (the weight by which each standardised original variable should be multiplied to get the component score) (Shaw, 2003).

PCA's fundamental concept is to reduce the data set's dimensions, which have many uncorrelated variables. From the geometric viewpoint, the computation of PCA is to minimise the variance. Its operations reveal the internal structure of data variance. If the original uncorrelated data set is imagined as a set of coordinates in high-definition data space, PCA can be used for a low-dimensional view, a projection of the objects when viewed from the most explanatory viewpoint. To achieve the projection by using only the first few principal components to reduce the dimensionality of the transformed data.

Like PCA, Factor analysis aims to reduce the dimensionality of the data set, but the approaches to achieve is entirely different from PCA. Factor analysis searches for joint variations in response to unobserved latent variables (the variable that is not directly observed but inferred from other measured variables). The information gained about the latent variables is used to reduce the variables in the data set. It incorporates more domain-specific assumptions about the underlying structure (Young and Pearce, 2013). As PCA defines a new orthogonal

coordinate system that optimally describes the variance in a single dataset similarly, canonical correlation analysis (CCA) defines coordinate systems that optimally describe the cross-covariance between two datasets (Barnett and Preisendorfer, 1987).

5.2.4.2 Principal Component Analysis Extensions

PCA with a wide variety of extensions have been proposed; this section discusses some of them. Feature extraction using PCA has been used in various applications such as pattern recognition, noise detection, and image indexing. Scholkop, Smola and Muller (2012) introduced a new method called Kernel PCA (KPCA)**.** KPCA is an extension of PCA to handle the data points in nonlinear space and better understand the extracted nonlinear features. KPCA uses the kernel method, which can be applied to any algorithms formulated for the *dot product*. The utility of KPCA is pattern recognition using a linear classifier. Nonlinear data is fed into high dimensional space, which allows it to behave linearly, and therefore, nonlinear mapping never happens. The use of a kernel means that all the points are represented using the distance calculated to all other points to form a kernel matrix. To this matrix, eigenvalue decomposition (EVD) has been applied. Since the kernel components are not linear, the limitation of KPCA is that it will not result in principal components by itself, but the data projection will be on those components (Scholkop, Smola and Muller, 2012; Oreifej, 2013). The advantages of KPCA are that nonlinear components result in better recognition rates and the possibility of using more components to increase performance with minimal computational cost.

The traditional approach to PCA calculation lacks the probabilistic model for the observed data set. So, Lawley (1953) and Anderson & Rubin (1956) investigated PCA from a probabilistic point of view, and their research was later extended to Probabilistic PCA by Tipping (Tipping and Bishop, 1999). They demonstrated

"*How principal component analysis may be viewed as a maximum-likelihood procedure based on probability density model of observed data*" that enables the comparison with other probabilistic algorithms.

The Gaussian noise model is applied with eigenvalues and eigenvectors of the sample covariance matrix. In addition, the Expectation Maximisation algorithm (EM) is used for

finding the principal axis by iteratively maximising the likelihood function. Therefore, EM makes Probabilistic PCA more efficient with larger data dimensionalities. Furthermore, probabilistic PCA demonstrates the capacity to handle the data with missing values since it is a generative model. The applications of the algorithm are the visualisation of data and image compression (Tipping and Bishop, 1999).

Many applications, such as computer vision and image processing, have problems with subspace segmentation. To overcome the problem, PCA has been extended to generalised PCA (GPCA). GPCA is the algebraic-geometric approach for subspace segmentation of the data points (Vidal, Ma and Sastry, 2005).

The GPCA approach includes the collection of polynomials from data and then evaluating their derivatives at data points to determine the subspaces passing through that point. PCA for noise in the exponential family, the method has probabilistic interpretation using a Poisson distribution which generates each data point with a mean parameter. It uses Poisson distribution with loss function without any constraints on matrices. PCA minimises the squared loss function as assumed in the Gaussian noise model. However, the Poisson distribution may better fit better integers, and the Bernoulli distribution may better fit binary data. The similarity of these distribution methods is the density function which can also be calculated as a member of the exponential family and could be extended to PCA algorithms (Collins, Dasgupta and Schapire, 2001; Oreifej, 2013).

All of the above extensions of PCA have certain limitations and associated problems. The drawbacks of KPCA are that it is highly dependent on nonlinear data and hence will never produce a linear PCA component. KPCA is also not suitable for identifying the kernel function. Probabilistic PCA lacks the distribution parameters of noise and is more beneficial for data compression. On the other hand, GPCA works efficiently on small data sets, and the robustness is not strong for outliers. Also, GPCA is more used in image processing. Therefore, it is essential to propose an algorithm to overcome the robustness of noise/outliers for efficient feature detection in the large 3D point cloud with low computation cost.

5.2.4.3 Principal Component Analysis Existing Applications

Nurunnabi, Belton and West (2012) proposed a method using PCA to segment point cloud data. Nurunnabi *et al*. statistically robust segmentation help to identify the underlying patterns in an unsupervised nonparametric fashion. The algorithm uses a minimum covariance determinant to produce a local covariance matrix. The PCA-based segmentation applications on terrestrial laser scanning datasets deliver good results for multi-planar surface extraction. However, the proposed algorithm does not potentially work for non-planar complex surface reconstruction. Belton and Bae (2009) proposed an automatic method for detecting roadside kerbs on urban point cloud data sets. The algorithm is divided into multiple phases to achieve detection.

1) The first step is segregating the road surface and other surfaces using statistical classification and segmentation. Next, the points that belong to the road surface are sampled and approximated using this technique.

2) After identifying the road surface points, the orientation and direction of the kerb at the candidate points are estimated using PCA on the local neighbourhood.

3) The candidate points are fitted with a kerb profile based on the derived cross-section and neighbouring properties.

4) The profile is incrementally chased along the kerb using candidate points to determine the kerb's path along the roads.

The algorithm results in a line representation of the kerb feature (Belton and Kwang-Ho, 2009). Bazazian, Casas and Ruiz-Hidalgo (2015) proposed sharp edge detection using a Gauss map clustering method. The algorithm uses the analysis from the eigenvalue of the covariance matrix defined by each point's *k*-nearest neighbour. First, PCA is applied to each cluster in local squares. After each point, a normal estimation of the *k*-nearest neighbour is applied to those sample points. Finally, the nearest neighbours are clustered by normal. The method is fast and accurate in small dihedral angles for detecting edges but is sensitive to the noise in the neighbourhood. The algorithm also lacks the threshold required for multi-scale analysis.

5.2.5 Summary

Section 5.2 analyses and evaluates the existing algorithms and methods for edge detection in point clouds. Followed by PCA in different fields and the existing algorithms using PCA extensions and applications have also been presented and reviewed.

It has been identified that there is a lack of algorithms or methods that can be used to produce adequate results on three-dimensional point cloud data. The findings include the lack of evidence of implementing the existing algorithms on large point cloud data. The algorithm must be cost-efficient because the point clouds could be massive with billions of points (disk storage of 150 GB or more).

The existing algorithms lack the verification of the processing time and demonstrate how large or small the point cloud data was. The existing algorithms also fail to produce results in the presence of different obstacles. Obstacles are very common in real-world point cloud data. Examples of obstacles include a gap, the shadow of the objects, the reflection of lights/rays, the elimination of unwanted feature detection, and the missing part of point cloud data.

Another common challenge is eliminating the outliers/noise while detecting features in the point cloud. The existing PCA methods and extensions overcome some of the outlier's problems; however, there is no evidence of a single method/algorithm working on different types of the point cloud, such as urban point clouds, terrain point clouds, handheld point clouds and airborne or LiDar point clouds with efficient computation for processing large point cloud data.

Therefore, it is necessary to propose, design, and develop a robust, accurate, and efficient method for edge detection in large 3D point cloud data with a minimum computational cost. Section 5.3 proposes a PCA-based edge detection method with real-time implementations on larger point clouds in the commercial environment.

5.3 A New PCA-based Method for Edge Detection

5.3.1 Overview

This section proposes a new PCA-based algorithm to detect the edges of different objects in the raw point cloud with efficiency, accuracy, and robustness. The procedure of the proposed algorithm is shown in Fig. 5.1.

The algorithm is designed to detect 3D edges on raw point cloud data, i.e., not filtered data, which may contain outliers and noises. The proposed algorithm consists of five stages: (1) the first stage is to sample the raw point cloud using the search sphere in a real-time large 3D point cloud (2) the second stage is to apply PCA to the sampled point cloud data in the search sphere and to extract the normal from PCA (3) the third stage is to categorise all the points as Plane1, and Plane2 through iterations (4) the fourth stage is to remove the points in both categories according to the threshold (5) the fifth stage is to identify the edges by intersecting two planes.

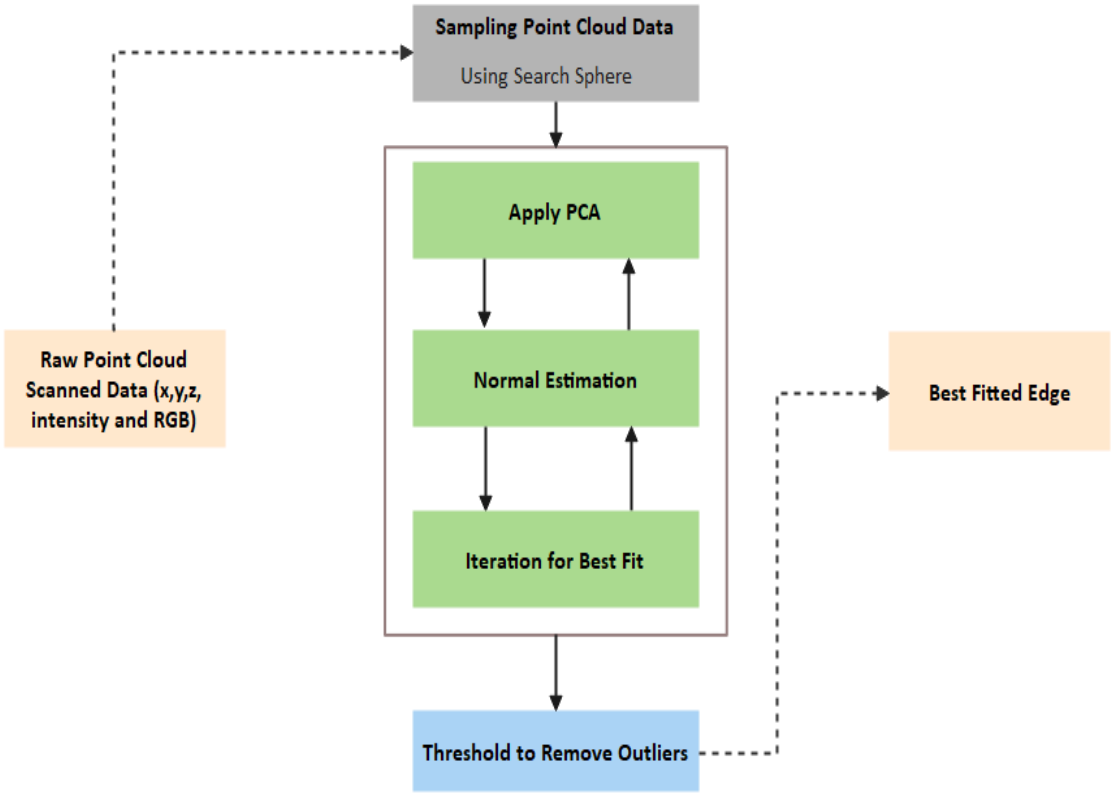
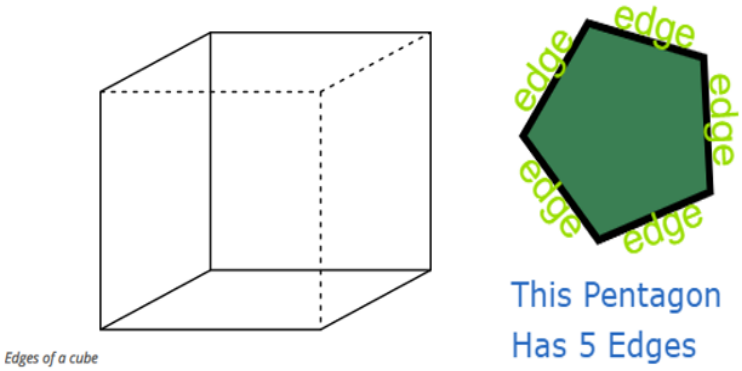


Figure 5.1 Procedure of PCA-Based edge detection algorithm

5.3.2 Important Terms

Different notations of edges have been studied in geometry, image processing, reverse engineering and topology. Mathematically in three-dimension, an Edge is defined as a line segment where the two surfaces meet. In geometrical 3D shapes, any line connecting two corners is called an edge, or any two connected surfaces also form an edge (Boster, 2016), shown in Fig 5.2 (a). Pierce (2018) defined an edge as a line segment on the boundary joining vertex (corner points) to another, as shown in Fig 5.2 (b).



(a) (b)

Figure 5.2 Edges defined by (a) (Boster, 2016) and (b) (Pierce, 2018)

In relation to point clouds, many studies have defined edges. Edges are defined as the curves along the surface directions that change abruptly—the edges by edge point representation are shown in Fig 5.3 (Du, 2020). These edge points are crucial for point cloud shape analysis. Wang and Shan (2009) defined edges in two types: jump and crease. Jump edges are defined as discontinuities in height values, and crease edges are formed when two surfaces meet. According to Farin, Hoschek and Kim (2002), an edge is a real analytic curve with finite length, whose limits of tangents are endpoints, and sharp edges are computed by surface-surface interaction. In addition, the extracted edge points are used in point cloud processing methods such as segmentation (Gilani, Awrangjeb and Lu, 2018) (Wang and Shan, 2009), mesh generation (Salman *et al.*, 2010) and resampling (Huang, 2013).

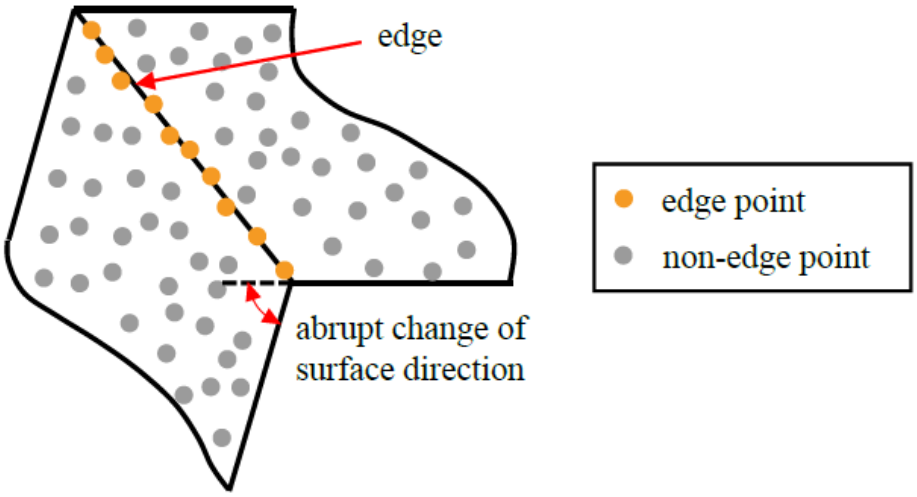
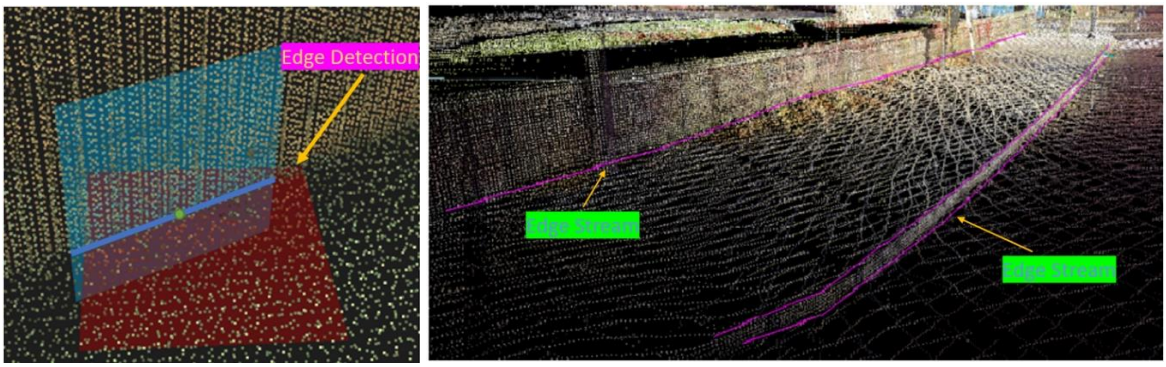


Figure 5.3 Du (Du, 2020) defined edges as curves along the surface direction

Using words like corners and boundaries can be confusing, and many interpretations of edges in different fields exist. To avoid any misunderstanding, in this thesis, two kinds of edges are defined for a typical point cloud as follows:

- The first kind of edge is called **Edge Sect**. An edge sect is an intersection line of any two planar surfaces**.** The intersection line can be formed on a planar surface, sharp edges line, break lines of a slope, walls and footpaths. Fig. 5.4 (a) shows the example of two different planar surfaces. The intersection of these surfaces is called Edge Sect.

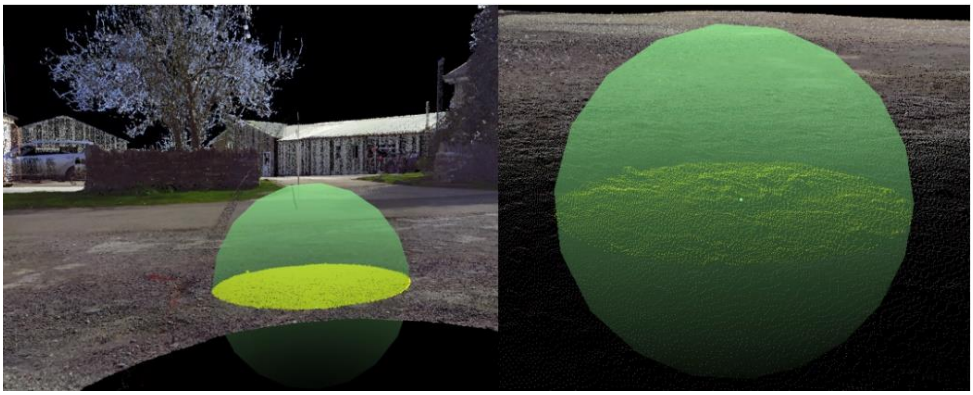
- The second kind of edge is called an **Edge Stream**. An edge stream is generally used to find edges without any gaps. The Edge stream could be defined as the series of intersection lines (edges) forming a stream of defined edges of consecutive planar surfaces. The point clouds have many disturbances; therefore, the edge stream must stop when there is any presence of disruptions like a large angular gap between two planar surfaces if the data is missing due to the shadow of other objects or due to abrupt changes in the direction of the normal of the planar surfaces in the point cloud. Examples of Edge Streams detection include road kerbs, building boundaries, roof edges, walls and floor plan outlines. An example of edge stream detection is shown in Fig. 5.4 (b) on the road kerb.



(a) (b)

Figure 5.4 Defined two types of edges (a) Edge sect and b) Edge Stream (the pink line)

Another essential method this thesis defines is the search sphere. A search sphere is a powerful tool used in various techniques in the thesis, such as the Search sphere used to delete points, as discussed in Chapter 4. This chapter uses the sphere to search and detect edge sects and streams. A **Search sphere** is a method of sampling the point cloud using a real-time 3D sphere. The search sphere moves along the cursor in 3D space in real-time, as shown in Fig. 5.5. The search sphere's size is user changeable, giving the flexibility to make the sphere big or small according to the point cloud data. The recommended size depends on the user's applications and the size of the feature to be extracted. All the points inside the search sphere are highlighted as the chosen points are apparent.



(a) (b)

Figure 5.5 (a) Search sphere on a given point cloud
(b) Magnified image with its inside points highlighted that are selected.

5.3.3 The Proposed Algorithm

In this section, the procedure of the proposed algorithm will be presented. A flowchart diagram of the proposed algorithm is shown in Fig. 5.6. The algorithm is applied to detect edge sects and edge streams in any given point cloud. The output is the intersection of two best-fit planes in the given point cloud data feature.

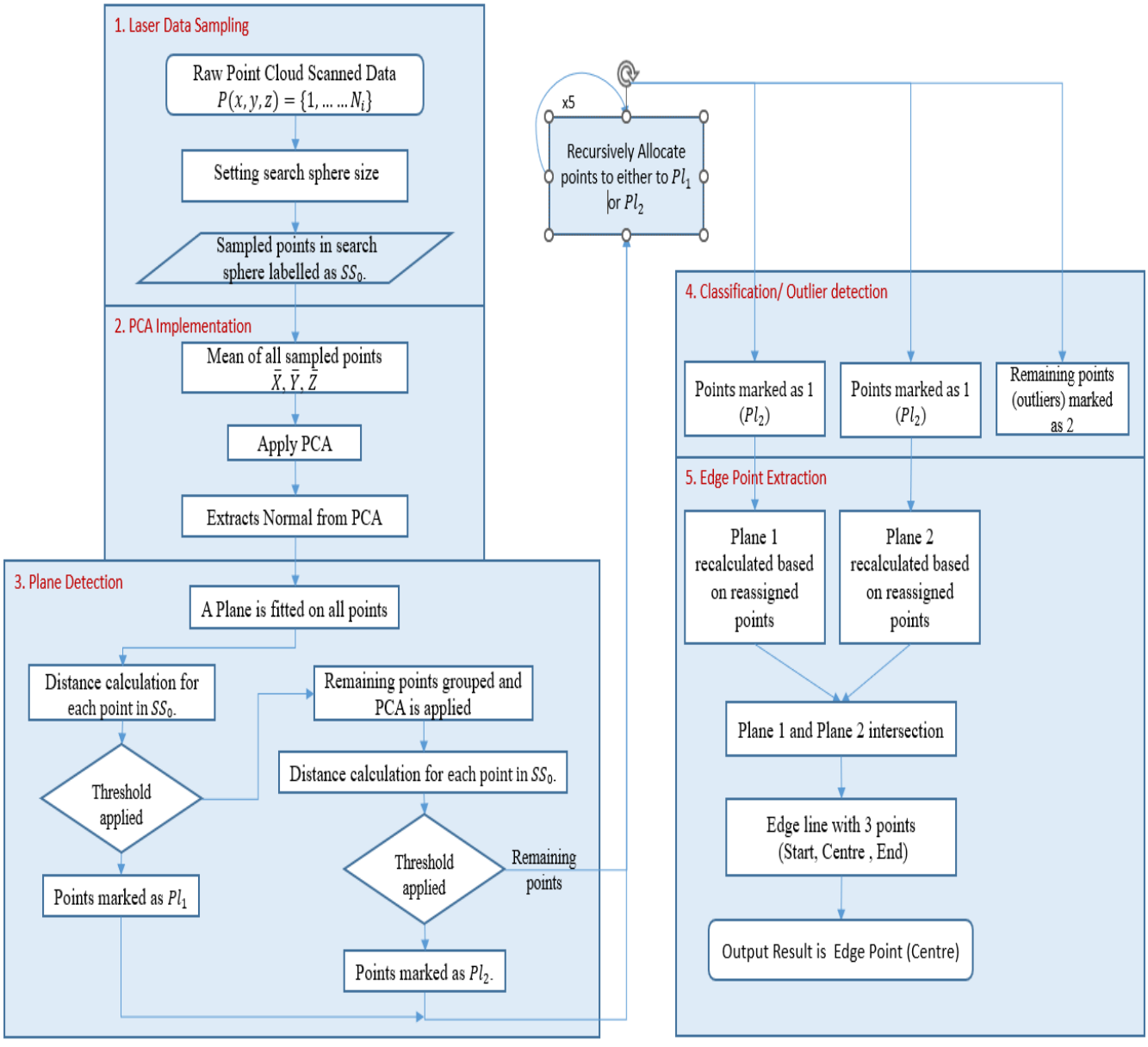
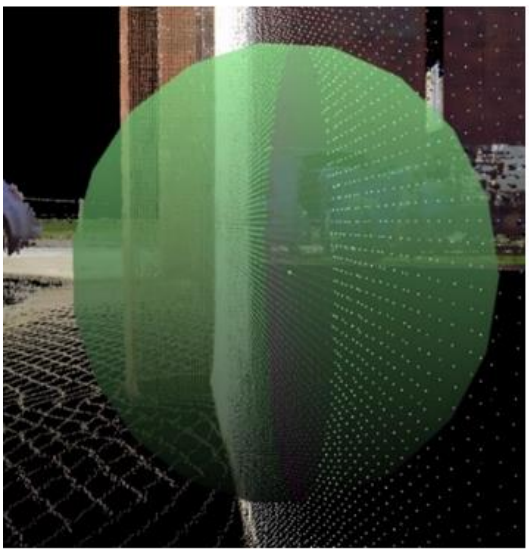


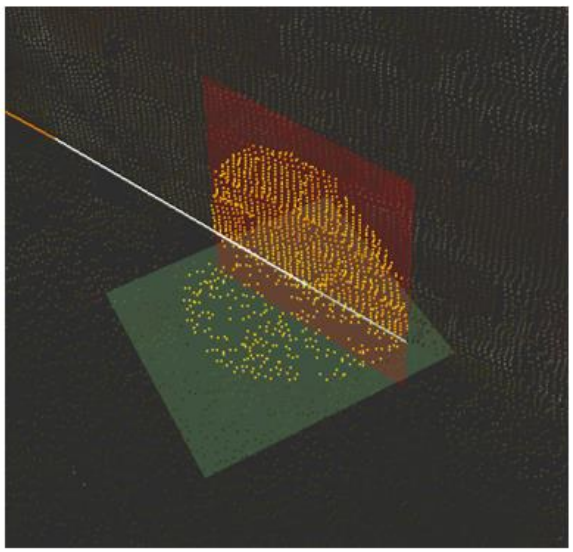
Figure 5.6 A flowchart diagram of the proposed algorithm.

5.3.3.1 Data Sampling

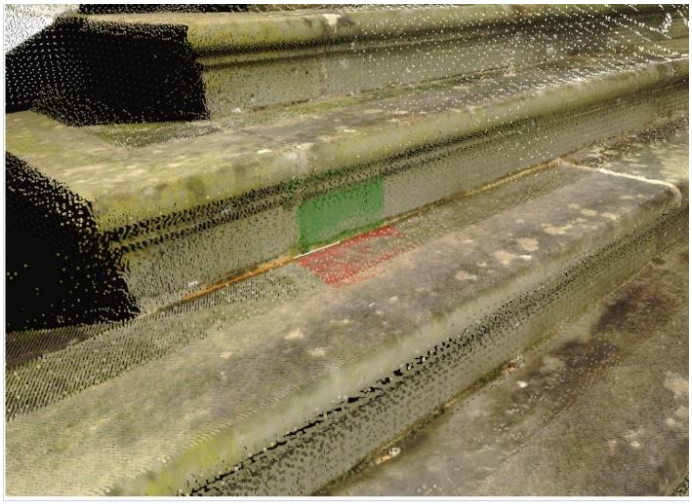
In the first phase, the whole raw point cloud data $P(x, y, z) = \{1, \dots \dots N_i\}$ is sampled in real-time by using the search sphere. All the points in point clouds have RGB and intensity values. However, the point's RGB and intensity are not used for implementing the proposed algorithm, just the X, Y, and Z values. Point clouds are generally large; therefore, point clouds must be sampled first. For example, a dataset used in this thesis for evaluation, i.e., University of Gloucestershire Park Campus data (Fullwood house), has 580.94 billion points and a church data set with 257 million points. For this purpose, the search sphere is used. A search sphere moves in real-time along the cursor wherever the mouse cursor is pointed to in the point cloud data. An example in Fig 5.7(a) shows that a green dot in the centre is where the mouse cursor is currently pointing in 3D space. The fuchsia-coloured points are the sampled points inside the sphere. The user sets the size of the search sphere, which directly depends on the size of the feature to be extracted. In the point cloud where the edge sects and edge stream are essential to detect are building walls, footpaths, kerbs, slope-break in any terrain data, stairs and windows. For example, edges found on the stair in the University of Gloucestershire Park Campus data set are shown in Fig 5.7 (b). All the sampled points inside the search sphere are labelled as SS_0. Also, these points are highlighted to make the selected sampled points evident.



(a)



(b)



(c)

Figure 5.7 Sphere in use (a) for sampling (b) to find the edges on stairs
(c) to find edges between wall and ground

5.3.3.2 PCA Implementation

The second phase is PCA implementation, which plays a significant role in the algorithm. PCA is defined by Jolliffe I.T (2002) as an "*orthogonal linear transformation that transforms the data to a new coordinate system such that the greatest variance by some scalar projection of the data comes to lie on the first coordinate (called the first principal component), the second greatest variance on the second coordinate, and so on.*"

The goal is to transform a given D-dimension of data set $X = \{x_i \dots \dots \dots . x_n\}$ of a new dimension p. The data is organised next. The data comprises a set of observations of p variables. To reduce the dimension of the data so that each observation can be described as $p < D$. Further, the data are arranged as a set of data vectors $\{x_i \dots \dots \dots . x_n\}$ where x_i representing a group of observations of p variables. Next, the mean is calculated for each dimension, as shown in Equation 5.1 for X.

$$\bar{X} = \frac{1}{N} \sum_{i=1}^{N} x_i \tag{5.1}$$

Then the **Standardization** of the data is performed by subtracting the mean from the original data (from each element) and represented as 'data adjusted'. Where the array is formed by $(x - \bar{x},\ y - \bar{y}, z - \bar{z}\ )$. The length of data adjusted is same as the original data.

The covariance matrix is a $m \times m$ symmetric matrix where m is the number of dimensions that have the covariances associated with all possible pairs of the initial variables (Jaadi, 2022). For this thesis, the data used is three-dimensional; therefore, the covariance matrix is a 3×3 matrix.

Covariance Matrix:

$$\begin{bmatrix} Cov(x,x) & Cov(x,y) & Cov(x,z) \\ Cov(y,x) & Cov(y,y) & Cov(y,z) \\ Cov(z,x) & (Cov(z,y) & Cov(z,z) \end{bmatrix} \tag{5.2}$$

The covariance matrix is denoted as Equation 5.3.

$$C = \frac{1}{N} \sum_{i=1}^{N} (p_i - \bar{p}\,)\,(p_i - \bar{p}\,)^T \tag{5.3}$$

where C is the conjugate transpose operator, N is the total number of points, p_i is the feature component and $\bar{p}$ is the mean of all the points, and T denotes the transpose matrix. Eigenvectors $\varepsilon_1, \varepsilon_2, \varepsilon_3$ and eigenvalues $\lambda_1, \lambda_2, \lambda_3$ are computed from the covariance matrix to determine the data's principal components. Principal components are new variables constructed as linear combinations or mixtures of the initial variables (Jaadi, 2022). The principal components are uncorrelated, with maximum information in the first component, then the maximum remaining information in the second and then the third. Once the eigenvalues are sorted such that $\lambda_1 \geq \lambda_2 \geq \lambda_3$.

The correlation between variables x_i and principal component φ_α is given by Equation 5.4 (Aluja-Banet, Morineau and Sanchez, 2018).

$$cor(\alpha, j) = \sum_{i=1}^{n} p_i \left(\frac{x_{ij}}{s_j}\right)\left(\frac{\varphi_{i\alpha}}{\sqrt{\lambda_\alpha}}\right) \tag{5.4}$$

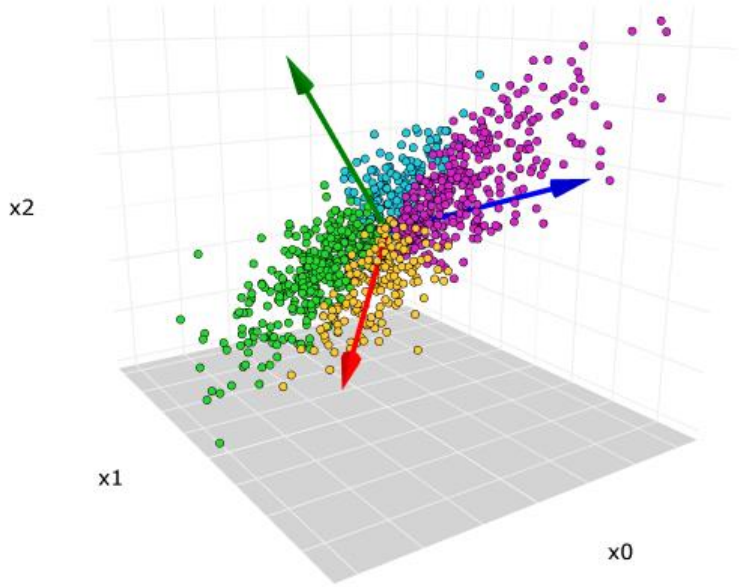


Figure 5.8 PCA in 3D with highlighted arrows in red (PC1), blue (PC2), and green (PC3). Source: (Cheng, 2022)

Example:

Suppose the data set is 2D with variable m, n, and the eigenvectors and eigenvalues of the Covariance matrix are as follows:

$$v1 = \begin{bmatrix} 0.677 \\ 0.735 \end{bmatrix} \qquad \lambda_1 = 1.28$$

$$v2 = \begin{bmatrix} -0.735 \\ 0.677 \end{bmatrix} \qquad \lambda_2 = 0.049$$

After sorting the eigenvalues in descending order $\lambda_1 > \lambda_2$ the eigenvector corresponding to the first principal component is $v1$ the second principal component corresponds to $v2$. To determine the variance in percentage in the above example, each component is divided by the total eigenvalues, which results in PC1 with 96% and PC2 with 4% variance of the data.

After applying PCA to the sample of the point cloud data set, the results are the following:

- Mean of the array of points inside the sample data set (sphere),
- Principal components (PC) – First PC is found where the maximum variation lies, the second PC with less variation than the first and third. All three PCs are orthogonal to each other and transform into a new coordinate system,
- Eigenvectors – direction cosine of each principal component,
- Eigenvalues – a scalar derivation from eigenvectors.

The implementation of PCA for the proposed algorithm starts by calculating the mean of all sampled points labelled as SS_0 . PCA is then applied to the sampled points SS_0. For all 3D points SS_0, the covariance matrix is computed by using Equation 5.1, and the eigenvectors $\overrightarrow{V_1}$, $\overrightarrow{V_2}$ and $\overrightarrow{V_3}$ and eigenvalues λ_1, λ_2 and λ_3 are also obtained. The three principal components $\overrightarrow{pc_1}$, $\overrightarrow{pc_2}$ and $\overrightarrow{pc_3}$. are derived through a transformation in a way that the first principal component $\overrightarrow{pc_1}$ has the most significant possible variance succeeding second $\overrightarrow{pc_2}$ and third $\overrightarrow{pc_3}$ with the highest possible variances. Since the three components are orthogonal to each other, the third principal component is the normal $\hat{n}$ to the plane of the first and second principal components. Based on the extracted normal and the origin, the first plane is fitted on all points in SS_0.

5.3.3.3 Plane Detection

The third phase is to find the best-fit plane on the sampled data in SS_0. After the extraction of the normal and the origin, a plane is fitted on the points SS_0. A plane is 3D is defined by Equation 5.5

$$Ax + By + Cz + D = 0 \tag{5.5}$$

and with the origin and normal, the plane equation is presented in Equation 5.6

$$\vec{n} \,.\, O(x, y, z) = 0 \tag{5.6}$$

where $\vec{n}$ is the directional vector of the normal and $O(x, y, z)$ is the origin. The best-fit plane is fitted on the sampled data inside the sphere.

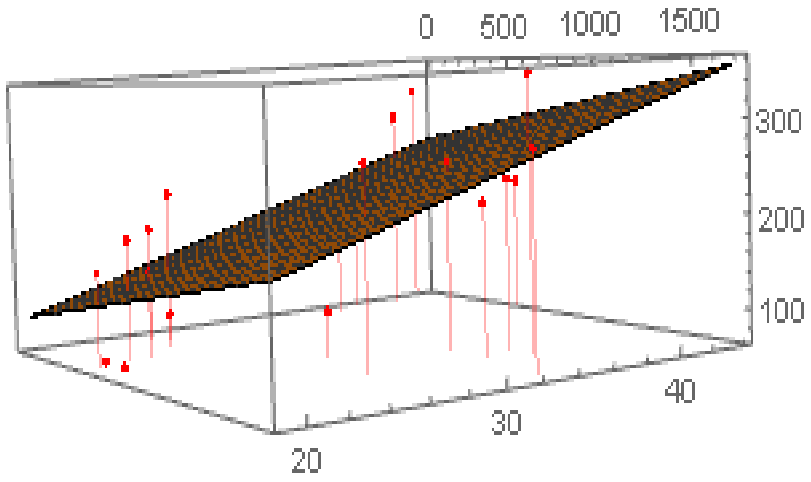


Figure 5.9 Best fit plane on the data presented as red points

The points which form the fitted plane could also contain outliers, which can cause the plane's misfit on the sampled data. Therefore, the next step is to filter all the outlier points. A threshold is applied based on distance calculation to achieve the outlier removal from each point to the fitted plane. The threshold is the standard deviation value. The result of the threshold is plane Pl_1. Furthermore, all the points remaining are classified as outliers. To these outlier points, PCA is applied again to find the normal $\hat{n}$ for the second plane. Based on the distance calculation threshold from all the remaining points (not including the first plane points), outliers are separated and resulting in the second plane Pl_2. The distance of the points (x_i, y_i, z_i) to the plane $Ax + By + Cz + D$ is Equation 5.7.

$$\Delta = \frac{|Ax_i + By_i + Cz_i + D|}{\sqrt{A^2 + B^2 + C^2}} \tag{5.7}$$

The algorithm includes the perpendicular least squares fitting method, which is superior to vertical and gives more accurate results.

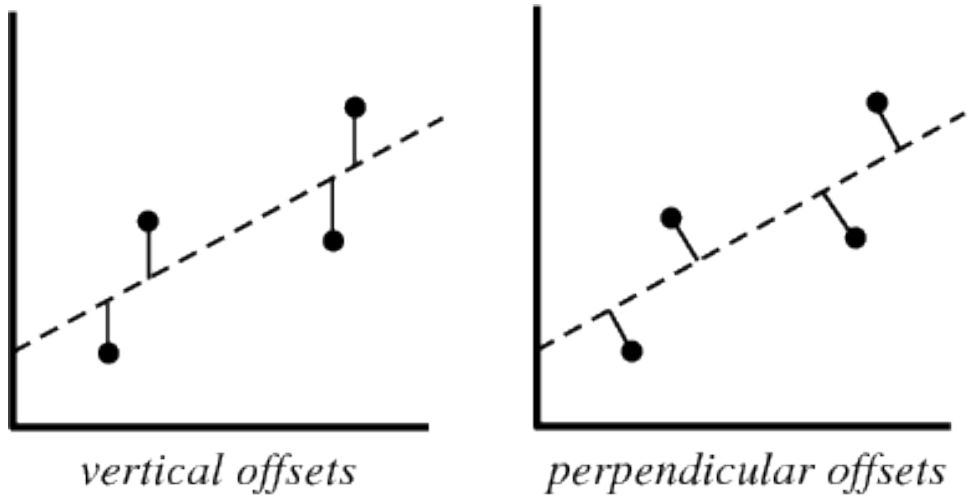


Figure 5.10 Least Squares Fitting Perpendicular offset

The perpendicular regression is implemented with the least squares fitting that calculates the distance of a point from a plane. Then, all points' distance is calculated to the planes as shown in Fig 5.11 and minimises the squared distance to the planes. The coefficient of perpendicular offset to the plane $z = a_0 + b_0 x + c_0 y$ is given by Equations 5.8 and 5.9 (Sampaio, 2006)

$$a = \frac{a_m}{n_{z0}} \quad , \quad b = -\frac{n_{x0}}{n_{z0}} \quad , \quad c = -\frac{n_{y0}}{n_{z0}} \tag{5.8}$$

$$n_{x0} = \frac{-b_0}{\sqrt{1 + b_0^2 + c_0^2}} \quad , n_{y0} = \frac{-c_0}{\sqrt{1 + b_0^2 + c_0^2}} \quad ,$$

$$n_{z0} = \frac{1}{\sqrt{1 + b_0^2 + c_0^2}} \tag{5.9}$$

where a_m is intercept after convergence, where $d_0 = \frac{a_0}{\sqrt{1+b_0^2+c_0^2}}$ is the distance of the plane from the origin and n_{x0}, n_{y0} and n_{z0} are direction cosine of the unit vector $\overrightarrow{n_0}$ normal to the plane.

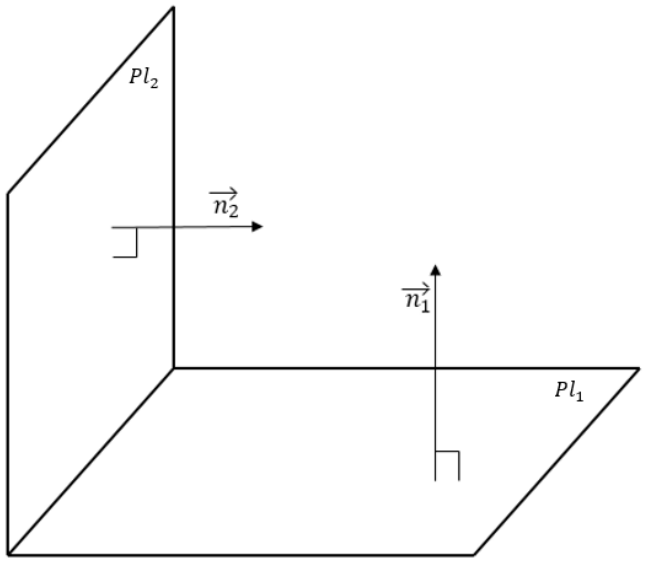


Figure 5.11 Two planes' normals $\overrightarrow{n_1}$ and $\overrightarrow{n_2}$

5.3.3.4 Classification

The fourth phase is to classify all the points through iterations. The points are classified into three groups. As discussed in Section 5.3.3.3, two planes are fitted on the sampled data as Pl_1 and Pl_2. The leftover points marked as outliers in the search sphere SS_0 are re-evaluated to fit the sample points. The outlier points are reallocated to respective planes based on the threshold. After a few iterations, the points nearest to either Pl_1 or Pl_2 are marked as 0 and 1, and all the remaining points are marked as 2. The details are explained in Section 5.5.4. The points marked with 0 are the best-fit plane points for the first plane Pl_1 , 1 for best-fit plane points for the second plane Pl_2 and 2 for the outlier points.

The final phase is to extract the edge. The intersection of two best-fit planes results in an edge sect. The edge sects consist of three points: starting, centre and endpoints. The three points play an essential role in edge stream detection, as explained in Section 5.3.5.

5.3.3.5 Best Fit Planes Algorithm

Algorithm for Finding Edges

Input: Point cloud =$P(x, y, z) = \{1, \dots \dots . Ni\}$.

1: Edge points {E}←∅

2: **For** i= 0 to size {[P]}**do**

3: Mark {[P]} as P_0

4: Calculate mean {M} ← $\bar{X}, \bar{Y}, \bar{Z}$

5: Calculate Covariance Matrix {C}←$\begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix}$

6: Calculate eigenvalues Matrix {E^v} ← {0, 0, 0}

and eigenvector Matrix {V} ← {0,0, 0}

7: Perpendicular offset ←$\overrightarrow{ost}$ and distance d

8: Mark P_0 based on Threshold ϵ (inliers) and rest P_1

9: Continue

10: **End If**

11: P_0← {0, 0, 0}, or P_1← {0, 0, 0}

12: Compute Pl_1 and Pl_2

13. Compute line l

14: Return Edge points{E}

5.3.4 Outlier Detection

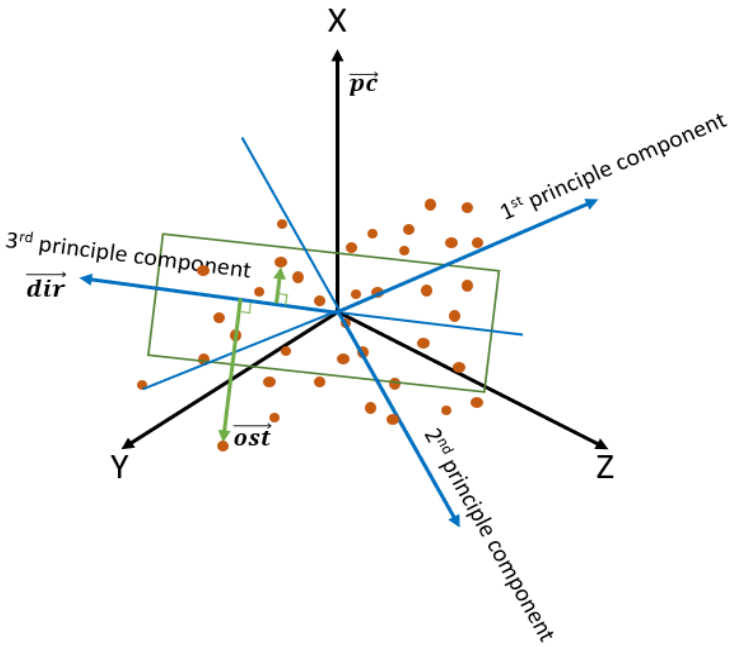
Once PCA has been applied to the sampled points inside the search sphere, as discussed in Section 5.3.3. Each point P_{Ni} inside the search sphere is analysed. Further, based on its distance to either plane, the point is included in the best-fit plane calculation for the respective plane. As PCA does not deal with the noise (Sengupta and Mitra, 1997), the proposed algorithm is combined with the least squares perpendicular regression method in order to obtain the best-fit planes Pl_1 and Pl_2 in the given data set. Perpendicular offset is calculated for all the points $p_i = (x_i, y_i, z_i)$ labelled as SS_0 using Equations (5.10) – (5.12)

$$Pl_{dir} = (\overrightarrow{dir}.x \times \overrightarrow{pc}.x_i) + (\overrightarrow{dir}.y \times \overrightarrow{pc}.y_i) + (\overrightarrow{dir}.z \times \overrightarrow{pc}.z_i) \quad (5.10)$$

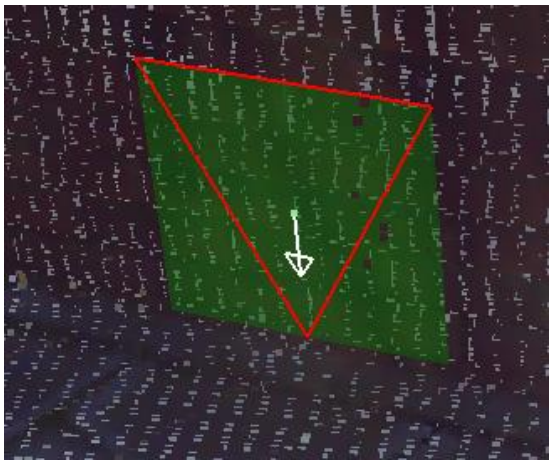
$$denominator = (\overrightarrow{dir}.x \times \overrightarrow{dir}.x) + (\overrightarrow{dir}.y \times \overrightarrow{dir}.y) + (\overrightarrow{dir}.z \times \overrightarrow{dir}.z) \quad (5.11)$$

$$\overrightarrow{ost} = \frac{((\overrightarrow{dir}.x \times p_i.x_i) + (\overrightarrow{dir}.y \times p_i.y_i) + (\overrightarrow{dir}.z \times p_i.z_i) - Pl_{dir})}{denominator} \quad (5.12)$$

where $\overrightarrow{dir}$ is the vector of the third principal component, $\overrightarrow{pc}$ is the vector of the point cloud data and $\overrightarrow{ost}$ is the vector of perpendicular offset of all points to the respective planes, as shown in Fig 5.12.



(a)



(b)

Figure 5.12 (a) The origin axis X, Y, and Z of the points cloud and three principal components and (b) the Best-fit plane with the white arrow showing the gradient of the plane.

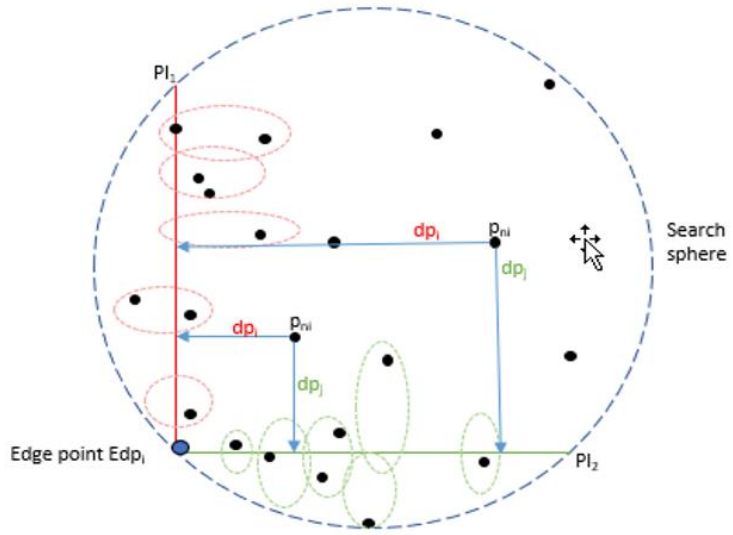
The $\overrightarrow{ost}$ vector of every point is calculated from the first fitted plane (dp_i = perpendicular distance of each point to plane one) and stored in P_0 and for the second fitted plane (dp_j = perpendicular distance of each point to plane two) stored as P_1. Standard deviation is calculated for the points inside P_0 and P_1. Based on the ϵ threshold, points are removed from P_0 and P_1. The threshold value is calculated using the standard deviation σ of all points by Equation 5.13.

$$\sigma = \sqrt{\frac{\sum(v_i - \mu)^2}{N}} \tag{5.13}$$

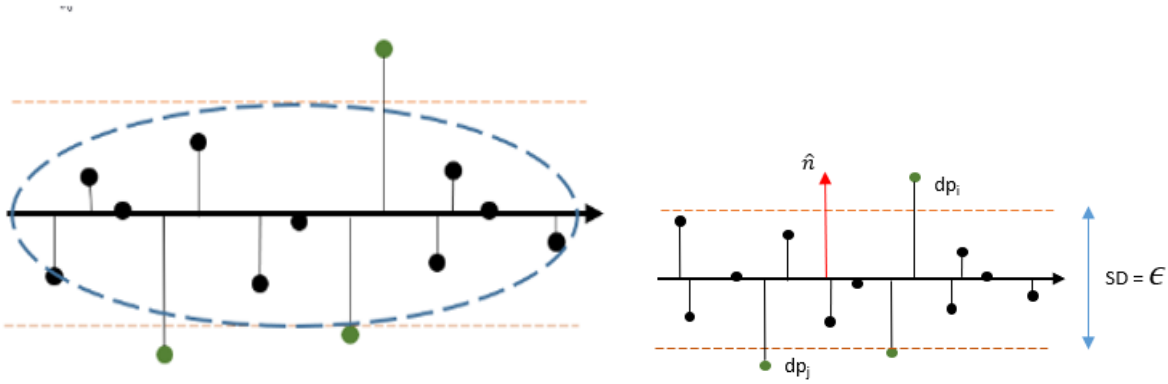
where v_i is the perpendicular distance of each point from the plane, μ is the mean of all the perpendicular distances, N is the total number of points' distances. All the removed points from P_0 and P_1 are re-evaluated by $\overrightarrow{ost}$ vector of all the points in the search sphere SS_0.

As shown in Fig 5.13(a), the blue dotted circle is a 3D real-time search sphere SS_0 where all the points are assessed based on the $\overrightarrow{ost}$ vector and a ϵ threshold. The point is then considered in the first or second plane calculation. After a few iterations, all the points in P_0 are labelled as '0', points in P_1 are labelled as '1', and all the remaining points are labelled '2'. The points

labelled as ‘2’ are considered outliers. Two best-fit planes are Pl_1 and Pl_2 generated from the points labelled ‘0’ and ‘1’, as shown in Figures 5.13 (b) and (c).



(a)



(b) (c)

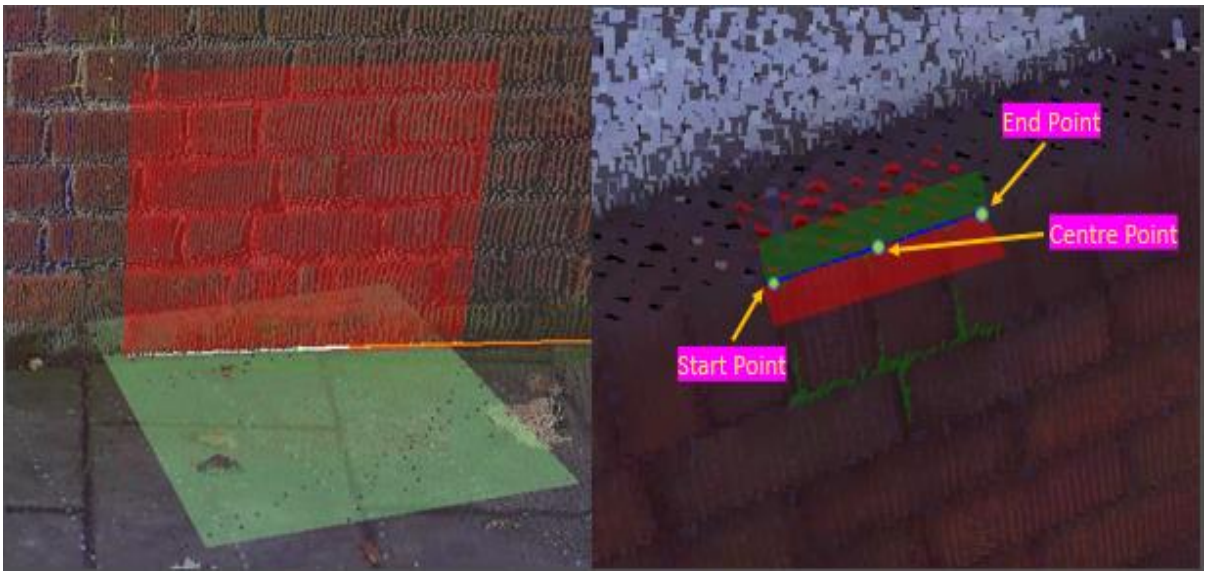
Figure 5.13 (a) Blue dotted circle represents a live 3D search sphere, the solid red line is the best-fit plane Pl_1, and the solid green line is the best-fit plane Pl_2. The black dots represent points inside the sphere $P_{Ni\ red}$, and the green ellipse or circles represent the points belonging to plane red or plane green. (b) Perpendicular regression method on each point dp_i and dp_j and the outlier (green coloured) points are removed (c) Combining the diagram shown in a) and b).

5.3.5 Edge Sects

As discussed above, the proposed algorithm implementations can be extended further to obtain two best-fit planes Pl_1 and Pl_2 from the sampled point cloud data shown in Fig.5.14 (a). Planes are called best fit as they best fit the given dataset until all the points satisfy the threshold and are grouped as mentioned above.

The points are grouped as P_0 and P_1 based on their offset distance dp_i and dp_j from each plane, respectively, as shown in Fig 5.13, the rest of the points are considered outliers. The intersection of two best-fit planes produces an edge sect. An edge sect is presented using an example to show the edge points, as shown in Fig. 5.14 (b) as follows:

a) A centre point,
b) Start point, and
c) End point.



(a) (b)

Figure 5.14 (a) The two planes, red and green, are the best fit planes derived from the proposed algorithm, and (b) The intersection forms a blue line, and the green dots are edge sect points

5.3.6 Edge Stream: Extension of Edge sects

Applying the proposed algorithm for edge detection method (PCA) inside the search sphere. The result is two planes, three edge points and two sets of PCs (principal components). From the edge start point, as shown in Fig 5.14, the edges are detected in real-time, and their results are shown in Fig 5.15.

- Three principal components
- Three eigenvalues
- X, Y, Z components of the start point x_1, y_1, z_1 , centre point and end point.

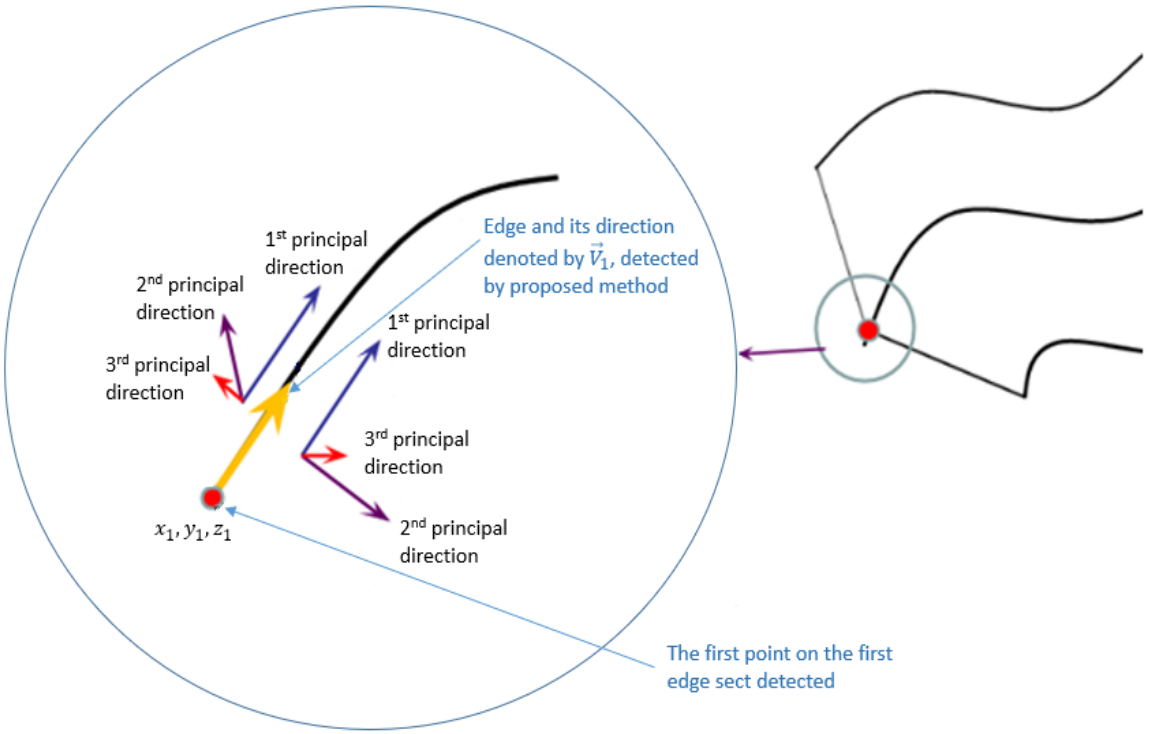


Figure 5.15 Resultants of PCA

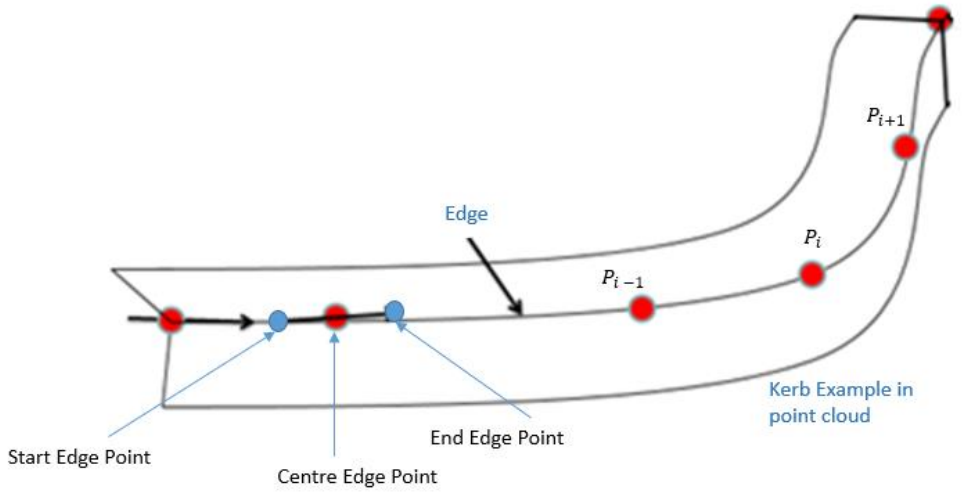


Figure 5.16 Example of a road kerb with red points being the centre edge points of edge sects

The edges have three points. Figure 5.16 presents centre points found from the edge line inside the sphere as input for edge stream detection. The extension of edge sects is applied by using the following equation:

$$\begin{pmatrix} x_2 \\ y_2 \\ z_2 \end{pmatrix} = T \begin{pmatrix} dl \\ dl \\ dl \\ 1 \end{pmatrix} \tag{5.14}$$

where T is a transportation matrix and is defined in Equation 5.15

$$T = \begin{bmatrix} \cos\theta_{1x} & 0 & 0 & x_1 \\ 0 & \cos\theta_{1y} & 0 & y_1 \\ 0 & 0 & \cos\theta_{1z} & z_1 \end{bmatrix} \tag{5.15}$$

The proposed algorithm is applied and results in the first centre edge point. If considering only centre points, the result points along the kerb are shown in Fig 5.16. The second centre point has another two sets of principal direction vectors, eigenvalues and x, y, z components (similar to the first centre edge point) and so on for each centre edge point. For the principal vector, Let

$\vec{V}_{1,f} = (x_{1,f}, y_{1,f}, z_{1,f})$ be the first principal vector of plane 1

$\vec{V}_{1,s} = (x_{1,s}, y_{1,s}, z_{1,s})$ be the second principal vector of plane 1

$\vec{V}_{1,t} = (x_{1,t}, y_{1,t}, z_{1,t})$ be the third principal vector of plane 1

$\vec{V}_{2,f} = (x_{2,f}, y_{2,f}, z_{2,f})$ be the first principal vector of plane 2

$\vec{V}_{2,s} = (x_{2,s}, y_{2,s}, z_{2,s})$ be the second principal vector of plane 2

$\vec{V}_{2,t} = (x_{2,t}, y_{2,t}, z_{2,t})$ be the third principal vector of plane 2

Obtain the angle between $\overrightarrow{V_1}$ and $\overrightarrow{V_2}$ by applying dot product

$$\cos\theta = \frac{\overrightarrow{V_1} \cdot \overrightarrow{V_2}}{\left|\overrightarrow{V_1}\ \overrightarrow{V_2}\right|} \tag{5.16}$$

Until $\theta < \theta_t$, the procedure will determine the next point where the sphere's centre is positioned. Thus, Equation 5.14 will now become 5.17, and Equation 5.15 will become 5.18.

$$\begin{pmatrix} x_{i+1} \\ y_{i+1} \\ z_{i+1} \end{pmatrix} = T \begin{pmatrix} dl \\ dl \\ dl \\ 1 \end{pmatrix} \tag{5.17}$$

$$T = \begin{bmatrix} \cos\theta_{ix} & 0 & 0 & x_i \\ 0 & \cos\theta_{iy} & 0 & y_i \\ 0 & 0 & \cos\theta_{iz} & z_i \end{bmatrix} \tag{5.18}$$

While continuing with the above procedure, certain checks need to be performed:

- Register the number of points in the sub-sampled point cloud within the sphere, which is denoted by N_{i1} and N_{i2}

- The average is calculated for N_{i1} and N_{i2} where T_{NOP} is the total number of points as shown below:

$$N_{1a} = \sum_{i=1}^{M_i} \frac{N_{i1}}{T_{NOP}} \tag{5.19}$$

$$N_{2a} = \sum_{i=1}^{M_i} \frac{N_{i2}}{T_{NOP}} \tag{5.20}$$

- Calculate the ratios:

$$\gamma_1 = \frac{N_{i+1,1}}{N_{1a}} \tag{5.21}$$

$$\gamma_2 = \frac{N_{i+1,2}}{N_{2a}} \tag{5.22}$$

To determine T when $\gamma_1 < \gamma_{1T}$ or $\gamma_2 < \gamma_{2T}$

- If $\gamma_1 < \gamma_{1T}$, it signifies that the shape of plane 1 changes sharply, and the point detected by Equations 5.17 and 5.18 is not an actual point in the point cloud. Thus, Equations 5.17 and 5.18 should be modified as follows:

$$\begin{pmatrix} x_{i+1} \\ y_{i+1} \\ z_{i+1} \end{pmatrix} = T_1 T \begin{pmatrix} dl \\ dl \\ dl \\ 1 \end{pmatrix} \tag{5.23}$$

$$T_1 = \begin{bmatrix} \cos\beta & -\sin\beta & 0 \\ -\sin\beta & \cos\beta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.24}$$

and β is the angle between $\overrightarrow{V_{P_i P_v}}$ and $\overrightarrow{V_{P_i P_{i+1}}}$ in the third principal direction at the point P_i

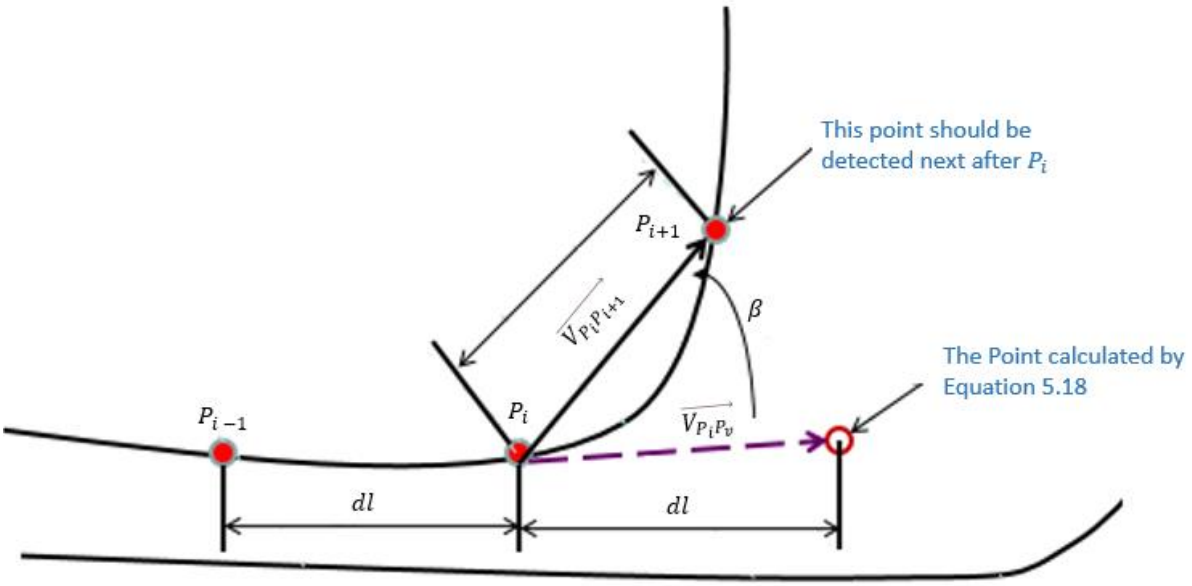


Figure 5.17 Edge points with the direction

- β can be determined by iteration trials. The process to determine β is as follows:
  - ❖ Let $\beta = 0$ for Vector $\overrightarrow{V_{P_i P_v}}$
  - ❖ Increase β by $d\beta$ ($d\beta$ is the radius of the user-defined size of the sphere)
  - ❖ Calculate γ_1
  - ❖ If $\gamma_1 < \gamma_{1T}$, then register the coordinates of the point P_{i+1}
  - ❖ Apply the edge detection method.

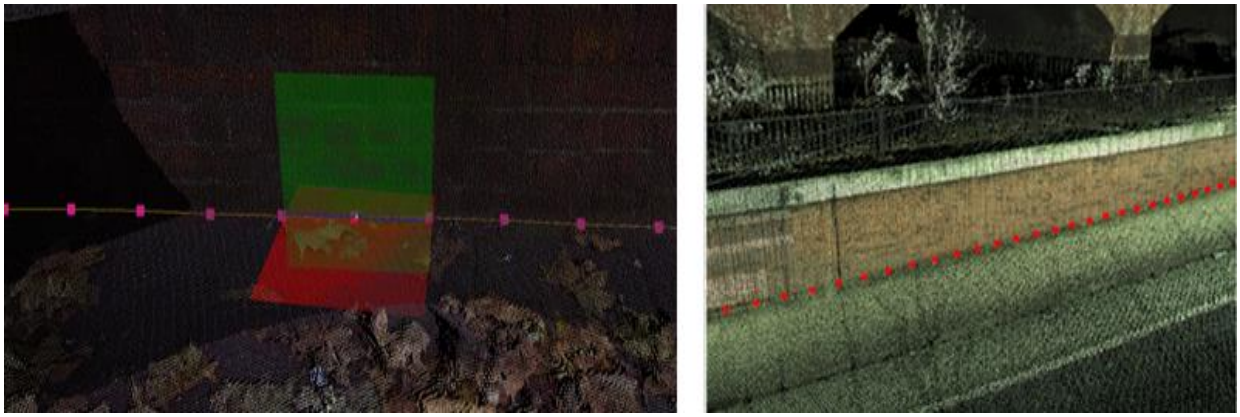
- If $\gamma_2 < \gamma_{2T}$, it signifies that the shape of plane 2 changes sharply, and the point detected by Equations 5.1 and 5.18 is not an actual point. The rotation axis should be the third principal direction vector of point P_i

- If both $\gamma_1 < \gamma_{1T}$ and $\gamma_2 < \gamma_{2T}$, it signifies that the shape of both planes changes sharply. The rotation axis to be used will be the cross product of the third principal direction vector of Point P_i

Once P_{i+1} has been detected by the above steps, the procedure is repeated until it can no longer find the points.

5.3.6.1 Stopping Criteria

The series of edge points are detected using an edge stream. The edge stream is powerful and continues even in the presence of obstacles. Therefore, a stopping criterion is implemented to control and provide accurate results. Figure 5.18 shows an example of centre points detected along the wall and the footpath.



(a) (b)

Figure 5.18 Highlighted centre points detected by the proposed algorithm (a) Shows detected planes with start, centre and end points (b) Stream of points detected

Table 5. 2 Two planes detected (orange and blue)

Points in Plane	Iter. (best-fit plane)	Standard Deviation (m)	Mean Perp Dist (m)	Max Perp Dist (m)	Min Perp Dist (m)	Time (Iter.)	Time (ms)
3236	1	0.0419408	0.0896429	0.3115862	9.27451181026254E-08	4	68
5105	2	0.0515273	0.0878875	0.3087326	5.22090422265999E-07	4	68
5074	3	0.0515421	0.0880621	0.3090510	3.07035415209336E-14	4	68
5062	4	0.05153760	0.0881201	0.3091568	8.06796118306324E-07	4	68
5059	5	0.05153700	0.0881372	0.3091868	1.42705135255466E-07	4	68
14142	1	0.06832822	0.0473381	0.2882325	8.25731507491804E-06	7	68
9036	2	0.03106678	0.0460698	0.2923764	3.25821342416892E-10	7	68
9068	3	0.03117444	0.0461259	0.2925754	4.30292573273945E-12	7	68
9080	4	0.03118604	0.0461348	0.2925759	1.00305969190162E-10	7	68
9083	5	0.03119782	0.0461314	0.292590697	2.38559955208919E-06	7	68

Table 5.2 shows group P_0 as an orange colour and group P_1 as blue colour going through each iteration as the number of points has been best fitted to each group. In this example, the iterations are set from 1 to 5. The iterations end when the number of points in the plane (either Pl_1 or Pl_2) starts repeating itself, i.e., all the points are successfully assigned to a group, and therefore no point in carrying on the iterations. The computer used to run the proposed algorithm specification is as follows: Intel Core i7 processor running at 2.60 GHz using 16 GB RAM running on 64-bit Windows 11 version 22H2.

The example shown in Table 5.2 stops at the fifth iteration. Other measures for quality control of best-fit planes are standard deviation, perpendicular distance: minimum, maximum and mean, time duration for finding points in each plane, time duration for each iteration and total time in milliseconds. The stopping criteria for edge detection are:

- The number of points in the planes is repeated in *i* or *i*-1 iterations.
- When the angle between the two planes is greater than the specified angle
- When the angle between consecutive edges is less than the specified angle
- When the total distance of the edge stream is not a multiplier of the search sphere size

5.4 Proposed Algorithm Implementation on Commercial Software

The application of the proposed algorithm on commercial software is discussed in this section. The commercial software is called “LSS - 3D Vision”. However, manually extracting and accurately identifying the edge features using the number of points along the edges of a surface in a point cloud proved tricky and time-consuming.

Therefore, the aim was to produce automated results of finding edges using a 3D sphere for kerb edges, walls, stairs etc. Hence, the edge streams are designed and developed. Edge steam is the first innovative method in the industry.

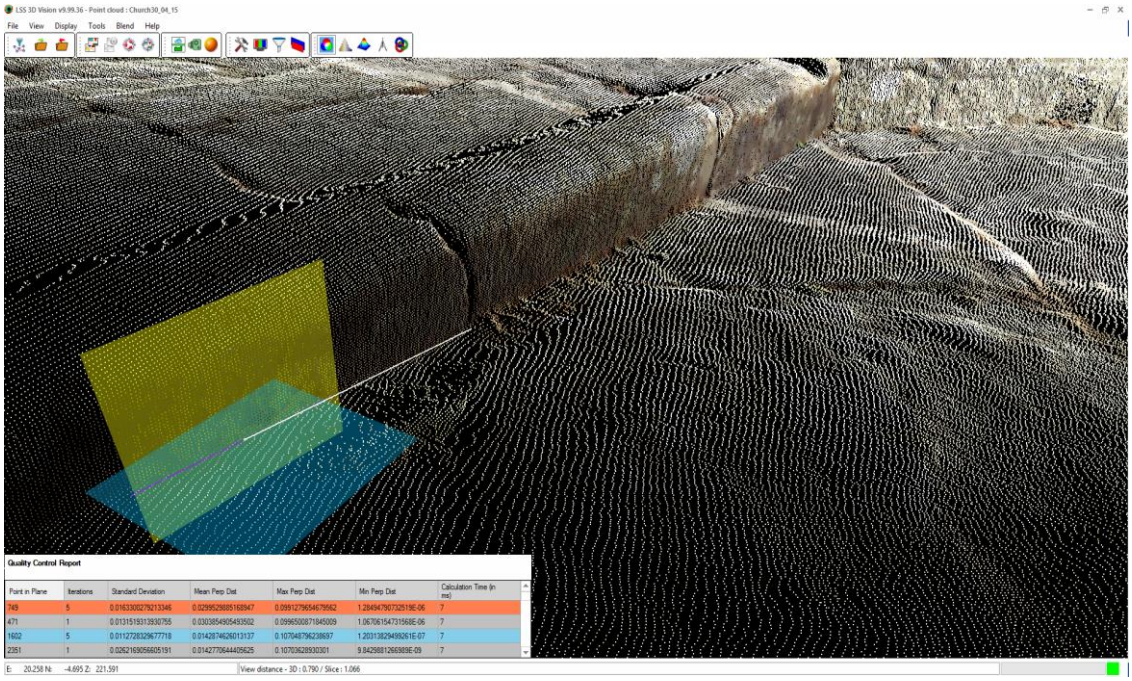


Figure 5.19 Edge sects extracted manually along kerb edges

5.4.1 Key Features

This section describes the application for finding the edges in LSS software and intends to provide the background of technical terms and reports used by the algorithm. An edge is calculated in real-time using the search sphere by moving the cursor in the 3D space within the point cloud. This search sphere is placed along the edge of a feature where two surfaces are present to identify an edge. All points inside the search sphere (commercially called "Searchphere™" in the LSS software) are divided into two planes, the intersection of which is identified **as an Edge.** The key features to operate the edge/edge stream commands within the software are:

- Search sphere options selection – The users must select the edge/edge stream option from the search sphere settings.
- Search sphere size – The users need to set the size of the sphere. Otherwise, the default size will be used.
- Edge detection Report – The edge report explains the edge/edge stream details, as shown in Figure 5.20.

The description of terminology used in the report of Find Edge/Edge Stream:

- Points in Plane –Indicates the number of points found in each iteration,
- Iterations – Iteration number for regression (maximum 5 per edge),
- Standard Deviation – Calculated with the perpendicular distance of each point in the search sphere to each plane,
- Mean Perp Dist – Mean of all the perpendicular distances,
- Max Perp Dist – Largest perpendicular distance,
- Min PerpDist – Smallest perpendicular distance,
- Calculation time – Total time taken for iterations of each plane.

Quality Control Report

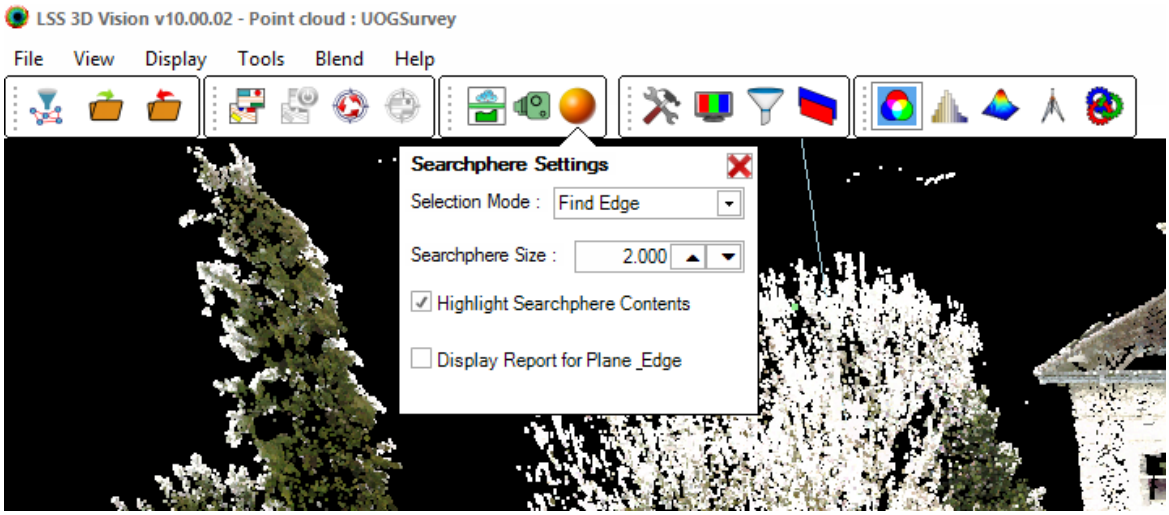
Point in Plane	Iterations	Standard Deviation	Mean Perp Dist	Max Perp Dist	Min Perp Dist	Calculation Time (in ms)
749	5	0.0163300279213346	0.0299529885168947	0.0991279654679562	1.28494790732519E-06	7
471	1	0.0131519313930755	0.0303854905493502	0.0996500871845009	1.06706154731568E-06	7
1602	5	0.0112728329677718	0.0142874626013137	0.107048796238697	1.20313829499261E-07	7
2351	1	0.0262169056605191	0.0142770644405625	0.10703628930301	9.84298812669 89E-09	7

Figure 5.20 This report is generated while the Edge/Edge stream option is in operation and updates in real-time

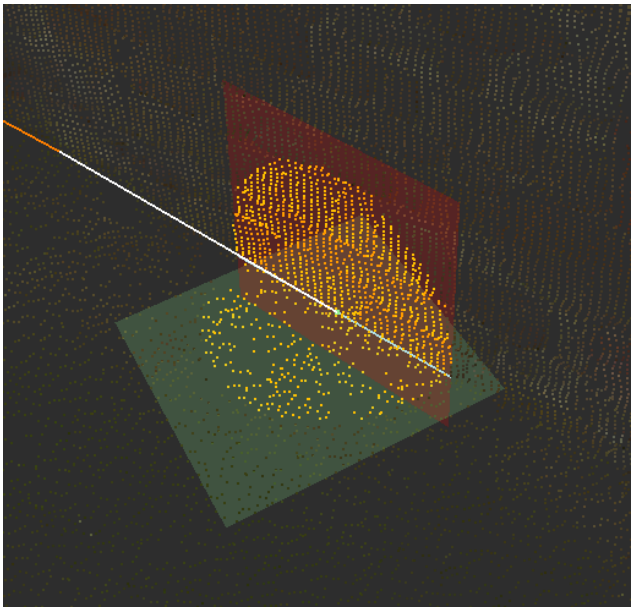
5.4.2 System operations

This section describes the commands in the 3DVision software of LSS. For the Edge option, the selection mode is selected as “Find Edge”, and a search sphere size is set. The setting of size and search sphere mode is shown in Fig 5.21 (a), with an example in Fig 5.21(b).

The user needs to select an appropriate search sphere size because the calculation is based on the points inside the sphere. Therefore, an appropriate size must be used to identify and extract the feature. For example, to find the top of a kerb, the sphere size is best kept below the height of the kerb itself to avoid points from outside the feature weighting the results.



(a)



(b)

Figure 5.21 (a) Select Find edge and Edge stream options from the dropdown menu of selection mode and set the required Searchphere size (b) Example of finding edge between wall and ground. The two colours represent defined best-fit planes, and the white line represents the edge with a green dot in the centre (edge point)

For finding edge streams, the selection mode is selected as “Edge Stream.” And search sphere size is set similar to the edge. In addition, the edge stream option has a separate list of options shown in Fig 5.22.

Algorithm Settings – The proposed algorithm in the commercial environment allows the users to set the parameters to find the best results in different data sets. The recommended or default parameters are shown in Fig 5.22; these settings could be changed according to the point cloud data used. This algorithm's feature has practical value for users, providing flexibility in different data sets. For example, a building wall data set differs from a quarry data set.

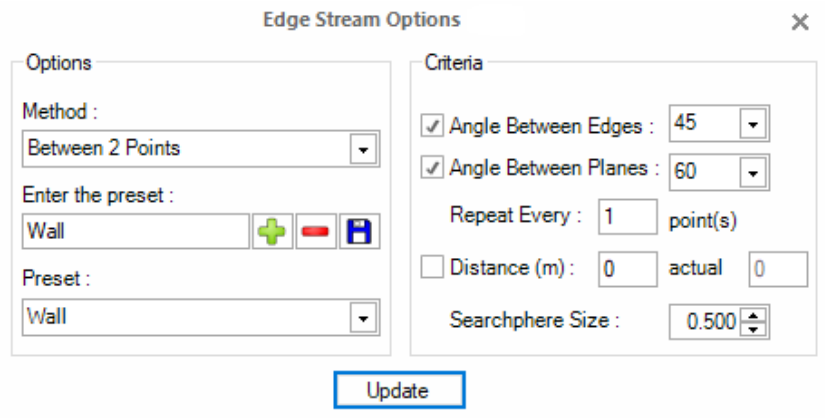


Figure 5.22 User-controlled options for the proposed algorithm in 3DVision

Setting Parameters – Figure 5.22 demonstrates the parameters that could be set according to the point cloud data used to detect the edge sects and edge stream. The first two parameters: (a) the angle between two planes that define the edge and (b) the angle between two consecutive edge sects is most important as it gives the flexibility of detecting different features (like the wall, kerb, slope, etc.) by manipulating these two parameters. Both parameters are used as stopping criteria for the proposed algorithm. The angles can be set to 30, 45, 60 and 90 degrees. The reason for choosing these angles was that they cover all the edge features to be detected in typical urban and quarry point cloud data. The third parameter (repeat every) gives the flexibility to obtain the edge points in the stream at a defined number. The fourth parameter provides the flexibility to obtain the edge points according to the set distance (metres). The fifth parameter is the search sphere size (metre), which plays a significant role in edge detection. The sphere size should be set according to the feature. The distance depends on the search sphere size; therefore, the actual distance between edge points is also shown in Figure 5.22.

5.4.3 SDE

The software development environment (SDE) used for the proposed algorithm is Microsoft Visual Studio 2022 version 17.4.4, the language is C# (pronounced as C sharp) version 10 and .Net Framework 4.8. The algorithm is implemented in the backend, which users access as the front-end interface, as shown in Fig 5.23.

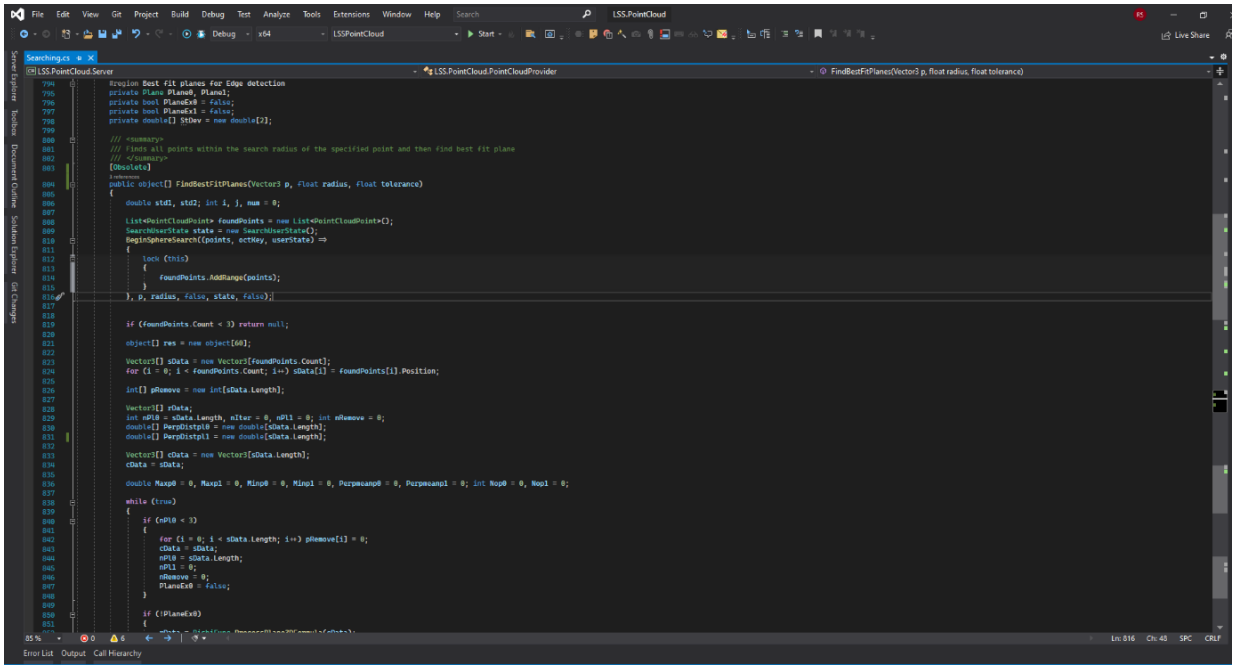


Figure 5.23 Visual Studio 2022 used for implementation of the proposed algorithm

5.4.4 RealWorld Scenarios

This section demonstrates how effectively the proposed algorithm detects edges in the presence of gaps, shadows, and obstacles. The challenges of existing methods are highlighted in Section 5.2. The important concern in civil engineering and surveying is reliably identifying gaps or obstacles in the point cloud data. When the scanner collects the point cloud data, the point density varies. The point density is the number of points collected per unit area. The area near the scanner usually has a high point density, and the area further from the scanner has a low point density. Higher point density means more points per unit area, which will help more accurate edge detection. However, the scanned points with low point density make it very

difficult to detect edges because of gaps or shadows in the data. The missing point cloud data could either be because of the shadow of an object or because the scanner density is decreased as the distance increases, creating a gap between the point cloud data. The other concern is the presence of obstacles. These obstacles could be part of urban data such as road furniture, lamp post, marker pole, electric unit, bins on the road, vehicles and vegetation.

5.4.4.1 Gap Shadow and Missing Point Cloud Data

Figure 5.24 illustrates the presence of low point density and a gap because of missing point cloud data. Edge detection could be challenging due to missing points, gaps, the reflection of points and scattered data (presence of trees). Therefore, relevant sanity checks are important to overcome these problems. An example of such data is shown in Fig 5.24. To overcome such problems, calculate the angles between two best-fit planes and the edges. For example, in Fig 5.24 (b), all the two best plane angles are calculated as the edges detected are along the wall and the path using the edge stream. If angles are greater than the average angles computed, then the edge stream detection is stopped—the criteria to effectively identify when the edge stream identification stops on its own.

(a) (b)

Figure 5.24 The point cloud data (a) shows the edge is difficult to identify because the second plane data is missing (b) shows a whole circle of data is missing

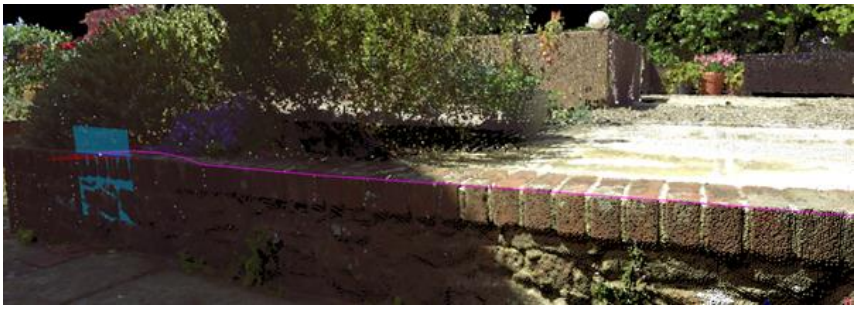


Figure 5.25 Example of the edge stream affected due to the presence of trees and shrubs

The other typical problem in the point cloud is the presence of vegetation. Figure 5.25 demonstrates a situation where vegetation, such as shrubs and small plants, can cause problems while detecting the edge. The real-time search sphere can trap vegetation and detect an inaccurate edge in these conditions. To avoid or prohibit false edge point results, the angle criteria play a very important role. The stopping criteria are the angle between two planes and the edges set according to the obstacles.

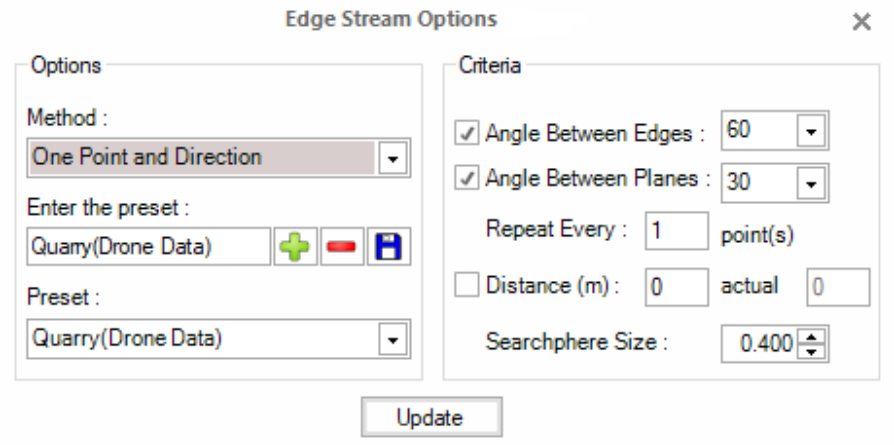
5.4.4.2 Obstacles Identification

The other common problem in urban point cloud data is various objects affecting edge detection. These objects include an electric pole, marker pole, electric box, bins, pillars on the wall etc., as shown in Fig 5.26. The solution for this problem is (i) calculating the angle between two best-fit planes and (ii) the angle between two consecutive edge lines.

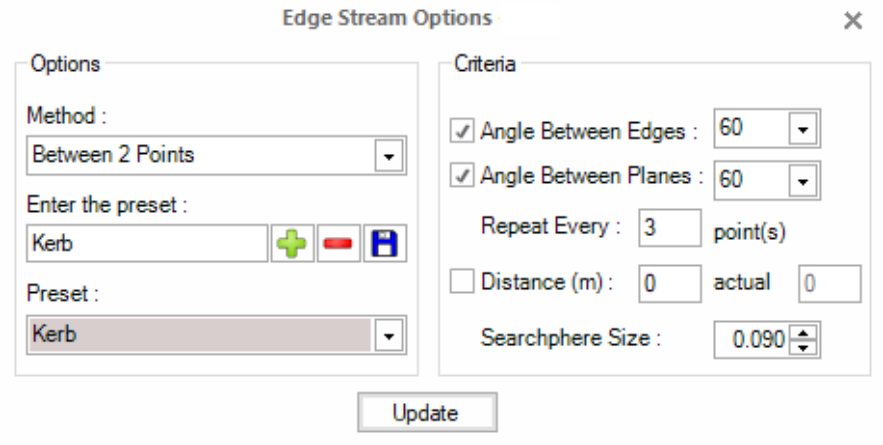
(a) (b)

Figure 5.26 Examples of obstacles such as (a) the lamppost and (b) the wall pillar

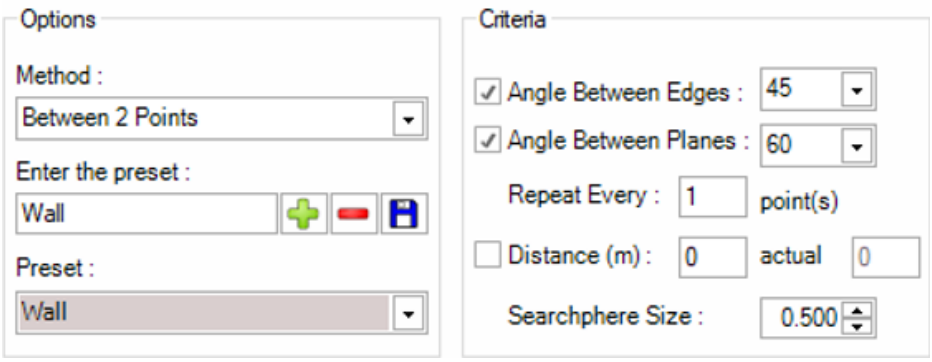
A few recommended settings have been tested on several point cloud data. The proposed algorithm detects obstacles and gaps for a wall when the angle between edges is set to 45 degrees, the angle between planes is set to 60 degrees, and the search sphere size is set to 0.5 metres. For kerb edge detection, the angle between edges and the angle between the planes is set to 60 degrees with a search sphere size of 0.09 metres. Setting a small search sphere size for a kerb is because the actual kerb size should be approximately similar to the search sphere size. Every three points are selected for a kerb detection repeat because the search sphere size is so small that it could detect hundreds of points. For quarry data (mineral extraction or mine), the angle between edges is set to 45 degrees, and the angle between planes is set to 60 degrees. The search sphere size is 0.4 metres; the repeat of every point is set to 1.



(a)



(b)



(c)

Figure 5.27 A tested predefined settings for obstacles and missing data according to the edge features in the point cloud

Together with the options and criteria, these settings are very useful for the users of LSS software in geographic information systems. The flexibility of the system allows them to extract edge and edge streams from difficult terrain to urban features in real-time. For example, the edges detected can be controlled by two values: 1) angle between edges and 2) angles between planes. In addition, the number of detected edge points can be filtered using the repeat every point option, and the distance between consecutive detected points can be set. Users save these criteria settings as personalised feature names for future use.

5.5 Evaluation

In this section, the proposed PCA-based algorithm is demonstrated and evaluated (validation) in terms of accuracy, robustness, breakpoints, classification of outliers and inliers, and computation speed using various point cloud datasets. The primary purpose of this section is to demonstrate the proposed algorithm's efficient computation time and an analysis of the accuracy of real-time point cloud data. The Edge stream is used for evaluation as it is a semi-automatic process and comprises edge sects.

5.5.1 Datasets for Evaluation: Point Clouds

The datasets used for the valuation and validation are generated using terrestrial laser scanners or LiDar scanners. For terrestrial laser scanners, the FARO scanner model FOCUS 350 is used. The focus scanner range is up to 350 metres for long-range measurements, and the measurement speed is up to 976,000 points/second. In addition, the focus has integrated GPS and Glonass, allowing position detection (Focus - FARO® Knowledge Base, 2016). The resolution of the scanner is up to 165 megapixels. For the LiDar laser scanner, Leica RTC360 3D Laser Scanner captures point cloud data for up to 130 metres with two million points/second measurement speed. In addition, the Leica scanner has multi-sensors GPS, compass, height sensor and dual-axis compensator (Leica RTC360 3D Laser Scanner | Leica Geosystems, 2018).

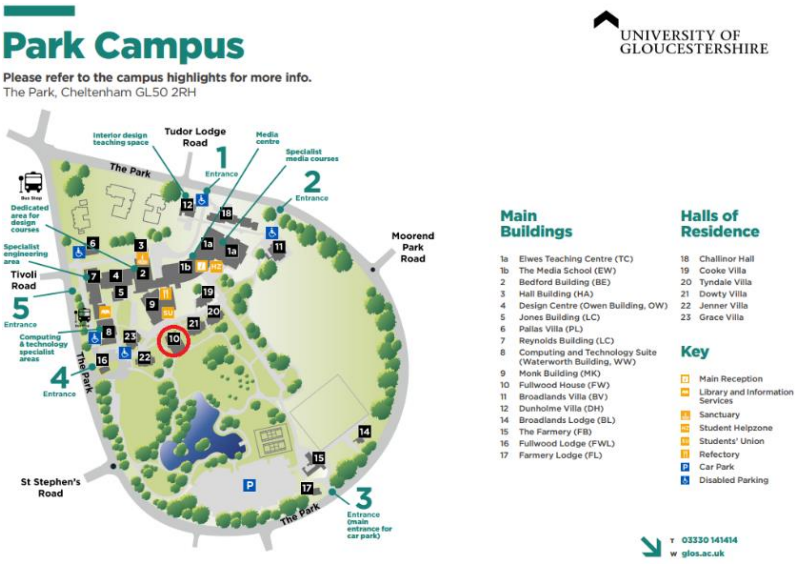


Figure 5.28 University of Gloucestershire Park Campus

Fig 5.29 illustrates five data sets that have been used to evaluate the proposed algorithm. Data sets are road, car park, church, quarry and University of Gloucestershire Park Campus data set (University data set). For this thesis, several point cloud data sets have been collected in collaboration with the University of Gloucestershire (UOG). The University data set was captured in front of Fullwood House of the Park Campus, as shown in Fig 5.28. The MTSL car park data set was captured in the front building of McCarthy Taylor Systems Ltd. (MTSL). The church data set was captured in Birdlip, Gloucestershire. The car park, church and

University data set were captured by the FARO laser scanner. The road and quarry data set were captured by the Leica laser scanner. The scanned datasets consist of points in 3D (x, y, z) along with each point's R, G, B and an intensity value. The point cloud data set description is as follows:

1. **Road data set** – an urban data set with features like tunnels, roads, marker posts, kerb etc. It has a total point count of 21.43 million.
2. **MTSL Car park data set** – a scanned car park with features like vehicles, buildings, roads and vegetation. It has 78.76 million points.
3. **Church data set** – a scanned church and surrounding vegetation. It has 257 million points.
4. **Quarry data set** – a scan of active quarry with 159 million points.
5. **University data set** – is captured in the University of Gloucestershire campus and has 580.94 million points.

(a)

(b)

(c)

(d)

(e)

Figure 5.29 Datasets used for evaluation of proposed algorithm (a) Road data set, (b) MTSL Car park data set, (c) Church data set, (d) Quarry data set, (e) University data set

5.5.2 Computation Parameters

This section implements the proposed algorithm for commercial software. The computation evaluation is based on algorithm settings, the performance of the proposed algorithm and default parameters.

Performance of Algorithm: The proposed algorithm has been tested on these five datasets. The algorithm's outcome in terms of performance is shown in Table 5.3. Furthermore, it is tested on the following point cloud data sets: (1) Church data set is a scanned church and surrounding vegetation. It has 257 million points, and the file size is 5.13 gigabytes (GB), (2) MTSL car park data set is a scanned car park with features like vehicles, buildings, roads and vegetation. It has 78.76 million points with 2.19 gigabytes file size, (3) Road data set is an urban data set with features like tunnels, roads, marker posts, kerb etc. It has a total point count of 21.43 million with 0.53 gigabytes file size, (4) Quarry data Set is a real-world quarry with stockpiles. It has 159 million points with 3.17 gigabytes file size, and (5) the University data set is the real-world campus with buildings, vegetation and terrain. It has 581 million points with 14.24 gigabytes file size.

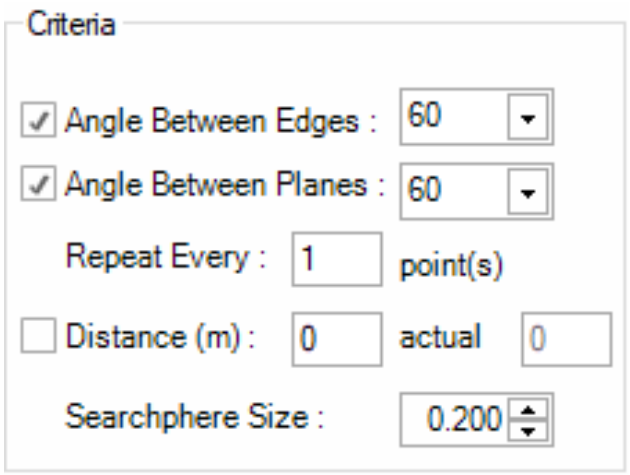


Figure 5.30 User-controlled criteria for the proposed algorithm in 3DVision

The parameters are different for different data sets. The points inside the search sphere depend on the selected size and point density. The sphere sizes in Table 5.3 are different as each data set has different features for edge detections. The size parameters are selected based on the detected feature to make edge sects evident. For example, the edge detected in the church data set was between the church wall and the ground; therefore, the size is set to 0.3 metres, which covers more area and is not very small. The wall and ground are almost perpendicular in the church dataset, the angle between planes is set to 90 degrees, and the angle between edges is set to 45 degrees as the wall and ground are almost flat. In this example, both stopping criteria are used for edge stream detection a) if the angle between planes is greater than 90 degrees and b) if the angle between the edges is less than 45 degrees.

In the car park data set, the edge feature is also the building wall and the road; therefore, the angle between planes is set to 90 degrees, the same as the church data set, and the edge angle is set to 30 degrees because of building wall beams.

In the road data set, because of various road furniture like a lamp post and a marker pole, the angle between the plane is set to 60 degrees, the edge angle is set to 45 degrees, and the search sphere size is set to 0.5. Both angles were set to 60 degrees in the university data set as the edge detection was inside a room to find the floor plan. The room has few obstructions and very high point count density; therefore, the search sphere size is set to 0.3 metres.

In the quarry data set, the angle between planes is set to 45 degrees, and the angle between edges is set to 60 degrees. The plane angle is less than the angle between edges as the edges are picked for a stockpile and break of slope. For these features, the search sphere size is set to 0.5 metres.

The proposed algorithm has been implemented to find edge sects and edge streams using the data with the parameter listed in Table 5.3. A difference in calculation time can be observed. The car park data with 4,489 points in 0.3 search sphere size takes 6 ms to find the edge compared to Quarry 380 points in 0.5 search sphere size takes 9 ms because of the space between the points. If the plane's surface points are not flat, the algorithm time is extended to fit the planes on the points.

Table 5. 3 Point Cloud data sets

Point Cloud	Average points in search sphere	Calculation Time in milliseconds	Angle between the edges	Angle between planes	Search Sphere Size	Distance in metres
Church	47,789	8	45	90	0.3	1.8 (2m)
Car Park	4,489	6	30	90	0.3	1.8 (2m)
Road	2,234	4	45	60	0.5	4.8 (5m)
University	16,262	14	60	60	0.3	1.8 (2m)
Quarry	380	9	60	45	0.5	2 (1m)

The algorithm is demonstrated in Fig 5.31 and Fig 5.32 as an example of edge stream detection. In Fig 5.31 example, the presence of the bin is an obstacle. Therefore, the stopping criteria are used to stop and exclude the noise points of obstacles. The parameters used in this example are 0.2 mm of search sphere as the points are dense, and the angle between edges and planes is set

to 60 degrees as there can be many obstacles. Similarly, in Fig 5.32 example, to determine the edge on the curve, the angle between the edge is set to 45 degrees and angle between planes is set to 60 degrees and the search sphere size is set to 0.5 mm as from two surfaces to find planes one surface has low-density points.

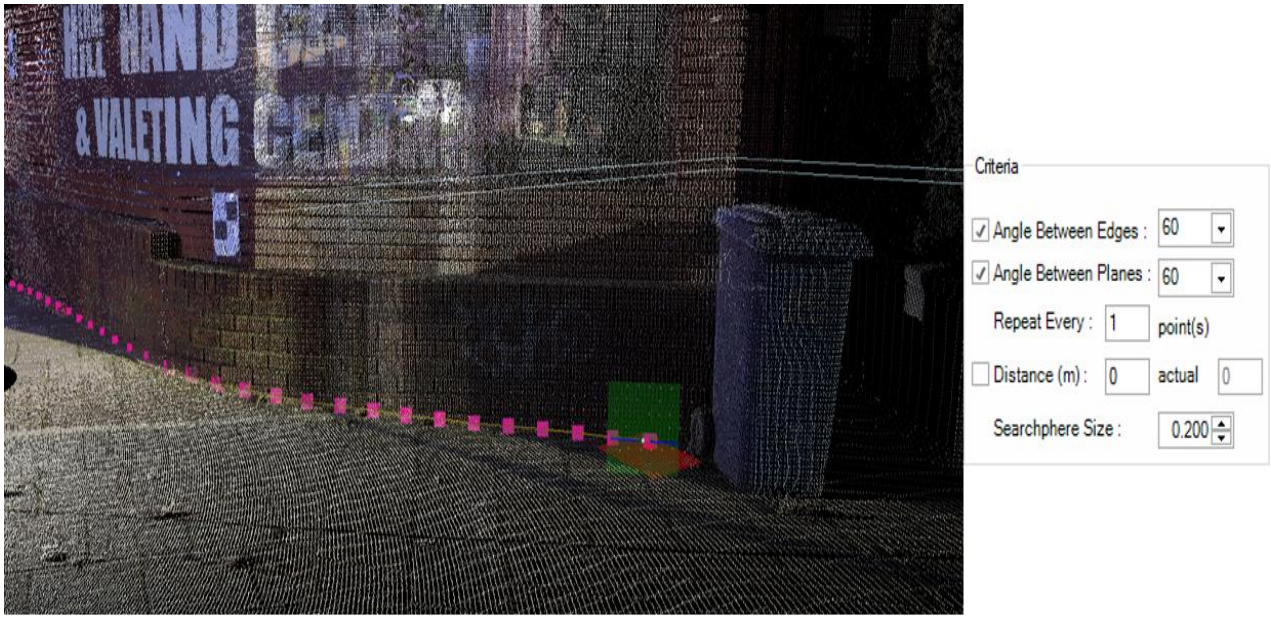


Figure 5.31 Stopping criteria for an obstacle

Figure 5.32 Edge Stream detection along a curve of the bridge

5.5.3 Comparative Analysis

The proposed algorithm is evaluated based on accuracy, breakdown points, robustness, computation speed, and classification into outliers and inliers from collected point cloud data sets. The proposed algorithm is compared with different edge detection algorithms RANSAC, MSAC, RSPCA, uLSIFand qSp (Nurunnabi, West and Belton, 2015a, 2015b). A planar surface is fitted on the sampled data with other methods to evaluate the proposed method's performance with other edge detection algorithms. To evaluate the plane's accuracy, the angle θ is calculated between the plane with outliers and the resultant plane without the outliers.

Nurunnabi, West and Belton (2015a) simulated 1000 sets of 50 synthetic 3D points with 10-20 % outliers created by the Gaussian normal distribution algorithm for fitting a regular plane. After which various descriptive measures of mean, minimum, maximum and Standard deviation of θ have been calculated. The proposed PCA-based algorithm is tested in two sets of 10 simulations to calculate mean, minimum, maximum and Standard deviation of θ similar to the provided data by Nurunnabi, West and Belton (2015a) to compare effectively. The proposed algorithm is evaluated on point cloud data set by generating real-time θ between the plane and without the outlier. The θ is calculated by the following equation,

$$\theta = arccos(\hat{n}_1 \times \hat{n}_2) \tag{5.25}$$

where $\hat{n}_1$ and $\hat{n}_2$ are two unit normals from the fitted planes with and without outliers. As the data set on which the algorithm is implemented is not synthetic, the number of points to fit the plane may vary. Further, based on the values of average θ for all the algorithms, the comparison of the accuracy and robustness is easily identified.

A regular plane is fitted on synthetic data in other methods. For the proposed algorithm, only the first (from the two best-fit planes detected) best-fit plane is used to evaluate and compare with other methods. Angle θ is calculated as the angle between the first plane and the best-fitted plane. In other words, the angle between planes with and without outliers. The real point cloud data simulation and theta calculation are shown in Table 5.4 and Table 5.5.

5.5.3.1 Test Set with Outliers

Two data sets have been used to analyse and compare with different algorithms. The datasets have been selected from 5 point cloud data sets mentioned in Section 5.5.1.

The two sets are:

1) The first data set has a cluster of points sampled using a search sphere with minimum or no outliers and noise in the data, shown in Table 5.4.
2) The second data set has a cluster of points with outliers shown in Table 5.5. The outliers present in the sampled data are vegetation, people passing by, and road furniture.

The proposed algorithm is tested and compiled on a computer with hardware and software configurations as follows: Intel Core i7 processor running at 2.60 GHz using 16 GB RAM running on 64-bit Windows 11 version 22H2. The proposed algorithm was also tested on a virtual machine and another machine with Windows 8. The timings were same and are not affected by the Windows version.

Table 5. 4 First set with uneven sampled data (5-10% outliers)

	NOP Plane 1	**NOP Plane 5**	**Theta (Cos)**	**Theta** $\boldsymbol{\theta = arccos(\hat{n}_1 \times \hat{n}_i)}$	**Speed in ms**	**Outliers**	**Inliers**	**Accuracy**
1	1926	179	0.99998611	0.301987701	2	1747	179	90.70612669
2	7665	309	0.99962216	1.575089629	6	7356	309	95.96868885
3	1399	13	0.99998855	0.274183167	1	1386	13	99.07076483
4	1648	73	0.99962931	1.560114531	2	1575	73	95.57038835
5	1375	55	0.9999945	0.19002869	1	1320	55	96
6	1485	107	0.999986	0.303181121	2	1378	107	92.79461279

7	2088	92	0.9999547	0.545366391	2	1996	92	95.59386973
8	20595	755	0.999999999	0.002562345	4	19840	755	96.33406167
9	1188	49	0.999554097	1.711094386	1	1139	49	95.87542088
10	2437	71	0.999971032	0.436111897	1	2366	71	97.08658186

Table 5. 5 Second set with uneven sampled data (50-55% outliers)

	NOP Plane 1	**NOP Plane 5**	**Theta (Cos)**	**Theta** $\boldsymbol{\theta = arccos(\hat{n}_1 \times \hat{n}_i)}$	**Speed in ms**	**Outliers**	**Inliers**	**Accuracy**
1	4209	93	0.99997776	0.38212533	8	4116	93	97.79044904
2	5519	113	0.99982064	1.085192839	4	5406	113	97.95252763
3	1957	52	0.99941825	1.954460433	3	1905	52	97.34287174
4	5561	59	0.99909865	2.432859272	5	5502	59	98.93903974
5	1203	32	0.99957931	1.662010429	1	1171	32	97.33998337
6	3783	163	0.99917771	2.323699148	4	3620	163	95.69125033
7	2127	45	0.97348845	13.2226765	1	2082	45	97.88434415
8	32613	1212	0.99967288	1.465558027	16	31401	1212	96.28369055
9	11475	313	0.99996876	0.452891608	7	11162	313	97.27233115
10	25921	486	0.99913442	2.384089572	20	25435	486	98.12507234

5.5.3.2 Types of Data

The planes fitted by different algorithms (PCA, RANSAC, MSAC, RPCA, uLSIF, qSp) and the proposed PCA-based method are shown in Table 5.6 and Fig 5.33. The following results have been taken from Nurunnabi, West and Belton (2015a) as a case study to evaluate the proposed algorithm against the other algorithms. All the existing algorithms have been evaluated by fitting the plane on the synthetic data set, but the proposed algorithm's fitted plane is tested on the real point cloud data. The real data are more challenging compared to synthetic data sets due to their large size. However, for comparison, the number of points is approximately the same. Given that the proposed algorithm is finding the edge in real-time compared to the one-off implementation by Nurunnabi *et al*.. The data sets mentioned in Section 5.5.1 are used that have several edge features than synthetic data. Therefore, for this thesis, the point cloud data used for evaluation is the same data used by surveyors and engineers regularly.

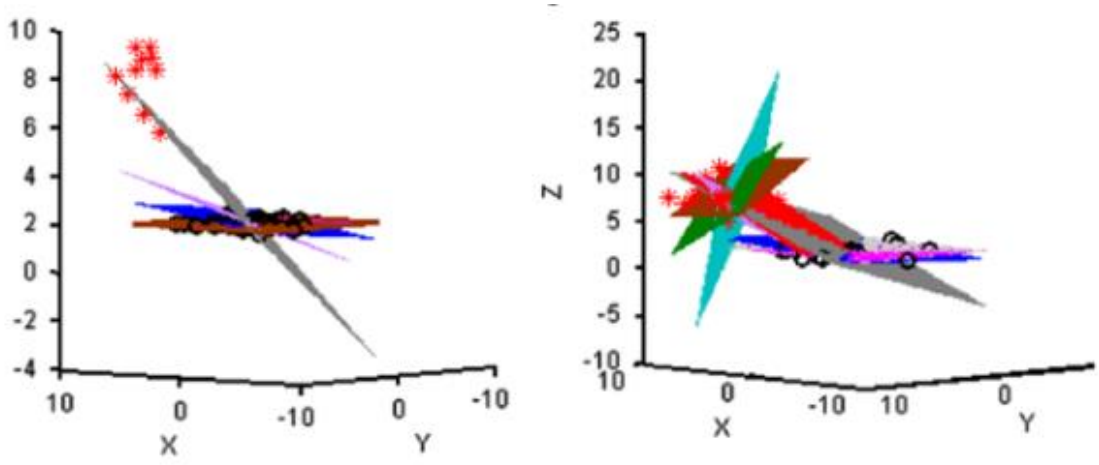
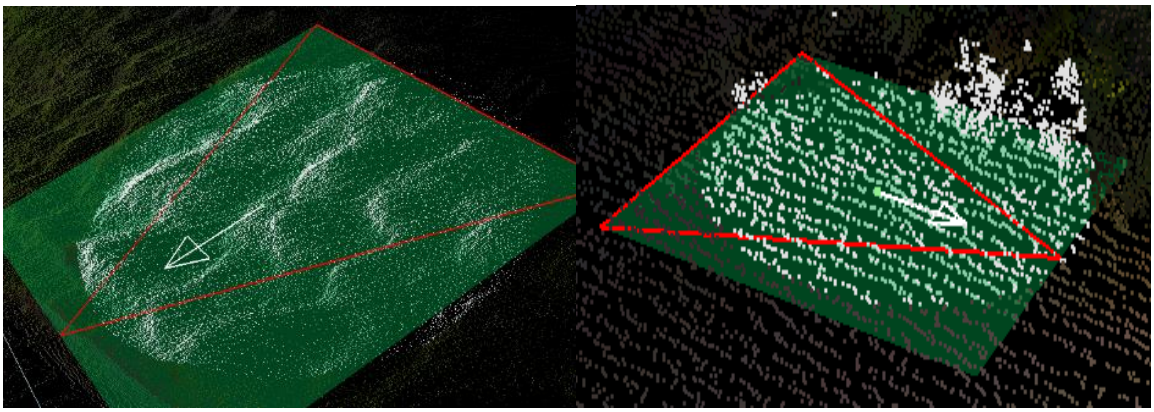


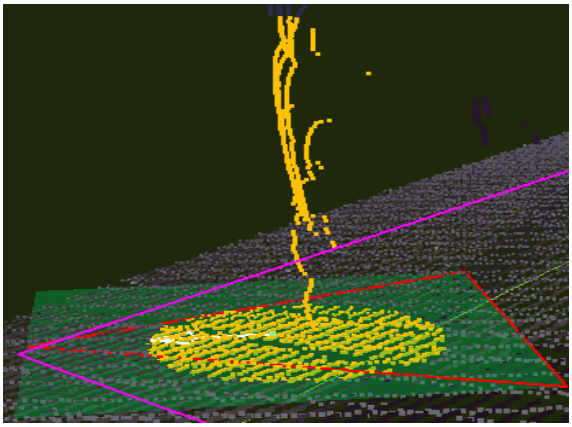
Figure 5.33 Nurunnabi, West and Belton (2015a) demonstrated a plane fitted by different algorithms with 20% cluster outliers. Planes: grey - PCA, red – RANSAC, green – MSAC, blue – uLSIF, pink - qSp

For the comparison, the size of the sphere has been set to 1 metre in the real-world point cloud. The reason for choosing a 1 metre size is that the number of points inside the sphere can be matched with the synthetic data. The angle θ between planes is calculated in degrees in two types of data sets. The first data set has a cluster of points sampled using search sphere (a)

without any outliers in the data, and the second set of data has a cluster of points (b) with outliers. The second data set has a plane fitted on sampled data using the proposed algorithm. An example of such data is shown in Fig 5.34.



(a) (b)



(c)

Figure 5.34 The fitted plane is displayed in green, and the number of points sampled is highlighted in white. A plane is fitted (a) on an uneven surface, (b) on the shrub and floor, (c) on part of a tree trunk better to provide the photos of three objects, then point clouds, then fitted planes with the point clouds.

The planes fitted by different methods are shown in Fig 5.34, in which the plane from the robust methods is only the one with the perfectly fitted plane. The planes fitted by the proposed algorithm are shown in Fig 5.34

(a) the presence of 5 to 10% outliers on real point cloud data with an uneven surface,

(b) shows the presence of vegetation such as small plants or shrubs when sampling the data with 50-55% outliers and

(c) shows the presence of an obstacle, such as a person or a marker pole, with 50% outliers.

Mean maximum, minimum, median and standard deviation are calculated from synthetic and real point cloud data simulation. The results, as shown in Table 5.6, indicates that the proposed algorithm has lower values than others. The PCA method has the largest measures of all the other algorithms. The proposed algorithm has a minimum quartile range of 0.253 (QR = 3^{rd} – 1^{st}). RPCA and the proposed algorithm result in better robustness than RANSAC, MSAC and uLSIF.

Table 5. 6 Algorithm's measures in angle

Methods	Mean	Min.	Max	Median	SD	QR
PCA	31.038	3.807	52.690	34.973	3.973	4.758
RANSAC	1.186	0.000	6.367	0.832	1.167	1.618
MSAC	1.1485	0.000	7.378	0.687	1.215	1.605
RPCA	0.694	0.022	2.698	0.599	0.489	0.550
uLSIF	4.3235	0.562	17.938	5.731	2.304	2.769
qSp	14.377	0.017	43.968	30.262	15.302	30.168
MCMD_MD	0.518	0.008	2.18	0.451	0.377	0.411
Proposed Method	0.689	0.002	1.711	0.369	0.55	0.253

The proposed algorithm has measured angles that are very close to RPCA, as shown in Table 5.6. However, calculating the angles (minimum, maximum, median, and quartile range) indicates that the proposed algorithm is better than RPCA. PCA has the worst results as it is sensitive to outliers. So, if a breaking point is calculated for PCA, it would be 0%, as it will stop even in the presence of just one outlier. After PCA, uLSIF results in 4.3 degrees mean and 17.9 degrees maximum. uLSIF and qSp also did not perform well. The maximum, median, standard deviation and quartile range are very high for qSp. MSAC and RANSAC results are almost equal. MCMD_MD performed well as compared to the proposed algorithm in terms of standard deviation however the method was not applied in real-time or on a large point cloud data set. The proposed algorithm has the best result minimum quartile range, which means that the proposed algorithm produces more robust results than other algorithms. Also, the proposed algorithm was implemented on real-world point cloud data and in real-time, which is more challenging. The next sections will evaluate the proposed algorithm accuracy analysis and speed for simulated data. The plotting of the two sets of simulations based on Table 5.4 and Table 5.5 is represented in Fig 5.35. The first set of sampled data had 10-15 % outliers compared to the second set of data with more than 50-55 % outliers.

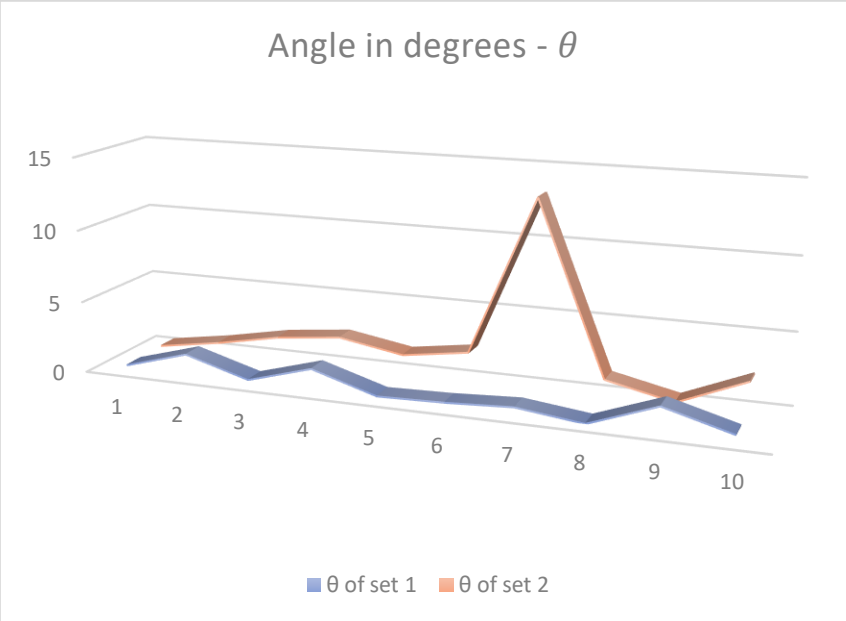


Figure 5.35 Two sets of data simulations are represented in blue and orange.

5.5.3.3 Evaluation of Accuracy in the Presence of Outliers

This section investigates the performance of all methods mentioned above and classifies them as inliers and outliers. The total points from the search sphere are classified as outliers or inliers. The data from Nurunnabi, West and Belton (2015a) are used for the accuracy analysis between the proposed algorithm and other methods. The accuracy of the algorithms is defined as

$$\text{Accuracy} = \frac{\text{TN} + \text{TP}}{\text{Total number of points}} \times 100 \tag{5.26}$$

where TP is true positive points identified as inliers and TN are true negative points identified as outliers

$$True\ Negetive = \ points\ identified\ as\ outliers \tag{5.27}$$

$$True\ Positive = \ \text{points identified as inliers} \tag{5.28}$$

where true positive is the number of points correctly identified as inliers in percentage, and a true negative is the number of points identified as outliers in percentage.

In the first set of data with 5-10% outliers, the accuracy ranges between 90 – 99% of the total points. The minimum is 90%, and the maximum is 99%.

The second data has 50 -55% outliers, and the accuracy range lies between 97- 98% of the total points. The minimum is 97%, and the maximum is 985. The data is shown in Table 5.4 and Table 5.5.

Therefore, the proposed PCA-based algorithm accurately rejected more points when the data sampled uneven surfaces to fit the plane best. As a result, the fitted plane by the proposed algorithm identifies the best-fit plane on every data despite the outliers. The results at different outliers present in the data demonstrate that the proposed algorithm is highly accurate.

5.5.3.4 Speed Analysis

The processing time in the point cloud is a very important aspect of any computing algorithm. The important advantage of the proposed PCA-based algorithm is computation time. The speed analysis is performed on the two data sets mentioned in Section 5.5.4.2. A regular plane is fitted through the data sets, and the calculation time to find the best-fit plane is represented in milliseconds in Table 5.7. Table 5.7 demonstrates the first set of data with 10% outliers and the second set representing more than 50% outliers.

Table 5. 7 Plane detection in Seconds

Methods	Detection Time in Seconds	
RANSAC	0.0934	0.3640
RPCA	0.7990	0.7937
uLSIF	0.0405	0.0409
MCMD_MD	0.0054	0.0395
Proposed Algorithm	0.001	0.02

The minimum time is one millisecond, i.e., 0.001s, and the maximum is 20 milliseconds, i.e., 0.02s. The comparison is performed with different algorithms, with 10% outliers. The resultant time of the proposed algorithm is 0.001 seconds, less than the other algorithms. RPCA has a maximum time of 0.799 and MCMD_MD in 0.005s, uLSIF in 0.0430s and RANSAC in 0.0734s (Nurunnabi, West and Belton, 2015a). The resulting time in seconds for 50 to 55% outliers of the proposed algorithm averages 0.02 milliseconds less than MCMD_MD of 0.0395. The other algorithms have similar results RANSAC in 0.3640 and ulSIF in 0.04909. RPCA has a maximum time of 0.7937, similar to 10% outliers. The computer used to run the proposed algorithm specification is as follows: Intel Core i7 processor running at 2.60 GHz using 16 GB RAM running on 64-bit Windows 11 version 22H2.

5.5.4 Accuracy Evaluation

The proposed algorithm accuracy is investigated by calculating the error distance measure, as listed in Table 5.8. The proposed algorithm has calculated two measures using the cross-section of sampled data inside the search sphere to analyse the resultant edge point.

The two measures are:

- the actual centre point (edge point) of the search sphere.
- the centre point generated by the proposed algorithm.

In Fig 5.36, there are two sections: the upper section is 3D point cloud data, and the lower section is the cross-section of the 3D data inside the green cube. The projection of the 3D green cube on the Y-X plane is represented in Fig 5.36, lower section. The green dots represent the edge points in the upper section (3D point cloud).

The proposed algorithm identifies all the green dots as an edge stream. In the lower section, the pink circle represents the search sphere. The red point of the pink circle represents the actual centre of the sphere, and the green dot inside the pink circle represents the edge point calculated by the proposed algorithm.

For evaluating the accuracy of the calculated (green point) by the proposed algorithm shown in Fig 5.36, a set of edge points have been tested on the real point cloud data to find the error distance Δd between the edge point generated by the proposed algorithm and the cross section's actual centre inside the search sphere. Δd represents the distance between the two points.

The edge point is a calculated point that may or may not necessarily be an actual 3D point but will best fit the available data. Table 5.8 lists five points of real point cloud data that compare the edge and actual centre points on the cross-section.

ΔX, ΔY and ΔZ are calculated, which is the difference between the two point's X, Y and Z components. The minimum distance is 0.038549, and the maximum is 1.0535, with ΔY of 0.019 and 0.053, respectively.

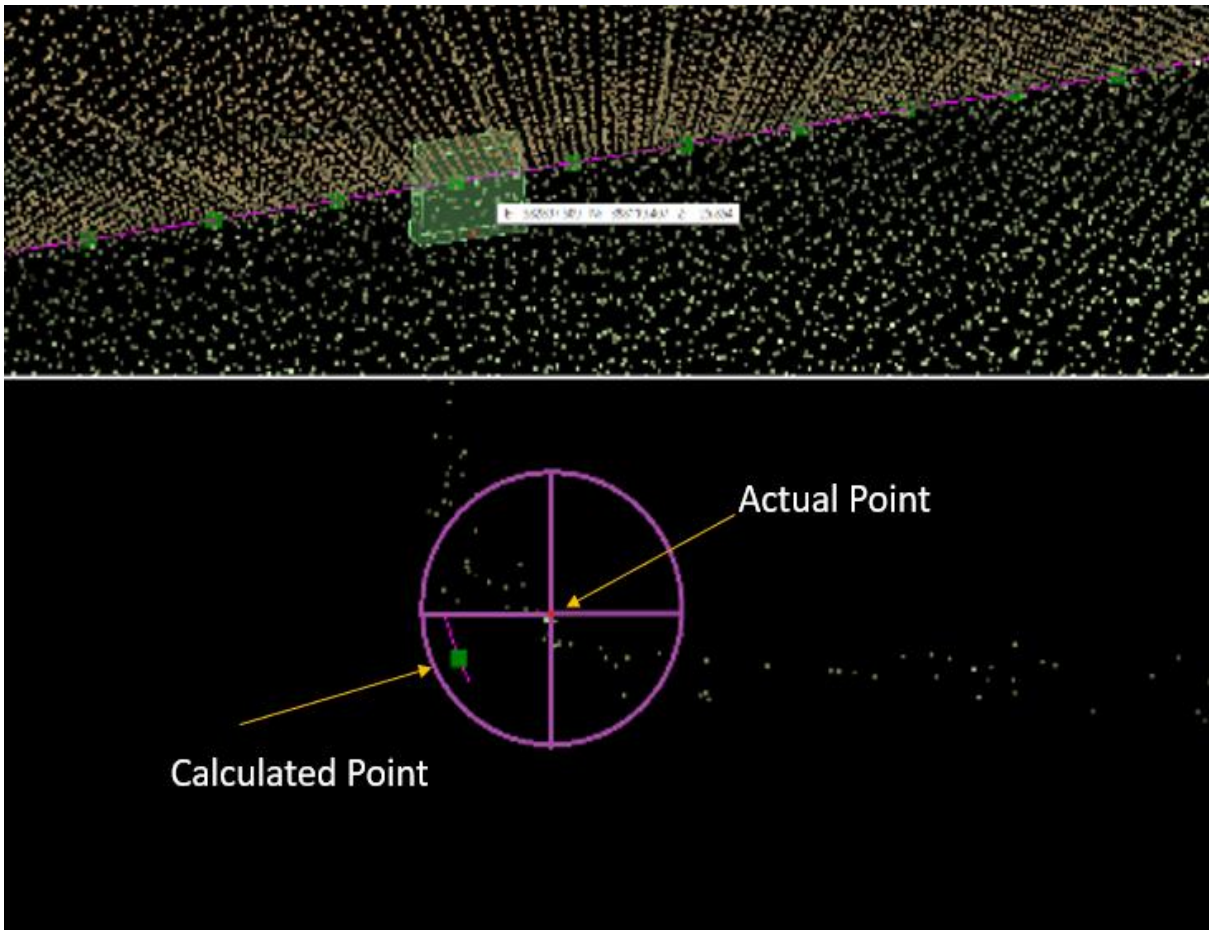


Figure 5.36 The upper section of the image describes the 3D point cloud, and the lower section demonstrates the 3D point cloud data cross-section. The green dots represent the edge points calculated by regression of the proposed algorithm, and the pink circle demonstrates the centre of the search sphere

The results shown in Table 5.8 display that the proposed algorithm efficiently best fits the planes on the data points to derive the edge points. Furthermore, the edge point is not influenced by the gravity of the points inside the search sphere. Therefore, the proposed algorithm proves the accuracy and correctness of the point detected.

Table 5. 8 Measures to Calculate the Accuracy of Edge point Detected

		Point 1	Point 2	Point 3	Point 4	Point 5
Actual centre point of Y-X cross-section plane inside the search sphere	X	382897.509	382897.668	382897.901	382898.081	382898.496
	Y	398119.407	398119.707	398120.073	398120.403	398121.129
	Z	25.854	25.881	25.881	25.885	25.905
Edge point generated by the proposed algorithm	X	382897.454	382897.662	382897.864	382898.069	382898.463
	Y	398119.343	398119.688	398120.047	398120.339	398121.076
	Z	25.848	25.848	25.873	25.881	25.905
Error	ΔX	0.055	0.006	0.037	0.012	0.033
	ΔY	0.064	0.019	0.026	0.064	0.053
	ΔZ	0.006	0.033	0.008	0.004	0
Error in distance Δd	Δd	0.084599	0.038549	0.045924	0.065238	1.053517

5.6 Chapter Summary

In this chapter, the algorithm applies a robust and statistical approach for identifying the edges between two surfaces in 3D point cloud data. The proposed algorithm is robust, efficient and accurate. The proposed algorithm first applies the best-fit plane to most of the data. Then, each local point is calculated to best fit the plane from the plane's point of interest. Secondly, each point is identified as an outlier or inlier based on its distance from the plane. Results show accuracy and fast computation times. However, the computation time is relatively greater when outliers increase. Also, the search sphere size plays an important role in Edge stream detection. The results are performed on real point cloud data and the algorithm is performed on large data sets. The advantages of the proposed algorithm are compared to other algorithms include (i)

the accuracy to identify the outliers, (ii) efficiency when applied on a large data set, and (iii) the faster computation time than PCA, RPCA, RANSAC, uLSIF and qSp, (iv) easy to use, (v) simplicity. The proposed algorithm could further be used for feature detection, segmentation and region-growing algorithms.

Edge detection can be further developed to find the edges in the whole point cloud automatically; the future of edge detection can also be extended to corner detection, where instead of two planes, it identifies three planes as it is very difficult to find a corner with the two planes.

One of the dependencies of the proposed algorithm is the user. The stopping criteria depend on the angle of planes and edges that have been set. The angles set completely depend on the user's experience using the algorithm for the obstacles and missing data in their designated point cloud.

Chapter 6 A New Voxel-Based Algorithm for Cylindrical Feature Detection in Urban Point Clouds

6.1 Introduction

Among different datasets, one of the essential ones is the urban point cloud. With technologies, there has been increased demand within the surveying industry to process these data automatically. Urban point clouds are detailed and contain various objects with meaningful geometrical and physical information that must be extracted. For example, buildings, vehicles, vegetation, street furniture, ground, road markings, and maintenance holes are all objects requiring identification. In addition, these objects have disparities: noise, size, incomplete structures, different point densities, holes and gradients. Therefore, the automatic extraction of these objects from 3D point clouds for urban cartography is extremely attractive and high in demand because it tremendously decreases the resources needed to analyse the data for subsequent uses in city management and planning functions (Lam *et al.*, 2010; Sahin *et al.*, 2012).

Among various urban objects, trees play an important role. Therefore, a tree survey report is beneficial in different ways. For example, a digital model of the street tree could play an essential role in environmental analysis or tree management, hazard analysis of a nearby building, urban landscape beauty, etc. Furthermore, poles are also important street furniture. These pole structures include marker poles, lamp posts, square poles, traffic signs and streetlights documented as road inventory (Landa and Ondroušek, 2016). Therefore, accurately detecting the location of the trees and pole structures is an essential aspect of the ecological and surveying field. In forestry, the volume of trees is often used to determine the health of the trees. In urban data, trees and poles are often surveyed for risk and hazard analysis, damage control, and logging accounts for tree growth.

This chapter proposes a new automatic voxel-based detection algorithm for cylindrical features such as tree trunks and pole-like objects in urban point clouds. The proposed algorithm detects trunks and pole objects with different shapes, girth, slopes, and low-density points. The algorithm consists of six stages:

1) Extraction of ground and non-ground points
2) Voxelization
3) Removing the ground points and detecting the seed layer
4) Clustering of the neighbourhood
5) Detection of cylindrical features
6) Classification of detected objects into trunk, pole and other features

The algorithm has been tested on different data sets. The analysis and evaluation of the related work for detecting the pole-like objects in urban data are given in Sections 6.2.1 - 6.2.4, and the detection of tree trunks in urban or forest data sets is provided in Section 6.2.5. A new proposed algorithm (voxel-based algorithm for cylindrical object detection) is presented in Section 6.3, with the classification of the objects in Section 6.3.7. The implementation of the proposed algorithm is presented in Section 6.4 with real-world scenarios encountered by surveyors and civil engineers. The evaluation of the proposed algorithm is presented in Section 6.5. Discussion is presented in Section 6.6 of the proposed algorithm's advantages and limitations. Finally, the chapter summary is presented in Section 6.6.

6.2 Analysis and Evaluation for Existing Methods: Pole-like Objects and Trees

Processing point cloud data consists of filtering, segmenting and transferring the information. The segmentation algorithms are widely used to divide the point cloud data into groups or regions to extract meaningful information belonging to a feature (Tombari, Cavallari and Stefano, 2016). Besides segmentation, shape and semantics-based methods are frequently used to detect trees and poles in urban data. In this Section, the existing methods are analysed and evaluated in detail, starting with pole-like structure detection by segmentation, semantics, slicing and shape-based, followed by individual tree detection in urban and forest data sets.

6.2.1 Segmentation and Clustering Methods (Model fitting)

This section discusses and presents the existing object detection methods based on segmentation. These segmented groups are clustered, or a model is fitted to extract the features from the point cloud. Lehtomäki *et al.* (2010) developed a method to detect pole-like objects in a road environment. The algorithm starts by segmenting the 3D scan line individually. Short distance adjacent points on the scan line are gathered in one sweep. The isolated poles (away from building vegetation) are captured by one sweep forming a profile point group. The possible pole-like structures are exacted by segmenting each profile into point groups. The *long* groups are discarded, and only *short* groups are considered. Next, the *short* point groups are clustered by searching groups in the horizontal plane to find the candidate pole cluster. A group is selected as a seed group on each profile.

All the groups adjacent to it are compared. If a group is found above a seed group, they become part of the same cluster. The clustering could divide the clusters into several sub-clusters, and therefore, the next phase of the algorithm merges clusters. The merging criteria are clusters that are pole-like and vertically oriented. The clusters classified as poles and non-poles are based on shape, length, orientation and point density. The pole candidate must satisfy the following criteria 1) the cluster should be along the main axis 2) a minimum of three sweeps in a cluster 3) pole-like shape 4) the main axis should be close to vertical 5) the presence of not too many points around the cluster in the local neighbourhood. The method can find 77.7% of poles and correctness of 81.0% in suburban areas. The disadvantage of using the scan line segmentation is that if there are outliers or shadows in a sweep, the point groups are split into many small groups. Therefore, poles with vegetation behind, parked cars, scattered points and oblique poles cannot be detected.

Pu *et al.* (2011) presented a method for recognising the road structures captured by mobile laser scanners. The framework starts by dividing the point cloud data into road sections. Then, each road is segmented with the surface growing algorithm based on planar seed surface detection in Hough space. Next, the parts are classified as ground surface, on-ground, and off-ground objects. The on-ground points are further used for detailed feature recognition. Next knowledge-based feature segmentation is applied to the segments based on geometric attributes as

1) size – length, width, height, area or volume
2) shape
3) orientation
4) position
5) colour
6) topological relationships – angle and intersect.

Objects with planar features are detected based on classes such as geometric fitting, recognising rectangles, circles and triangles. The pole-like structures are identified by a percentile-based algorithm that slices vertically in 2D. A rectangle is fitted in each slice. The rectangle's centre point and diagonal length are compared to all the slices in percentile. The difference in neighbouring slices is checked if the centre position and diagonal length are within the threshold and the slice is part of the segment. Finally, if the count reaches a certain number, the object is classified as pole-like, and if the rectangle diagonal exceeds the maximum length, the slice is discarded. The objects on the ground are identified, such as traffic signs, trees, building walls and barriers. The results show that the recognition is 86% for poles and 64% for trees. However, the problem with the algorithm is that rough classification assumes that ground segments are large planes across the data, which might not be true for all point cloud data sets.

Yokoyama *et al.* (2011) proposed a method for recognising pole-like structures from scanned point clouds. The process is based on Laplacian smoothing using a k-nearest neighbours graph. The first step is the segmentation of input point clouds. Golovinskiy, Kim and Funkhouser (2009) used the segmentation method by connecting the nearest neighbour points. The outcome is a sequentially generated k-nearest neighbour graph. Next, Laplacian smoothing is applied to each segment to improve the recognition rate. Smoothing helps in removing noise and shape degeneration. The disadvantage of smoothing is that the algorithm cannot identify pole-like objects, such as trees, as the branching information will be lost. Laplacian smoothing is applied to overcome the preserving endpoints, which is an operation that relocates a point to the centre of the neighbours. Then each point is classified into points belonging to pole-like, planar or other objects. Each segment's degree of pole-like shape is evaluated, and pole-like segments are selected based on the threshold. The threshold is the height of the pole and the minimum number of points. The degree is calculated by:

$$f_n = \left(w_1 \frac{C_n}{S_n} + w_2 \frac{D_n}{C_n}\right) \times \frac{100}{w_1 + w_2} \quad (6.1)$$

where w_1 and w_2 are weights, S_n is a set of points in a segment, C_n is a set of points on pole-like objects included in S_n and D_n is a set of points that are vertical and included in C_n (Yokoyama *et al.*, 2011). The average accuracy is 97.4 %. However, the disadvantage of the algorithm is that the pole with a nearby hedge or tree was undetectable as pole-like objects' detection depends on the correct segments.

Yokoyama *et al.* (2013) further extended the method to classify pole objects more effectively. First, the attached parts of the pole are recognised by applying RANSAC. The points within a distance threshold from a fitting line are identified as supporter points. The iterations of the k-nearest search extract various segments of the attached parts. Then, shape features are evaluated based on the membership value of the utility pole, lamp post, and street sign. The value is calculated by:

1) the number of attached parts (utility pole has more attached parts than lamp post and street signs)
2) the height
3) the part types.

The next step is to use the context features evaluated using the pole's relative position and local distribution. Both shape and context features contribute to the calculated membership value to classify the pole-like objects into utility poles, lamp posts and street signs. The classification accuracy for pole-like object detection using shape features is 66.7%, and the classification accuracy using shape features and context features is 81.5%. The disadvantage of the algorithm is that the detection depends on specific criteria such as height and shape; therefore, it is not flexible enough to accurately identify all pole-like objects.

Tombari *et al*. (2014) proposed another algorithm to detect pole-like structures in an urban environment. First, using RANSAC, the point cloud is reduced by removing all the planar surfaces, such as building façades and ground regions. More than one plane is detected due to noise; therefore, the plane with the most normal consensus is the dominant plane. Second, on

each non-removed point, a local feature is computed to emphasise the *poleness* of a point's neighbourhood and is categorised by a support vector machine (SVM). The SVM provides a score for a point's probability of either belonging to a pole or not. After the categorisation, a semantic clustering depending on Markov Random Field is conducted using a connected point cloud graph. Finally, all the points in the reduced point cloud are clustered as poles or not poles, based on the constant classifier's output and point connectivity. Spin image descriptors were computed to reject the presence of false positives. The advantage is that the plane could be fitted on one point instead of 3 points and the point's neighbourhood. The algorithm's disadvantage is that the spin image computation to define pole and non-pole structures does not work in the presence of vegetation near the pole structure. Also, the algorithm is tested on a limited point cloud dataset.

Cabo *et al*. (2014) proposed an algorithm to identify pole-like objects from street furniture. The 3D point cloud is divided into 3D cubes called voxels. A centre point and the number of points in voxels are stored. The algorithm is implemented on voxel's generated simplified version of the point cloud. Horizontal fragments of voxel points are analysed and segmented. Next, 2D fragments referenced for pole-like objects are grouped. The 2D fragments are isolated as potential elements using criteria. The criteria are calculated by fitting rings of two different radii. If the isolated voxel cluster is within the inner and outer ring, it is considered a pole object. Finally, these 2D fragment groups of a pole are converted to 3D voxel representation from the original point cloud. The advantage is that the algorithm uses voxels for efficient access to points. The disadvantage of the algorithm is that it works in 2D to identify poles; hence, there is no account for poles with gradients. Furthermore, the poles are not detected in overlapped regions and the presence of other closer objects.

Another voxel-based method by Hackel *et al*. (2016) proposed a method to extract semantic information about objects in scanned data and convert point clouds into geometric representations using changing density. The method first down-samples the point cloud (with a voxel-grid filter and replacing points inside the voxel with their centroid) to generate a multi-scale pyramid and computes separate search structures per scale level. Approximation makes the process of multi-scale neighbourhoods fast. The Neighbourhood calculation is achieved using the method to configure the most suitable k-d tree. Next, 3D features based on eigenvalues and eigenvectors are extracted, which leads to the description of surface properties. Hackel *et al*. (2016) method modified SHOT and SC3D methods and used a reduced point

cloud to speed up the process. An approximate Signature of Histogram of Orientation (A-SHOT) and approximate Shape Context 3D (A-SC3D) are created for more complex objects, especially near contour edges. Next, a random forest classifier was applied to predict conditional probabilities of different class labels and mark every point based on semantic class. These classes are building façades, ground, cars, motorcycles, traffic signs, pedestrians and vegetation. Although mobile mapping data results had high accuracy of 97.6%, the accuracy achieved on Terrestrial Laser scans with base features is 90.3%, further reducing accuracy when larger classes are tested. Furthermore, the computation time is not great for the test data set as it takes 90 minutes to segment 30 million points.

Wu *et al.* (2017) presented a super voxel-based method for automatically locating and extracting street light poles. The method has five steps (1) preprocessing, (2) localization, (3) segmentation, (4) feature extraction, and (5) classification. First, raw point clouds are divided into segments along the scanner line trajectory. Then, RANSAC is used to remove the points that are part of the ground. The remaining point segments are voxelized by VCCS (Papon *et al.*, 2013). Second, the localization method is proposed to identify the pole objects in three steps: localization map generation, the ball falling and position detection. Third, super voxels are segmented using guided localization. Then, the pole objects are extracted based on their characteristics and the calculated barycenter. This helps in adding and expanding voxels to obtain the lamp part. Fourth, feature extraction is done by dividing the feature into the pole and global features. Finally, pole-like objects are classified using a support vector machine and random forest. The advantage of the method is that it is tested on a large point data set of 701 million points with 98.8% localization achieved. The disadvantage of the method is that the processing time is too long.

Yadav *et al.* (2015) proposed an automatic method for Pole-shaped Objects (PSOs) on the LiDar point cloud by MLS. The method is divided into three parts: (1) gridding, (2) vertical segmentation and (3) region growing. First, the input point cloud data is projected onto a plane. Then, regular square grids of the projected data are generated at a predefined size of $m \times m$ in 2D along the X and Y axis. Second, the segmentation is performed on the grouped data by rearranging the data points in increasing Z values. The sorted Z values are segmented and divided into the minimum and maximum Z values. The user defines the number of segments and the height of the segment. The third step is a region growing method, applied by finding the neighbouring points of the seed point as the centre using the k-d tree. PCA is implemented

to analyse the data with cylindrical clusters based on the user-defined threshold of maximum normalized eigenvalue and angle between the z-axis and eigenvector. Finally, pole objects are detected by applying the second and third steps on each square grid. The method can detect with 95.12% correctness. However, the disadvantage of methods is the inexperience of users in deciding the number of segments and their height. The outliers are removed by only considering the 100 points from the top, which is not a powerful way to deal with outliers. The limitations are the detection of pole structure overlapping with other objects, pole's upper part extraction and automatic recognition classification of objects.

Xiao *et al.* (2016) presented a system for car park monitoring. The method starts by classifying the points into the ground, building façades and street objects and then segmenting using state-of-the-art methods. Then, each segment is used to extract the geometric features by fitting the vehicle model to obtain its orientation and position. Then the vehicle features are classified. The system is able to find and locate the vehicles. Also, categorise them. However, the method is based on supervised learning and is only limited to the training sets.

6.2.2 Semantic-Based

This object detection method is based on rules on prior knowledge of objects. Lam *et al*. (2010) proposed an approach for extracting features like roads and attaching features of interest to the road. The method assumes that the road is not flat and divides it into sections to fit the plane. The method addresses this; the method implements two solutions 1) applying RANSAC followed by least squares to estimate the local plane on a 3D subsection bounding box, and 2) Kalman filter is applied to monitor the changes of the local planes. For extraction of road structures, the least squares fit an estimated 3D line. If the 3D line with enough number of points is located, it is identified as a pole structure. In addition, a threshold value of radius and if it is perpendicular to the road surface is applied to determine the poles. The advantage of the method is that it extracts the road and other features even in the presence of cars and trees. The disadvantage of the method is that the plane fitting does not work on curved roads due to the scale factor. In addition, the method is implemented on a 1.4 million point cloud. Hence, there is no proof of its effectiveness in terms of accuracy and time on a larger dataset.

Fan, Yao and Tang (2014) proposed a method based on a priori knowledge of urban point clouds. The method is achieved in three stages 1) pre-processing, 2) detecting seed points of man-made objects and 3) Distinguishing and identifying the seed points belonging to different types of objects. First, the point cloud is divided into three layers in terms of vertical height using a height histogram. Second, each layer's seed points belonging to man-made objects are identified using a line filter called binarizing spatial accumulation map. These seed points are further analysed to be classified. Finally, points belonging to an object are retrieved based on categorized seed points. The advantage of the method is that the detection rate achieved is up to 83%, and classification accuracy is up to 92.37%. The limitation of the method is that objects are identified wrongly because of height criteria (truck identified as a house), fences are scattered points which are unable to detect, and low-density features are unable to be identified.

Teo and Chiu (2015) proposed a coarse-to-fine approach for extracting pole-like objects from the point cloud. The method works in 3 stages – data pre-processing, coarse-to-fine segmentation and pole-like object detection. First, the pre-processing stage focuses on the region of interest (ROI) selection and building façade removal. Parts of the road are identified based on pre-defined road width and length, and points located 15m above the ground are selected as façades. Second, coarse-to-fine segmentation is accomplished, which involves three major steps: (1) voxel scale segmentation, (2) point-scale segmentation, and (3) overlapped object segmentation. The methods at this stage extract man-made and non-man-made pole-like objects. Voxel scale segmentation simplified irregular points through voxel space and removed any non-pole-like objects.

All voxels which do not contain any points are deleted. Next, euclidean clustering of voxels is applied, after which points in the voxels are segmented into individual objects. Point scale segmentation takes the output of voxel scale segmentation and removes local ground points using the plane detection method based on RANSAC. Then Euclidean clustering of points is applied to extract individual objects based on tolerance distance. Point scale segmentation cannot address overalled objects, which are then processed by overlapped object segmentation step. In this step, based on ground height, the method selects points lower than breast height to detect individual stems. Coarse-to-fine segmentation separates objects on the ground but may contain non-pole-like objects, like walls and vehicles. Next, pole-like road object detection in which pole parameters such as location, radius, and height are calculated, and non-pole-like objects are filtered based on height, position, shape and cross-section. The method achieved an

accuracy of about 90%, which is more efficient than a single-scale framework. However, false positives were caused due to mixed objects, occlusion, and circular man-made columns like bus stations. Due to insufficient point density, different objects were mixed and misclassified as poles. The method also falsely detected low vegetation and banners, and errors were caused due to the complex road area environment.

A method to use hierarchical extraction to detect urban objects by Yang *et al*. (2015) is proposed. The method first generates multi-scale voxels. Then, all the voxels are traversed to calculate the distance between each voxel centroid. If the distance is less than the threshold, the voxels are grouped. These voxels are then analysed to find linear, planar, and volumetric geometric shapes. Next, the normal and principal directions are calculated, followed by RGB and intensity. There is no restriction on super voxel size; the smaller voxel estimates are correct compared to larger voxel sizes. Therefore, two different voxels are created and integrated from different original voxel sizes to overcome the size problem. Hence, this results in similar geometric structures in each super voxel. The next step segments the super voxels by combining graph-based segmentation with multiple cues. A set of rules is defined to merge the segments into units with similar urban objects. The calculated saliency includes the height of the segment, the angle between the vertical and normal direction, the angle between the principal and vertical direction and the number of neighbouring segments. Finally, it extracts and classifies urban objects in hierarchical order placed by the saliency of segments. The advantage is that the proposed method is very effective as it does not just extract the objects but also classifies them in order. The disadvantage is that the super voxels are segmented using the geometrical attributes that do not account for overlapping structures in the voxel. Also no indication of the reason for threshold distance selection.

Yan *et al*. (2016) presented a workflow for automatically extracting highway poles and towers. The method starts with automatic filtration to separate ground and non-ground points. The filtering is based on measuring slope changes in points and their neighbouring points and the skewness of all points. Skewness is calculated as

$$sk = \frac{1}{N.\sigma^3} \sum_{i=1}^{N} (s_i - \mu_\alpha)^3 \tag{6.2}$$

where N is the total number of points with $i = \{1,2,3 \ldots\ldots, N\}$, s_i is the length of the total number of points, σ and μ_α are standard deviation and arithmetic mean (Yan *et al*.,2016).

If sk is greater than zero, the point is classified as the non-ground point. The isolated points data produces a digital elevation model to normalise all the above-ground points. A density-based spatial clustering algorithm of application with noise (DBSCAN) groups non-ground points into clusters. DBSCAN requires two parameters:

- the number of minimum points to form a cluster and
- the radius to contain that minimum number of objects.

Next, a set of rules based on the structure's height and projected horizontal area are set to identify potential poles and light towers. Then the least squares circle fits to find the circle of the pole. Finally, cleaned light poles and towers are extracted. The method uses a very sophisticated way to separate the ground points. However, the problem with the method is that the road signs and light towers can be of various sizes, and the set of decision rules is not flexible. Furthermore, the large dataset implementation means more computing time for clustering the points in the algorithm.

Yan *et al*. (2017) also presented a workflow for detecting and classifying pole-like structures in motorway environments. The first step is data processing, separating the ground points based on local minimum height and clustering non-ground points into segments. K-d tree is used to organize the points and is clustered based on the Euclidean distance between the neighbours. The overlapped segments containing pole-like road objects are further separated by comparing the minimum boundary rectangle and the candidate's height. Next, a weighted graph is calculated. An iterative min-cut-based segmentation approach is applied to minimize the sum of all weights. The object is divided into ten slices across Z into the foreground and background clusters. After the clustering, pole-like object detection is achieved using prior and shape information. Prior information is the object's size, and shape information is where the objects must be vertical. Finally, pole-like structure classification is achieved using objects' features and a random forest classifier. The algorithm's accuracy for the two datasets is 94.9% and 97.8%. However, due to their short height, the proposed algorithm failed to detect pole-like objects in the presence of vegetation and signs. The height criteria should be flexible to accommodate all pole height structures to detect them.

6.2.3 Slicing-Based Methods

This section analyses existing methods like structure detection based on horizontal or vertical slicing. Like Tombari *et al*. (2014), Huang and You (2015) proposed a system that uses an SVM classifier. The method starts with the horizontal slicing of the Z layer. Next, clustering is performed based on the Euclidean distance to each slice. Each slice could have potential characteristics identified by a pole seed generation. Pole seed depends on two criteria based on the bounding box property of each candidate cluster. The generation is calculated by:

i. the cross-section area
ii. the segment length

The result is the candidate poles. Next, bucket augmentation is performed on pole seeds to locate the attached structures. The seed trunk segment's centre is considered a pole bucket, and other points in the range are added to the pole cluster. The next step is to ensure that the candidate clusters contain just pole points, not the ground or other closer objects. In order to fulfil the criteria, the ground (lowest part of the pole) and the disconnected regions are trimmed. Finally, the clusters are classified into four categories lights, poles, signs and others, according to their height. The disadvantage of the method is that it assumes that the horizontal slice always has a pole-like structure. Therefore, lacking an extensive search for contextual information on detected features. For example, the trees are confused with poles if they have similar structures.

Landa and Ondroušek (2016) proposed another method to detect pole-like structures by horizontal sections. Firstly, the point cloud is processed by eliminating the lowest points, ground points, and highest points, anything above 12 metres (pole height). After that, outliers are removed based on a statistical analysis of neighbourhood points. Next is the division of point clouds into horizontal sections. Finally, each section is segmented by Euclidian distance. The resultant 3D points are based on centroid $C_m(x,y,z)$ and the maximum distance between the centroid and any subsequent section point. Points inside the sections are then classified as pole structures based on directional vector, the difference in z coordinates and distance from the centroid. The algorithm reduces false positives. The advantage of the method is the interconnectivity of all the objects inside the point cloud. However, the disadvantage is that more pole object detection was missed due to the similarity because the false-positive algorithm

was implemented. The method could be more effective by segmenting or analysing the feature to classify it more efficiently. Also, the processing time was excessive (2 hours 35 minutes).

6.2.4 Shape-Based Methods

This Section analyses the existing methods of pole-like structure detection based on their shape in the point cloud. Golovinskiy, Kim and Funkhouser (2009) proposed a system to recognise objects in a 3D point cloud of the urban environment. The method consists of 4 steps – Locating, segmenting, extraction, and classification. The method uses multiple alternatives for each step and evaluates all alternatives to find the most efficient and accurate method for each step. The system was trained using truth data set and then was used to recognise objects in a 1 billion point data set. The results indicated labelling 65% of the small objects. Firstly, the localisation step cluster nearby points to form sets of potential objects of interest locations where density is highest. Next, points close to the ground are filtered out and removed from isolated points. Any points belonging to buildings (large connected components) are removed too. Next, for each location, the algorithm tries to differentiate between points near the object from those that are background clusters through segmentation. Segmentation also identifies object shapes used in the next step and assigns points to objects when they have been classified.

After segmentation, potential objects are extracted, describing the shape and context. This step also distinguishes objects from one another and their backgrounds. To identify the shape, multiple quantities are evaluated at this step, like the number of points, volume, average height, and standard deviation. Next, to evaluate context, the position of the object relative to its environment is used as a cue to identify its type. After extraction, objects were classified with respect to the manually labelled training set of object locations. The result is that easy object shape features were identified with 54% precision which was enhanced to 64% by adding segmentation and contextual features. The major bottleneck of the method recognition performance is feature extraction and classification. Better shape descriptors, contextual cues and classifiers need to be enhanced to improve the performance of the system discussed. The entire process of recognising objects in 1 billion points of point cloud took 46 hours on 3GHz PC – 15 hours on pre-processing, 6 hours for the location step, 15 hours for segmentation, 6 hours to extract features and 4 hours to classify. This clearly shows that the system can

recognise with decent accuracy but will take a long processing time for real-world point cloud data sets.

El-Halawany and Lichti (2011) proposed a method of point cloud processing to detect road poles and evaluate their dimensions from the unorganised point cloud. The method applies 3D segmentation and extracts poles using local neighbourhoods and analysing eigenvalues. The method consists of 2 phases – the first phase examines the effect of the density of point cloud and neighbour size on segmentation and the second phase is the extraction of different poles and determining their radius and position. A K-d tree is applied in the first phase to organise the point cloud and accelerate the search process. Then, PCA analysis is performed on the neighbourhood to classify groups as linear and planar features. In the second phase, each pole is extracted using distance-based region growing. To deal with noise, two methods are implemented (1) intensity-based outlier removal and (2) the relation between the radius of the pole and its eigenvalue. In the first method, low-intensity points were removed by analysing the distribution of intensity values for the whole point cloud, improving cylinder fitting results. The second method does not remove any low-intensity points, but many circles are simulated with different radii and compared with eigenvalues of the cross-section to evaluate the mean radius difference.

The result showed dense point clouds gave better segmentation results than low-density point clouds. The results were analysed with different *k*-neighbourhood sizes and the accuracy of pole detection, which varied a lot. The method cannot assign a neighbourhood size to all data sets as it is relevant to the density of point cloud data. Further, this method is also inefficient in extracting pole features close to the ground due to the inclusion of pavement points in neighbourhood search, which affects eigenvalues. In addition, after isolating linear features, the pole structure was incomplete. The evaluation of the radius of poles based on intensity succeeded in giving the exact radius for the flagpole and sign pole, while eigen-radius successfully gave the correct radius for the street pole. However, both methods failed to evaluate the correct radius for the traffic light. The method concludes that the intensity-based method can be used in lower radius poles, and the eigen-radius method can be used for higher radius poles. The method is tested in a relatively small point cloud dataset (1 million and 4.8 million points) with no proof of processing time, which makes it unclear whether this method is efficient enough to be used in bigger point clouds and its impact on processing time and accuracy.

Bremer, Wichmann and Rutzinger (2013) presented a method to classify pole-like objects using rotation and scale invariant points. Single objects like poles are classified by connected components and Dijkstra-path analysis, while tree and artificial objects are separated using a graph-based approach. The method achieved an accuracy of 90%. The method focuses on artificial pole-based objects like lamps and traffic signs and natural pole-like structures like trees. Each point local neighbour based on a pre-defined radius is identified and encoded into a covariance matrix from which three eigenvalues and three eigenvectors are computed. Based on eigenvalues, the points are classified into primitive classes (linear, planar and volumetric) based on the eigenvalue pattern of each class. Some classes like linear and planar were further categorised based on orientation (horizontal, vertical and other). Object classification is then accomplished using primitive classification to aggregate and separate semantic groups. Minimum cluster sizes were also introduced to reduce noise.

Eigenvalue-based classification has its advantages and disadvantages. With a 0.1 metre radius search, classification was appropriate for small geometrical patterns but strongly sensitive to scanning patterns and differences in point densities. For example, detecting thick tree trunks is impossible through 0.1 metre search radius classification. On the other hand, a 0.5 metre radius offers better results for larger poles. It also deals with lower point densities and detects planar features. However, planar objects were incorrectly classified as undefined in areas with low point density. Furthermore, the method demonstrated classification to separate eight classes and focus on pole-like objects. As a result, planar object groups were handled less accurately and would require a more intelligent and thresholdless solution.

Rodríguez-Cuenca *et al.* (2016) proposed a method to detect and classify urban objects. The methods start with pre-processing in two stages:

1) transforming the reference frame into a local cartesian coordinated system and
2) removing the parts of the cloud that will not be used for detection.

These points are removed by indexing vertical and large surfaces, and the connected components are segmented on the same surface. The geometric indexing merges the information of each point's normal vector and roughness value. The indexing is then extracted using two threshold values from the vertical and horizontal surfaces. Next, an octree level and a minimum number of points per segment are used to segment based on prior knowledge

resulting in three groups – original, building facade and road point clouds. Next, the point clouds obtained are analysed in a 3D vertical pillar pattern to connect the ground level to detect pole structure. The *Reed and Xiaoli* anomaly detection algorithm applies the height difference and spatial dispersion vertically and horizontally. The results of the algorithm are vertical urban elements. Finally, the vertical elements are classified by descriptors of the roughness and cylindrical coordinate scattering of radial distance. The quality rates tested in the two data sets are 94.3% and 95.7%. The advantage of the method is that it considers the slope and is not dependent on the scan trajectory as the original coordinates are transformed. The disadvantage is that the method could not detect the tree structures with low-density point clouds and the trunks found on the tilted ground.

Wang, Lindenbergh and Menenti (2017) proposed an algorithm to identify lamp posts and traffic signs in urban road environments gathered by mobile laser scanning. Before the method, the raw data is pre-processed in two parts, 1) tiling and 2) separation of ground and non-ground points. Then, these tiles are created, dividing the data in the direction of the scanning trajectory. SBET is Smoothed Best Estimation of Trajectory whose points are used in tile calculations.

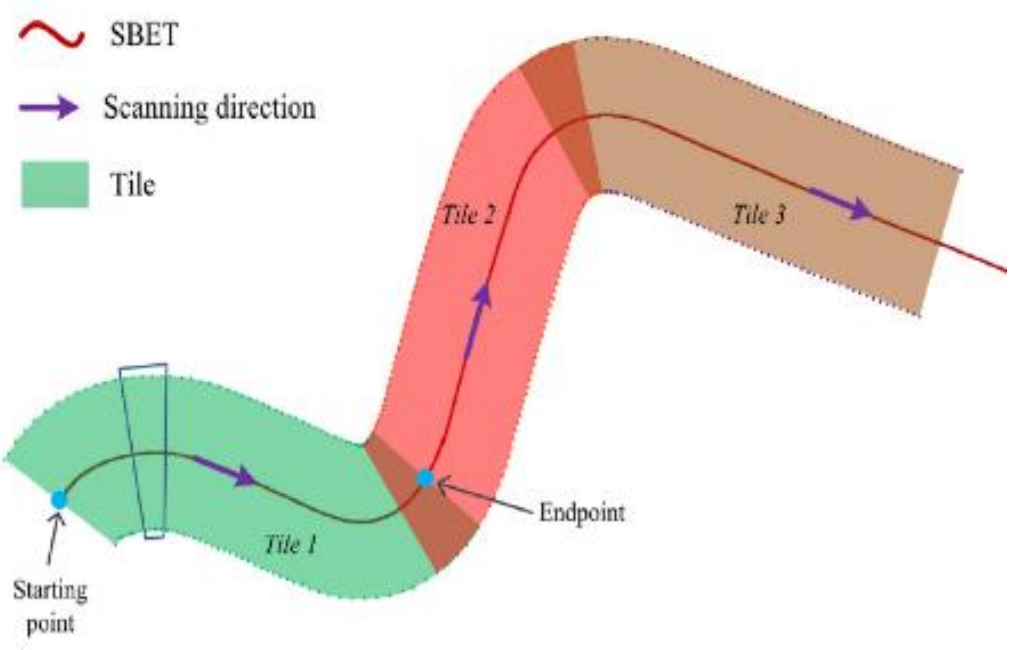


Figure 6.1 Wang demonstrated different tilling along the road (Wang, Lindenbergh and Menenti, 2017)

The next step is the voxelization of non-ground points, and the connected voxels are grouped into clusters to form candidate objects. The next is the dimensionality analysis of each voxel, followed by eigen-based shape descriptors. Then, significant eigenvector points in each voxel are mapped into a triangle of sphere icosahedron, also called the eigen sphere of the voxel cluster. Finally, the repeated subdivision of voxels is performed until matched candidates, and training objects are found. The algorithm works well with the given descriptors of poles and road signs; however, point density and other closer objects affect the pole and sign identification. Therefore, the method will not work if the point density is low or the presence of noise. For example, a lamp or a tree on the roadside is not detected because of their closeness.

A similar approach to detect pole-like objects by Shi *et al*. (2018) started with pre-processing point clouds that remove outliers, voxelization, downsample and filter ground point. The outliers caused by the laser beam and differences in object surface are essential to remove, resulting in wrong surface normals and curvature. The outliers are removed by calculating the mean distance and standard deviation. A k-d tree is used to structure the points. Afterwards, the model is gridded into voxels, and the point nearest to the voxel's centre is kept. The next step is filtering ground points using fabric on an inverted surface. The interaction between the distribution node and LiDar points generates a surface compared to the original points to classify ground and non-ground points. On non-ground points, a second voxelization is performed. Pole-like structures extracted by spatial independence analysis, i.e. analyzing the number of empty neighbourhood voxels. The next step is applying RANSAC for cylinder feature detection. The voxels are verified to either belong to pole-like objects by PCA or not. Moreover, the region growing method is used to extract the complete structure of the pole-like object. Finally, detected pole-like structures are automatically classified into three categories streetlamps, traffic signs and poles by 3D shape and height matching. The advantage of the algorithm is that the dataset is downsampled and cleared of any noise. Also, the second voxelization is helpful for the *k*-nearest search of neighbouring voxels. The disadvantage is similar to Tombari's method as the practical implementation is only tested on limited datasets with a lack of variety of data sets that the algorithm can use. The method also assumes that all pole and cylinder objects are stand-alone and have no vegetation or building points near them. The template of the pole-like shapes is fixed; hence if there is a variation in shape or height, it cannot be detected.

6.2.5 Individual Tree Detection Methods in Forest Point Clouds

Tree detection is essential for various applications like 3D construction for the city. In this section, the existing method of tree detection is discussed and analysed. Existing methods to detect individual trees (Trunk and foliage) are proposed and discussed by Monnier (Monnier, Vallet and Soheilian, 2012), Wu (Wu *et al.*, 2013), Li (Li *et al.*, 2016), Safaie (Safaie *et al.*, 2021), Wu 2018, (Lalonde, Vandapel and Hebert, 2006) and Belton (Belton, Moncrieff and Chapman, 2013). Other methods to detect trees and poles together discussed by Pu (Pu *et al.*, 2011), Cabo (Cabo *et al.*, 2014), Yang (Yang *et al.*, 2015), Rodríguez-Cuenca (Rodríguez-Cuenca *et al.*, 2016), Li (Li *et al.*, 2019), Li (Li and Cheng, 2022), and Kang (Kang *et al.*, 2018).

Lalonde, Vandapel and Hebert (2006) proposed an approach to process point clouds to identify tree stems for forest inventory. The approach starts with point-wise classification by feature extraction, feature distribution and online classification. The input point cloud is divided into linear, solid, and scattered classes. First, the expectation-maximization algorithm calculates the centroid, principal axes and covariances. The second step is point cloud segmentation based on the direction of features, normals and regions. The third step is an interpretation based on the data's context knowledge, such as size, smoothness, continuity, direction, spatial and classification and relationship. The last step is high-level scene modelling by fitting geometric shapes to extracted components. The cylinder is fitted to the points in 2D projection and 3D. The 3D fitting increases estimation accuracy. The advantage of the proposed method is that it is generic and could be applied to ariel and ground scanned data. The disadvantage of the method is that it lacks the evaluation of the filtering, and the criteria to remove ground points are not explained.

Belton, Moncrieff and Chapman (2013) introduced the method for processing trees using gaussian mixture models. The method first classifies the features in the point cloud by PCA. PCA gives the variance in the direction of eigenvectors and the distribution of neighbourhood points. A variable neighbourhood size (radii value) determines the feature resolution. Next, the points are clustered using the Gaussian mixture model. After applying different neighbourhood radii values, the resulting cluster is manually analyzed to categorize them as leaves, tree trunks, and branches. The multiple clusters are combined into a single model for the same class. Further, the classified data is examined based on two characteristics 1) tree structure generation

and 2) volume measurement of carbon capture. For tree structure generation, a horizontal slice produces a skeleton. Each slice is examined to be clustered, and an ellipse is found. The ellipse centre is used as a node graph, a cyclic detection algorithm to find common nodes and merge them to find a tree structure. The total volume is calculated by cylinder fitting as it is an important attribute for carbon capture and is used in predicting climate change. Belton's method delivers a Gaussian mixture model to separate leaves from the rest of the tree but does not provide evidence of tree detection in any large point cloud data or the applications of the method on various types of data sets.

Amiri *et al.* (2017) proposed a three-tiered tree detection method in forested areas that work on point, segments, and object levels. In the first level, all the points in the forest scene scanned data are evaluated to belong to a tree or not. The features are grouped into their categories 1) point feature histograms is a local neighbourhood shape descriptor to distinguish between different surfaces using surface normals, 2) covariance features derived from eigenvalues of the local neighbourhood covariance matrix, and 3) normalized heights. Next is the segment level, where the cylindrical neighbourhood is applied by orthogonal distance regression. The results are classified into positive groups that could contain a part of the tree stem and negative groups that could contain vegetation and branches. Then, segmented features are divided into a modified version of the cylindrical shape context and angular deviation of the segment axes from the Z-axis. The last level is the object level, where the segmented and point-level features are merged to recreate the individual tree. The positive segments are merged using hierarchical clustering based on the aggregate distance matrix between clusters. The final step is stem line fitting using orthogonal distance. The method's classification precision is achieved at 0.86 and 0.85 for two samples. However, the method lacks to present cylindrical object detection on low-density point clouds. Also, the clustering of the segments could be enhanced if the tree locations are considered.

Wu Rongren *et al.* (2018) presented a method for detecting tree stems and diameter breast height (DBH) estimation in a forest environment on the point cloud data captured by the terrestrial laser scanner. The method starts by preprocessing, i.e., removing ground points from raw point clouds. Next, the stem points are differentiated from non-stem points by calculating normals. The normal vector of *k*-nearest points belonging to a tree is a small absolute value compared to tree crowns and bushes. Then, the derived tree points are divided into voxels. A downward growing method within voxels is implemented to identify tree stems that stop at the

bottom layer. The derived stems are horizontally sliced at a height of 1.3 metres. All the point's normal vectors in the slice are computed. Finally, the tree stem is considered whose normal vectors are perpendicular to the plane. The method could detect 17 out of 21 trees. However, the method does not account for the reflected noise points and low-density tree stems; therefore, the proposed method lacks to detect trees with noise and fewer points.

6.2.6 Urban or Street Trees Detection Methods in Urban Point Clouds

Monnier, Vallet and Soheilian (2012) presented a method to detect trees in complex urban environments. The method is divided into two parts, separating the points belonging to trees and the individual trees within these points. Firstly, the scan points are classified according to their geometrical shape. A local descriptor is used for the classification by Demantke *et al.* (2011), which started by applying a principal component analysis of their neighbourhood. Each neighbourhood is categorized by three descriptors linear, planar and volumetric. The linear descriptor is used for small trunks and posts, the planar descriptor is used for building façades, and the volumetric descriptor is used for tree leaf foliage and balconies. The results of descriptors are very noisy, so probabilistic relaxation is applied. The probability depends on the distance between the point and its neighbours and a compatibility matrix. The descriptors are not robust enough to identify tree trunks and posts; therefore, next step is to apply another cylindrical descriptor. Next, tree detection is achieved in four steps:

1) Vertical accumulation of each descriptor into a horizontal grid is achieved by creating accumulation maps,
2) Spatial filtering using smoothing by Gaussian kernel, hysteresis thresholding, connected component computation, size filtering, and morphological dilation,
3) Combination of masks to retrieve individual trunks,
4) Tree foliage separation from other objects and other trees by associating each pixel to the nearest trunk.

The accuracy of the method is 80%. The disadvantage of the method is that each point is analyzed with descriptors or the distance between its neighbours. The calculations are extremely cost-inefficient (computing time) as the point cloud could be very large.

Wu *et al.* (2013) presented a voxel-based marked neighbouring search (VMNS) method for detecting street trees. The method starts with the voxelization of point cloud data dividing the space in cuboids. The location of voxels is indexed based on their length, width, and height. The voxel height varies till layer six corresponds to 1.2 - 1.4 metres. The values for each voxel unit are calculated. Next is the voxel's neighbourhood search, which leads to the seed voxel selection. The search for tree objects starts with seed voxels. The voxels are traversed to find the connected voxels, marked and grouped. The grouped tree voxels are selected based on 1) geometric properties such as the area and shape of voxels and 2) morphological attributes such as the number of voxels in a group and compact index. The next step is top-down radius constrained searching and marking to identify the tree's trunk and further by bottom-up neighbourhood competing for search and marking to identify the tree crown. The competing search is applied to differentiate the overlapping tree crowns. The marked voxel group results in potential trees are estimated based on tree height, crown diameter, breast height, and base height. The other pole-like objects are eliminated based on crown diameter and tree height threshold. The case study resulted in 98% of correctness rates. The method cannot detect trees with lower heights and if they have multiple stems.

Li *et al.* (2016) proposed a similar method to Wu *et al.* (2013) that follows the dual growing methods for separating individual trees. The method is divided into three stages: trunk shape growing and segmentation, crown voxel growing, and refinement to obtain trees. The method starts by separating ground and non-ground points. The individual trees are extracted from non-ground points. Next, the candidate cluster containing trees is extracted by local surface normals based on their Euclidean distance. Then, the candidate tree cluster is voxelized. The seed layer is found by dividing the candidate tree by half. The artificial objects are rejected from the candidate tree cluster by comparing a crown with a round cross-section and a trunk with a smaller diameter and cylindrical shape. The horizontal convex hull is calculated from the points in the voxel group at each layer. The upward cross-sectional analysis is performed to determine the tree trunks. The convex hull of seed and up traced voxels are compared by area, perimeter and diameter. Also, the crown's minimum required diameter, minimum roundness degree, and a maximum ratio of geometric parameters are used to find the crown. The down tracing process is used to determine the trunks.

Next, the crown parts are further analyzed for crown voxel growing. The voxels are considered crown voxels if the layer has the same row and columns and unusual shapes in cross-section

areas. After the voxel grows, two overlapping crowns are segmented based on fewer area increments as the changes are observed in cross-sections. Finally, the seed selection and trunk point growing are applied in iterations to extract individual trees. The advantage of the method is to distinguish between trees and other objects and to be able to extract the whole tree. The correctness in tree extraction is 96%, and completeness in tree detection is 98%. The disadvantage is that the tested datasets have high-density points, directly affecting the crown separation process. The method for separating candidate tree clusters depends on surface normals which are not cost-efficient as the point clouds could be very large.

Kang *et al.* (2018) proposed a voxel-based method for extracting and classifying pole-like objects. The method starts by dividing data into 3D grids called voxels. Each point is grouped and indexed. Next, PCA is applied to analyze the dimensionality of the voxels and find the predominant direction for linear, planar and spherical. MRF shape detection optimizes the voxels, i.e., the contextual information for the direction and classification (labels linear and non-linear). Pole-like are vertical and isolated compared to building façades and tree canopies. Therefore, the non-ground points are divided into slices on the selected interval. Adjacent slices are clustered together based on the circular model with an adaptive radius. The model has two concentric circles that compare the geometric centre and any point inside the inner circle to a threshold to determine a pole-like object. The individual poles are extracted by the vertical region growing method as 1) Starts with one voxel of the pole-like object, 2) Vertically growing from the seed voxel, and 3) Growing continues until the distance between the segmented object and nearest voxel exceeds a threshold, and 4) Repeated until all voxels of pole-like are traversed.

Further, the extracted objects are classified by semantic rules. The classification is based on height (classify trees and poles). Another rule is 2D projected point distribution; if the distribution exceeds the threshold, it is considered a tree, otherwise considered poles. The precision of the method's detection is 85.3%, 94.1% and 92.3% for the three datasets. The disadvantage is that the voxel-based recognition fails to capture trees of large trunk diameter as it is focused on pole-like objects. The method also assumes the poles as isolated objects without noise, which might not be true in the urban point cloud.

Safaie *et al.* (2021) proposed a method for efficiently creating a tree inventory of roadside trees in point clouds using raster images. The method filters the ground points to start with by using

tile sections. The points in each Section are divided into left-side and right-side. Then, each side's low height points are filtered. The next step is trunk Extraction 1) trunk positioning and 2) elevation range. A trunk portion is considered for locating tree positions at adjacent horizontal intervals. The detection of circles is done from raster images by the Hough transform algorithm. The circles are validated as trunks by range and number of layers. Elevation range extraction is achieved by cylindrical buffering in trunk position, elevation sectioning, raster binary image creation, altitude density histogram and density threshold. Next is foliage extraction, which is based on the following 1) determining the initial range – A Voronoi tessellation is applied on the extracted trunk 2) extracting the foliage points 3) altitude sectioning 4) density image generation for each section 5) precise boundary detection using active contours. The geometric region-based active contour is applied to the groups. After separating trunk and foliage points, the final step is characteristics measuring. The characteristics measures calculated for each tree are:

a) planimetric coordinates, which is tree location found by the lowest height circle,

b) trunk height calculated by the difference between the maximum and minimum trunk heights,

c) trunk diameter calculated for each tree,

d) foliage height calculated by the height difference between the maximum and minimum foliage heights,

e) maximum foliage diameter, which is the maximum diameter of all peripheral circles,

f) total tree height

g) distance from the road edge

The proposed method works well by extracting the trees in the urban road dataset. The algorithm identified all the trees in the dataset despite the foliage overlapping. The disadvantage is that the ground removing method assumes the ground to be flat, whereas the ground could have a gradient in the real world. Also, the method lacks considering the scenarios where trees could be of different shapes and have low point density.

6.2.7 Summary

In Section 6.2, the existing methods are analysed and discussed. The section starts with the methods to detect pole-like structures in point clouds divided into four categories: segmentation-based, semantic-based, slicing-based and shape-based and followed by reviewing the methods to detect trees in the forest and urban or roadside scenarios. The common problems with methods have been identified. For example, the existing methods lack the ability to identify pole-like structures in low-density point clouds, are affected by noise (the presence of noise and other close objects), do not elaborate on the specific criteria and do not account for the presence of a gradient in the data. Another challenge is computation time. In reality, point clouds can be very large, with billions or trillions of points, so the methods were shown to be practically inefficient in terms of time.

A common issue with terrestrial scanners is that the point density thins out as the scanner scans to the farthest points. Therefore, the analysis of existing methods concluded a particular problem: the poles or trunks furthest away from the scanner are undetected as the point density is very low. Another common problem with existing methods is that they fail to detect pole-like structures and trunks in the presence of noise. When the other objects were present close to the pole or trunks, they were undetected, i.e., the presence of noise or outliers. Also, they are undetected when other objects close to the pole and trunk overlap.

Another common challenge is not specifying and providing evidence on selecting individual criteria and threshold values to explain the selection reason for these values. The point cloud data is scanned data from the real world and therefore has a high probability that the ground level has slopes and dips; however, the existing methods assume that the ground level is always flat. Hence, there is no account for the presence of a gradient in any of the methods. Therefore, the voxel-based algorithm is proposed, designed and developed as a robust, accurate and efficient method to detect trunks and poles in large point cloud data to overcome the drawbacks of existing algorithms as stated above. Section 6.3 presents the proposed algorithm on large-scale data in a commercial environment.

6.3 Proposed Algorithm for Trunk/Pole-like Object Detection

6.3.1 Overview

This section proposes an algorithm to detect cylindrical objects in urban point cloud data. These objects can be tree trunks or man-made street objects like utility poles, lamp posts, traffic lights, streetlights, etc. The data collection is accomplished by various laser scanning systems like airborne, terrestrial and mobile. Moreover, the algorithm is generic, i.e., can be applied to any scanned data. Most poles and trees have a generally circular cross-section and a cylindrical shape. Therefore, the algorithm aims to detect cylindrical objects. A typical tree and a pole in the point cloud are shown in Fig 6.2. The tree is divided into two parts: tree trunk and tree foliage. The proposed algorithm detects the trunk and pole and classifies them.

Figure 6.2 Example of a typical tree and man-made object, i.e. pole

The proposed new voxel-based algorithm addresses the disadvantages of existing methods discussed in Section 6.2.7. The functional requirements of the proposed algorithm are that it works on terrestrial and LiDar scanned point clouds, works with low-density points, identifies the slopes on the ground level and is efficient in terms of computation time. The proposed algorithm is divided into six stages:

1) **Pre-processing** – Terrain extraction, Ground and Non-ground Points Classification.

2) **Voxelization -** Voxel Bounding Extent, Voxel 3D Indexing and Generating 3D grids.

3) **Seed layer -** DBH (Diameter of Breast Height) estimation to find the seed layer.

4) **Clustering -** Neighbourhood Searching and Approximation and Marked Grouping.

5) **Cylindrical feature extraction -** Potential clusters filtration by Compact Index, Area and Geometric Primitive Modelling - Circle fitting in 2D.

6) **Trunk and pole classification**
    a) Shape-based rules – Pole and tree isolation and 2D Voxel distribution,

    b) Intensity and colour-based rules,

    c) Semantic rules – Voxel Dimensionality analysis, Adaptive radius to classify pole or tree, upward region growing.

The overall workflow of the proposed algorithm is shown in Fig 6.3. The input data is either raw point cloud data or sampled data, and the output is the centre and radius of the detected cylindrical object classified as a trunk or pole-like object.

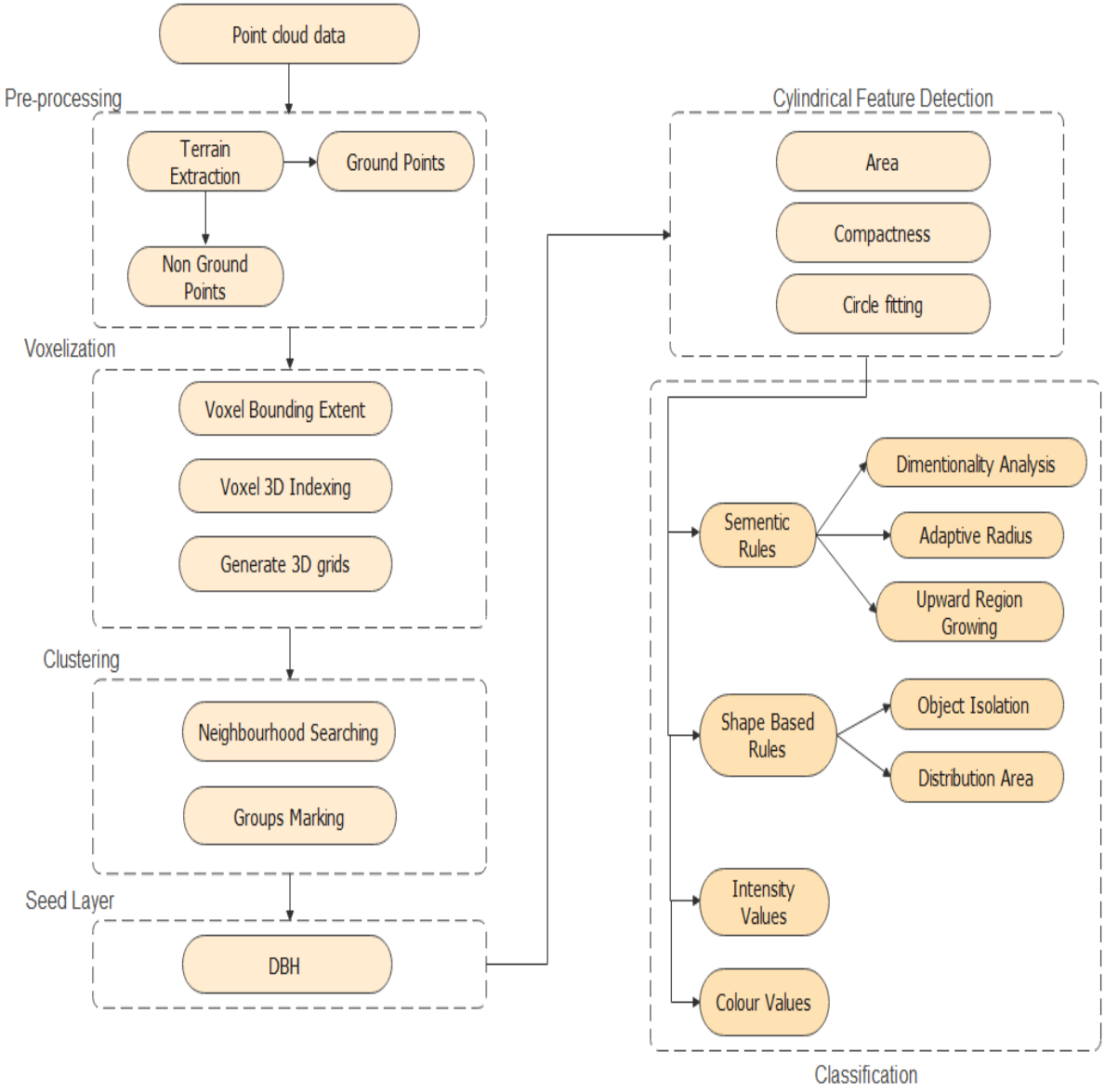


Figure 6.3 The workflow of the proposed algorithm

6.3.2 Terrain Extraction: Classification into Ground and Non-ground Points

The first stage is *Terrain Extraction*, which uses gridding to identify the ground and non-ground. The point cloud consists of extraordinarily complex data with various objects and information. As the proposed algorithm focuses on extracting cylindrical features such as tree

trunks and pole structures that are not part of the ground, the ground level is removed for downsampling and reducing the points. The removal of ground points i.e., reducing the total number of points leads to the proposed algorithm's high computation efficiency and thus saves time.

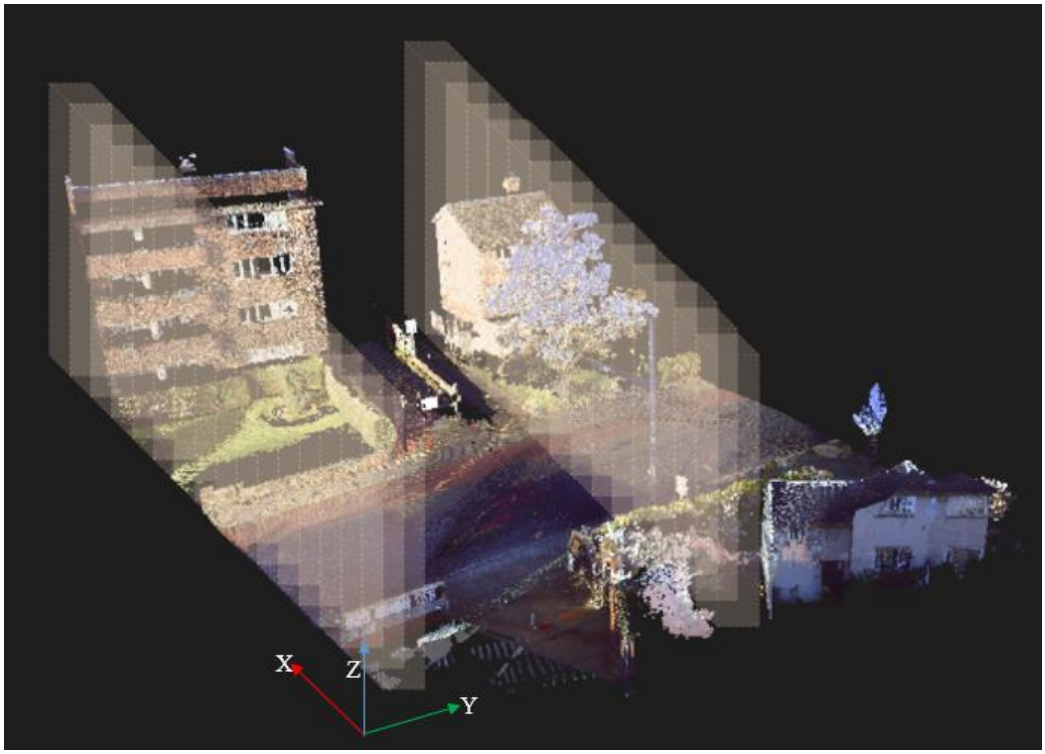


Figure 6.4 Example of gridding in a point cloud

The example in Fig 6.4 shows the grids implemented on the real point cloud data. Fig 6.4 does not show all grids to prevent confusion. The ground points are always inlined with the x and y-axis of the reference coordinate system. The problem is that there is no procedure to identify the ground level on Z-axis. The minimum Z level will not necessarily be the ground as the point clouds are similar to the real world, which means there are good chances for the presence of a dip or slope on the ground. Therefore, an approach allows users to click on the ground near the tree or pole structure to derive the Z value. Then, the user clicked Z value is fed as input for the next phase of the algorithm. However, there are problems with ground user selection are:

- The ground level picked may not be the same for the whole point cloud, i.e., it can have gradients.
- Users may have to click several times to input the Z value; hence, more clicks in terms of usability.
- Lack of experience of users as to where to click to derive the correct Z value. For example, if the click should be under the tree or on the street to get the correct ground level.
- When the point cloud data set has flat ground, the algorithm works perfectly; however, when the data has a gradient, the algorithm will not work.

This thesis proposes "*Terrain Extraction*", a new technique designed first time to extract terrain from point clouds. The solution for the above problems is Terrain Extraction applied to classify ground and non-ground points to overcome the problems. The terrain extraction automatically extracts the ground level with or without slopes. The functional requirements and advantage of applying terrain extraction is that the algorithm:

1) Works automatically without the user intervention.
2) Fewer user clicks.
3) Detects terrain with gradients.
4) Classify ground and non-ground.
5) Lowest and Highest Z levels.

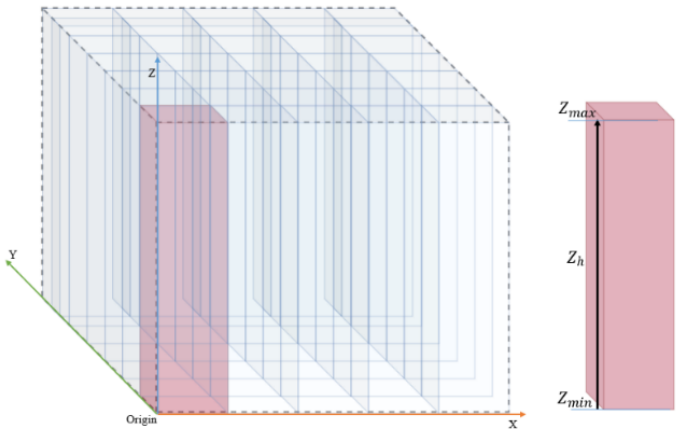


Figure 6.5 The Gridding example

The process starts by gridding the whole point cloud data vertically, as shown in Fig 6.5. A grid has eight corners with length, width, and height. The height Z_h of the grid is

$$Z_h = Z_{max} - Z_{min} \tag{6.3}$$

The length and width are set similar to the voxel size discussed in Section 6.3.3. During gridding, points inside grids are searched for all grids. The algorithm for searching points inside the grid is adapted from Sunday (2021). The algorithm starts by accumulating all the points in the grid along the x, y and z-axis. Then, every point inside the grid is traversed to find the lowest point inside the grid. Finally, the derived lowest points of all the grids are passed for the next stage, i.e., Voxelization.

6.3.3 Voxelization

The second stage is *Voxelization*. Voxels are used widely for spatial sampling, grouping, and partitioning. Voxelization is used for different purposes such as detecting non-static objects (pedestrians) by Schauer and Nüchter (2018), voxels and *k*-means clustering (Tazir, Checchin and Trassoudaine, 2016) used for colour-based reduction and segmentation. Many scholars used voxels to segment point clouds, such as Xie, Tian and Zhu's (2020) point-based labelling method through voxels. Xu *et al.* (2018) used a probabilistic connectivity model for segmentation. Voxel-based four planes congruent set is proposed for estimating transform by Xu *et al.* (2019) and voxel-based shape recognition in point clouds by Wang *et al*. (2016). Voxels are also used in griding and region growing methods, as explained by Li and Sun (2018).

In this thesis, Voxelization is a spatial partitioning technique that divides the whole point cloud data into 3D cubes, similar to Wu *et al*. (2013) and Cabo *et al*. (2014). These 3D cubes are called voxels. Each voxel has a length, width and height equal to the voxel size. In addition, each voxel has an index value to identify among the other voxels. All the empty voxels are removed, and the voxels are created from the lowest points derived from the first stage. The details are explained in Section 6.3.3.1 – 6.3.3.3.

6.3.3.1 Voxel Bounding Extent

Size defines the bounding extent of a voxel. Each voxel has a length, width and height that defines the voxels, as shown in Fig 6.6. The size is chosen based on the type of the point cloud. For example, quarry data with fewer details can be divided into large voxels, but urban data with many intricate details to capture should have smaller voxels. The terrain extraction grid size is the same as the voxel size to save computation time, and the search within the grid is accurate and fast. If the size is different, the search must be performed repetitively, costing more as the point cloud could be very dense.

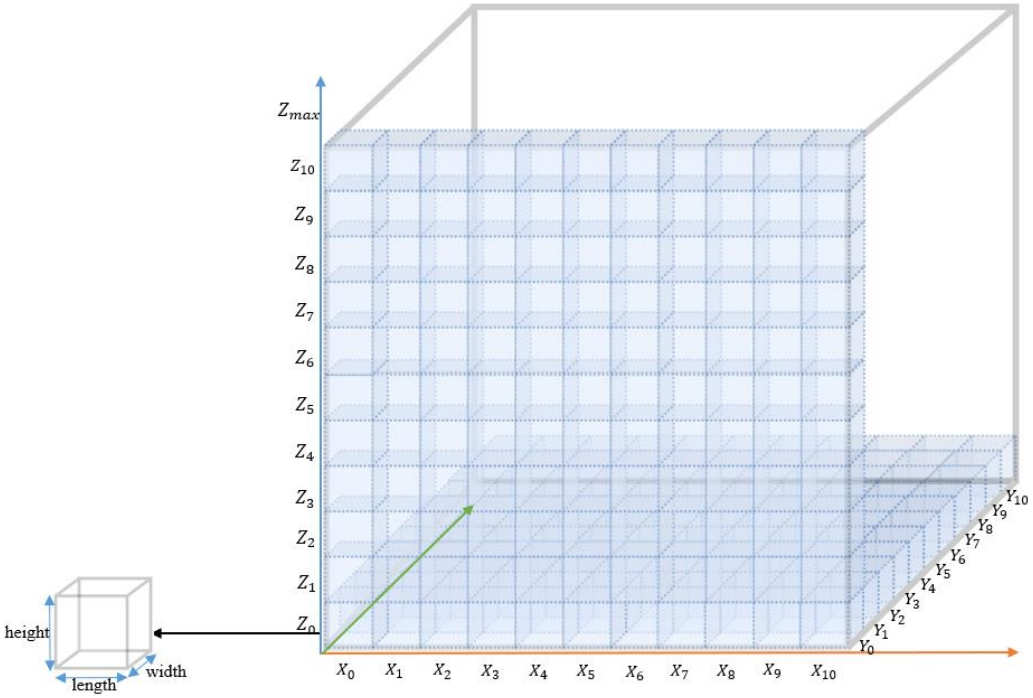


Figure 6.6 Voxel grid representation along the X, Y and Z axis

6.3.3.2 Voxel Indexing

Voxel indexing is important to identify the voxels in the point cloud. The indexing also provides the facility to capture points inside the given voxel. The voxel index is based on the voxel position in a 3D grid and is denoted by $V_{(i,j,k)}$ where i is for columns, j is for rows, and z is for layers. The columns i are along the X-axis, rows j are along the Y-axis, and layers z are

along Z-axis. The voxel index helps identify any voxel in the 3D grid and its points. Voxel indexing plays a key role in clustering for the neighbourhood.

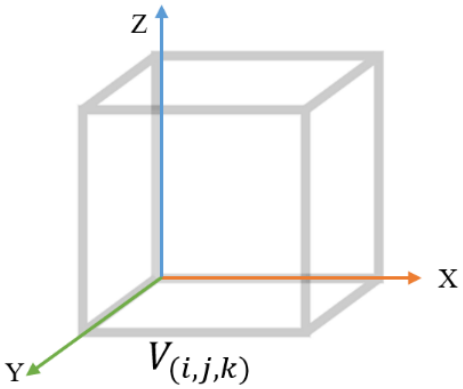


Figure 6.7 Voxel Indexing represents its position in 3D by using x,y, and z values, as shown in Figure 6.6

6.3.3.3 Generating 3D Grids

The voxels in the 3D grid are generated based on the size and indexed based on location, as shown in Fig 6.8. The voxels are generated from the resultant lowest points of Terrain Extraction for the proposed algorithm. The different colours represent the voxels on different layers.

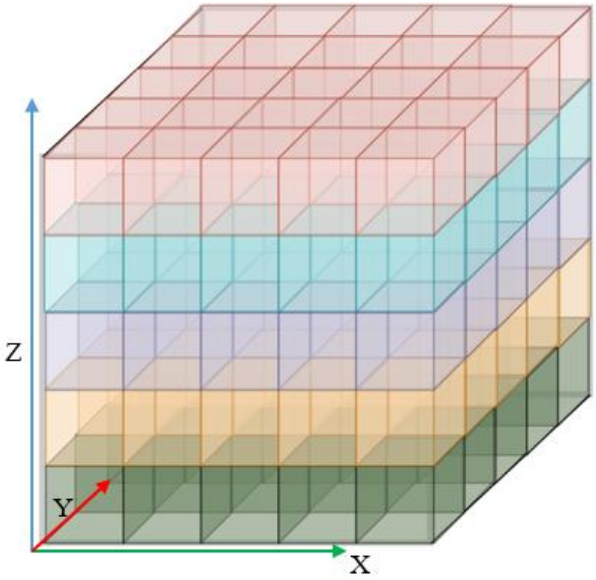


Figure 6.8 Voxel 3D grids represented by different colours

The voxels are generated using Equations 6.4 – 6.6 presented below:

$$vx = (integer)\left(\frac{(X_{max} - X_{min})}{voxelSize}\right) \tag{6.4}$$

$$vy = (integer)\left(\frac{(Y_{max} - Y_{min})}{voxelSize}\right) \tag{6.5}$$

$$vz = (integer)\left(\frac{(Z_{max} - Z_{min})}{voxelSize}\right) \tag{6.6}$$

where vx is the voxels generated along the x-axis, vy is the voxels along the y-axis, and vz is the voxels along the z-axis. X_{max}, Y_{max}, Z_{max} are the maximum coordinate bounds and X_{min}, Y_{min}, Z_{min} are the minimum coordinates in the point clouds. The voxel size is selected based on the point cloud. The voxels are generated along the three-axis; however, there are conditions when the three axes are not the same sized. For example, a point cloud of the road or railway is generally longer along one axis. Therefore, padding of the voxel is implemented to solve the problem using Equations 6.7 – 6.9. For padding the grid, an extra voxel is created as follows:

$$Ex = (X_{max} - X_{min}) - vx * voxelSize \tag{6.7}$$

$$Ex > 0\,, vx \rightarrow vx + 1$$

$$Ey = (Y_{max} - Y_{min}) - vy * voxelSize \tag{6.8}$$

$$Ey > 0\,, vy \rightarrow vy + 1$$

$$Ez = (Z_{max} - Z_{min}) - vz * voxelSize \tag{6.9}$$

$$Ez > 0\,, vz \rightarrow vz + 1$$

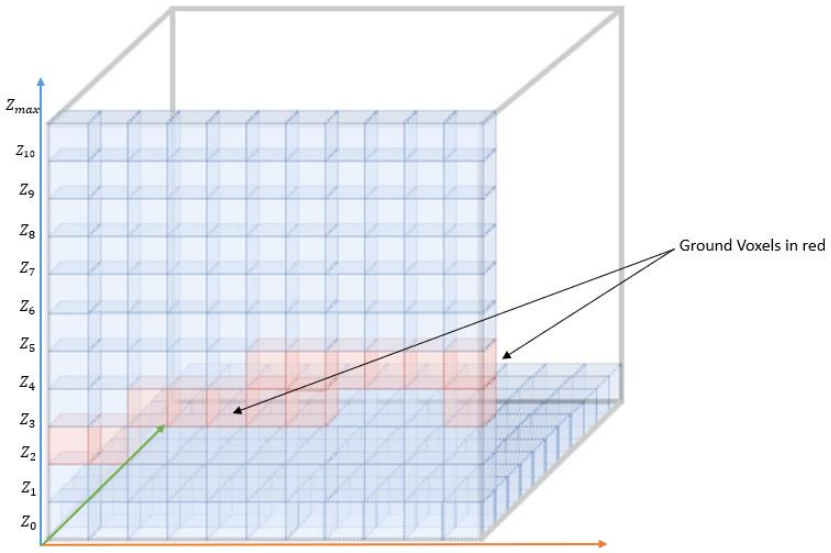
where Ex calculates along the x-axis, Ey calculates along the y-axis, and Ez calculates along the z-axis. If Ex, Ey and Ez are greater than zero, a voxel is added along the axis.

6.3.4 Seed Layer Identification

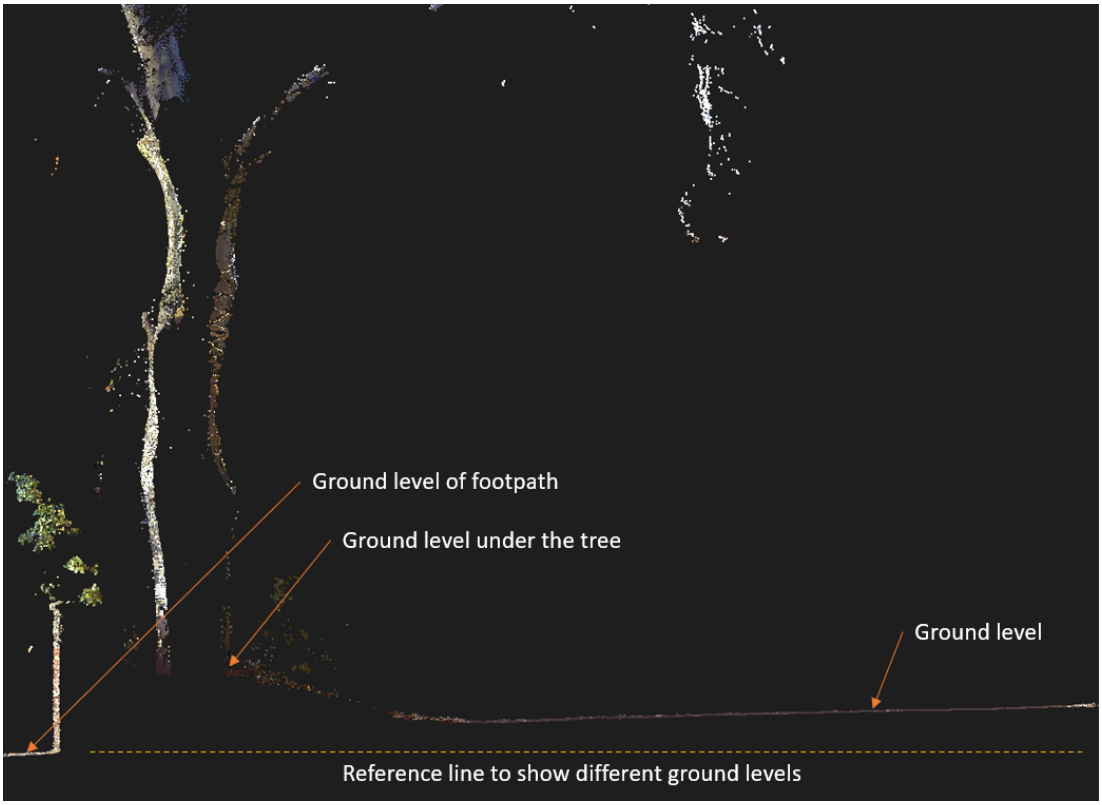
The third stage is seed layer identification—the first stage results in the lowest ground points. In the second stage, the voxels are formed from all the lowest points found as a result of stage 1. In the third stage, all the lowest ground voxels (points) are removed, and the DBH level is estimated from each lowest ground voxel to identify the seed layer voxels. The seed layer voxels play an important role in identifying trunks and poles. Sections 6.3.4.1 and 6.3.4. explains the usage and working of the seed layer in detail.

6.3.4.1 Removing the Ground Points

After implementing *Terrain Extraction*, all the lowest points are captured. Those lowest points are then included in the bounding extent of a voxel during the voxelization stage. All the grids created by *Terrain Extraction* are traversed to find all the lowest voxels with the lowest point of the grid, as shown in Fig 6.9 (a) as red-coloured voxels.



(a)



(b)

Figure 6.9 (a) Ground voxels represented by red

(b) Different ground levels shown on a cross-section of a point cloud

Red coloured voxels represent ground voxels, and everything above the red voxels is considered non-ground points. Once the ground voxels are derived, every point inside them is removed as they are considered ‘ground points’. Removing ground points saves a lot of computation time and fewer points to the cluster that belongs to the trunk or pole. Further, it will also help identify the trunks, as shown in Fig 6.9 (b) and poles that are not on flat ground.

Fig 6.9 (b) presents the example of different ground levels in a typical point cloud. Therefore, Terrain Extraction extracts these different ground levels or lowest points in this example and is classified as ground voxels. Finally, the proposed algorithm is implemented on non-ground voxels.

6.3.4.2 DBH Seed Layer

Next, the seed layer is identified after extracting all the ground voxels. The voxels are divided along the z-axis, as shown in Fig 6.9(a), to demonstrate the ground voxels in layers. In the example, the ground voxels are between Z_2 and Z_4 layer. A tree in the typical point cloud is defined by a tree trunk and a crown, shown in Fig 6.2. Compared to the crown part and trunk is composed of fewer voxels. The points on the trunk and poles are spatially separated from other trunks and poles. Therefore, a seed layer divides the tree into two parts and the poles.

The trunk diameter is measured using DBH in arboriculture for different purposes of urban and forest trees. By British standards, the DBH is 1.4 metres from the ground level. Therefore, the seed layer is calculated at 1.4 metres above the ground layer height. As shown in Fig 6.10, the seed layer, denoted by green coloured voxels calculated using DBH 1.4 metres above the ground layer, is denoted by red coloured voxels. After voxel extraction on the seed layer, these voxels are fed for the next part of the proposed algorithm: clustering. A dotted yellow line is drawn in Fig 6.9 (b) to demonstrate different ground levels and the importance of extracting the ground with slope to detect the trunks on the DBH height.

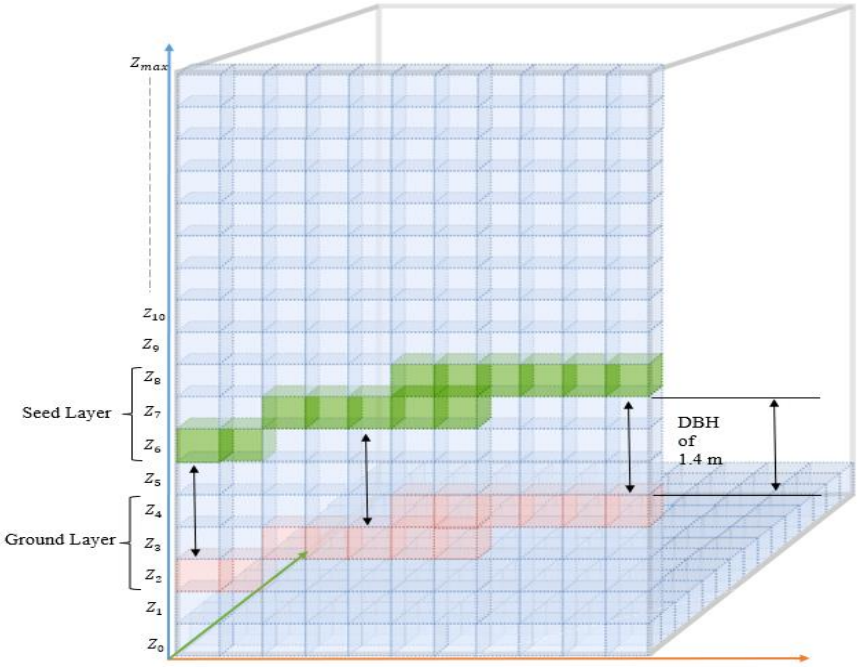


Figure 6.10 Voxels on seed layer represented in green colour

6.3.5 Clustering

The fourth stage of the proposed algorithm is clustering. Clustering is used in various fields like social networks, market behaviour analysis, risk assessment, robotics, and recurring patterns in financial transactions. In geoinformation systems, clustering divides or groups the points with similar observations for data analysis (Poux, 2020). Various authors accomplish the clustering of points in point clouds in different ways. An example of clustering is normal difference clustering by Ioannou *et al*. (2012) to process unorganized point clouds, spectral clustering by Teng *et al*. (2010) to connect each point with neighbours based on similarities, clustering based on the geometric description by Weinmann *et al*. (2017), clustering based on colour (Tazir, Checchin and Trassoudaine, 2016) and density (Aljumaily, Laefer and Caudra, 2017).

The clustering in this thesis is based on voxels. The resulting seed layer voxels are clustered based on the nearest neighbour search.

6.3.5.1 Neighbourhood Approximation and Grouping

The voxels are created in columns i, rows j and layers z along the X, Y, and Z-axis. The origin of voxelization is $V_{(0,0,0)}$ which is the three-axis minimum point. The clustering of voxels is performed by neighbourhood searching, where a voxel is denoted by $V_{(i,j,k)}$. The first step is to mark all the voxels as 0 for empty and 1 for non-empty. By quick traversing of voxels, the voxels with points are marked as 1 and voxels without points are marked as 0. All the empty voxels are removed.

Next, the search starts with the origin in all the voxels marked as 1. The voxels are searched for their eight neighbours on a single layer k. The eight neighbours are shown in Fig 6.11. If the voxel $V_{(i,j,k)}$ is searched along the x-axis or columns i and along the y-axis or rows j the searching of voxels can be divided into two types

a) edge voxels (shown in Fig 6.11)
b) corner voxels (shown in Fig 6.12)

The edge voxels are those voxels that are connected by edges like $V_{(i,j+1,k)}$, $V_{(i-1,j,k)}$, $V_{(i+1,j,k)}$ and $V_{(i,j-1,k)}$. The corner voxels are those which share a corner with the voxel like $V_{(i-1,j+1,k)}$, $V_{(i+1,j+1,k)}$, $V_{(i-1,j-1,k)}$ and $V_{(i+1,j,k)}$.

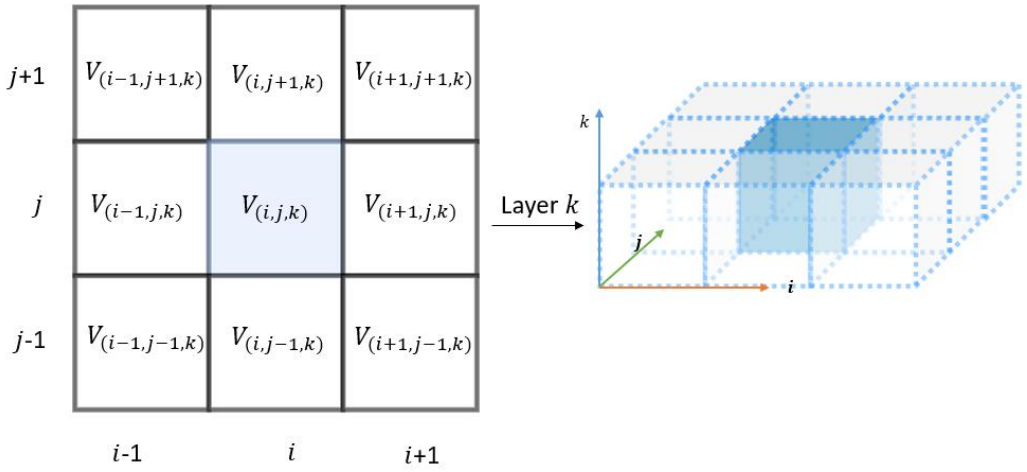


Figure 6.11 Voxel neighbour search on single k layer

The search must be extended to multiple layers as the ground has a gradient. The proposed algorithm focuses on identifying the trunks and poles on the ground with a slope. Therefore, the search is performed on three layers $V_{(i,j,k)}$, $V_{(i,j,k-1)}$ and $V_{(i,j,k+1)}$. The original layer of the voxel, plus a layer above and below, as shown in Fig 6.12.

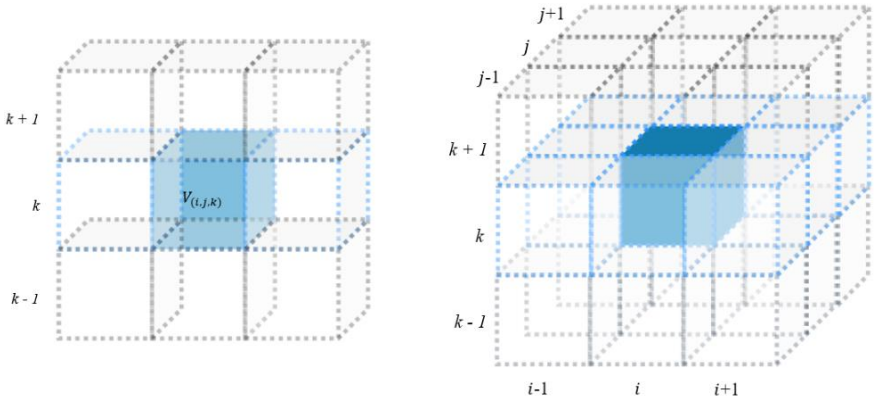


Figure 6.12 Voxel neighbourhood search on k, k+1, k-1

Every voxel shares a maximum of eight neighbours on a layer and 26 neighbour voxels when searched on three layers as mentioned above. The search must be fast and accurate; therefore, few measures are taken, as shown in Fig 6.13. The rules are if the voxel for the nearest neighbour search is the origin, then instead of searching eight neighbours, only three neighbours are searched, i.e., two edge voxels and one corner voxel. Similar rule for $V_{(i_n,j_n,k)}$ the last corner voxel in i columns and j rows. The voxels on the edge of the 3D grid search for three edge voxels and two corner voxels, as shown in Fig 6.13 (c) and (d).

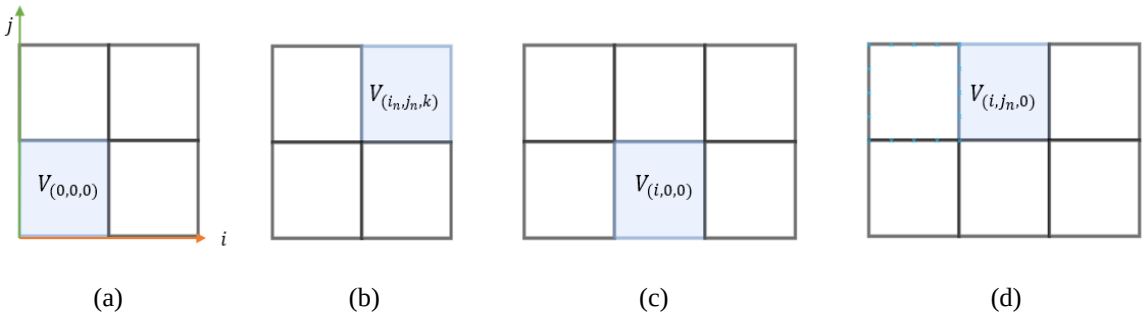


Figure 6.13 Voxel neighbourhood search

The clustering of voxels focuses on the points while removing ground points. However, the clustering results in incorrect ground removal in the case of hollow features. The main cause of hollow features is the scanning systems that capture an object's surface points. For example, any feature in the point cloud is hollow when it does not have points inside, so the lowest points inside the feature would not belong to the actual ground, as shown in Fig 6.14.

Therefore, the grouping of voxels checks for the sudden change in the Z layer direction and distance to overcome the problem. The ground layer voxels are compared with neighbouring ground voxels based on a distance threshold Th_d in the Z-axis direction. An example is shown in Fig 6.14, a vertical section of a tree displays that a tree is hollow in the middle. The terrain extraction results produce ground points around the tree with a similar z value, but the lowest point inside the tree does not belong to the ground. To overcome the threshold Th_d is implemented. Any lowest points greater than the threshold are not considered ground points.

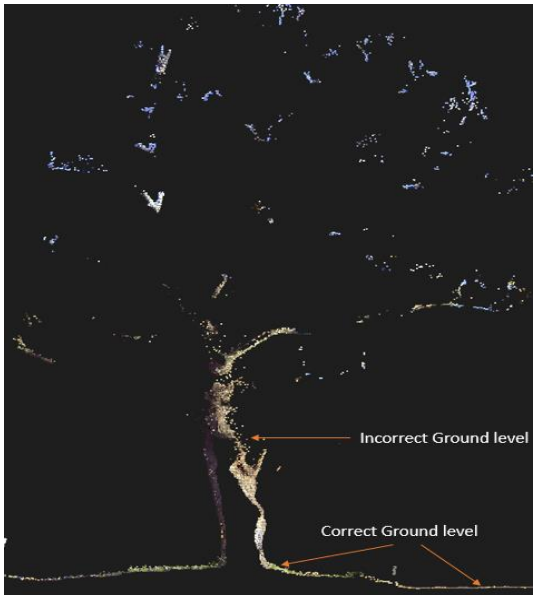
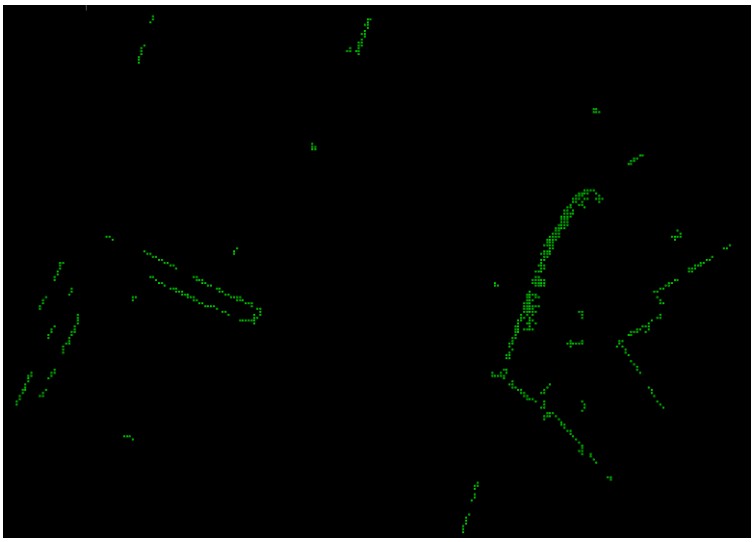


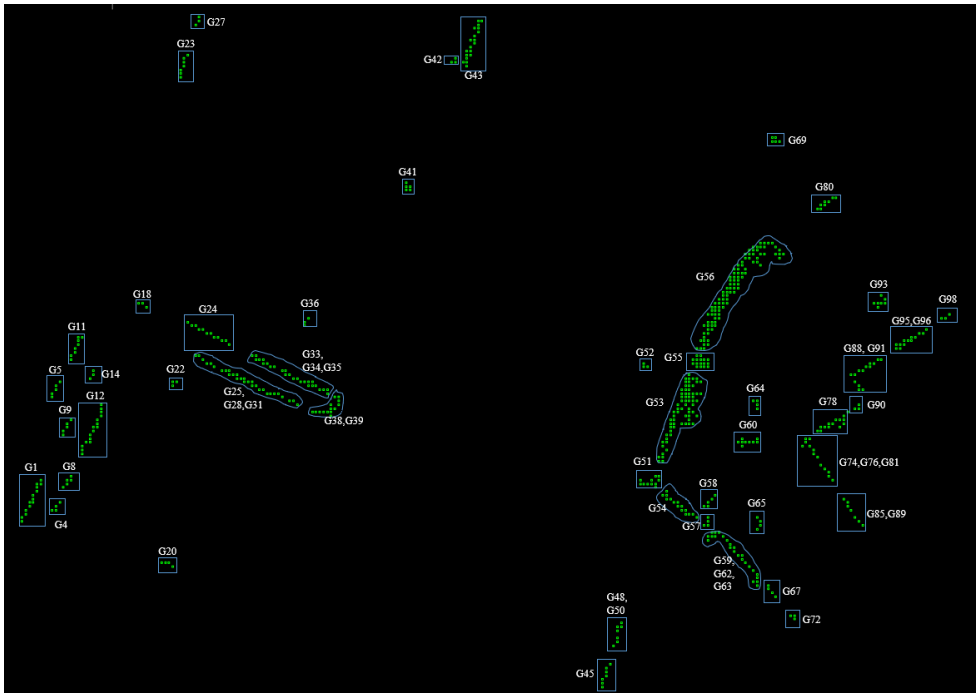
Figure 6.14 A section view of a tree in a point cloud

6.3.5.2 Cluster Groups

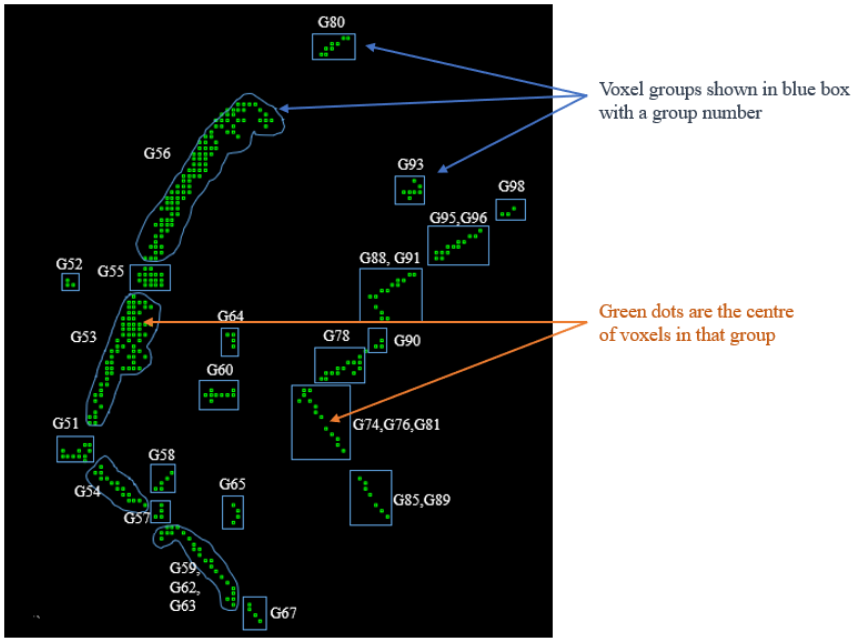
The nearest neighbour search of voxels is implemented, resulting in the voxel clusters on the seed layer. Furthermore, the voxel clusters are assigned a group number to identify the clusters. To demonstrate the clustering and grouping, Fig 6.15 (a) shows the centre of all voxels on the seed layer, and Fig 6.15(b) shows the grouping of clusters on the seed layer.



(a)



(b)



(c)

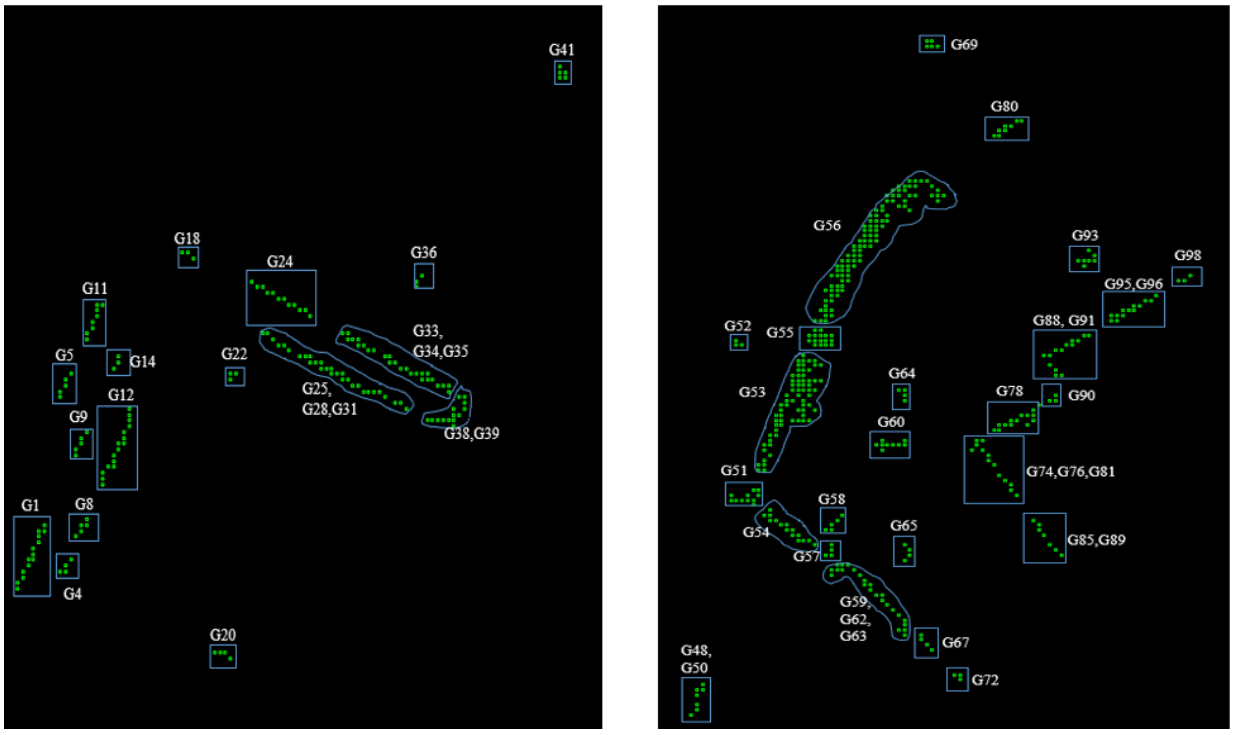
Figure 6.15 (a) The centre of the voxels, (b) Clustered voxels on the seed layer, (c) Zoomed small area of (b)

Voxels are grouped in the seed layer along the column (X-axis), row (Y-axis) and layer (Z-axis) of the voxel grid. The proposed algorithm starts from the origin to cluster the voxels, as

explained in Section 6.3.4.1. While searching in the voxel grid, every voxel is grouped into a cluster. Each cluster has a unique group number to identify it among the other clusters. The naming of groups starts from $\sum G = \{ G1, G2 \ldots \ldots GN\}$ in column vice manner. The rule for allotting the groups in the cluster is that:

a) if a current cluster has an assigned group or not; if it doesn't, it is assigned one.
b) if the nearest neighbour in the cluster has a group assigned, the current cluster will have the same group number.
c) if the nearest neighbour in the cluster has no group assigned, then the group number is assigned by an increment of one.
d) the voxels on a layer above and below are also searched and assigned a group number.

The first group is assigned as G1. Then, as the cluster progressed, the group numbers were assigned to each cluster, as shown in Fig 6.16. The cluster groups in this example are 59 groups in total.



(a) (b)

Figure 6.16 Voxels centre are clustered in groups of Fig 6.15 (b) (a) Zoomed left side (b) Zoomed right side

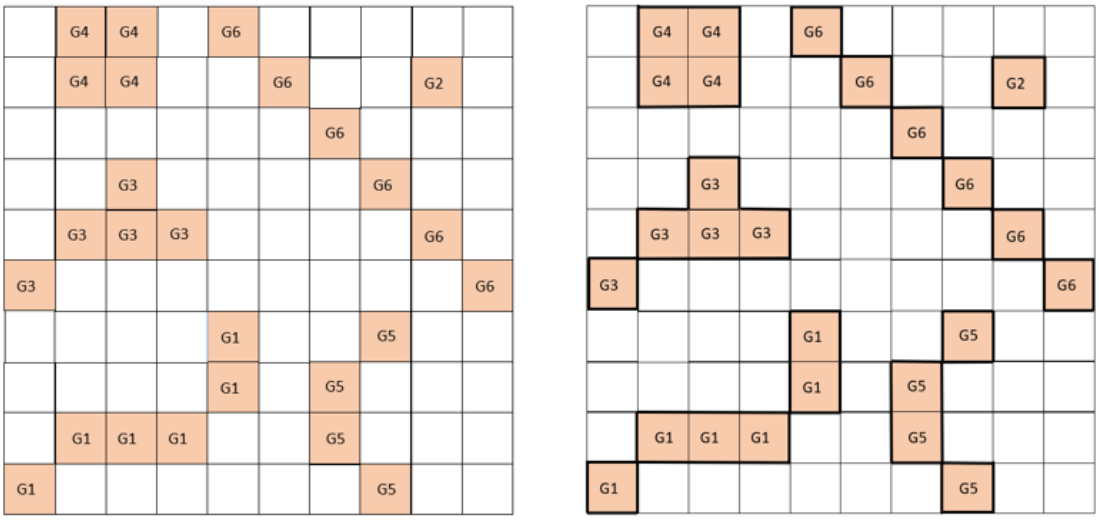
6.3.6 Extraction of Cylinder Objects

The next stage of the proposed algorithm is the extraction of potential clusters that belong to a trunk and pole. Then, the clustered group from the previous stage are analysed and filtered. These filtered groups are potential groups belonging to the trunk or pole, as presented in Sections 6.3.6.1 and 6.3.6.2.

6.3.6.1 Extraction of Potential Voxels

In the example shown in Fig 6.16, there are 59 cluster groups. The clusters on the seed layer could belong to many point cloud features, such as a building, trees, vehicles, shrubs and ghosts. To make the proposed algorithm faster and more efficient the voxel clusters are filtered based on

a) Area
b) Compactness
c) Number of voxels (NOV) in a cluster



(a) (b)

Figure 6.17 (a) Hypothetical example of clusters on a single layer
(b) Shows the perimeter of the clusters to calculate the area and compactness

Fig 6.17 shows a hypothetical example of the cluster groups along the x and y axis on the seed layer. The potential trunk and pole clusters have less area than the clusters that belong to buildings or other objects. Each voxel cluster is counted for the number of voxels in it. If the number is more than the threshold Th_{nov} the group is not considered. The threshold is knowledge-based and selected based on different test point cloud data for trunk and pole clusters.

The area and compactness of clusters belonging to a trunk or pole are comparatively less than other voxel groups. As both are cylindrical shapes, a circle is the best shape fitted along the x and y axis. The two attributes are computed for each voxel group. For the area A, the following parameters are calculated X_{min}, X_{max} along x-axis, Y_{max}, Y_{min} along y-axis and n is the number of voxels in each voxel group as shown in Equation 6.10.

$$A = n * ((X_{max} - X_{min}) * (Y_{max} - Y_{min})) \tag{6.10}$$

$$dia = \sqrt{(X_{max} - X_{min})^2 + (Y_{max} - Y_{min})^2} \tag{6.11}$$

$$CA = \pi * \left(\frac{dia}{2}\right)^2 \tag{6.12}$$

$$Compactness = \frac{A}{CA} \tag{6.13}$$

The second attribute is compactness, defined in Equation 6.13. For compactness, dia the diameter and the circle area CA are calculated by Equations 6.11 and 6.12. The compactness value is higher when a voxel group is close to the circular shape. Therefore, if the threshold Th_{com} for the compactness is close to 1, it is a perfect circle, or if the compactness is close to 0, it is not a circle. The compactness is calculated using equations 6.8 – 6.10. An example of voxel groups on a hypothetical seed layer is shown in Fig 6.18. In Fig 6.18, groups G6 and G1 are non-circle whose compactness is less than the threshold Th_{com} and G5 and G4 compactness are greater than the threshold Th_{com} which means the G5 and G4 are clusters that belong to a trunk or pole.

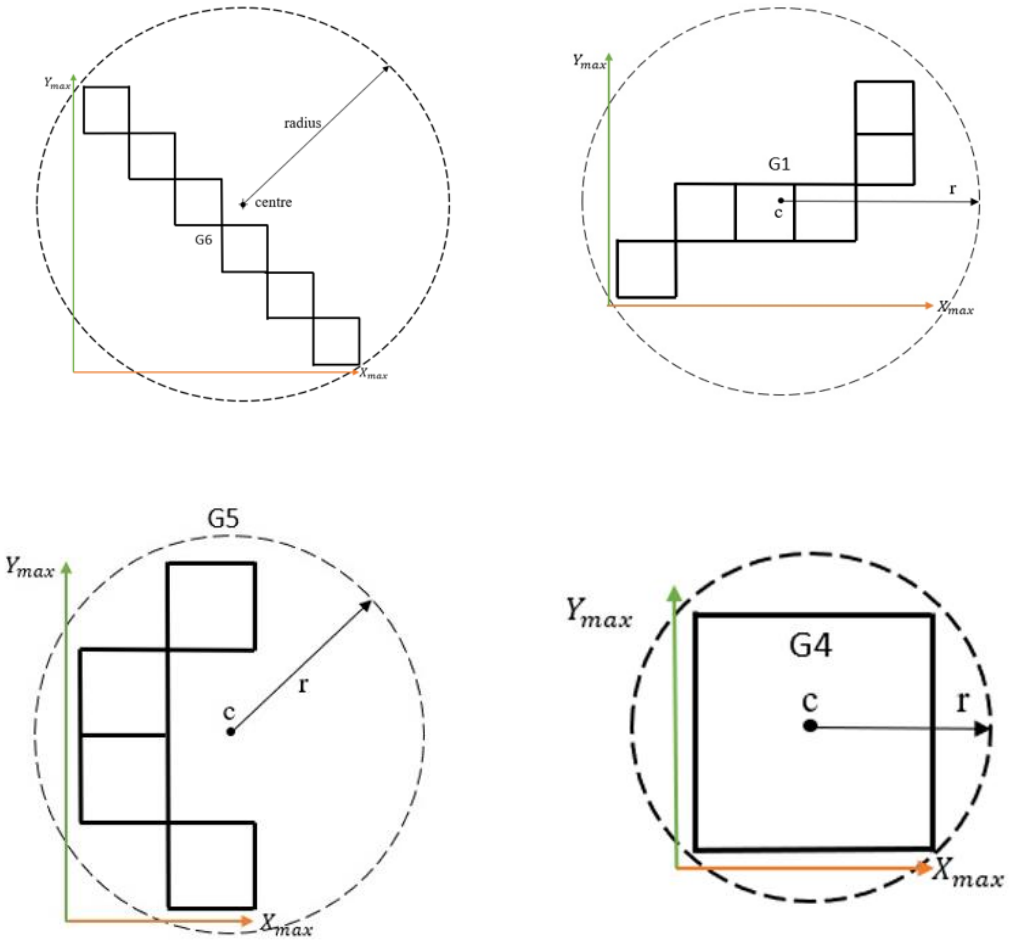


Figure 6.18 Area calculated to measure compactness

Following the calculation of all attributes for each cluster, the potential voxels are selected:

1) if the number of voxels is smaller than Th_{nov} i.e., Th_{nov} is a given threshold to limit the number of voxels and
2) if the compactness is greater than Th_{com} i.e., Th_{com} is a given threshold for compactness.

When implemented in the example shown in Fig 6.15, results are shown in Fig 6.19. As a result, the cluster groups are reduced from 59 groups to 11 potential groups. Therefore, area compactness and the number of voxels of each cluster play an important role in the proposed algorithm for increasing efficiency and robustness.

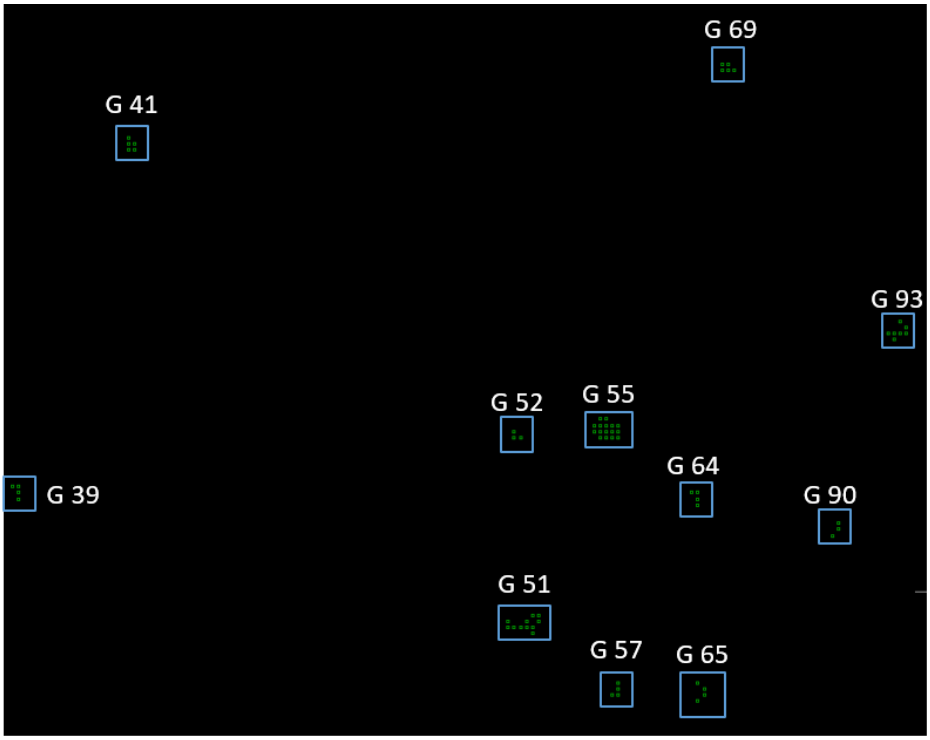


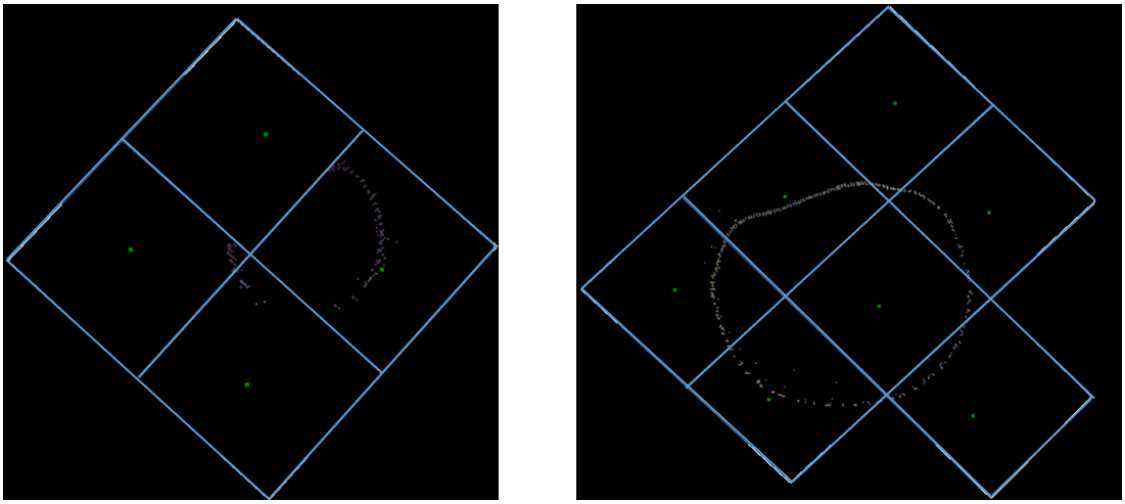
Figure 6.19 Potential Clusters belonging to tree or pole

6.3.6.2 Circle Fitting

The next stage is circle fitting on the potential groups. The trunks and poles can be differentiated from other objects based on cylindrical shapes. As the clusters can be on different layers and have up to three z-layer values (the clustering is implemented on k, k+1 and k-1), the circle is fitted in 2D, considering only the x and y values.

Various researchers implement the techniques of circle fitting methods. For example, Pratt circle fitting is more an extension of Kasa's (1976) circle fit which is more biased towards the small circle or incomplete arc. However, Kasa's is not robust as Pratt's method. Taubin circle fit is like Pratt's method but is comparatively faster and more accurate. Hence, for this reason, the combination of methods is implemented for circle fitting in this thesis.

The 2D circle fitting is implemented using a hyper fitting algorithm by Chernov Nikolai (2012), a combination of Pratt's (1987) and Taubin's (1991) circle fit algorithms.



(a) (b)

Figure 6.20 (a) Circle fitting on a pole (b) Circle fitting on a trunk

An example of a hyper circle fitting on the voxels belonging to the possible tree trunk and a pole is shown in Fig 6.20 (a)(b). Further, Fig 6.21 demonstrates the cluster groups filtered by circle fitting in green and the rejected groups in red. From 11 potential groups, circle fitting results in 6 cluster groups.

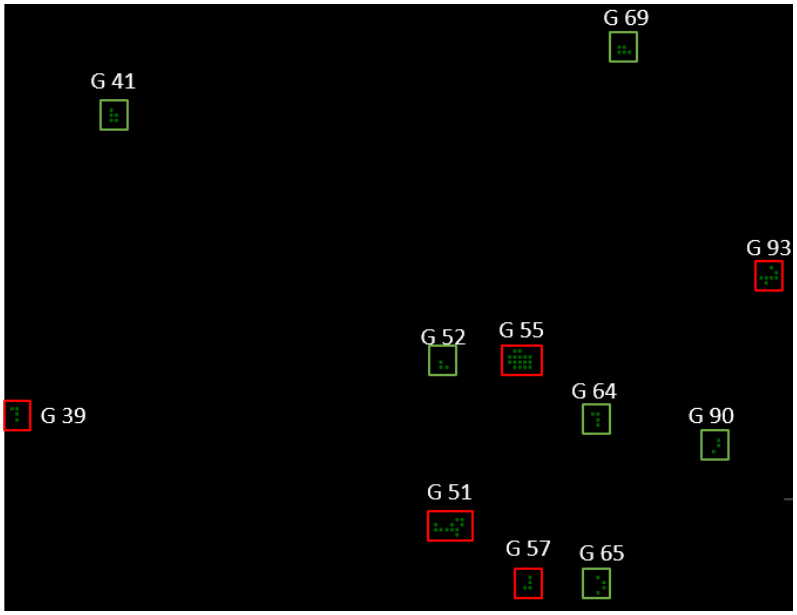


Figure 6.21 Potential trees and poles in green after circle fitting algorithm

6.3.7 Tree and Pole Classification

The last stage of the proposed algorithm is classification. The section focus on classifying the voxel groups that are filtered as potential groups by earlier stages of the algorithm. The classification is demonstrated in the example shown in Fig 6.21 with the six resulting cluster groups. The point cloud has various objects similar to the trunk and pole, such as a moving person, a part of a building, a tripod used for scanning the point cloud data or bushes/shrubs. The human eye can detect these objects much more quickly based on knowledge and understanding of the feature's shape; however, the proposed algorithm needs to learn to identify and classify the objects based on certain rules. Each voxel cluster is analysed based on the rules as follows:

1) Semantic rules
2) Shape-based rules
3) Intensity and colour-based rules

6.3.7.1 Semantic Rules

Semantic rules are based on the prior knowledge of the objects, i.e., trunks and poles. To differentiate between trunks and poles, voxel dimensionality S_{VD}, the adaptive radius of the voxel cluster S_{AR} and upward region growing S_{RG} are applied in this section.

1) Voxel Dimensionality Analysis

The first rule applied is Principal component analysis (PCA) to evaluate each voxel group's dimensionality S_{VD} (Demantké *et al.*, 2011). PCA is widely used to analyse and reduce the dimensionality of datasets with interrelated variables while retaining the variation present by principal components (Dubey, 2018). In addition, PCA is used as a popular approach to identify the structure of points as linear, planar or volumetric. For example, PCA was used for shape identification by Yang and Dong (2013), Kang *et al.* (2018), Shi *et al.* (2018), Yokoyama *et al.* (2013) and Yang *et al.* (2015), whereas PCA was used for dimensional identification by Demantké *et al.* (2011), Monnier, Vallet and Soheilian (2012) and Huang and You (2015).

In this thesis, PCA is applied in Chapter 5 for edge detection. In this chapter, PCA is applied to analyse the dimension of the points inside the voxels. The first step is to calculate the covariance matrix using Equation 6.14

$$C = \frac{1}{N}\sum_{i=1}^{N}(p_i - \bar{p})(p_i - \bar{p})^T \quad (6.14)$$

where p_i is the i^{th} point in N , and $\bar{p}$ is the mean calculated by $\bar{p} = \frac{1}{N}\sum_{i=1} p_i$ and N is the number of points in voxels. The eigenvalues λ_1, λ_2 and λ_3 are obtained from C where λ_1 is the largest variant than λ_2 and smallest is λ_3. If $\lambda_1 > \lambda_2 > \lambda_3 > 0$, PCA is applied to determine the dimensions inside the voxels as linear, planar and scattered. The dimension analysis symbol is S_{VD}. The voxel is considered linear if $(\lambda_1 - \lambda_2) / \lambda_1$, planar if $(\lambda_2 - \lambda_3) / \lambda_1$ and scattered if $(\lambda_3) / \lambda_1$. The following rules are applied:

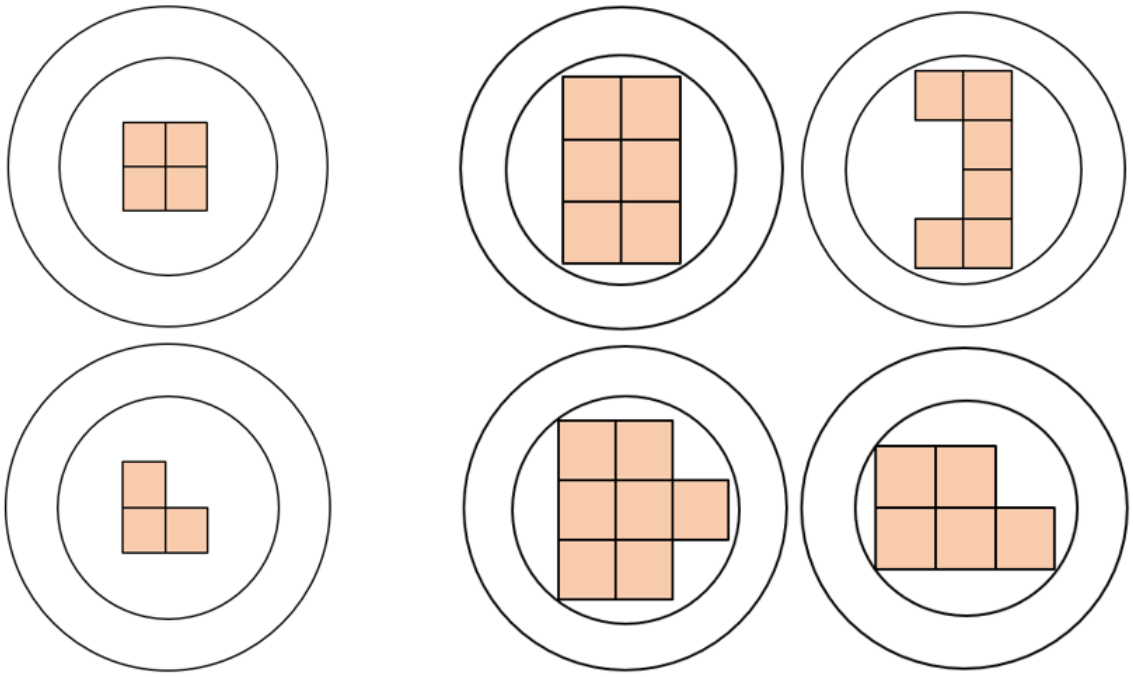
- If cluster groups have scattered data – rejected,
- If cluster groups have all planar data – considered,
- If cluster groups have all linear data – considered,
- If cluster groups have linear and planar data – considered,
- If cluster groups have linear and scattered – rejected.

The volumetric voxels are discarded as the trunks and poles never have scattered data. Only voxels with tree foliage and noise have the scatter dimensions. All the linear and planar dimension voxels are considered, and the rest are discarded.

2) Adaptive Radius of Trunk and Pole

The second rule applied is the adaptive radius S_{AR}. A circle is applied from the centre of the voxel clusters, and the radius is stored to compare. As poles are man-made objects, the radius for poles is fixed, as shown in Fig 6.22 (a). After testing on various datasets, the pole clusters

are always a cluster of four voxels or less. On the other hand, the voxel cluster of tree trunks is variable. Fig 6.22 (b) shows that the radius of trunks is usually larger than poles.



(a) (b)

Figure 6.22 Voxel groups of (a) Poles (b) Trunks

3) Upward Region Growing

The third rule applied is an upward region growing S_{RG} on the seed layer of potential voxel clusters. The upward region growing starts from the seed layer L_s along Z-axis, analysing all the voxels clusters vertically on each layer. Poles are always symmetrical and have an almost equal number of voxel clusters on each layer. In contrast, the trees start scattering due to the presence of foliage and branches. Fig 6.23 shows the example of the voxel clusters on the pole and trunk from the seed layer (red) compared with the clusters above the seed layer.

If the voxel cluster growing above the seed layer L_s is equal to the number of voxels in the seed layer identified as a pole. If the voxel cluster growing upward from the seed layer L_s is increasing, i.e., the number of voxels in L_{s+1}, L_{s+2} are increasing, then it is identified as a tree trunk.

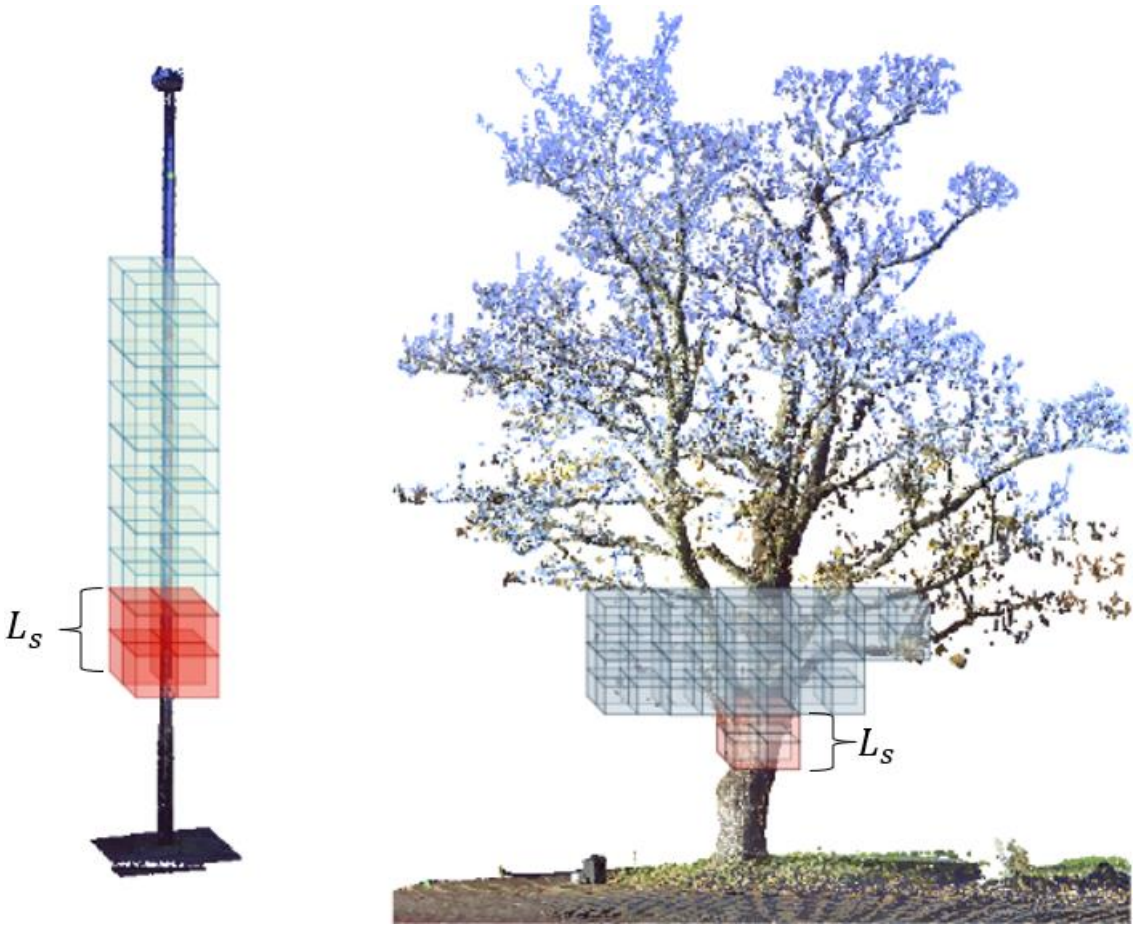


Figure 6.23 Upward region growing from the seed layer L_s shown for pole and tree

6.3.7.2 Shape-Based Rules

Shape-based rules are based on the shape of the objects, i.e., trunks and poles. To differentiate between trunks and poles, object isolation criteria S_{ISO} and the distribution area of voxels S_D are applied in this section.

1) Object Isolation Criteria

The first shape-based rule is the isolation criteria S_{ISO}. Many researchers used the isolation criteria in the study for identifying objects, especially poles (Arastounia & Oude Elberink, 2016; Li *et al*., 2019; Ordóñez *et al*., 2017; Li *et al.*, 2016; Wu *et al.*, 2017). The pole is usually isolated from other road features. If the potential voxel group are separated and isolated, it is classified as a pole.

2) Distribution Area of Voxels

The second rule is the distribution area of the voxel cluster S_D. If observed orthogonally along the x and y-axis, the voxel clusters will have a distribution that could indicate whether the voxel belongs to the pole or a trunk. If the voxel distribution is more, it is classified as a trunk or if smaller than it is classified as a pole.

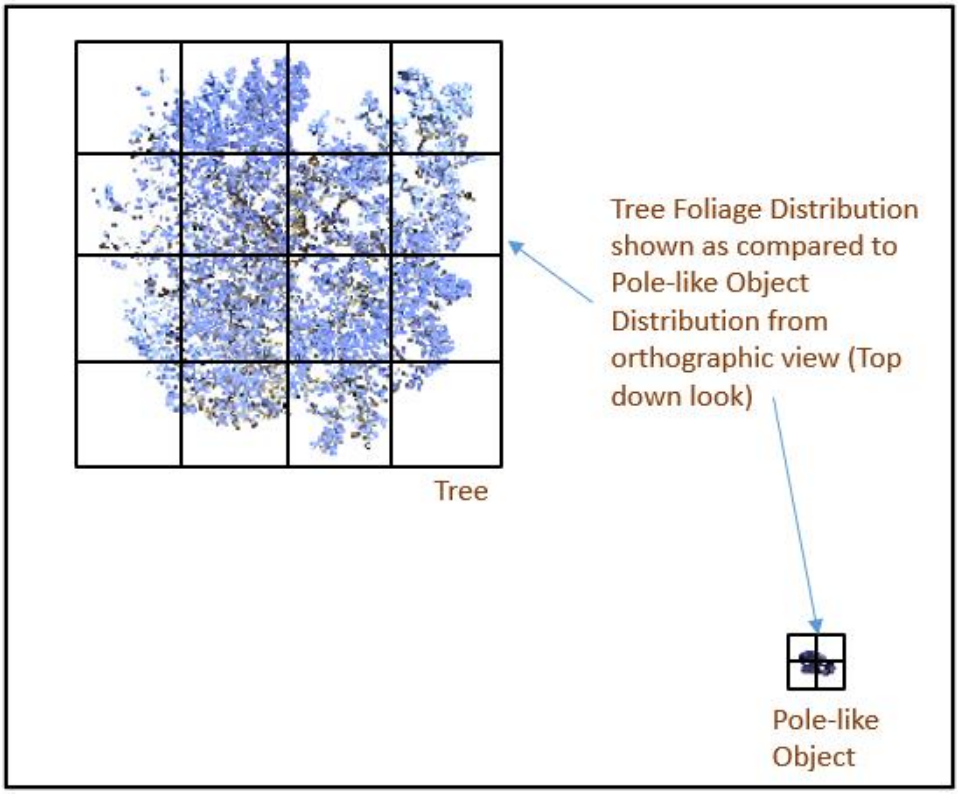


Figure 6.24 Distribution of Tree and pole cluster

6.3.7.3 Intensity and Colour-Based Rules

The differentiation between the trunk and pole rule is based on intensity and colour values. The sum of all the intensities S_I and colours S_C are calculated in potential voxel groups. The pole as a man-made object is more reflective and therefore has more intensity value, whereas the trunk has lower intensity. Hence, colours are often a shade of brown for the trunk, whereas poles have more distinctive colours than trunks.

The result of potential voxels is shown in Fig 6.25 after applying trunk and pole classification. Implementing the same example shown in Fig 6.21 with six potential groups, the groups are reduced to 3 voxel groups. Section 6.4 demonstrates the proposed algorithm implementation presented in this section on commercial software 3D Vision.

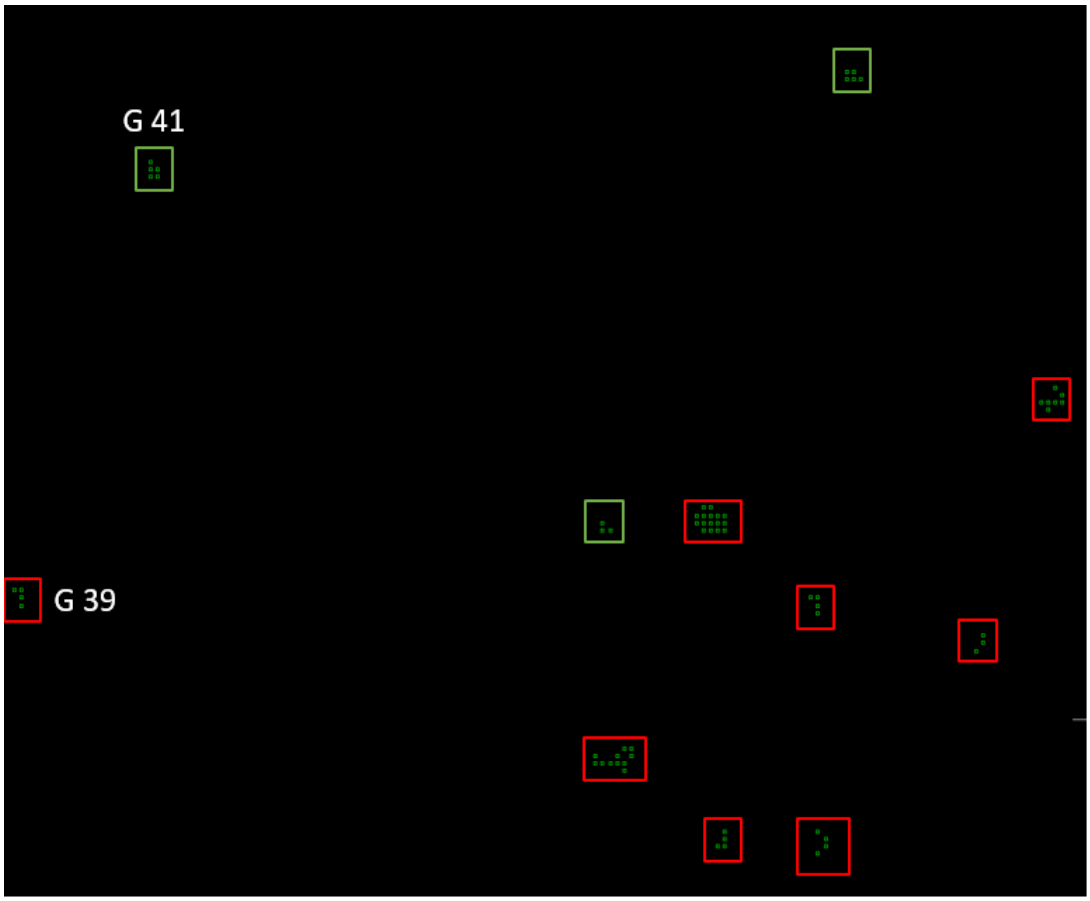


Figure 6.25 Result of classification and true trees and pole detection

6.3.8 Pseudo Algorithm

The proposed algorithm can robustly identify cylindrical shapes with various radii and tilt angles in the point cloud. The workflow of the proposed algorithm is shown in Fig 6.3. In this thesis, the voxel size is fixed for any input point cloud. After many iterations, the voxel is selected as 0.2 metres for the proposed algorithm. A size bigger than 0.2 did not have enough details captured in it, and less than 0.1 was time-consuming.

Algorithm for Trunk and Pole Detection

Input: Point cloud =$P(x, y, z) = \{1, \ldots \ldots Ni\}$.

1: For algorithm 0.2, voxel size is selected because a higher number affected the tree girth detection as the hollow ground beneath can produce no centre point. At 0.2, it worked efficiently for almost all data types.

2: Terrain extraction - Scan each voxel from the bottom and find the first one with data, which is recorded as ground level

3: Whole data is viewed vertically to find all the points as ground and add to the data.

4: It starts from Z0 to Z6 or Z8 as the seed layer. According to DHB, the trunk width is usually taken at 1.4 metres.

5: Seed voxels are clustered on the same layer and a layer above and below by neighbour search. The clusters are then marked.

6: The DBH does not consider the seed layer's points if the ground is hollow. Therefore, the seed layer clusters are extended on the same layer.

7: The clusters use a circle fitting algorithm to find whether the actual circle can fit. As a tree might not be a perfect circle, only 70-80% circle is considered.

8: All the groups with near circles, compactness and area are selected. Then, the centre and the radius are calculated.

9: Each cluster is then classified as trunk, pole and others

Algorithm for Trunk and Pole Classification

Input: Voxel clusters

1: Check if the voxels are within the circle radius threshold input by the user (Adaptive radius)

2: Check for vertical upward region growing on the seed layer

If (voxels are present)

{

3: Check the voxel cluster's RGB and intensity (input by training data)

4: Checks for the area distribution of voxels

5: PCA analysis and Standard deviation (for checking distribution linearity)

6: Isolation criteria

}

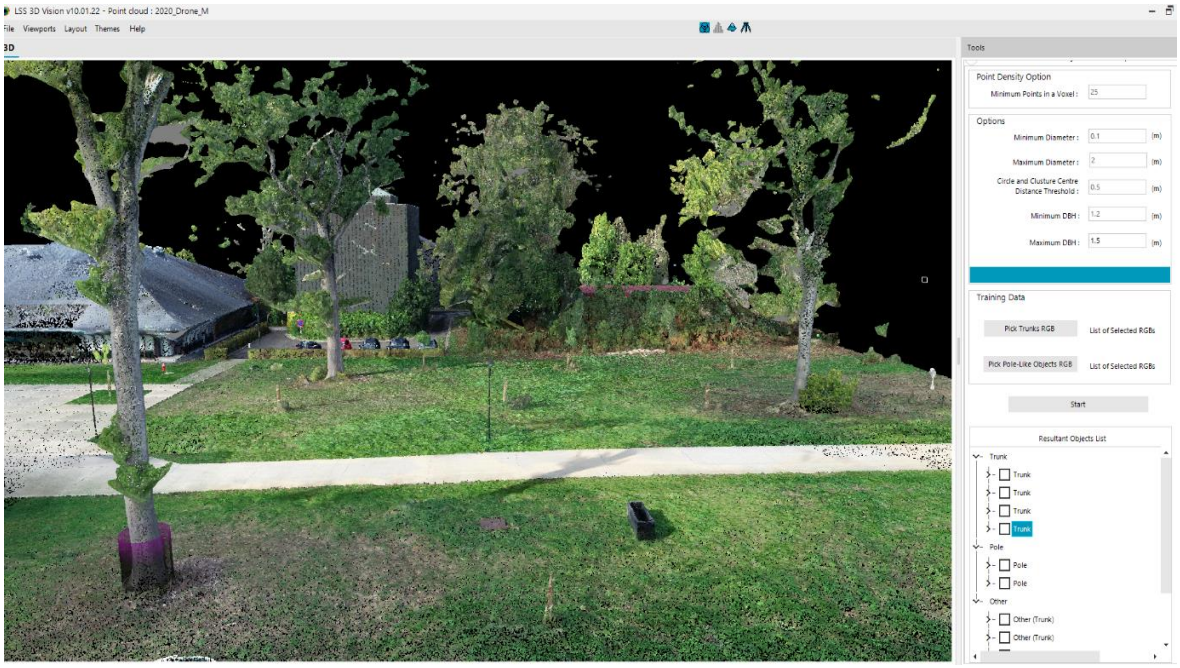
6: Result: Pole, trunk or other

6.4 Proposed Algorithm Implementation on Commercial Software

The popular and most captured point clouds are for urban sites. These data cover the street scene or a city block. However, the level of detail can be very complex and processing these data is challenging. This section aims to demonstrate the implementation of the proposed voxel-based algorithm on commercial software, "3D Vision".

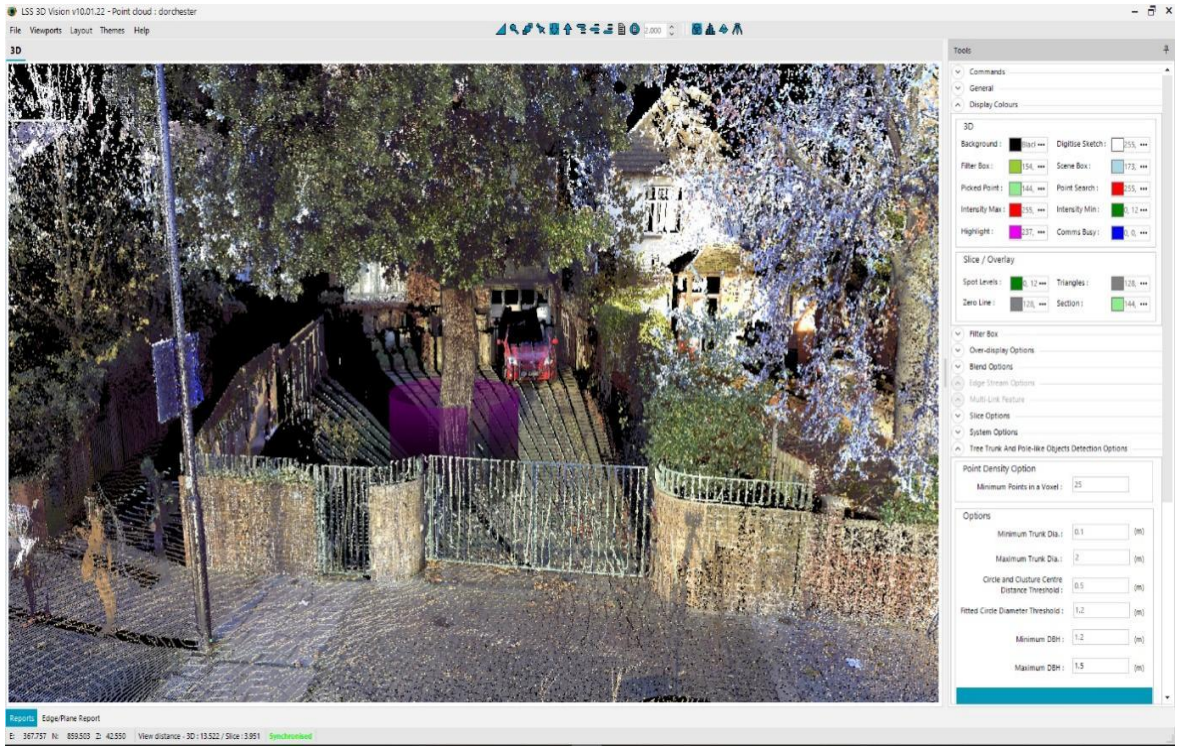
For demonstrating the implementation, the following datasets are used 1) Dorchester and 2) Car park. In Section 6.4.1, the key features of the proposed algorithm are presented on the software 3D Vision and followed by the software's UI parameters to control and manage the algorithm according to the point cloud type presented in Section 6.4.2. Next, Section 6.4.3 shows the software development environment used to implement the algorithm. Finally, in

Section 6.4.4. demonstrate the real-world scenarios and challenges encountered by users, and the proposed algorithm can handle and extract the cylindrical features efficiently.



(a)

(b)



(c)

Figure 6.26 Trunks and poles detected in 3D Vision shown in a fuchsia-coloured cylinder (a) trunk detected (b) pole detected (c) trunk detected

6.4.1 Key Features

This section presents the key features of the 3D Vision software for triggering and using the proposed algorithm. The trunks and poles are detected by clicking on a start button under the "*Tree Trunk and Pole-like Objects Detection Options*".

The algorithm is then triggered and starts the process; meanwhile, the users are shown a progress bar indicating that the algorithm is taking place in the background. Once the algorithm finishes, it lists the number of trunks, poles and other objects detected in the point cloud. The list presents a centre and radius for each detected cylindrical object. These centres and radii are then sent to the LSS DTM survey for modelling by users (surveyor and civil engineer).

How are the cylindrical objects detected in point clouds?

- It starts with terrain extraction as the separation of ground and non-ground points are essential for further process.
- All the non-terrain data is segmented using voxels (After many iterations, the voxel is selected as 0.2 metres for the proposed algorithm. The size bigger than 0.2 did not have enough details captured in it, and less than 0.1 was time-consuming).
- The seed layer is detected by implementing the DBH.
- The clustering of voxels takes place on the same layer and a layer above and below.
- The compact ratio and circle fitting algorithm are applied.
- Results in all the cylindrical objects.

The results of cylindrical objects are not very useful to the user as cylindrical objects in a typical point cloud can be anything. Therefore, proper classification is required for the users to be able to model the feature extracted. The classification divides all detected cylindrical objects into 1) Trunks, 2) Poles, and 3) Others. The classification starts by

- Semantic rules – voxels dimension, adaptive radius and upward region growing.
- Shape-based rules – isolation criteria and distribution of voxels.
- Colour-based and intensity-based.

6.4.2 System Operations

This section describes the commands in the 3D Vision software of LSS. For trunk and pole detection, a list of parameters is set. If users do not change and select, it will be set to default values. The appropriate parameters may change from the point cloud of a city scene to the other point cloud of an urban scene in residential development.

Algorithm Settings – The proposed algorithm in the commercial environment allows the users to set the parameters to find the best results in given data sets. The parameters enable flexibility for the user to use the algorithm according to their requirements. For example, a user might be interested in just the trunks so they can specify the parameter to accommodate that; on the other hand, a user might be just interested in pole-like structures. The recommended default parameters are shown in Fig 6.27.

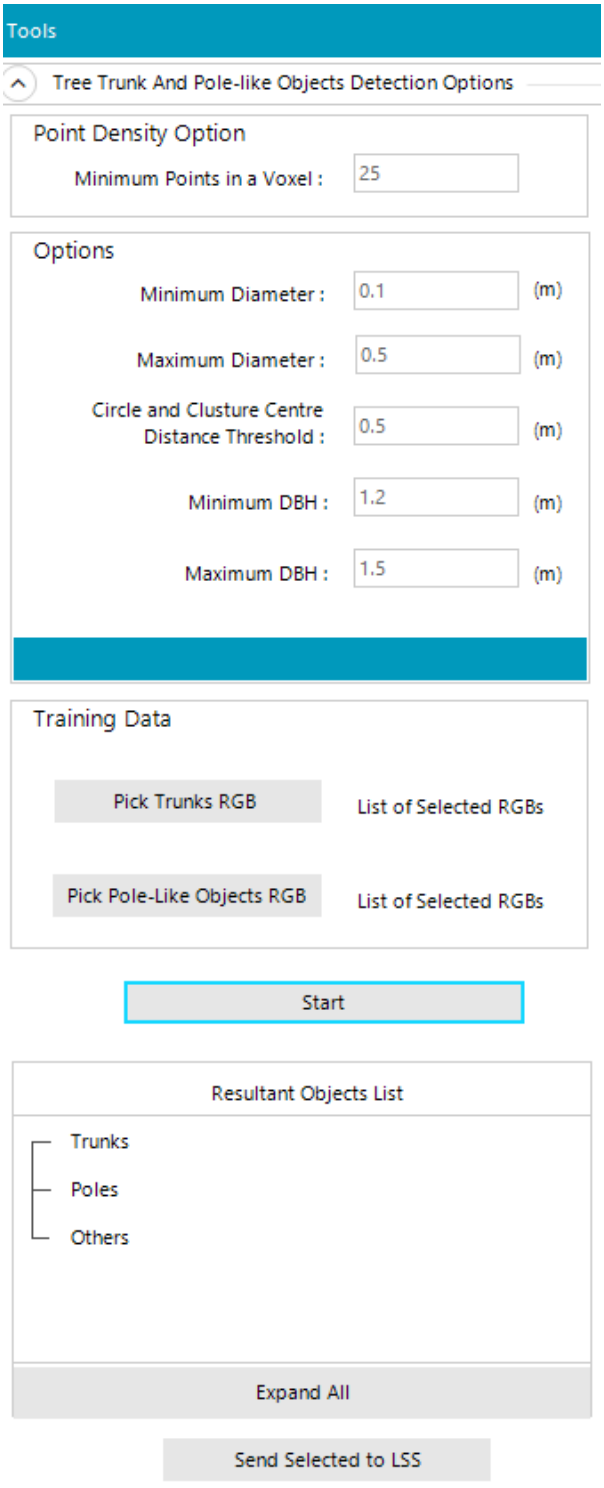


Figure 6.27 User-controlled options for the proposed algorithm in 3D Vision

Setting Parameters – Figure 6.27 demonstrate the parameters that could be set according to the point cloud used to detect the trunks and pole-like objects. For these, users must select the parameters in UI.

The first parameter is 1) ‘Minimum Number of Points in a Voxel’ for handling point densities inside the voxel. This parameter allows the trunks and pole-like objects to be detected even with minimum point density. The laser scanners collecting point cloud data sets use laser beams that fall on the object’s surface and return the points. The objects close to the scanner have full coverage, i.e., high-density points, and the farthest object surface has less coverage, i.e., low-density points. Therefore, point cloud points have variable density and distribution of points. One of the major problems with the existing methods is that they are unable to detect trunks and poles where the point densities are low. The parameter ‘minimum number of points in a voxel’ will allow the user to set the number as low as possible to detect variable densities.

The other options are 2) ‘Minimum Diameter’, 3) ‘Maximum Diameter’, 4) ‘Circle and Cluster Centre Distance Threshold’, 5) ‘Minimum DBH’, and 6) ‘Maximum DBH’.

The parameters minimum and maximum trunk diameter allow users to control the width (cylindrical) of the detected objects. The trunks generally have variable widths, whereas the pole-like structures have similar widths. Depending on the type of the point clouds, the user can either have mature or new trees. This parameter will help differentiate between mature trees to conserve and new trees for a user such as a tree surveyor.

The next parameter is distance thresholds. The threshold specifies the distance between a fitted circle on the clustered group of points and the actual centre of the group. The fitted circle on the clustered group could either be very large or very small. Therefore, this parameter is used to control the trunks that are detected. For example, if the distance between the fitted circle and cluster centre is large, it is either an anomaly or a very large tree, whereas if the distance between the fitted circle and cluster centre is small, it is a comparatively compact tree trunk.

Lastly, two parameters control the Diameter Breast Height (DBH) of the tree trunk measured by surveyors and civil engineers. The British standard DBH is 1.4 metres above ground level (Measuring Trees · The Tree Register, 2022). This parameter will help the user specify the range above ground level to measure the trunk and pole diameter at that height.

The next option is two buttons 1) Pick trunks RGB and 2) Pick pole-like objects RGB. This parameter is used to train the algorithm with the set of RGB present in that typical point cloud. RGB is very powerful and is used to differentiate between different features.

The last section is the Resultant Objects list. All the cylindrical objects detected in the point clouds are listed in this section and are classified into three groups (a) trunks, (b) poles (c) others. This classification helps the user to choose the object of interest. Furthermore, the user can choose the items and send them to LSS for creating a DTM survey by using the "*Send Selected to LSS*" button.

6.4.3 SDE

The software development environment (SDE) used for the proposed algorithm is Microsoft Visual Studio 2022 version 17.4.4, the language is C# (pronounced as C sharp) version 10 and .Net Framework 4.8. The algorithm is implemented in the back-end, which users access as the front-end interface, as shown in Fig 6.28.

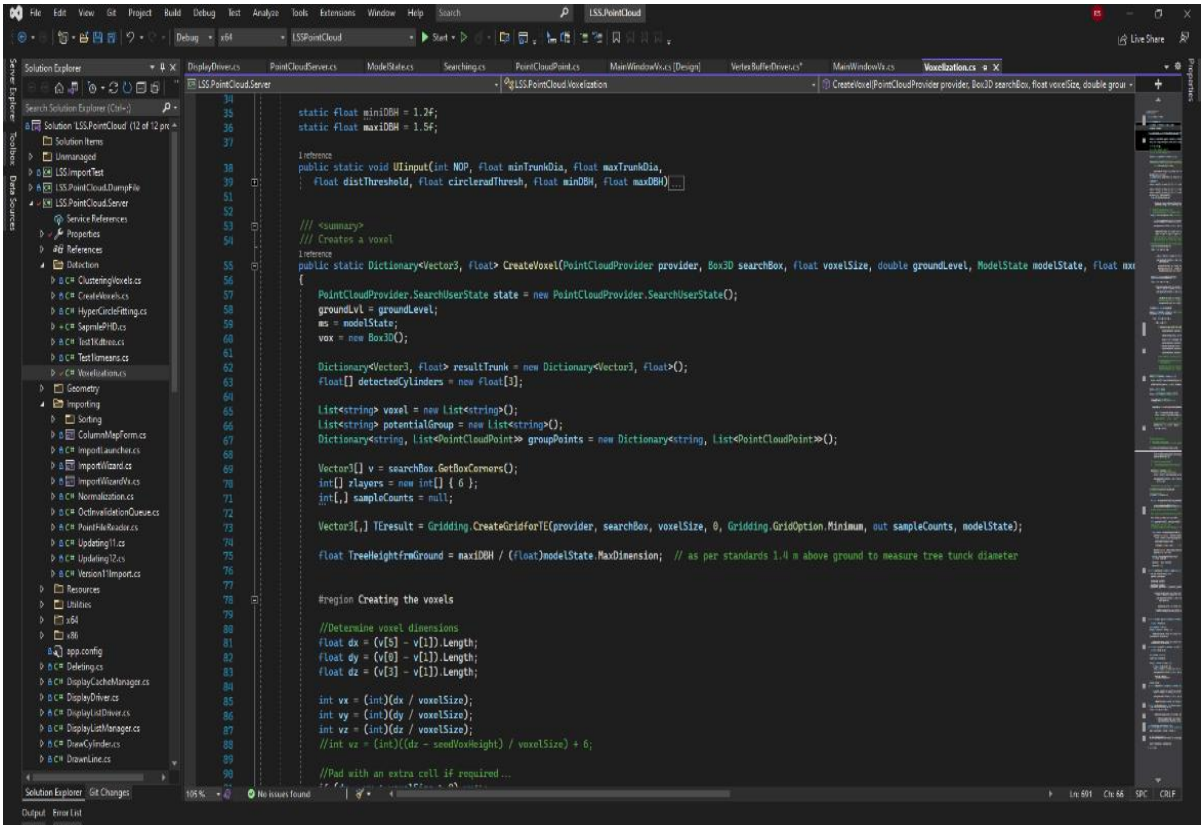


Figure 6.28 Visual Studio 2022 used for implementation of the proposed algorithm

6.4.4 RealWorld Scenarios

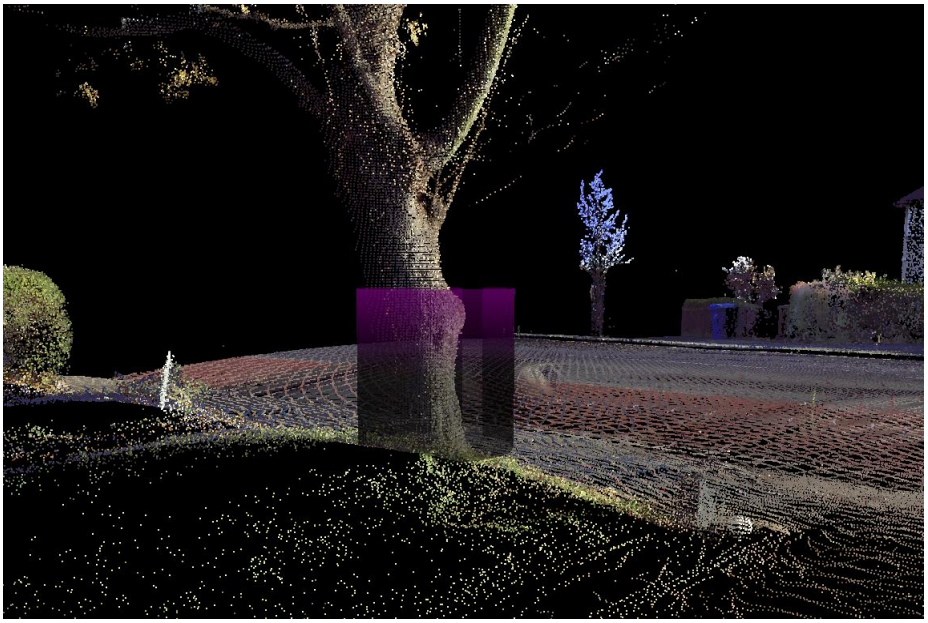
This section demonstrates how effectively the proposed algorithm works to detect the trunks and pole-like objects in real-world scenarios, such as the presence of gaps, detection in slope, low point density, and objects closer to the trunks. An example of real-world data is shown in Fig 6.29. In addition, this section aims to demonstrate that the proposed algorithms can be used to overcome the existing methods' challenges highlighted in Section 6.2.7.

Figure 6.29 A typical real-world user data in 3D Vision

6.4.4.1 In the Presence of Gap

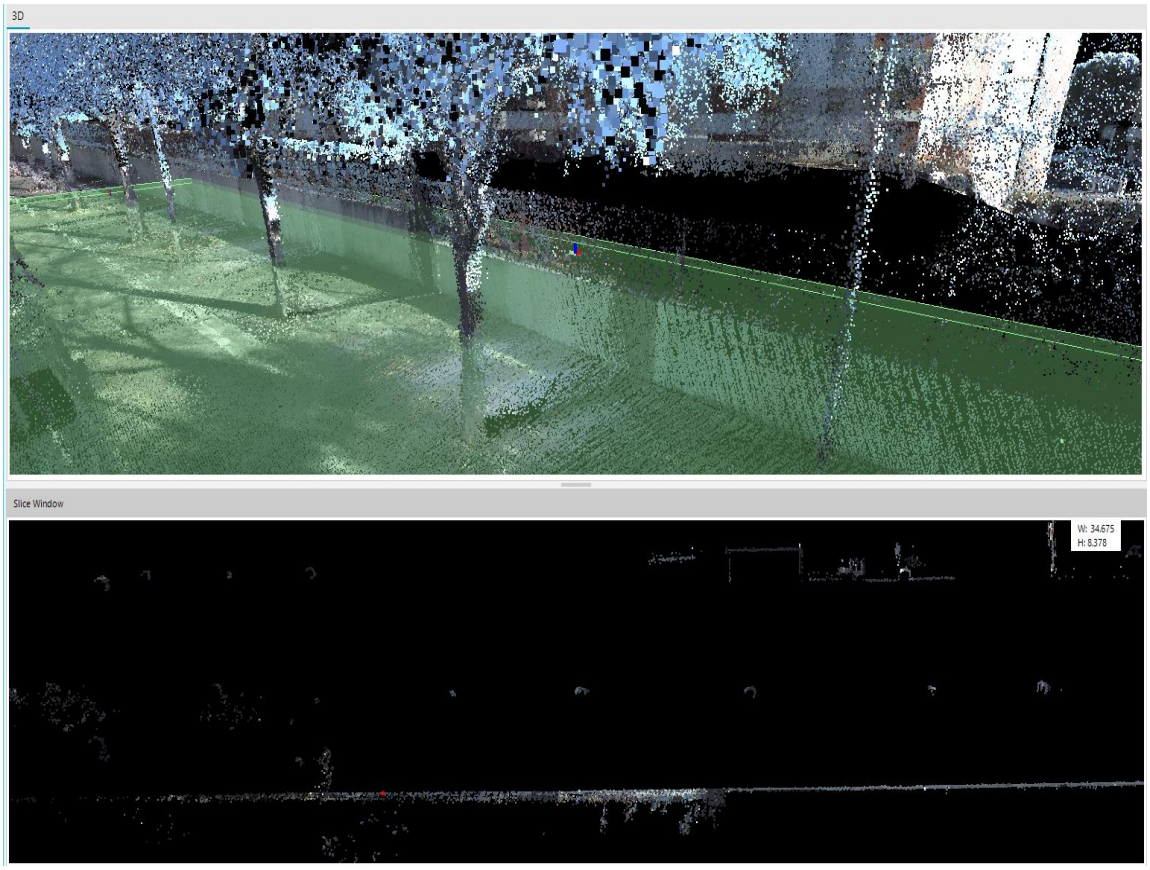
Figure 6.30 illustrates the presence of a gap or data missing. This is very common due to the nature of laser scanners. As the laser scanner picks the surface of objects and depending on the scanner position, often the objects are not fully captured, i.e., only part of the object visible from the direction of the scanner is captured. The examples of half trunk and pole are shown in Fig 6.30. To overcome such problems, the proposed algorithm uses terrain extraction in order to capture all the points on the same level, which then highlights the gaps. These are

further analysed within voxel clusters to determine the gap/missing points by fitting the circle and calculating the compact ratio (the compactness close to a circle). For example, Fig 6.30 (a) shows the trunk that has been detected (as a fuchsia colour cylinder is present around it) that has a hollow ground. To show the details of the captured ground, Fig 6.30 (b) and (c) presents the data from a section through the trunk. The proposed algorithm was able to detect the trunk with zero points below it.



(a)

(b)



(c)

Figure 6.30 Trunk detection (a) shows the hollow trunk, (b) shows a section through the trunk in (a), and (c) shows the series of trunks and poles with the horizontal section through it

6.4.4.2 Detection on Slope

Figure 6.31 illustrates the presence of a slope. Another challenge of existing methods is that they assume that all the trunks and pole objects will be on flat ground. Whereas in the real world, this is not the case; the reason includes terrain being on a slope or the ground under the trunk being covered in a pile of leaves (depending on which time of year the point cloud is captured). To overcome such problems, the criteria minimum and maximum DBH are very effective as the seed layer is between them. The example is shown in Fig 6.31 (a), where the scanner has not properly captured the trunk points due to the wall in front or the pile of leaves. Fig 6.31 (b) shows a section through this tree, showing the trunk above ground level.

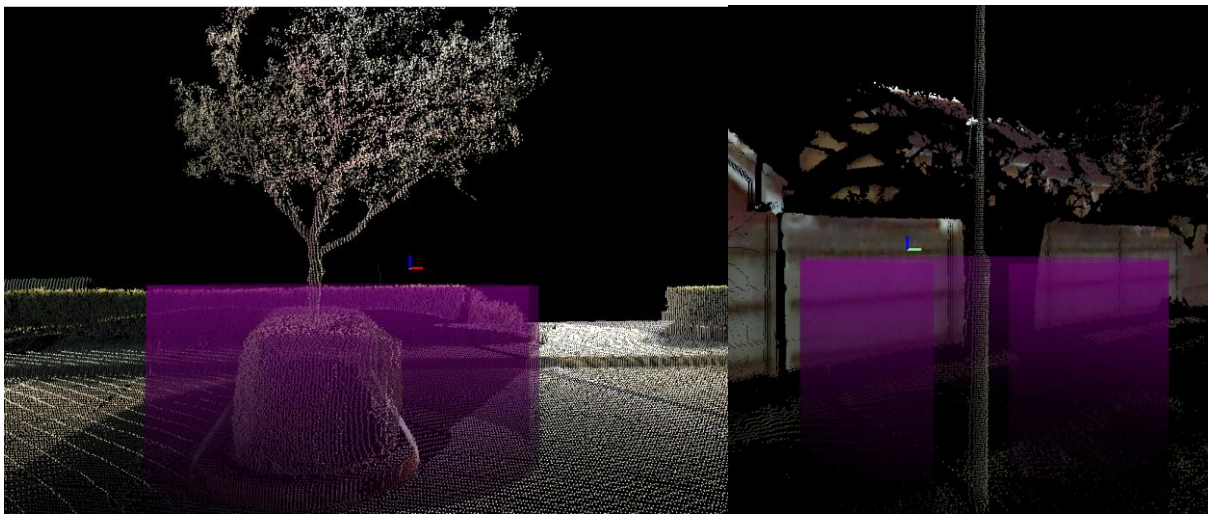
(a)

(b)

Figure 6.31 The trunk is present slightly above the ground in (a) and (b)

6.4.4.3 Low Point Density

One of the most common challenges of existing methods is they are unable to identify pole-like objects and trunks in cases of low-density data. The point density option (minimum number of points in a voxel) is implemented to overcome this problem in the proposed algorithm. When performing clustering, the point density specifies the number of points within each voxel. The example shown in Fig 6.32 demonstrates that the proposed algorithm is able to detect the trunk Fig 6.32 (a) with a point density of 5 and a pole example in Fig 6.32 (b) which is detected with a point density of 6. Hence the point density option provides the flexibility to the user to choose the number of points that they want to be detected by the proposed algorithm.



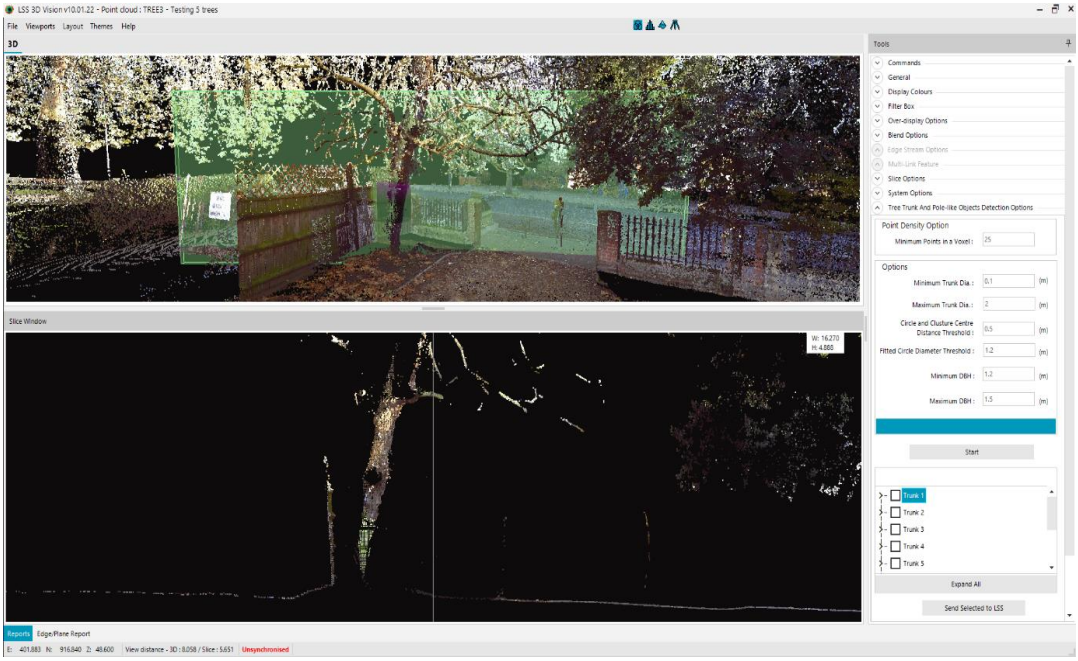
(a) (b)

Figure 6.32 (a) Trunk detected with a point density of 5, (b) Pole detected with a point density of 6

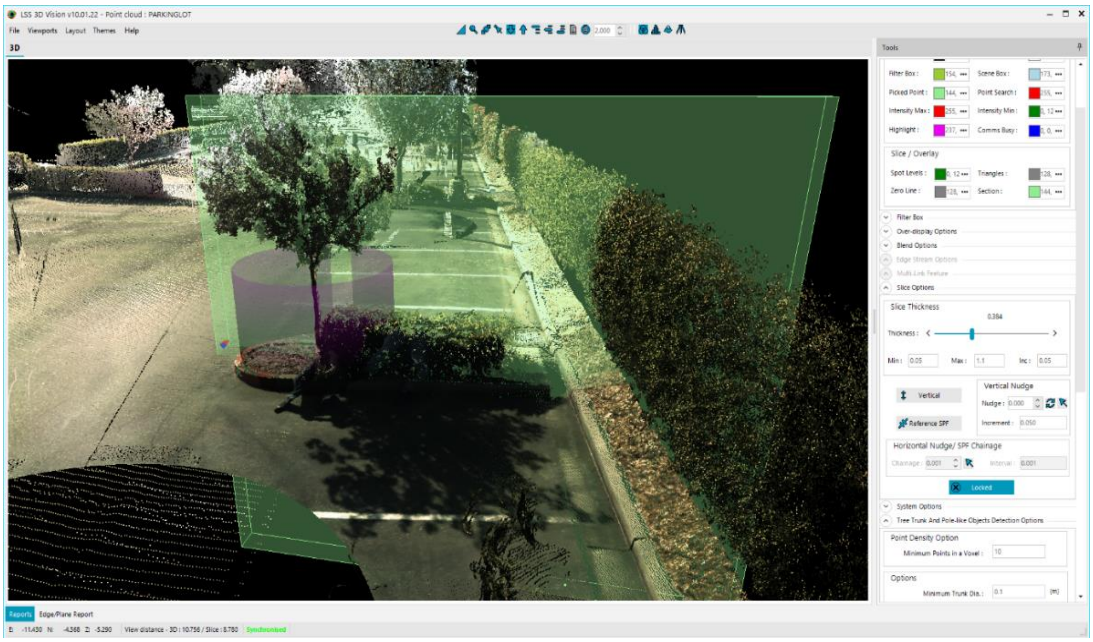
6.4.4.4 Objects closer to trunks

The proposed algorithm is able to detect the trunks and pole-like objects closer to other objects in point clouds. One of the challenges of the existing methods is that the detected objects that are closer to other features, i.e., not isolated, are not detected. Non-isolated objects are difficult to find as the points are to be differentiated between the required feature and others. The proposed methods of clustering and the training data of the objects allow for the differentiation

between two objects. The example in Fig 6.33 (a) shows that the proposed algorithm could detect a trunk closer to a fence. A similar example, in Fig 6.33 (b), shows that the proposed algorithm was able to detect the trunk close to shrubs or vegetation near it. A detailed section view of Fig 6.33 (b) is shown in Fig 6.33 (c).



(a)



(b)

(c)

Figure 6.33 Examples of trunk detected that is (a) closer to a fence, (b) closer to vegetation and (c) vertical section of (b)

6.5 Evaluation and Validation of Proposed Algorithm

This section evaluates and demonstrates the proposed voxel-based algorithm for precision and recall quality for the number of trunks and poles detected. Further, the detection speed and classification accuracy are analysed and compared with the existing methods.

6.5.1 Test Datasets

Both terrestrial and aerial laser scanners capture datasets used for testing. The FARO scanner model FOCUS 350 is used. The focus scanner ranges up to 350 metres for long-range measurements, and the measurement speed is up to 976,000 points/second. Focus has integrated GPS and Glonass, allowing detecting positions (Focus - FARO® Knowledge Base, 2016). The resolution of the scanner can also be changed. LiDar laser scanner, Leica RTC360

3D Laser Scanner captures point cloud data for up to 130 metres with 2 million points/second measurement speed. Leica scanner has multi-sensors GPS, compass, height sensor and dual-axis compensator (Leica RTC360 3D Laser Scanner, Leica Geosystems, 2018). The test dataset includes an urban scene or city block (trunk and pole structures). Data sets that have been used to evaluate the proposed algorithm, as shown in Fig 6.34 (a), (b) and (c), are:

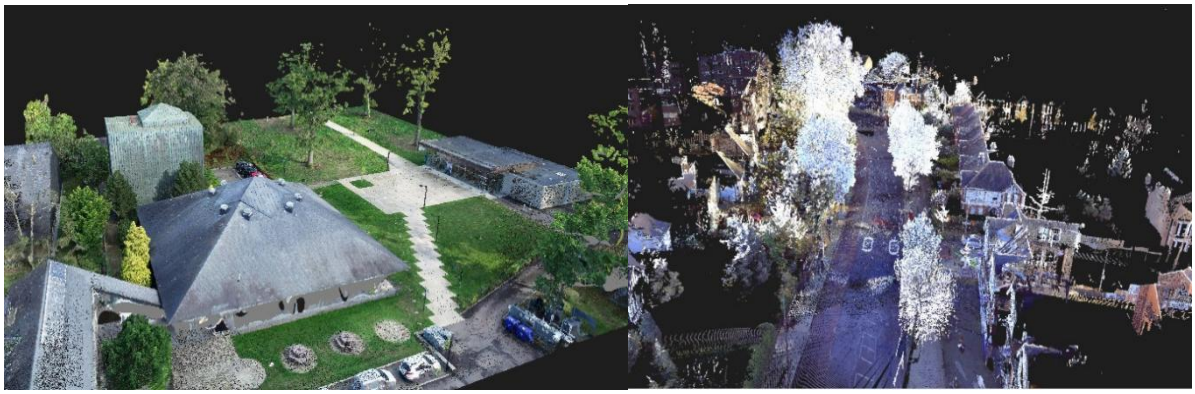
(a) **Chateaudo LiDar data set** was captured by Faro and has 27.1 million points.

(b) **Drone data set was** captured by photogrammetry and aerial LiDar and has 20.2 million points. It is secondary data set by Zegaoui (2018) downloaded for the evaluation of the proposed algorithm.

(c) **Dorchester data set** was captured by Leica and has 161 million points.

The scanned datasets consist of points in 3D (x, y, z) along with each point's R, G, B and an intensity value.

(a)

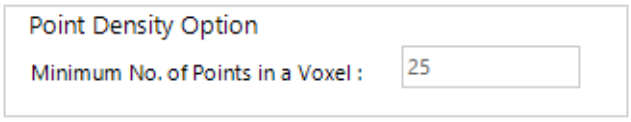


(b) (c)

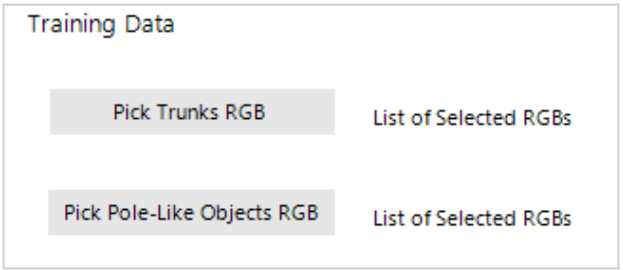
Figure 6.34 Datasets (a), (b) and (c) used to test and evaluate the proposed algorithm

6.5.2 Computation Parameters

This section demonstrates the parameter that can be set and controlled by users for the performance of the proposed algorithm. Figures 6.35 and 6.36 show the default values of the parameters in the commercial environment. The parameter settings allow the flexibility of detecting the trunks and poles, which could differ according to the different point cloud data sets.



(a)



(b)

Figure 6.35 User-controlled parameters to control the proposed algorithm

The “Point Density Option” parameter is shown in Fig 6.35 (a). Point density is important because it will allow the users to detect cylindrical objects in various ranges of density. The laser scanner beams pick a higher density of closer objects and a lower density of distant objects, i.e., the trees or poles that are picked far off could have lower point densities. The higher-density trunks and poles are easier to detect than low-density ones (as the algorithm will have enough points to detect the trunk/pole). In addition, this is one of the great challenges of existing methods to be able to find trees and poles with lower densities. Hence, this parameter helps users specify the minimum number of points in a voxel in order to detect trunks and poles with very low densities (which helps pick the trunk/pole even with minimum points shown in Section 6.4.4.3). Practically testing the minimum number of points in voxel parameters on various point clouds, the default value is set to 25, and the lowest density on a trunk/pole was 5.

The next parameter is "**Training Data**”, shown in Fig 6.35 (b), which allows the user to train the proposed algorithm according to the RGB values captured by the scanner. This group contains two buttons 1) Pick trunks RGB and 2) Pick pole-like objects RGB. This parameter is used in the classification of pole objects and trunks. The point cloud RGB can vary due to many factors like environment, weather and obstacles, especially when captured outdoors. Therefore, colours are not always realistic; however, they can be used to classify different features. For example, a tree can be white, purple or green, which depends on the time of day, weather and reflection etc.

Therefore, to overcome this problem, these parameters will allow the user to tag the objects and define the colours to identify them. The process is simple when the user triggers the pick trunks RBG or pick pole-like objects button, and it will allow the user to navigate within a scene in the point cloud and identify the tree trunks or pole-like objects by clicking on it. This process will register the colours for trunks and pole objects as training data which is then used with other criteria to classify all detected objects in three (trunks, poles and other) categories. Further, this option allows the user to detect the tree and pole-like objects in various point clouds with different coloured objects as it is not restrictive.

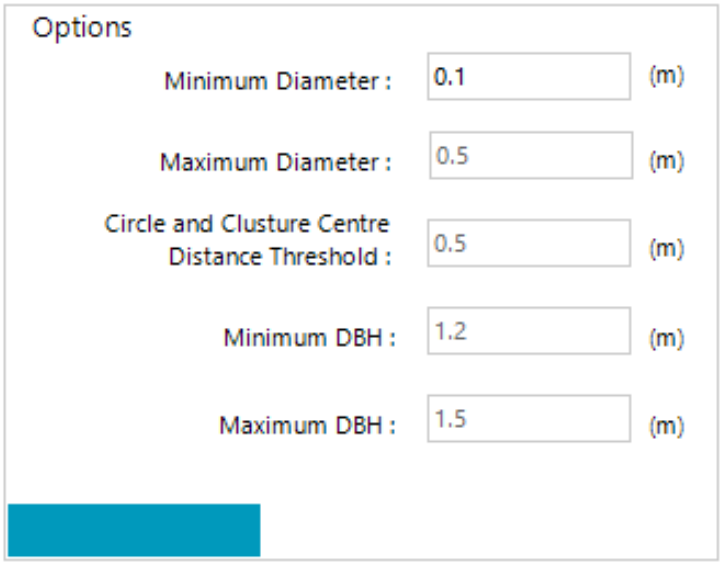


Figure 6.36 Software Parameters to control the detection algorithm by users

The next parameters are a set of "**Options**" shown in Fig 6.36 grouped together to control the detection. In this group, the first two parameters are maximum and minimum diameters. These two parameters allow users to choose the diameter of detected trunks or poles. This parameter is more useful to the tree surveyor as they might want to differentiate between an old tree and a new tree. The default for the minimum diameter is 0.1 metres as it can also detect slim pole-like objects with trunks. The maximum diameter default is set to 0.5 metres which can accommodate the trunks in urban areas.

The next parameter is the distance threshold between the fitted circle and the voxel cluster centre. The voxel cluster centre is calculated by the gravity of all points in the cluster, and on the other hand, the fitted circle on the points can be small or large depending on the points' alignment. Therefore, this parameter helps the user to specify the difference and control the type of trunks and poles they want to be detected. The default is set to 1.2 metres, which is big enough to accommodate all types of trunks and poles.

The next two parameters are maximum and minimum diameter breast height (DBH) values. This parameter will provide the flexibility to users to detect trunks and poles at what height. The British standard in the surveying industry is 1.4 metres; therefore, the default is set between 1.2 – 1.5 metres.

After implementing the proposed algorithm and classification algorithm, the resultant objects are listed in the software's user interface for visualization, as shown in Fig 6.37. The resultant objects are listed in 3 groups:

- Trunk,
- Pole
- Other

When the user clicks on any listed object, the object is highlighted by a fuchsia-coloured cylinder to represent the detection. The other category, further analysed by the proposed algorithm, lists the objects that fit the pole or trunk description. This classification result enables users to override objects' detection and classification according to the point cloud data.

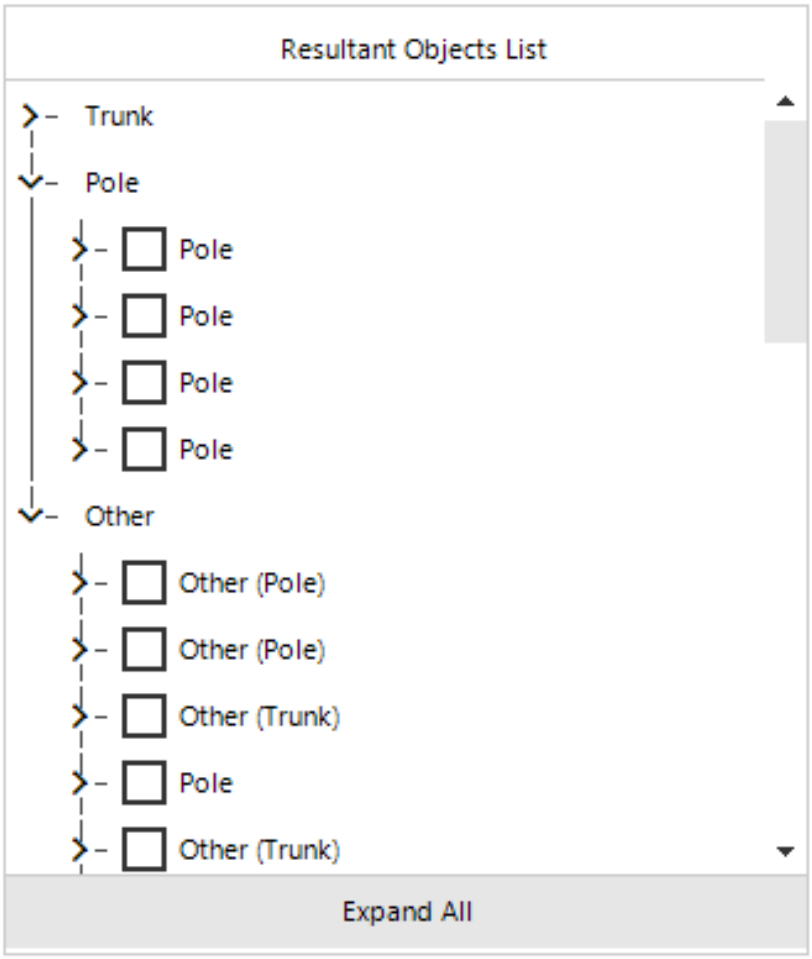


Figure 6.37 List of objects detected by the proposed algorithm and classified into trunk, pole and others

6.5.3 Comparative Analysis: Detection and Classification Results

The algorithm results are tested using standard metrics described by Yan *et al*. (2017), Yang *et al*. (2015), Landa and Ondroušek (2016), and Kang *et al.* (2018). The detected features are categorised as True Positive (TP), False Positive (FP), False Negative (FN) and True Negative (TN). The thesis uses Yan *et al*. (2017) and Wu *et al.* (2017) methods to evaluate the proposed algorithm using precision, recall, quality and $F_1 - measure$ values.

1) TP – The features are correctly detected.
2) FP – The features are falsely detected.
3) FN – The features are undetected.
4) TN – The features correctly detected as others, neither trunk nor pole.

Based on these values, precision, recall, quality and F1 measure (Wu *et al.*, 2017) are calculated

$$Precision = \frac{TP}{(TP + FP)} \% \tag{6.12}$$

$$Recall = \frac{TP}{(TP + FN)} \% \tag{6.13}$$

$$Quality = \frac{TP}{(TP + FP + FN)} \% \tag{6.14}$$

$$F_1 measure = \frac{2 * Recall * Precision}{Recall + Precision} \tag{6.15}$$

The correctly detected trunks and poles are considered true positives, and undetected pole-like objects and trunks are considered false negatives. The objects that are falsely detected as trunk/poles are considered false positives. The number of objects present in each dataset is listed in Table 6.1. The objects are divided into two categories high-density objects (HDO) and low-density objects (LDO). The objects that are fully captured and have a high density of points are HDO, and the objects with very few points and not fully captured are LDO. The objects in the datasets are listed below:

a) **Drone Dataset** – 20.2 million points -7 trees and 10 poles
b) **Chateaudo LiDar Dataset** – 27.1 million points - 49 trees, 13 poles and 5 other objects – persons, buildings etc
c) **Dorchester Dataset** - 161 million points – 23 trees, 18 poles and 15 other objects - buildings and enclosures

Table 6. 1 Point Cloud data sets with urban objects

Point Cloud	Marker poles, Utility Pole, Traffic Signs (LDO)	Marker poles, Utility Pole, Traffic Signs (HDO)	Slim Trees (LDO)	Mature Trees (HDO)	Others (Building, enclosure)	TOTAL
Drone	7	3	1	6	2	17
Chateaudo	13	0	49	0	5	67
Dorchester	6	12	5	18	7	41

The detected objects are listed in Tables 6.2, 6.3 and 6.4 for three datasets. The interested user objects are classified into poles and trunks. The precision, recall, quality and $F_1 - measure$ are calculated using Equations 6.12, 6.13, 6.14 and 6.15.

Table 6. 2 Precision, recall and overall accuracy of Dataset 1 – Drone data set

Pole				**Trunks**			
TP	8	Precision	80.0%	**TP**	7	Precision	100.0%
FP	2	Recall	100%	**FP**	0	Recall	100.0%
FN	0	Quality	80.0%	**FN**	0	Quality	100.0%
TN	0	$F_1 - measure$	88.9%	**TN**	0	$F_1 - measure$	100.0%

Table 6. 3 Precision, recall and overall accuracy of Dataset 2 – Chateaudo set

Pole				Trunks			
TP	9	Precision	81.8%	**TP**	43	Precision	89.6%
FP	2	Recall	81.8%	**FP**	5	Recall	95.6%
FN	2	Quality	69.2%	**FN**	2	Quality	86.0%
TN	0	$F_1 - measure$	81.8%	**TN**	0	$F_1 - measure$	92.5%

Table 6. 4 Precision, recall and overall accuracy of Dataset 3 – Dorchester set

Pole				Trunks			
TP	16	Precision	94.1%	**TP**	21	Precision	95.5%
FP	1	Recall	100.0%	**FP**	1	Recall	91.3%
FN	0	Quality	97.1%	**FN**	2	Quality	87.5%
TN	0	$F_1 - measure$	97.0%	**TN**	0	$F_1 - measure$	93.3%

A comparative analysis of previous data has been performed. The studies focused on detecting multiple poles classification and tree classifications. Even though the data sets used in their studies are different, the comparison is performed with the proposed algorithm.

Cabo *et al*. (2014) achieved an average recall of 92.3% and a precision of 83.8% for detecting pole-like objects on two datasets with 4.6 and 41.5 million points. Li, Li and Li (2016) achieved recall values between 94.6% to 97.7% and precision values between 79% to 100% on three datasets with 12, 7.1 and 8.3 million points. Teo and Chiu (2015) achieved a recall of 95.5% and a precision of 95.6% on two datasets with 5 and 6 million points. The pole detection of the proposed algorithms achieved recall values of 100%, 81.8% and 100.0% and trunk detection recall values of 91.3%, 95.6% and 100.0% in the three datasets mentioned above with 20.2, 21.1 and 161 million points. The precision values achieved pole detection of 80.0%, 81.8% and

94.1% and achieved trunk detection precision values of 89.6%, 95.5% and 100.0% in the three datasets mentioned above. The recall and precision value ranges are slightly higher compared to the other methods, and the datasets are bigger than the existing methods.

The accuracy achieved by Rodríguez-Cuenca *et al.* (2016) is 94.35% and 95.0% in the two datasets. Guan *et al.* (2016) achieved an accuracy of 88.9% for classifying the traffic light poles and light poles. Yan *et al*. (2016) achieved an overall accuracy of 91% in detecting multiple pole-like objects. Yang and Dong (2013) achieved a recall of 84.2% for tree trunks and the recall values for poles is 89.6% and 90.2%. The achieved precision value of 85.4% for trunks and 86.3% and 89.2% for poles. Yang *et al*. (2015) achieved a recall of 91.0% for tree trunks and 94.1% for poles. The achieved precision value for the trunk is 91% and 93.5% for the poles. The proposed algorithm achieved an accuracy of 70.0%, 80.0% and 94.1% in pole detection and an accuracy of 100.0%, 86.0% and 87.5% in trunk detection with large datasets.

The $F_1 - measure$ achieved by the proposed algorithm for pole is 81.8%, 88.9% and 97.0%, and for trunk detection, the values are 91.5%, 93.3% and 100.0%. The proposed method can identify all kinds of poles with low and high-density points with overall high accuracy.

6.5.4 Processing Time

This section records the computation/processing time the proposed algorithm achieves in minutes. The method is implemented in C# on a desktop-based application. The processing times of the proposed algorithm are recorded using the commercial software 3D Vision. Wu *et al.* (2017) subdivided the data set into ten subsets for faster evaluation, whereas the proposed algorithm is implemented on the whole dataset without subsampling. The size of the datasets is not the same in this comparative study. However, times can be compared in terms of the number of points taken to detect and classify objects in the point cloud. Landa and Ondroušek's (2016) method's processing time was 2 hr 35mins, Hackel *et al*. (2016) 90 min for segmenting 30 million points, and Yan *et al*. (2017) presented a computation time of 121.55 minutes for processing 180 million points. On the other hand, the computation time of the proposed algorithm for 161 million points (Dorchester Dataset) is **6.5 minutes**.

6.6 Discussion

The proposed algorithm shows promising results for automatically detecting cylindrical objects in urban point clouds. The advantages of the method are: (1) the use of terrain extraction for fast and quick retrieval of ground points. (2) elimination of lengthy segmentation processes such as euclidian distance segmentation, which does not work in highly populated environments (Landa and Ondroušek, 2016). The proposed algorithm implements voxels (voxel grids), a relatively quick segmentation technique. (3) Voxel-based clustering method works effectively on a current layer and accommodates the voxels in a layer above and below. (4) Effective in real-world scenarios.

During the implementation of the proposed algorithm on the commercial software, real-world scenarios are tested, such as the presence of gaps (no points present), the presence of slope on the ground, the presence of low-density point objects and other objects closer to the objects of interest (trunk and pole-like objects). As a result, the algorithm works efficiently to detect and classify the trunk and polelike objects of point clouds.

Furthermore, the parameters provide the flexibility to detect the features of interest in several kinds of point clouds. The evaluation showed the recall, precision and quality values, indicating that most pole-like objects and trunks are detected. Based on the three datasets, the average recall values of the pole and trunk are 91.6% and 92.8%, the average precision values of the pole and trunk are 91.5%, and 97.2 and the average quality values of the pole and trunk are 84.4% and 90.6%. These are very high numbers, given that the dataset was large.

The limitations of the proposed algorithm are missing ground points and RGB-based classification. The trunks or pole objects that do not have ground points are unable to detect as the terrain extraction algorithm cannot detect any points to implement DBH for the seed layer, as shown in Fig 6.38 (a). Furthermore, the wrong classification was achieved because the user-selected trunk RGB is very close to the building pillar RGB as shown in Fig 6.38 (b).

(a) (b)

Figure 6.38 (a) Trunk with no ground points (b) Building pillar detected as a trunk because of similar RGB

6.7 Chapter Summary

The proposed algorithm detects trunks and pole-like structures by segmenting the ground and non-ground points and removing the ground. First, Voxelization is applied on the non-ground points to fasten the search. Next, the seed layer is identified to start the neighbourhood search to cluster the voxel groups. Next, the potential cluster groups are selected by filtering based on area, compactness and distribution. Finally, the resulting voxel groups are classified as trunk, pole or other in the point cloud. The proposed method works efficiently and quickly as it is not dependent on prior knowledge and is automated by implementing various parameters and segmentation. However, the algorithm does not work when the ground points are absent. However, this can be enhanced by traversing the voxel cluster groups on the same level. The results conclude that the proposed method is robust, fast, efficient and automatically detects and classify trunks and poles. Moreover, the algorithm works well with low-density points and features on gradients.

Further enhancements and research of the proposed algorithm could be extended to offer a more detailed classification of objects in the classes, such as the type of tree, type of pole, vehicles and buildings.

Chapter 7 Software Implementation, Application and Case Study

7.1 Overview

This chapter discusses the implementation and application of this thesis's proposed algorithms and presents a case study on commercial software. The software is called LSS and 3D Vision (*https://www.dtmsoftware.com, Accessed: 18 June 2022*).

Section 7.2 presents the proposed algorithm projects' implementation and models used to design and develop. Section 7.2 presents the software life cycle model implemented, including the requirement, design analysis, development, testing, and release. The presentation of the programming language C# and software development environment visual studio was chosen as it supports good 3D graphics libraries. The section also discusses the execution process involved with each software project life cycle, including project management, quality assurance, program UI intuitiveness, and reusability of programming. The next Section, 7.3, introduces the commercial software 'LSS', its brief history, and the journey to point clouds product '3D Vision' by the company McCarthy Taylor Systems Ltd (MTSL). Section 7.4 presents the market research in the surveying and civil engineering industry, the workflow used, and the software provided by different companies and users in the UK. Section 7.5 presents the case study on the 3D Vision software product and its implementation of point cloud processing using UI parameters. Finally, Section 7.6 provides the chapter summary.

7.2 Software Implementation

The field of computer science software engineering is well-developed. Software engineering can be defined as follows by Tucker (2021).

"*Software engineering is the* discipline *concerned with the application of theory, knowledge, and practice to building reliable software systems that satisfy the computing requirements of customers and users.*"

Software engineering is used for software development with proper tools, techniques and methods. However, development is a large task requiring accurate engineering and management techniques to make it cost-efficient and reliable.

Large-scale software projects are usually divided into phases for the effortless delivery of the product. Since this thesis is also part of a commercial product, the proposed algorithms were separate projects delivered using software models presented in Section 7.2.1, followed by programming languages and SDE decisions (Section 7.2.2), and the execution process involved within projects (Section 7.2.3). This section explains and summarises the software development and design of the proposed algorithms.

7.2.1 Software Development Models

According to Ruparelia, the models are linear, iterative, and combination. Linear models are sequential models, meaning when one stage finishes, the other stage starts, whereas iterative models enable revisiting all the stages in the future (Ruparelia, 2010). Software development models are used to design, develop and release this thesis's proposed methods/algorithms. Examples of software development models are waterfall, unified, rapid, incrementing, spiral, v and w models.

- Waterfall Model – is a linear sequential where each phase is specialised to a task and is dependent on the result of the previous one. Waterfall has six stages: analysis, requirements, design specifications, development, testing and integration, and deployment shown in Fig 7.1

- B-Model – is an extension of the waterfall model to ensure the constant improvement of the software in the development stage (Ruparelia, 2010)

- Incremental Model – is an iterative model and can be perceived as a 3D waterfall model where the z-axis represents the number of iterations to improve the functionality (Ruparelia, 2010)

- V-Model – is folded in half. The left half represents the evolution of user requirements, and the right half represents the integration and verification of the system (Ruparelia, 2010)

- Spiral Model – is used for risk management that combines an iterative development process and the waterfall model (Boehm, 1988). It consists of four quadrants, as shown in Fig 7.2:
    - identifying and understanding the requirements,
    - performing the risk analysis
    - building the prototype
    - evaluation of software performance

- Wheel and Spoke Model – is a bottom-up approach that establishes the system's requirements and initial design. Then, it creates a prototype for implementation and verifies against the requirements, then feedback is constant during the development cycle and the stages after use to create a more refined prototype (Ruparelia, 2010)

- Rapid Application Development Model – is used for prototyping and iterative with no specific planning. The model emphasises coming up with the prototype rather than planning tasks (*What Is Rapid Application Development?, (Accessed: 19 June 2022)*)

- Unified Process Model – is a user-driven and iterative model. It specifically addresses problems related to object-oriented software consisting of four stages: inception, elaboration, construction and transition.

- Agile – is the practice of managing a project by breaking it into several phases. The development happens in small projects and is released with small changes. Agile method types are extreme programming (XP), joint application development (JAD),

lean development (LD) and scrum. Scrum is short sprints, and progress is monitored daily (Ruparelia, 2010)

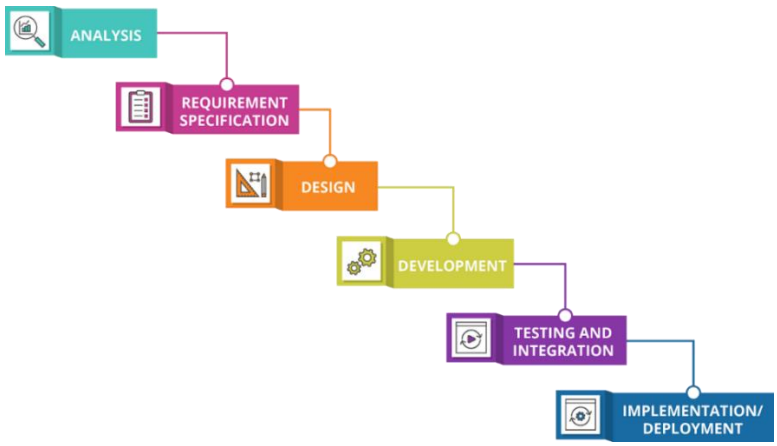


Figure 7.1 Waterfall model (Jones Justin & Waddel Scott, 2019)

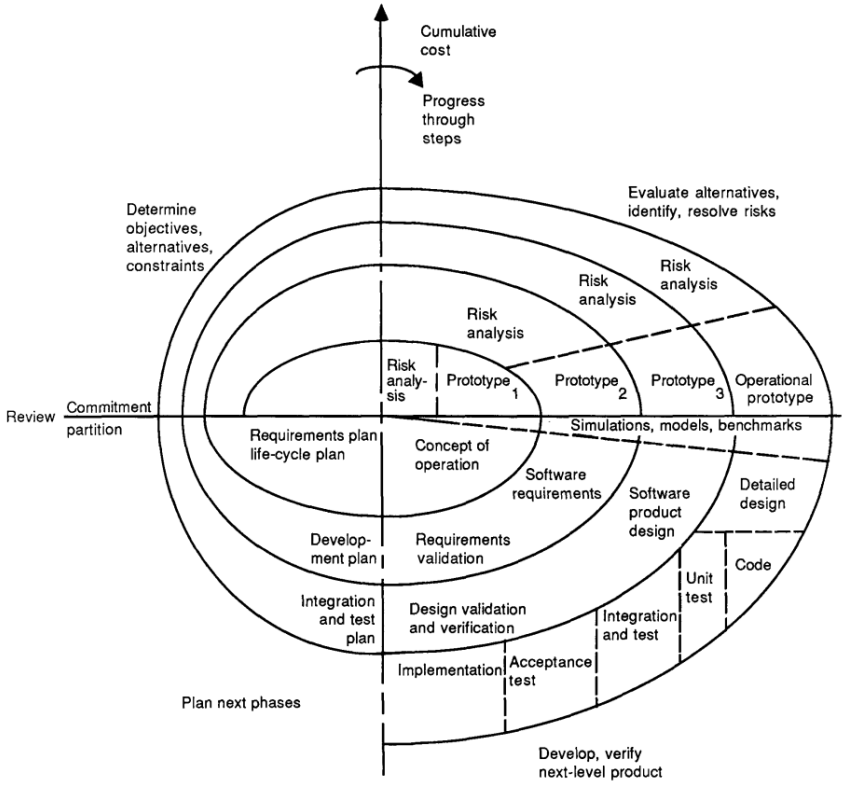


Figure 7.2 Spiral Model (Boehm, 1988)

A brief introduction to the software development models mentioned above are commonly used to develop the software life cycle. After careful analysis and consideration for this thesis, both the spiral model and agile scrum are used to develop the proposed algorithms. The advantage of the spiral model includes that the software prototype is produced at an early stage of development, risk handling, flexibility, good for complex projects, strong approval and documentation and customer satisfaction (Upadhyay Raj Kumar, 2020).

On the other hand, the advantage of Agile Scrum is that the development cycle is iterative, provides a learning experience, can be revisited, projects are delivered quickly and tested, and is customer feedback oriented (Mixing Agile and Waterfall, 2021). The models constitute a system which defines the software design phase. The software design phase is responsible for the overall software architecture and execution of software functions.

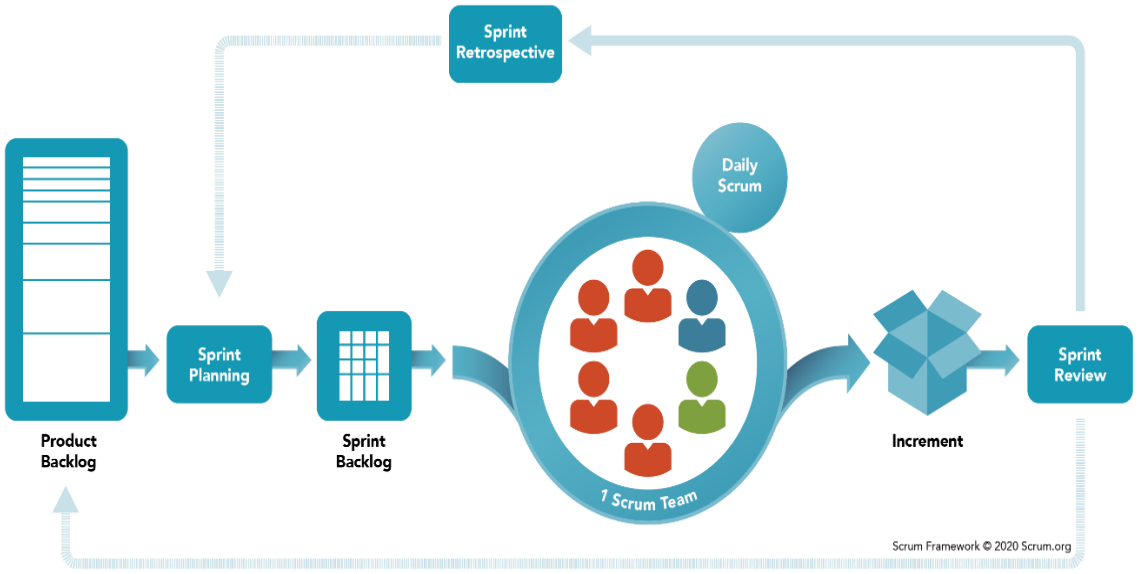


Figure 7.3 Agile Scrum in a nutshell (*What Is Scrum?, Accessed: 19 June 2022*)

7.2.2 Design Decision

7.2.2.1 Programming Language

A programming language is a computer or constructed language designed to communicate instructions to computers (Ogala, Ogala and Onyarin, 2020). Syntax is the set of rules that defines the instructions, a combination of symbols and words in a structure's statements or expressions (Woz U, 2020). For this thesis, C# is used as a programming language for coding the proposed algorithm. C# is read as C sharp. It is an object-oriented programming language. The primary architects of C# were Anders Hejlsberg (lead architect at Microsoft), Peter Golde, Eric Gunnerson, Peter Sollichy and Scott Wiltamuth. It was first introduced in 2000 at the Professional Developer Conference (PDC) (Ogala, Ogala and Onyarin, 2020). Microsoft also introduced C# with the .NET framework and Visual Studio. In 1999, Anders Hejlsberg formed a team to build a new language called COOL (C-like Object Oriented Language) (Hamilton, 2008). Later, before publicly announcing, Microsoft renamed the language as C#, inspired by a musical notation of a sharp symbol that indicates the note should be a semitone higher in pitch (Kovacs, 2007).

The reason for choosing C# is to be consistent with the commercial software in MTSL (3D Vision) that already uses C# in desktop applications with the abovementioned benefits. Furthermore, C# also provides metaprogramming, classes with properties, methods, functions, namespace, memory access, exception, polymorphism, functional; programming, inheritance, and supports different libraries.

Benefits of using C# (Hejlsberg *et al.*, 2011) are:

1. Component-oriented programming,
2. Garbage collection – unused objects in the memory are automatically deleted,
3. Exception handling – structural approach for error detection,
4. Unified type-safe system – all primitive types inherit from the root *Object* type,
5. Supports both user-defined reference types and value types, permitting dynamic allocation,
6. Supports versioning,
7. Portability, as it supports Common Language Infrastructure (CLI).

Another reason for choosing C# is its cross-platform, simple, modern OOPs language which is very popular in the game development industry. Furthermore, it supports 3D graphics functionality, providing a full-featured gaming development platform. As point clouds are 3D data, the development platform must provide options for navigation, viewports and projections. In addition, C# works well with OpenGL. The 3D aspect is handled by OpenGL, which is cross-language and cross-platform programming for rendering 2D and 3D graphics. This thesis's algorithms (projects) are handled using the OpenGL library, which provides functionality like rendering and drawing shapes and setting camera positions and projections in 3D.

C# applications can be desktop-based. The high-end programs in the 3D better run on desktop vs on mobile or tablet. The surveying and civil industry users are using more windows OS applications; therefore, C# was the right fit. Furthermore, the language lessons are easily accessible to anyone on the internet.

C# Syntax

The basic syntax for introducing the C# language is shown in Fig 7.4. In OOPs language, a program consists of various objects interacting with each other using actions. These actions are methods (*C# - Basic Syntax, Accessed: 14 July 2022*). In addition, C# has reserved predefined words called keywords. The program always starts by including the "using" keyword. Using is used to call system libraries or any third-party library that supports C#. These libraries give functions and classes for various actions. Next, the namespace is used to arrange the classes, structures, interfaces, enum and delegates (V.S. Rajesh, 2005). Finally, the class is declared using the keyword "Class" and a unique name within the namespace. Inside the class, the methods, functions and variables are declared. The output of the example program in Fig 7.4 is "Thesis on Point Clouds", as it uses the system library to call the Console.Writeline, which prints any text in double quotes.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Sample_Class
{
    public class Sample
    {
        public static void Program(string[] args)
        {
            System.Console.WriteLine(" Thesis on Point Clouds");
        }
    }
}
```

Figure 7.4 C# syntax example

C# Example

The source code file can be saved on a computer with the '.cs' file extension. The file extension helps open the file from anywhere on the computer. The example in Fig 7.5 is saved as 'Testing.cs'. The example presents a class called CompareNumbers that is used to compare two numbers. The 'result' method passes any two numbers, and the class compares them. The output in this example is 'Number2 which is higher.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Sample_Class
{
    0 references
    public class Sample
    {
        1 reference
        public void CompareNumbers(int number1, int number2)
        {
            if(number1 > number2)
            {
                Console.WriteLine(" Number1 is higher");
            }
            else
                Console.WriteLine(" Number2 is higher");

        }

        0 references
        public void result()
        {
            CompareNumbers(5, 12);
        }
    }
}
```

Figure 7.5 Pseudo Code example for comparing two numbers

7.2.2.2 Programming Environment

The programming environment allows the combination of hardware and codes to be built into applications. The developer typically uses an Integrated Development Environment (IDE). For this thesis, the IDE used is Visual Studio 2019 and 2022. Visual Studio was announced at the same time as C# and .Net in 2000 at the Professional Developer Conference (PDC) (Ogala, Ogala and Onyarin, 2020). The Visual Studio environment with C# coding is shown in Fig 7.5.

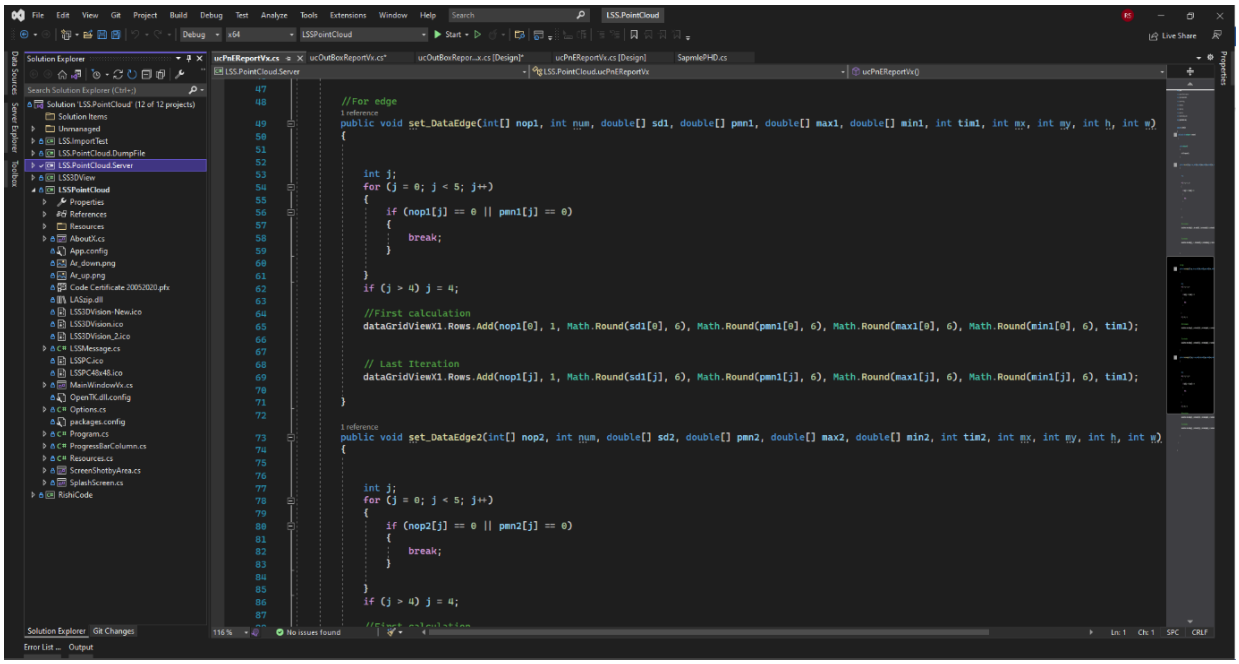


Figure 7.6 Visual Studio and C# coding

7.2.3 Execution/Process

The software development method generally consists of a ‘life cycle’ for the final product. Hence, they are called software development life cycle (SDLC). The SDLC has several phases: requirements gathering, analysing, specification, design, outputs, developing, validation, deployment, testing and maintenance. Several SDLC models are used in software development. They describe the steps involved in the cycle. In this thesis, the execution of software projects is divided into phases or stages to manage the elements of the projects more efficiently (Jevtic, 2019). As shown in Fig 7.7 for this thesis, the projects are divided into nine stages as follows:

1) Planning,
2) Requirement gathering,
3) Design,
4) Develop,
5) Build,
6) Test,
7) Document,
8) Release
9) Maintain

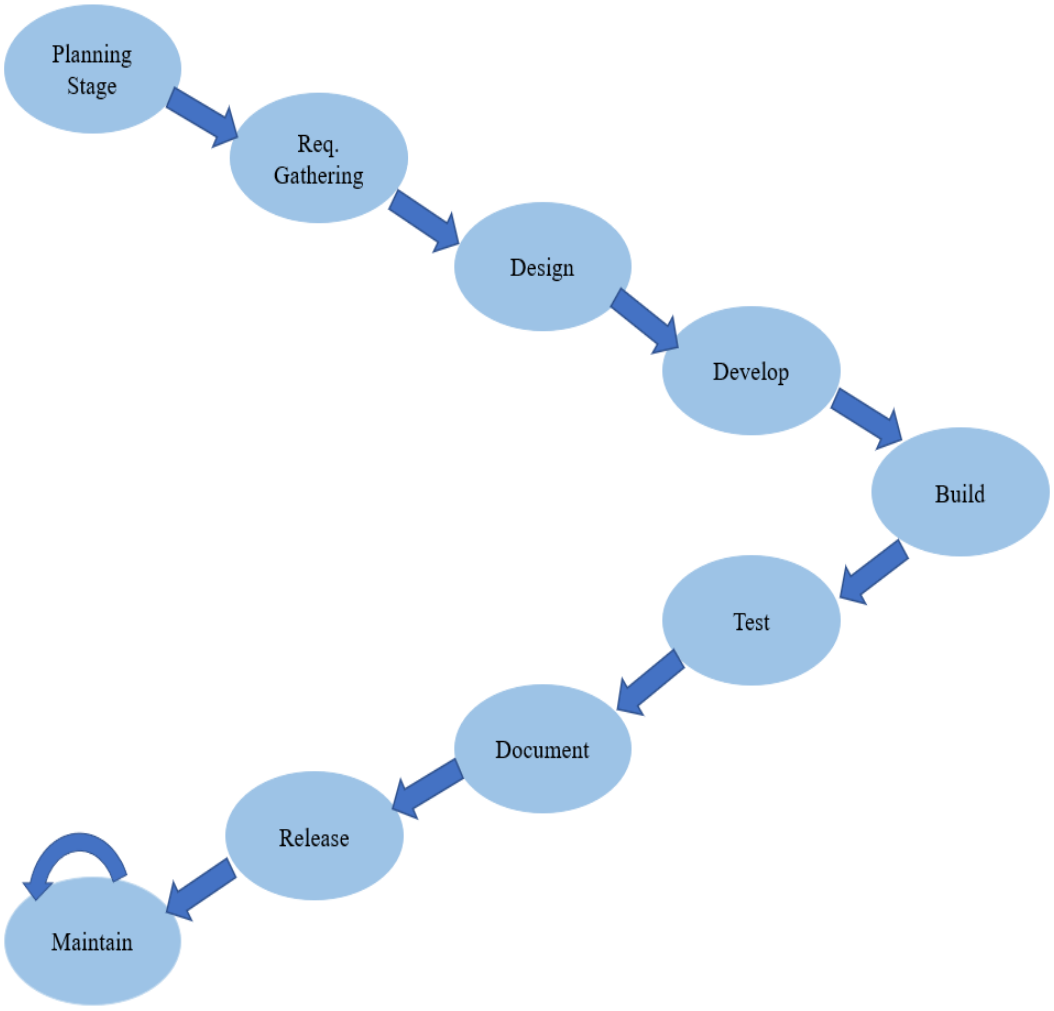


Figure 7.7 Software Development Cycle

The development phase involves the users for constant feedback. Then, the methods and algorithms are refined and fine-tuned until a satisfactory system is developed. Therefore, the software development is gradual and goes through these stages shown in Fig 7.7.

For this thesis, an outline of the development phases is explained, from planning to maintenance of the software. The process is iterative and was applied to each project while working on the proposed algorithms.

The first stage is the planning stage, where general market research is accomplished. It is very important as the point cloud is an emerging technology with a competitive market. The next stage is requirement gathering, accomplished by speaking to the users, reviewing the wishlist items and involving the stakeholders. The overall project design is dependent on the requirements. Once the requirements are listed, a prototype design is initially fabricated. The design involves the UI options, messaging between applications and intuitiveness of the software. The next stage is development, where the software design is converted into programs. Before the next stage of software build, the written code in the development stage is tested by the developer, called unit testing. Once the program passes the unit testing, the various units are integrated to form a build. The build has a unique version number supplied to other testers, and the stakeholder and each unit are tested individually. The initial build helps with the evaluation at an early stage, including any design (UI) changes.

Next, the project's documentation is processed with explanations and flow charts for other developers to understand. Finally, after all the stages are processed without impediments, the developed project is released as a final version to every user with a version number. The last stage is maintaining the software's released version, including support and bug reports (if any) found.

Apart from the development cycle, some important aspects of software development are project management, design quality, program reliability and efficient coding.

7.2.3.1 Project Management

Project management is important and plays a crucial role in software development and delivery. With respect to the business, the goal is to complete the project on schedule and within budget.

The commercial environment mainly drove this thesis project which enforces prompt and reliable software delivery. The project had intangible deadlines because of the pressure from customers and their expectations. The thesis project timeline was difficult to estimate, particularly when using new technologies. The user feedback and input were also very important, which took a lot of time. Thus, designing, developing and estimating were iterative to achieve the project milestones. The development progress of the projects was monitored by using Agile scrum.

7.2.3.2 Quality

Two of the important factors in this project are quality and accuracy. Land surveyors and civil engineers rely on the software to provide them with accurate results. The proposed algorithms are modified and tested for accuracy. The quality check reports are presented for the users to check and verify.

The quality also depends on understandability and adaptability: 1) Understandability means understanding the design, documentation, and complexity, and 2) Adaptability means easy changes. This project is both understandable and adaptable. The variable name in the code aims to be clear and presented by comments for in-depth understanding. In order that a third party (or any other developers) can understand the code, comments and documentation are included.

7.2.3.3 Program Reliability or Intuitiveness

Software intuitiveness and reliability are very important aspects. The software's prominence is at stake if it fails to impress the users. In general, a program can be reliable by avoiding crashes and bugs. Having said that, software completely free from bugs is an exception; therefore, the solution has to be interactive, informing the users about the bugs and adding facilities to handle them in the system. Certain programming techniques, such as defence programming, combine checks for faults and fault recovery in the program (Sommerville, 1996). This thesis uses a defensive programming approach to aim and anticipate errors and potential bugs before and accommodate those in the code. For example, all the inputs would have a default value if the

users did not set it. C# is also very good at handling exceptions and type-safe (for casting objects).

Software intuitiveness helps users with little or no experience understand the logistics and access the software commands. All the commands and operations are accessible either by mouse clicks or user input. This allows the user to be in control.

For this thesis, the proposed algorithms are provided with minimum user clicks and user input values for edge detection and trunks and poles detection, as point clouds can be unpredictable given the variety of data. Therefore, the users are in control of the data. Also, all parts of the program must work to maintain a high level of reliability and user perception. For example, if an error occurs continuously and the program fails to work, this may hurt user confidence and willingness to use the software.

7.2.3.4 Reusable Code

The algorithms and methods for this thesis aim to provide code that can be used again. In order to achieve this, individual reusable components are produced that are accessible on a global level. It is called block coding, where important calculations are coded in functions and methods separated (in blocks) that are accessible globally to other developers. For example, the code is made into functions so that other developers can use it as needed, such as mathematical calculation functions.

7.2.3.5 Testing

All algorithms and methods presented in this thesis are thoroughly tested. The testing is performed on three levels 1) Unit Testing, 2) Acceptance Testing and 3) User-based testing.

The unit testing is at the developer level, where a unit, i.e., a function, is tested. The functions are the basic building block of the software, and many functions are written together to perform a functionality (Singh, 2020).

Next is acceptance testing, which can be performed by any stakeholder involved in the development process. For this thesis project, colleagues in McCarthy Taylor Systems Ltd performed acceptance testing. Acceptance testing is carried out by a series of commands to achieve the desired results and satisfy the business requirements. The tester performs this and checks the working and performance of the software function.

Last is user-based testing. User-based testing involves real users after the first deployment of the software functions. These users are selected to perform testing on a beta version of the software that includes new functionalities before the release. This is very helpful as it provides insight into how the user interacts with the product, which helps design better UI elements, fine-tune functionality and identify bugs. The feedback from users is used to improve the functionalities following which the alpha release of software happens.

This thesis used all three levels of testing before releasing the methods and algorithms to the wider user base. The user-based testing is also part of the spiral model mentioned in Section 7.2.1, which ensures that the software is maintained, improvised and amended based on the requirement changes from the users.

7.3 Application in MTSL Software

McCarthy Taylor Systems Ltd (MTSL) is an independent software company developing software for a wide range of industries such as land surveying, mineral extraction, hydrographic design, surveying, civil engineering, construction, landscape architecture, consultancy, air and collision investigation, geotechnical engineering, waste management and archaeology (*https://www.dtmsoftware.com, Accessed: 18 June 2022*). LSS is the DTM software, and 3D Vision is the point cloud product.

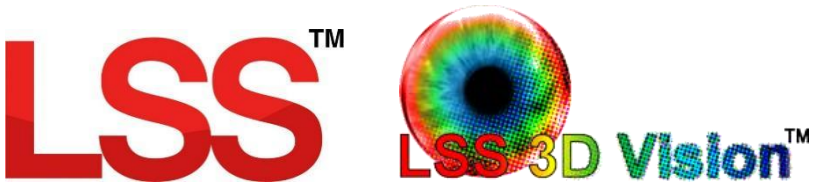


Figure 7.8 LSS and 3D Vision Logo

7.3.1 Brief History of MTSL

MTSL was started in 1985 with the “LSS” software, which stands for Land Surveying Software. LSS is a cost-efficient, straightforward software package for surveyors, designers and engineers (Civil and mineral). In 1995, MTSL evolved and became popular in several industries. The company is headquartered in Birdlip, Gloucestershire.

LSS is a powerful windows-based software to produce a digital terrain model supporting Electronic Distance Measurement (EDM) or Global Positioning System (GPS) instruments and Computer-Aided Design (CAD) systems imports. These models could contain contours, break lines, elevation data, vegetation and building/building footprints. LSS offered an efficient way to convert all that data into a 3D terrain model. LSS pioneered the concept of real-time 3D terrain modelling, which was only possible through its high-speed triangulation algorithms.

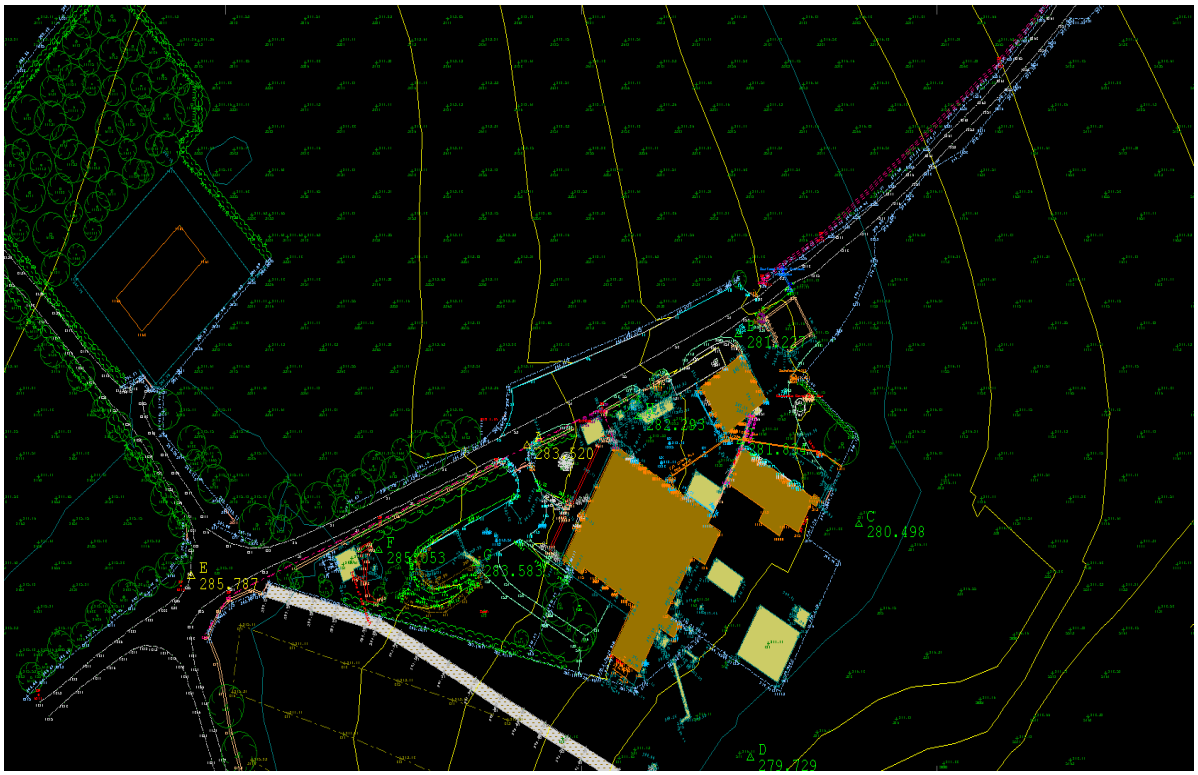


Figure 7.9 A Digital terrain model in LSS

In the early 2000s, a technology arrived which would transform the survey market, “laser scanners”. Laser scanners were capable of collecting a cloud of 3D points in a fraction of the time it took to survey previously. The latest scanners can collect more than one million points a second. Although the challenge now was to process these huge amounts of data to generate a 3D CAD drawing (the ultimate deliverable).

At this point, the laser scanner manufacturers had a stranglehold on the software market because they were the only ones who could process their own scanner’s data. As a result, they charged many £’000 for each copy, in addition to the £50,000 - £100,000 cost of the scanner.

By 2012, LSS started to work with these datasets, and the company embarked on an ambitious product development project. In 2015, a Knowledge Transfer Partnership (KTP), funded by the UK government (Innovate UK), began to start the point cloud’s feature detection project. I joined MTSL as part of Innovate UK and began my journey with point clouds and in the company. Point Cloud solution called “3D Vision” was released in April 2017 to widespread acclaim and is available with LSS. The largest point cloud so far processed by LSS contains 100 billion points.

Drones started to capture point clouds, which photogrammetry software then processes. LSS can read these files to generate a 3D terrain model and allow the surveyor to draw lines in 3D directly from the point cloud. Building Information Modelling (BIM) systems for all construction, maintenance and refurbishment contracts is a high priority for the UK Government, but there is a great deal of confusion over its practical implementation. LSS bridges the gap between engineer and client by providing leading-edge data exchange solutions.

As a result, LSS has become an essential business tool for hundreds of organisations. LSS products are LSS Solo (basic level), LSS Vista (mid-level), LSS Elite (advanced level), 3D Vision (point clouds), 3D View, LSS Unity, LSS Education, LSS Police (used for collision and police incident investigation) and LSS Toolkit.

Figure 7.10 World heritage site “Gorham’s Cave” (Copyright @DroneSurv) visualised and preserved with the help of LSS 3D Vision

7.3.2 3D Vision: LSS point Cloud Software

The 3D Vision supports various import formats. For example, users can create Point Clouds from E57, LAS, LAZ, FARO FLS, Leica Geosystems PTS, XYZ and ASCII. The data can be with or without Intensity (a single numeric value which can be used to colourise the data) and with or without RGB colouration. Point Cloud provides the real-world context for recreating and extracting valid information about objects. For example, in Geoinformation systems for landscape design, planning, and urban scenes, point cloud data is usually collected to capture the scenes. Then, the scenes are processed to extract important and useful information. Some commands in 3DV (*https://www.dtmsoftware.com, Accessed: 18 June 2022*) are:

1) Eraser tool to remove unwanted points, 2) Extract a terrain from the Point cloud to create an LSS DTM survey 3) Use the Point Cloud application as a 3D digitiser to extract 3D lines and points 4)Create elevations and topographical surveys 5) “LSS 3D Vision^{TM}” application is a free Point Cloud viewer 6) Generate Orthophotos from any Point Cloud 7) World’s first “Searchphere^{TM}” technology to find points 8) Display an LSS survey in 3D 9) Query coordinates and distances 10) View vertical and horizontal slices 11) Vertical and horizontal

heatmaps that show deviation from vertical or horizontal planes and 12) The over-display helps users see the extracted information in 3D

LSS is widely used for landscape design. The point clouds extract information, such as trees, hedges, buildings, kerbs, crest and toe of banks, walls, roofs, marker poles and lamp posts, vehicles, break lines, etc. LSS will represent these features on-screen and on the final plotted output as user-defined symbols and line styles to create elevation and topographical surveys.

Fig 7.11 shows an example of LSS on the left-hand side with extracted DTM model, and on the right-hand side is 3D Vision with a point cloud open. LSS and 3D Vision are two separate applications. Both are connected via a pipeline that sends and receives messages to sync the location and various functions. The features are extracted from the point cloud into LSS in the form of points and links. For example, a survey or DTM in LSS represents contours, levels, buildings, trees and roads.

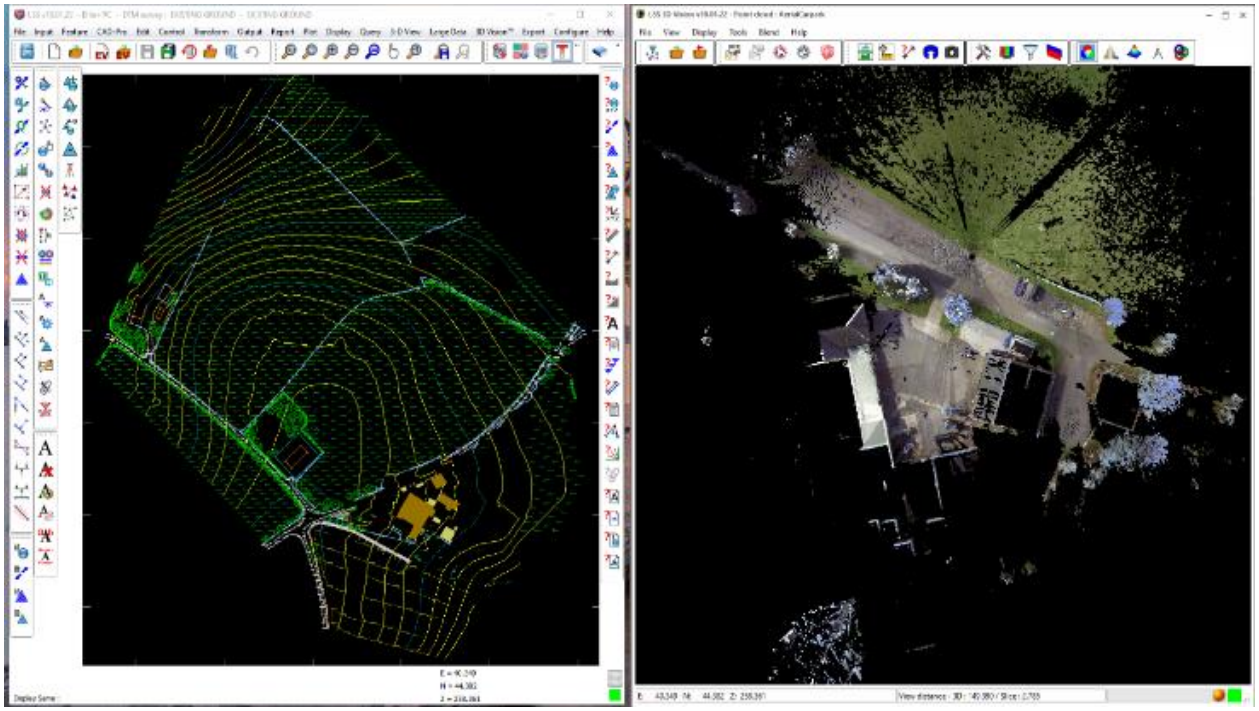


Figure 7.11 Left hand is LSS, and the right hand is 3D Vision (point clouds)

7.4 Market Analysis

The global 3D scanning market is projected to grow by 16.3% at a CAGR, valued at $3.72 billion in 2020, to $16.66 billion by 2030 (Wood, 2022b). Europe's 3D scanner market was valued at $316 million in 2020 and is expected to grow to $669 million by 2027, translating into a CAGR of 11% (Wood, 2022a). A major part of the market would be in North America (29%), followed by Europe (28%) and Asia (28%).

Growth in the scanning market is predominantly due to the surging need for highly accurate 3D data and the increasing need to capture a large volume of 3D data for analysis and modelling (Wood, 2022b). Growth can also be attributed to the wide application of point cloud in industries from engineering and manufacturing to healthcare, high investment in research and development, and advancement in 3D modelling and mapping technology. The 3D scanners market size, however, decreased during the global pandemic but is now back on track and expected to grow from 2021 onwards (Wood, 2022a). Given the high demand for 3D scanners and advancements in modelling technology, the 3D point cloud processing software market is bound to grow at a similar rate or higher as software demand is highly correlated to 3D scanner use. Furthermore, point clouds are becoming increasingly popular, and there is a "point cloud boom" at the moment by the recent developments in point cloud software (Cropp, 2021).

Each software is different as the new capabilities, and innovative thinking spurred. The market is very competitive. The new scanning technologies are becoming integral to many diverse projects (Cropp, 2021). As for the software, innovation is vital to accommodate the new scanning technologies. With a wide range of software with various options to process point clouds, it is essential to understand the industry's workflow.

7.4.1 Workflow

The workflow of a typical user in the software industry (surveying, civil engineering) to process the data from a scanner to a model has to pass the data through various software and solutions, as shown in Fig 7.12. First, the data is collected using a scanner or drone. Then the user inputs these raw files into the scanner or drone company's software provided by hardware companies.

These are used to register different raw files together, called registration. After registration, they can view the file and manipulate the data using cropping and deleting points. When the user is happy with the file, the data is exported into desired file formats. Second, the data is imported into point cloud processing software, which is used to extract meaningful information of the features or terrain to create models such as DTM or DSM (Digital Surface Model). Third, the data is imported into CAD from this processing software. CAD is popular because it allows the user to accurately visualise and present information on the particular area they are surveying. AutoCAD (Autodesk Corporation) is a common software used within the industry; therefore, it is easier to pass the model to clients, other surveyors, and engineers using its formats. Finally, users end up with a CAD model.

Figure 7.12 Workflow from scan to model for processing point clouds

This thesis concentrates on point cloud processing, which converts point clouds into models.

7.4.2 Software in the Market

Point cloud processing software tools are globally available to analyse point clouds. However, for this thesis, the market analysis and software market focus solely on UK-based companies. The point cloud processing market is generally divided into two parts 1) Hardware company products and 2) Software company products. As LSS is commercially used to implement this thesis's algorithms, the focus is on software companies.

Following is an overview of the software provided by hardware and software companies to handle point clouds from raw files to models.

7.4.2.1 Hardware Companies

The hardware or scanner companies also provide solutions to import raw files after scanning, register multiple clouds, clean the data, and export the files in their supported formats. Examples of such software are:

- Autodesk – Recap,
- Revit and AutoCAD,
- Bentley – Pointools,
- Leica – Cyclone,
- Faro – Scene,
- Riegl – RiSCAN PRO,
- Trimble – RealWorks,
- Trimble Business Center (TBC),
- GeoSLAM,
- NavVis and
- Z+F.

Hardware companies are producing software so that the user can handle their raw point cloud files. However, to maintain customer stickiness, they provide basic functionalities such as point cloud registration, removing outliers, chopping the data, viewing the data in 3D, etc. Point clouds are captured from multiple scanner station sites.

In order to produce a single point cloud, registration is performed to merge multiple station data into one. The problem with software is that they are mainly focused on viewing and managing the data; they are not capable of performing the complex options for feature extraction and other point cloud processing process. Hence, software products are required for point cloud processing.

7.4.2.2 Software Companies

Software companies provide solutions to process and extract information from the exported files from hardware solutions, as data will be nothing without appropriate software to process it. The software solutions support the file formats from the hardware solutions to import the files, and then the commands and options allow the user to extract the information, save and export it into CAD to model or deliver as it is. Some software in the UK market are as follows:

- Vercator,
- TopoDoT,
- 3DReshaper,
- EdgeWise,
- PointCAB,
- PointFuse,
- VEESUS,
- Terra3D and
- LSS – 3D Vision.

Software companies provide complex functionality, unlike hardware companies' software. However, the problem with the software company products is that they are more focused on particular features, i.e., not all the software can perform all tasks a user might need. For example,

1) EdgeWise is an exclusive solution that provides automatic building floor extraction,
2) PointCAB is a tool that is focused on the extraction of sketches such as elevation profile, panorama view, measure distance etc.,
3) PointFuse is software focused on extracting pipes from the building and industrial sites,
4) Vercator is focused on the segmentation of points,
5) 3DReshaper focused on segmentation as well,
6) TopoDOT specialises in break line extraction and rail,
7) VEESUS is a software focused on visualisation than the extraction of features and
8) Terra3D focus only on railway asset extraction and management.

On the other hand, LSS 3D Vision implements the proposed algorithms and methods that will allow the user to filter out an outlier, noise removal, edge detection, trunk and pole-like object detection, segmentation, sampling, and modelling.

7.4.2.3 Free

There are free software other than hardware and software companies that are not commercial and are available for free. Open source or free point cloud processing software are standalone packages.

- Software such as
    - CloudCompare – open-source software for viewing, editing and processing point clouds.
    - MeshLab – open-source tool for creating 3D meshes and triangulation of point clouds.
    - Euclidean – 3D graphics engine that renders point clouds to images.

- Libraries such as
    - PCL – Point cloud Library is a standalone large-scale open project for 2D/3D image and point cloud processing. It is written in C++.

CloudCompare and MeshLab are open-source software which is contributed by various developers around the world. However, these free software and libraries are effective but not as compared to paid software as they lack speed and efficiency, and there is no proper documentation to use these.

7.5 Case study

The case study is conducted to demonstrate the practical application of the three proposed algorithms and methods in this thesis using the 3D vision software. This case study focuses on key problems in a typical point cloud processing and answers the research questions presented in Chapter 1.

A point cloud can be very large, and the processing of the point cloud will depend on the size and complexity of the data (*What is a Point Cloud Survey?, 2021*). A point cloud can contain many millions or billions of points; therefore, processing can take many hours or days. To address these problems, removing noise and outliers is essential to speed up the process, as the number of points can be reduced significantly by focusing only on the good points. After cleaning up, the data is ready to extract information. The information is the features such as kerb, road, building floorplan, trees, etc. The solutions in the software (3D Vision) are user oriented. The main requirement throughout is to maintain a high degree of accuracy, but it must also be **fast**, **robust** and **easy to use**. The case study is performed on a dataset to demonstrate 3D Vision for *Point Cloud Processing*. Firstly, by removing noise and filtering outliers, data is cleared of bad points or points that are not part of relevant features. Secondly, extracting edge sects and edge streams on the data and finally using segmentation to identify the tree trunks and pole objects.

7.5.1 Dataset

The point cloud data “Dorchester” used in this study was captured with the Leica RTC360 3D laser scanner. The scanner was released in 2018, and the data was captured in 2019. The scanner takes less than two minutes to complete a full dome scan at 6mm point spacing at 10 metres. The field of view is 360 degrees with a range of 0.5 - 130m. It collects 2 million points per sec. The accuracy is 1.9mm at 10m, 2.9 mm at 20m and 5.3mm at 40m. The data are generated in E57 file format, with a medium density of 6mm @10m. Other options are low 12mm @10m and high density 3mm @10m. The data is automatically registered using a VIS app that tracks multiple scans in the correct position, and by selecting the scans, they can be

linked together. The VIS app also puts the scans together in the correct orientation and alignment.

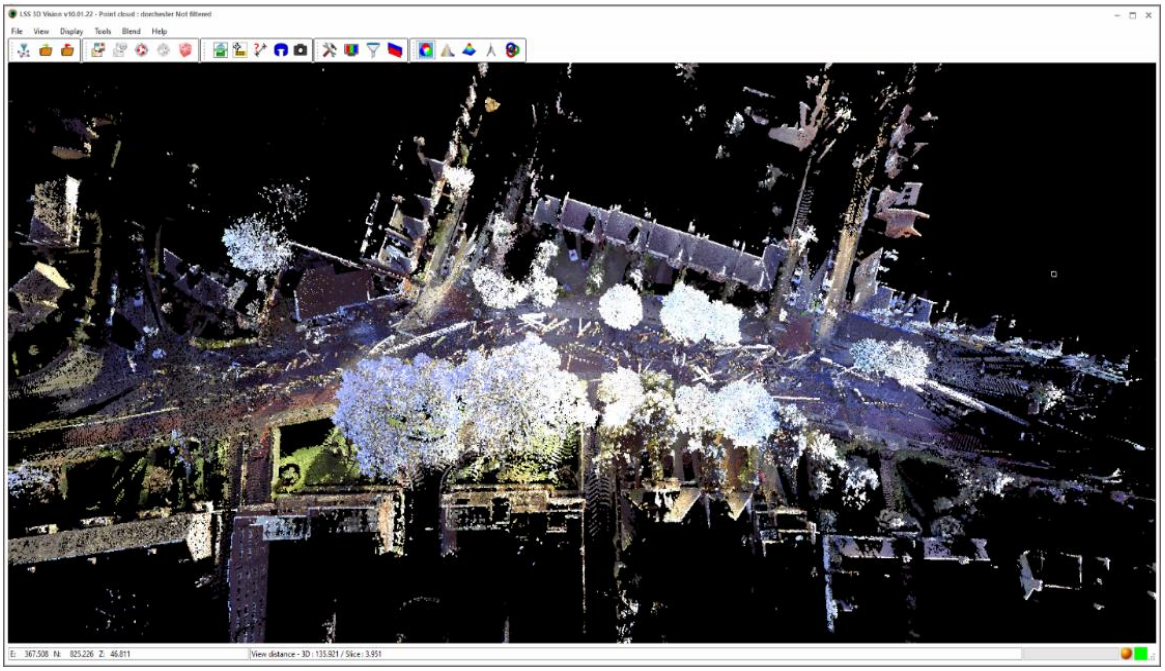


Figure 7.13 Point cloud “Dorchester” in 3Dvision

7.5.2 Point Cloud Processing in 3D Vision

For this section, urban point cloud “Dorchester” is used to demonstrate the implementations of the proposed point cloud processing methods on the commercial software 3D Vision. Dorchester is a town in Dorset, England. The point cloud is captured on a busy road in Dorchester. It is an urban scene with features such as roads, buildings, street and street furniture, bins, lamps, trees, bus stops, etc. The raw point cloud was captured in several different areas on the site. Lastly, they were linked together to form this dataset. The Dorchester data set has 161 million points.

As discussed in Chapter 3, this thesis presents point cloud processing methods/algorithms for filtration, edge detection, feature extraction and modelling. The Dorchester data set is used to show each of these processes in 3D Vision.

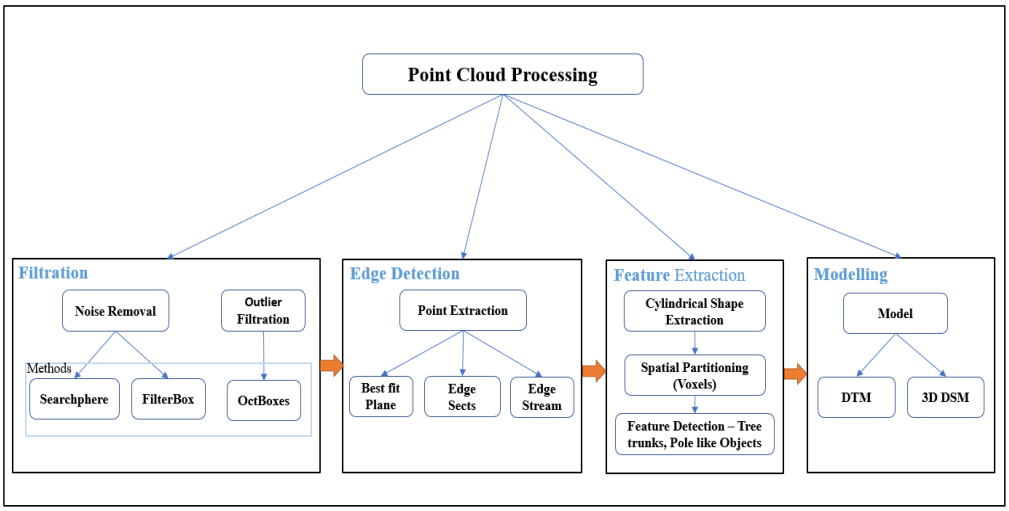


Figure 7.14 Point cloud processing in this thesis

The outline of the point cloud processing is as follows:

Filtration

- Identify the outlier and noise as mentioned in Chapter 4 according to their distribution, proximity and position.
- The noise removal and outlier filtering are performed using NR-S, NR-F and OF-OB.

Edge Detection

- Once the data is clean, it is ready for feature extraction.
- Select points using the search sphere to identify the edges.
- Set the parameters for the edge stream.
- Apply edge stream to the features where long line tracing is required.

Feature Extraction

- Set the parameters for the trunk and poles.
- Segmentation using voxels.
- Apply the algorithm to extract the tree trunk and pole-like objects.

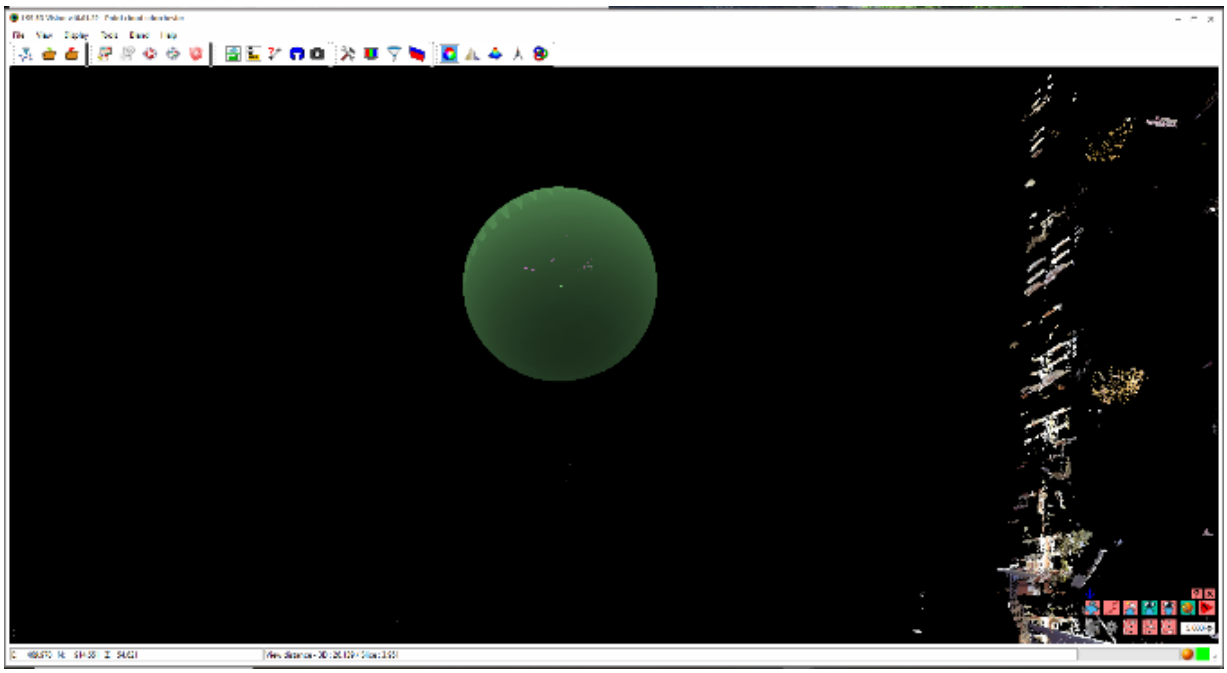
Modelling

- Repeat until the points are extracted into DTM.

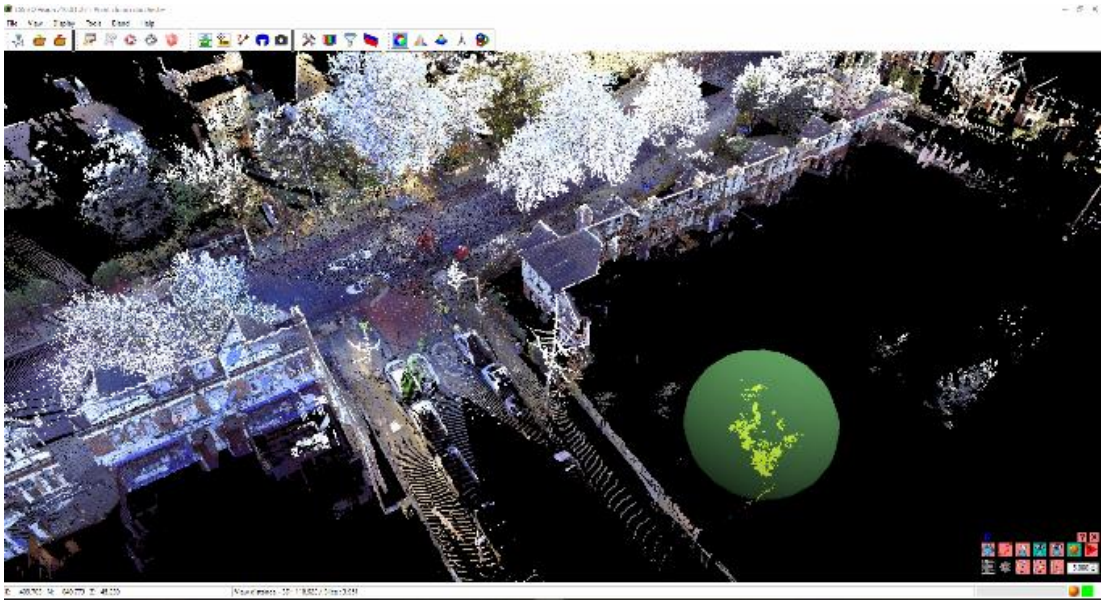
1) Outlier and Noise Filtration

For processing, the first stage is to remove the noise and filter the outliers from the point cloud. The Dorchester point cloud is shown in Fig 7.13. The scanning can cause the presence of outliers and noise. Generally, outliers and noise are due to reflection on the surfaces like mirrors and moving vehicles or people. The search sphere is used to remove points, as shown in Fig 7.15 (a); a set of outlier points are deleted using NR-S by selecting a 5m size. The sphere size 5m is selected as the outliers were far apart and proved to be a quicker and more efficient way of deleting points.

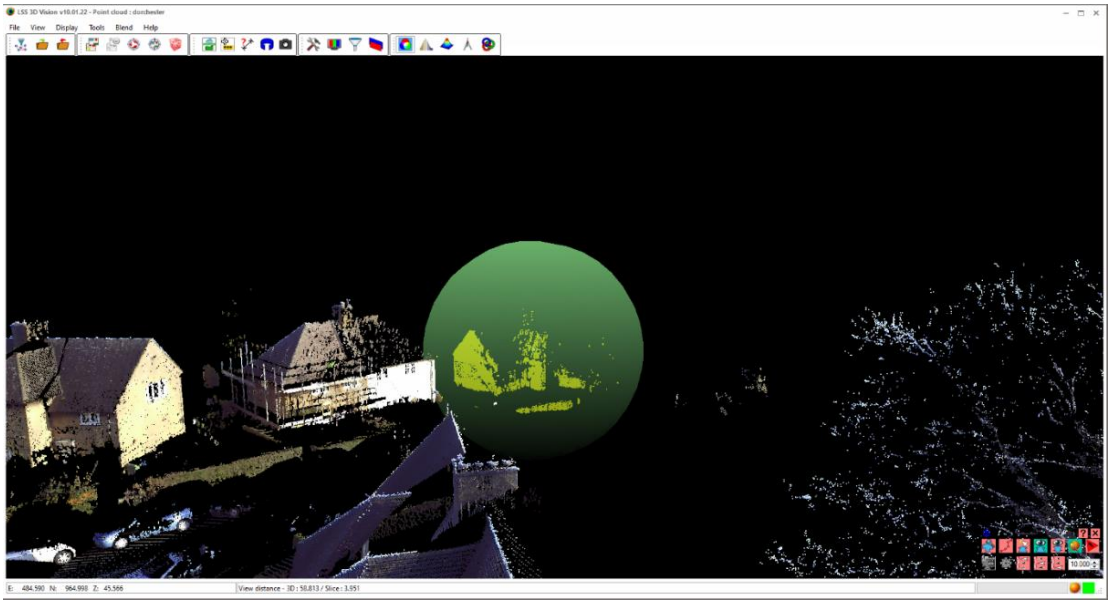
In Fig 7.15 (b) and (c), the search sphere size is 10m because a larger set of noise points is selected for deletion. For example, Fig 7.15 (b) presents a large set of tree points picked by a scanner between the buildings. As the tree points shown in the example are outside of the study area (where objects beyond the surveyed street are glimpsed by the scanner), they are classified as noise and have to be removed for efficient feature extraction. Similarly, the example shown in Fig 7.15 (c) is deleted as it is a part of the building that is not required as the main feature and is far away from the actual scanning site.



(a)



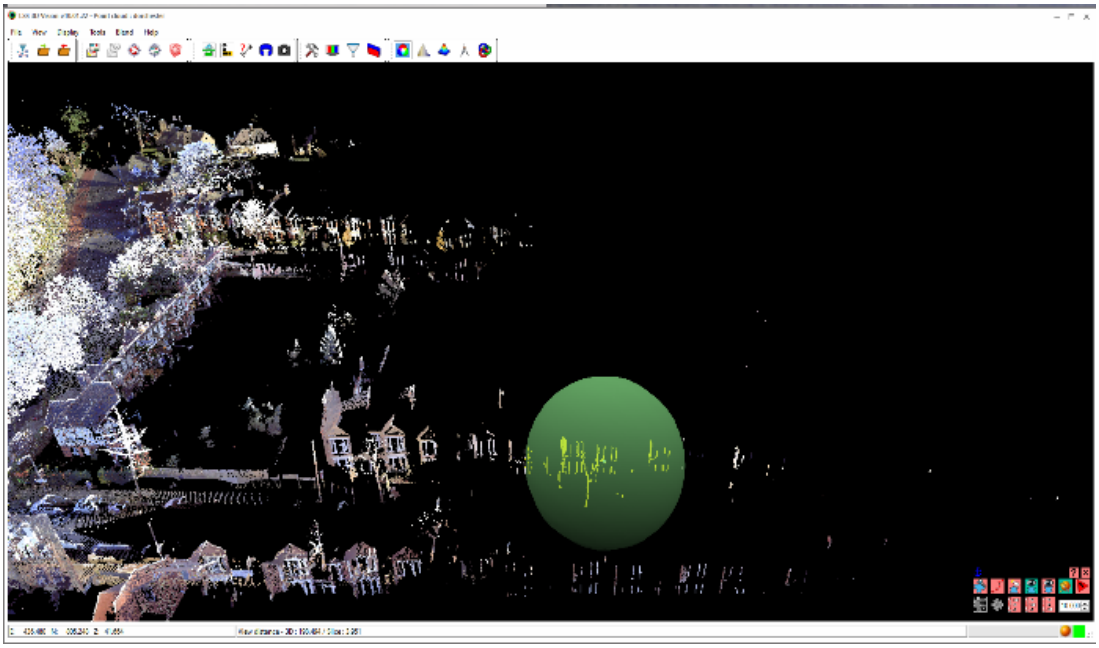
(b)



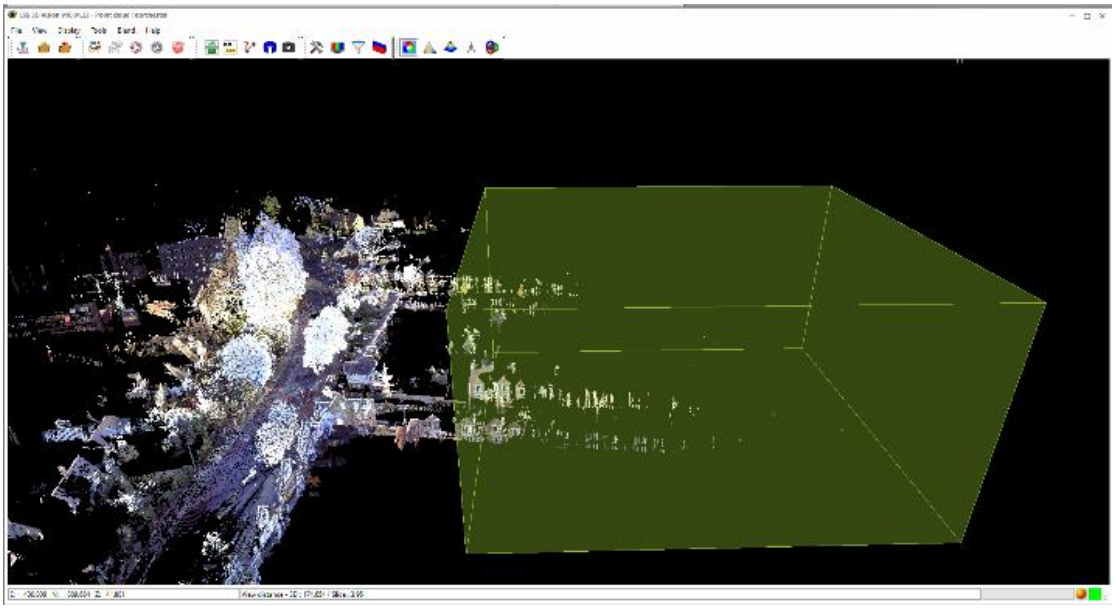
(c)

Figure 7.15 (a), (b) and (c) demonstrates the “Search sphere” to remove noise

The NR-B is used next to remove the noise spread on a larger area. In addition, the NR-S is click-heavy for the users and too manual for the deletion of points in a large spread. An example of NR-B is shown in Fig 7.16 (b) to cover a larger area at once, which is difficult to cover by NR-S in Fig 7.16 (a).



(a)

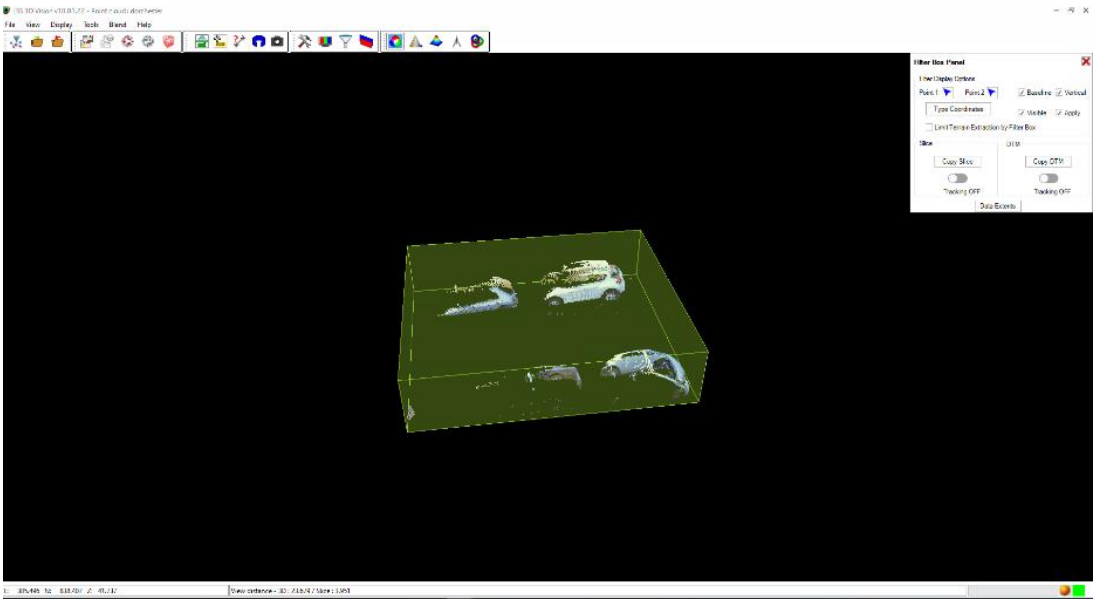


(b)

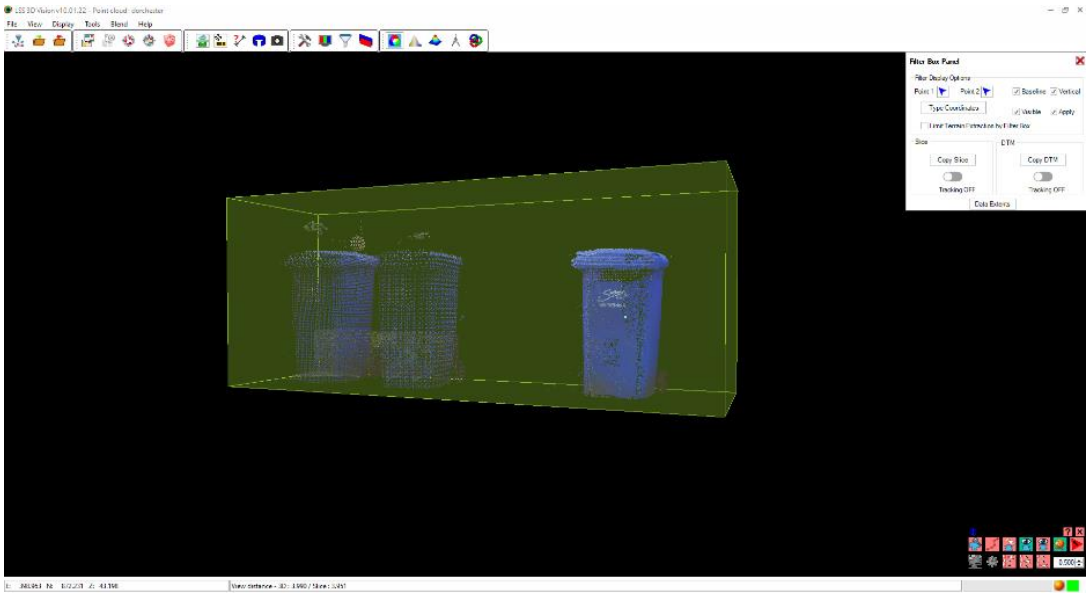
Figure 7.16 (a) Search sphere data inclusion (b) 3D Box data inclusion

NR-B is ideal for deleting the large area and for deleting the noise closer to the important features. The NR-B is also used heavily to remove unwanted features treated as noise, such as

vehicles and man-made road objects like bins. The examples of using NR-B to remove vehicles and bins are shown in Fig 7.17 (a) and (b).

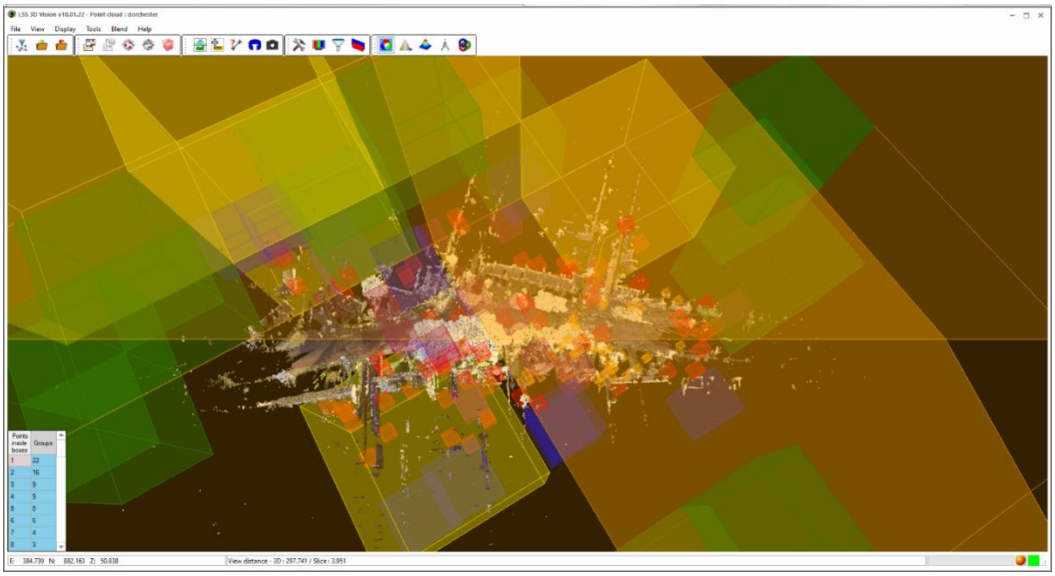


(a)

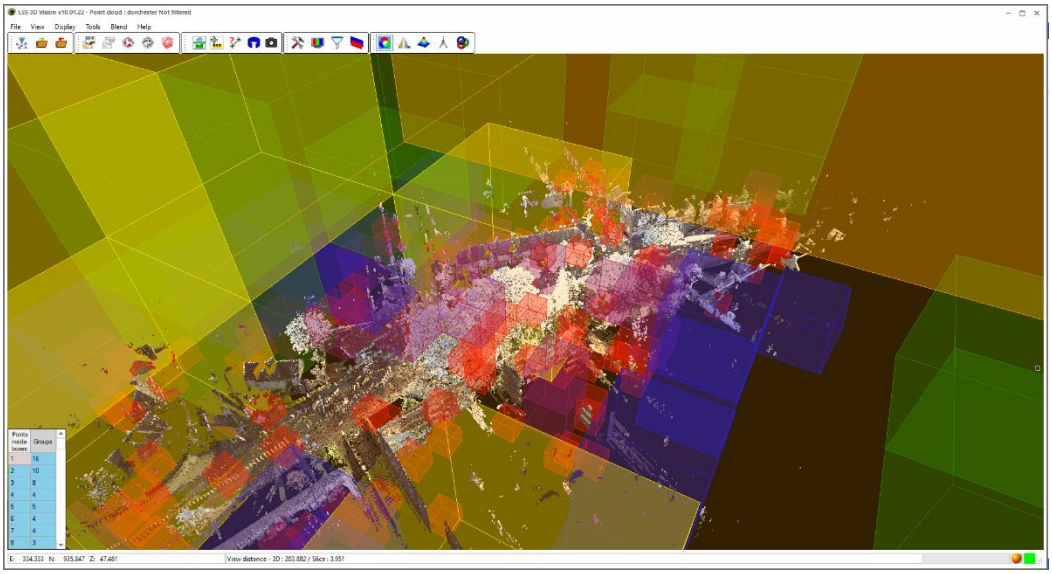


(b)

Figure 7.17 Box used to remove noise close to other objects (a) removal of cars (b) removal of bin



(a)



(b)

Figure 7.18 Filtering outliers using Oct Boxes demonstrating in (a) and (b)

Next, the outliers that are scattered all over are deleted using OF-O, shown in Fig 7.18. The points inside the box are set to 100 based on Dorchester point cloud data experience and detected points approximately by zooming in the outlier's groups. The number varies for every point cloud; therefore, the system provides users with the flexibility to alter the number accordingly. A comparison illustrates how the proposed methods worked with noise removal

and outlier filtering. Fig 7.19 shows the data before implementing proposed methods with bad points (outlier or noise), and Fig 7.20 shows the data after using the NR-S, NR-B and OF-OB.

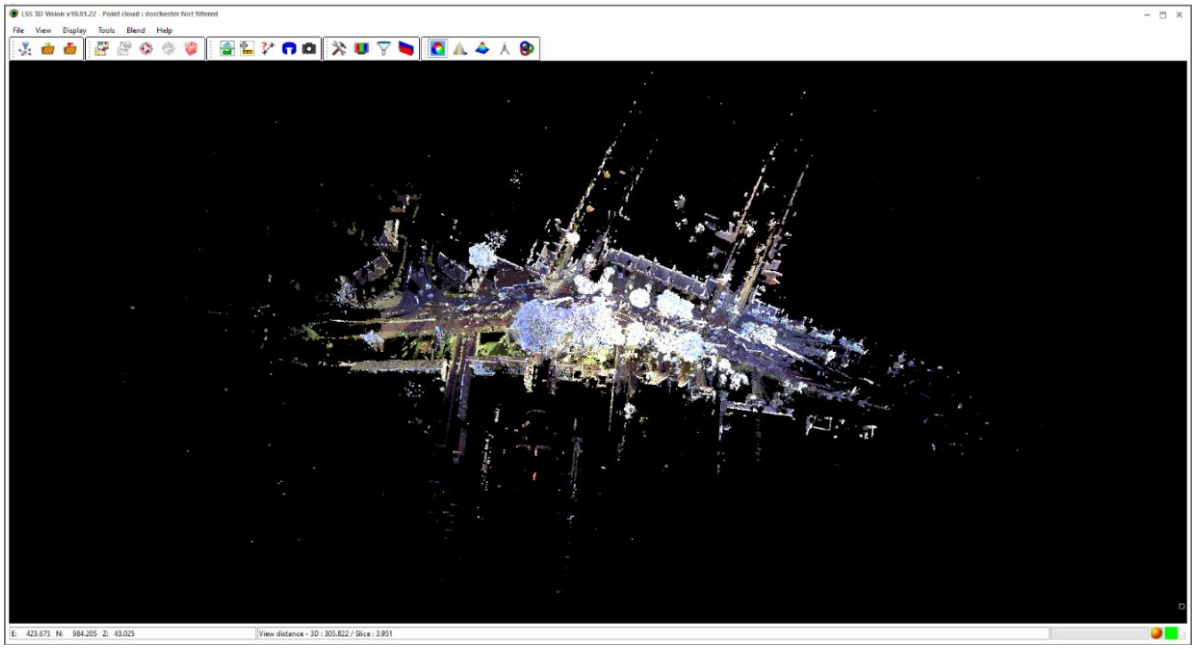


Figure 7.19 Data with outliers and noise (before NR-S, NR-B and OF-OB)

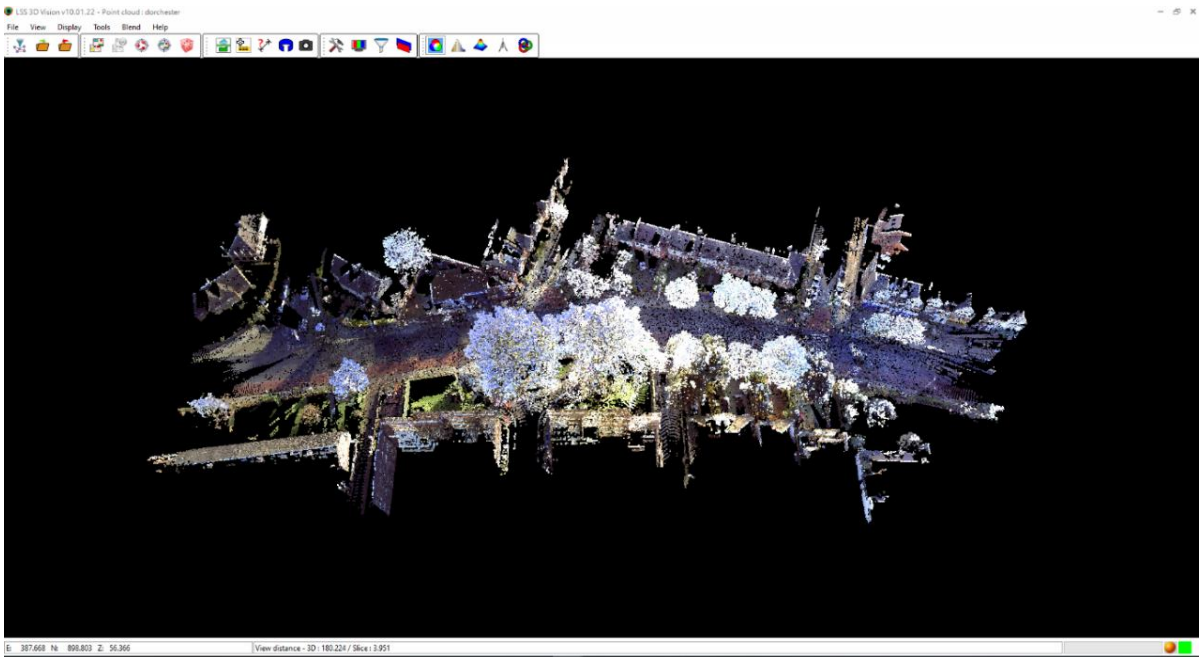
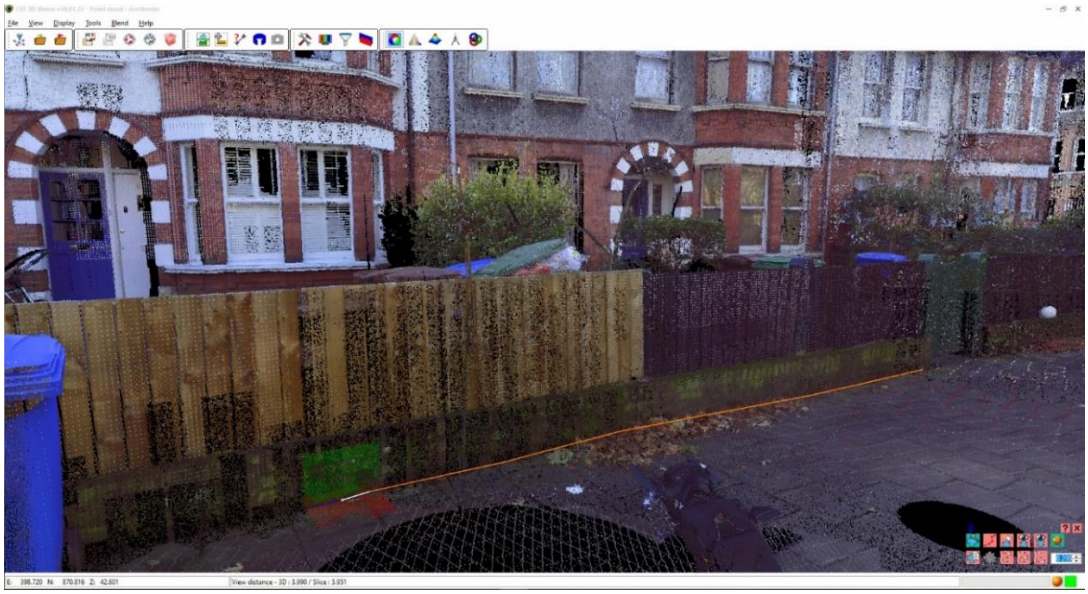


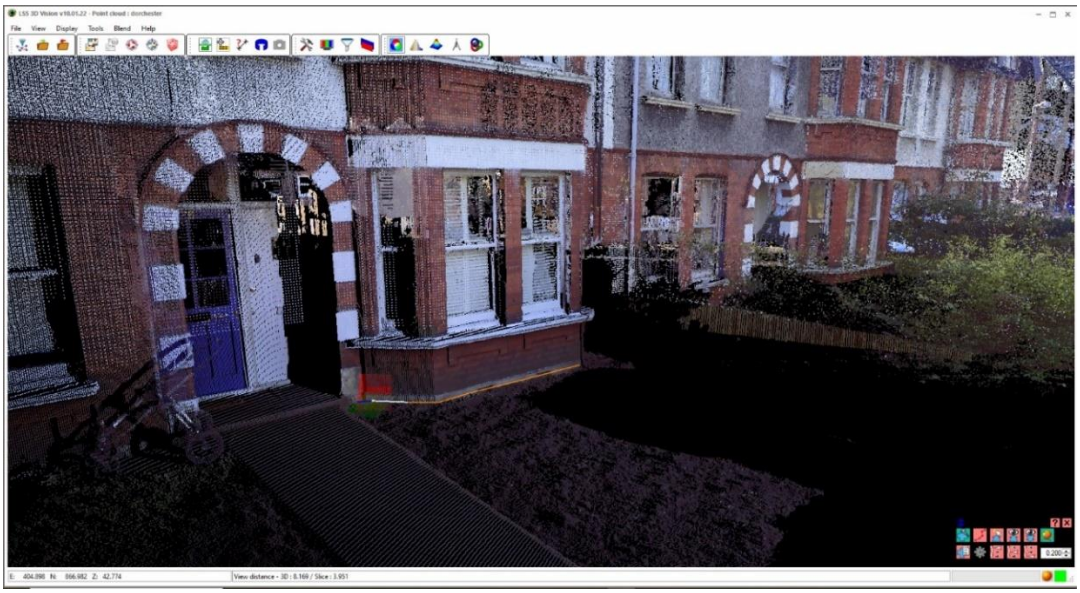
Figure 7.20 After using NR-S, NR-B and OF-OB

2) Edge Sects and Edge Stream

Once the outliers and noise points are deleted, the data is clean and ready to extract essential features. One of the essential features of point clouds is edges. The edges are extracted from different parts of the point cloud.



(a)

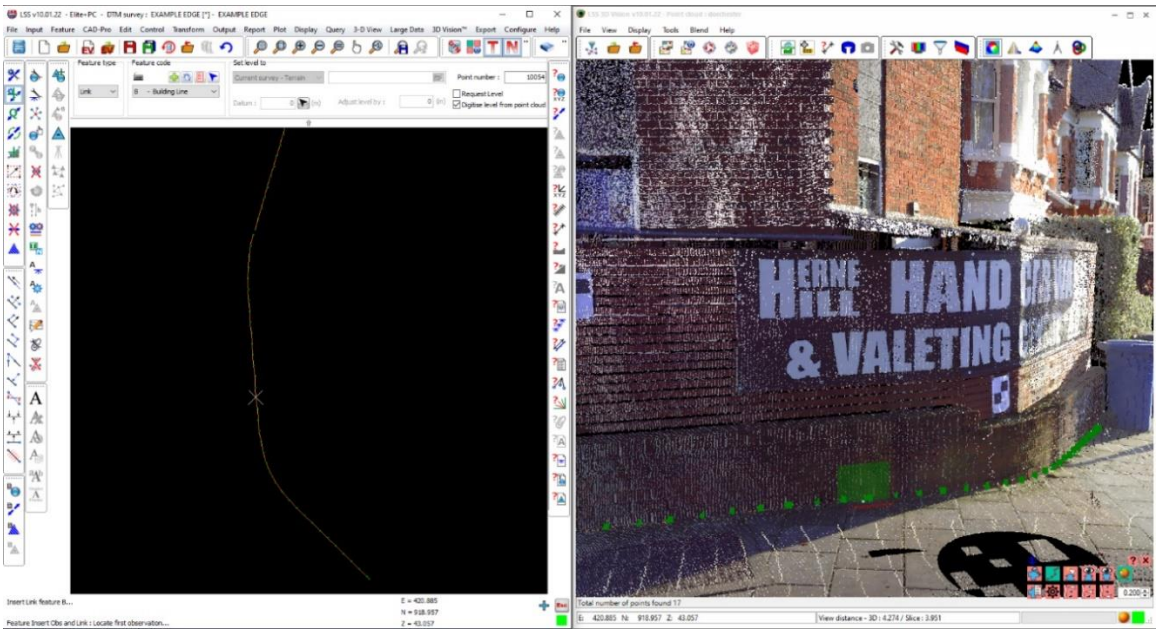


(b)

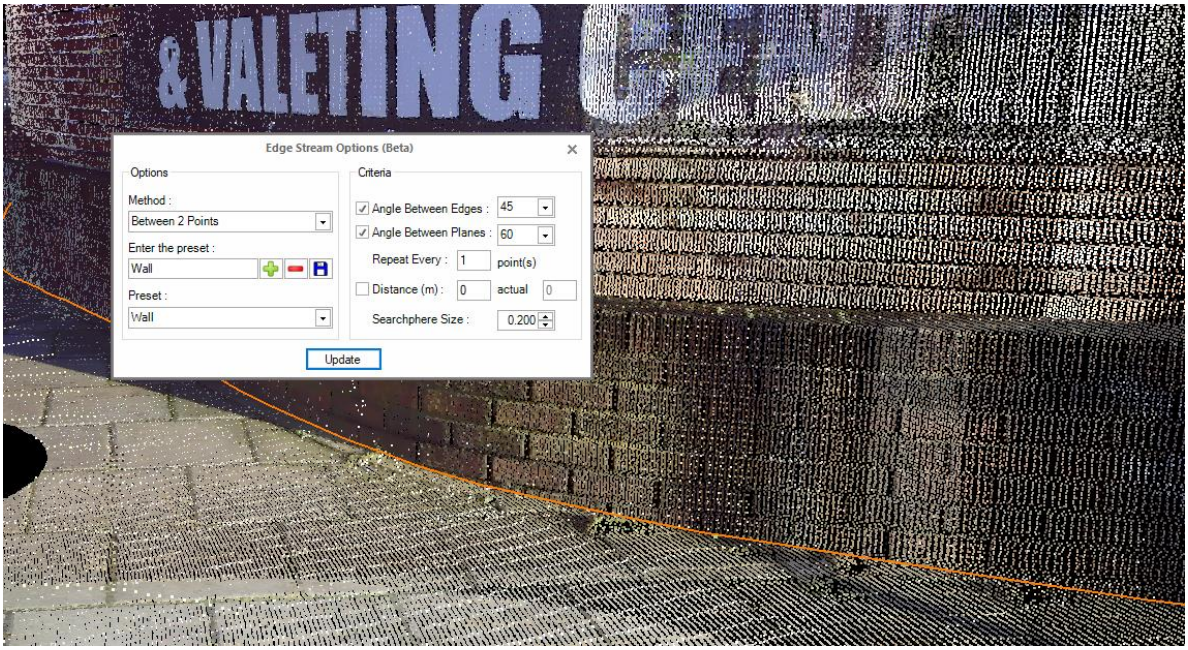
Figure 7.21 Edge detection (a) along the fence and footpath (b) along the building footprint

In the Dorchester point cloud data, an example is presented where the edges are extracted between a house fence and footpath shown in Fig 7.21 (a); the search sphere size is set as 0.5m as the fence and footpath are large areas that may not need very detailed points extracted. Furthermore, the points extracted depend on the users and their projects. The edges between the fence and footpath or wall and footpath are generally used for maintenance and hazard analysis. Another example shown in Fig 7.21 (b) is the edge extraction of the building footprint. Building footprints are of interest to surveyors and engineers. The search sphere size is set to 0.2m as detailed edge point extraction is required.

The edge stream example is demonstrated in Fig 7.22, with edges that run from one point from a bin to another point with a curve. The proposed algorithm is able to detect all the points on the edge of a feature. A total of 17 edge points are detected in the example as green points. In Fig 7.22 (a), the right side represents the edge stream option in the point cloud, and the left side represents the extracted edge points in DTM. Fig 7.22 (b) shows the settings used to detect the edge points for Fig 7.22 (a). The angle between edges and planes is set at 45 and 60 degrees, respectively. The reason is that walls are nearly perpendicular to the footpath, although they have a curve, so in order to follow the curve, the angle between edges is set to 45. The sphere size is 0.2 m to get the detail of the curve along the wall.



(a)



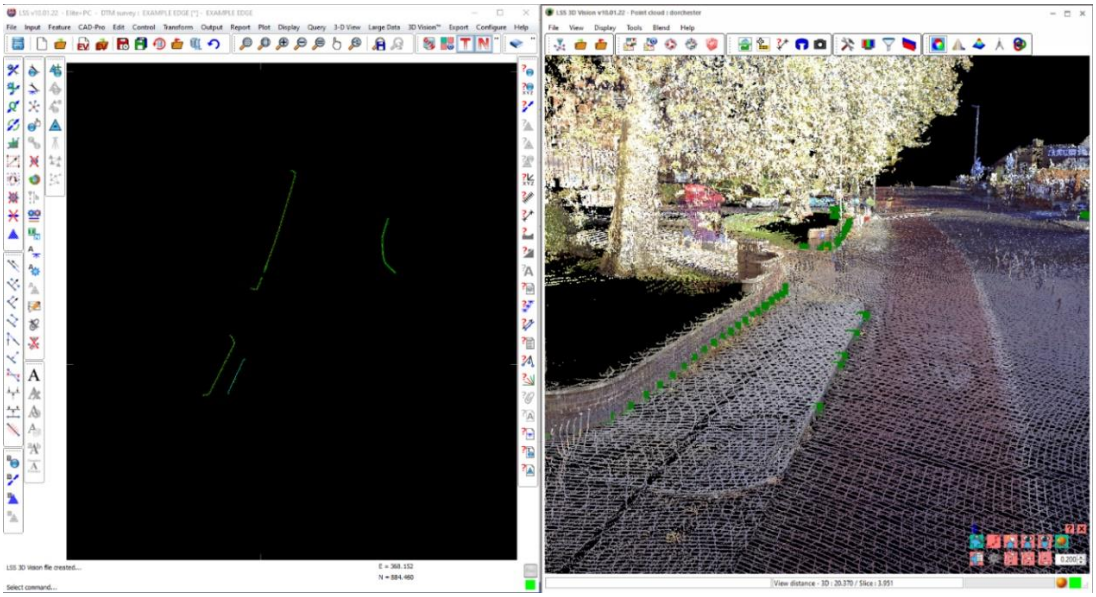
(b)

Figure 7.22 Edge stream (a) Wall and footpath (b) Settings

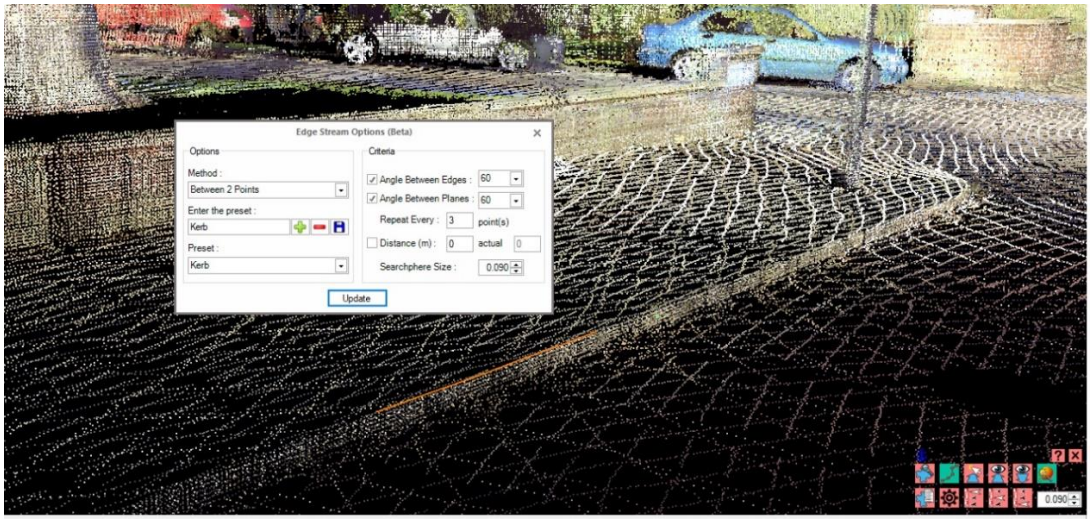
Another example is shown in Fig 7.23, which extracts the edge points from the top of a kerb. Again, the extracted edge points are shown on the left side, and the right side demonstrates the extracted kerbs in point cloud data.

Kerbs are popular features of urban point clouds. The users (surveyors and civil engineers) are interested in extracting both the top and bottom of the kerb. The kerb features are a large part of city management projects. In this example, the top of the kerb is presented.

The kerb is a relatively small feature compared to the wall in Fig 7.22; therefore, the sphere size is set to 0.09m which fits correctly on the edge of the kerbs. The angle between edges and planes is set to 60 degrees as the kerbs are straight and without sudden changes or obstacles. The ‘Repeat Every’ parameter is set to 3 as with the small sphere, there could be hundreds of points within a small distance. This parameter is a user preference as to how many points each metre could have. The settings allow changing and saving user-defined features and their angles on every point cloud data.



(a)



(b)

Figure 7.23 Edge stream (a) Top of the kerb (b) Settings

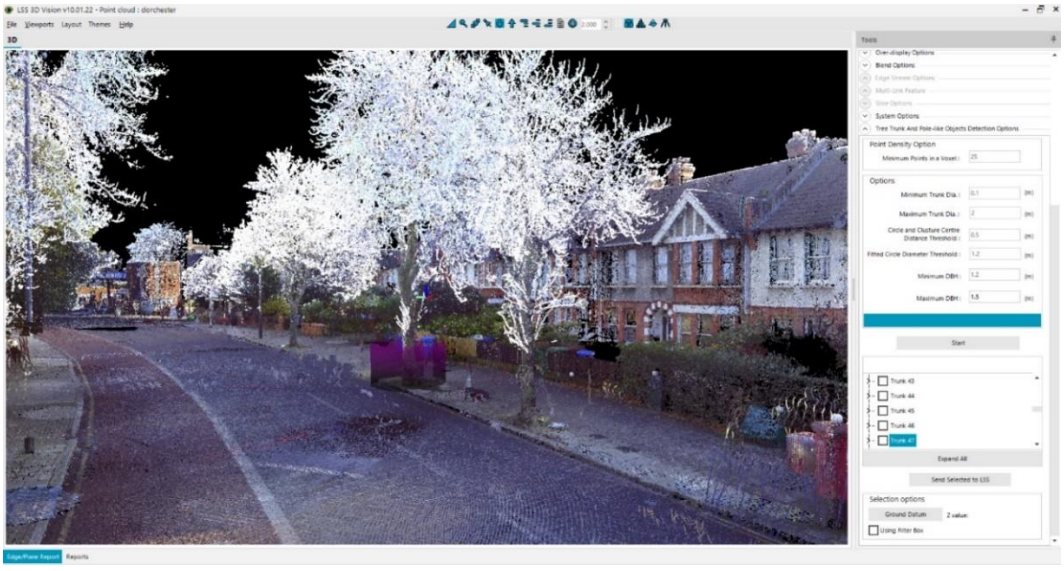
3) Tree trunks and Pole like objects

Next, after edge points detection and extraction, the next popular feature in urban point clouds is trees and poles. Taking advantage of 3D point clouds and visualisation, the trees and pole objects are used extensively in city models, city management, risk analysis, classification and

road management. The Dorchester point cloud is used to present an example of tree trunk detection in Fig 7.24 and Pole object detection in Fig 7.25. Each detected trunk and pole centre and radius is extracted into the DTM. The tree trunks and pole parameters are set as follows;

- The minimum number of points in the voxel is set as 25 because, in this dataset, it is clear that trees/poles have high-density points and therefore do not need any lower than 25.
- The minimum and maximum trunk diameters are set as 0.1m to 2m as some Dorchester trees are old with a huge girth.
- The distance threshold between the circle and the voxel cluster centre is set as 0.5m. Anything bigger will include larger tree trunks, and anything lower will include slim trunks and poles with a smaller radius.
- The DBH (Diameter at breast height) for trunks and poles are set between 1.2m and 1.5m, as the average British standard for DBH is 1.4m.

Fig 7.24 (a) shows a roadside tree trunk, and Fig 7.24 (b) shows a tree inside a property. Fig 7.25 shows a pole structure detected on a roadside. The parameters for poles are the same as the proposed algorithm accommodates both trunks and poles within the parameter settings.



(a)

(b)

Figure 7.24 Detected tree trunks highlighted by fuchsia cylinder

Figure 7.25 Detected pole structure highlighted by fuchsia cylinder

4) DTM

Lastly, all the information is extracted into a DTM from point clouds. DTM is a digitised version of a map generated using points, links and surfaces. The point clouds are data in 3D space, but the extraction into DTM defines those features and maps them so that they can be

registered and used for analysis. The surveyors and civil engineers utilise the results of DTM to distribute them to their clients to show and demonstrate data before commencing any projects. The example of DTM is shown in Fig 7.26, and Fig 7.27 shows the Dorchester point cloud overlapped by the DTM in Fig 7.26.

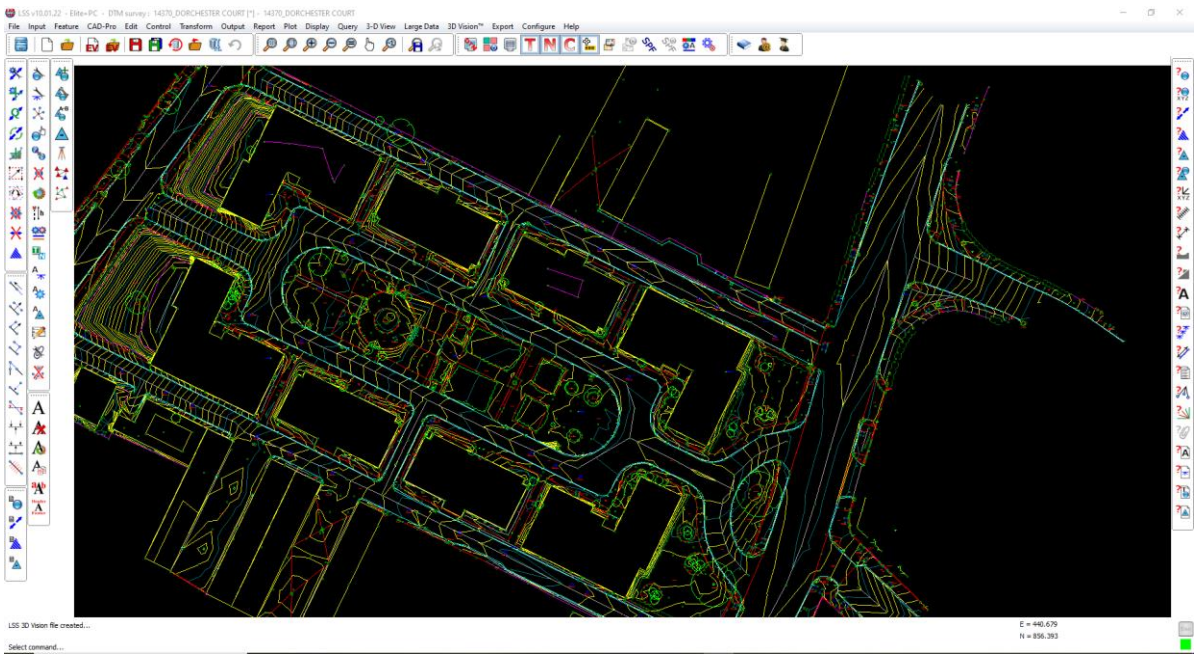


Figure 7.26 Digital Terrain Model (DTM) shown in LSS

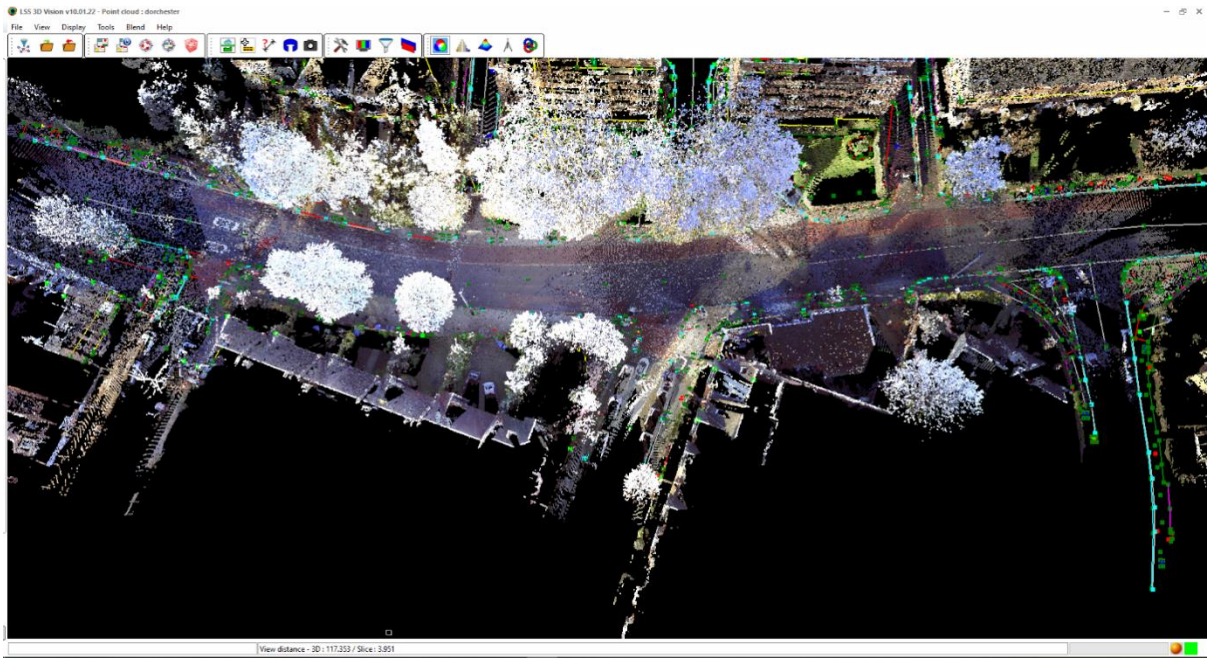


Figure 7.27 Overlapped DTM in 3D Vision

7.6 Chapter Summary

This chapter demonstrates the implementation of the proposed methods/algorithms on the commercial software LSS and its point cloud product 3D Vision. Each project (proposed method) was implemented using a software development cycle and a model. The spiral model has been used for the software development project previously, and recently agile scrum has been implemented to deliver software projects in sprints. The programming language C# is discussed, which is used to accomplish the coding for all the proposed methods and algorithms. The code reusability was kept in mind for future proofing of the solutions used by other developers. Project management and quality analysis have been achieved when executing each project. While designing the user interface, user intuitiveness is the priority, with less user intervention and easy-to-use commands while running the program.

The proposed methods and algorithms are implemented on software called 'LSS', a powerful Land Surveying Software and its point cloud product '3D Vision'. The market analysis is provided to show the understanding of the market. The types of software used by surveyors and civil engineers in the industry in the UK are presented by categorising as hardware companies, software companies and free (non-commercial) software solutions for point clouds.

Lastly, a case study is presented by implementing various point cloud processing stages to process point cloud data. The processing starts with noise removal and outlier filtration. Once the data is clean, the feature detection and extraction methods are applied, leading to the next stage to detect the edge and edge stream. Furthermore, the trunks and pole objects are detected from the point clouds to be modelled into the DTM.

Chapter 8 Conclusion and Future Work

This thesis aims to understand the challenges current point cloud processing methods face and design and develop solution/s to accommodate them. Point cloud processing became essential with the emergence of scanning technology, which generates huge datasets. Point clouds can contain anything from a few hundred to billions of points. As technology advanced, these numbers rose, the challenge being handling and using these data for rightful purposes. However, the process of extracting information from geometrical and non-geometrical features from ubiquitous point clouds is challenging. To this end, a set of research questions emerged by undertaking a literature review of existing methods, as presented in Chapter 2, which directs the research to find solutions to the problems. The problems that formed the research question are as follows:

1. The challenges for surveyors and civil engineers to process 3D point clouds.
2. The challenges in feature detection from point clouds.
3. The challenges during the filtration, classification, and edge detection.

The conclusion of these research questions is described in Section 8.1. After which, eight key contributions of this thesis are described in Section 8.2 and followed by future works of proposed methods and algorithms in Section 8.3.

Point cloud processing is the process of extracting features from a point cloud into a meaningful model. This thesis divides point cloud processing into a number of techniques and methods. First is **Filtration,** where the outliers and noise points are filtered. The second is **Edge Detection**, where edges of different features are detected. The third is **Feature detection** by segmentation and extracting the features, and finally extracting all information into the DTM model.

8.1 Achievements

The surveying industry is one of the professions that require continuous data acquisition at every step of the project lifecycle, from building plans to road surveys. Therefore, properly accessing, implementing and managing the data is very important. Unfortunately, the current algorithms have limitations and issues that prohibit the users from working efficiently. The research and research objectives (carried out in Chapter 2) resulted in designing and developing the proposed algorithms to solve the challenges. As a result, the proposed novel and robust algorithms and methods will allow the users to overcome the challenges and use the application in point clouds.

The first research objective was to identify and evaluate 3D point cloud processing challenges with current methods. Surveyors and civil engineers use laser scanning as the preferred tool to capture the data because of its portability, comprehension and precision. More and more, laser scanning has become popular without a proper workflow. Processing these data (point cloud) requires accurate analysis and procedures. The scanner technology innovations and detailed capturing of data resulted in very large point clouds. The common challenge is to process these huge datasets accurately and quickly.

The most obvious and immediate step for processing large point clouds is to remove the bad points. The bad points are not part of the important features and deleting them would reduce the point cloud size and enable faster processing. These bad points are noise/outliers that are relatively present within the good points (points that belong to an important feature). The challenge of the existing method is to remove these noise/outliers efficiently to preserve the primitive shapes and geometrical features. This results in the second research objective, which is to design and develop a method to overcome the problems of separating outlier/noise. Despite saying that separating is the key, the problem lies with the points that are in very close approximation to the primitive features in the point cloud. This led to the development of the tools mentioned in Chapter 4 NR-S, NR-B and OF-OB to remove noise and filter outliers.

Then after the cleaning of data, the important element is extracting the features; this becomes the third research objective to research/investigate the existing methods and identify their limitations. A considerable amount of point cloud acquisition is in the built environment, such as city scenes, residential areas, construction sites, city roads etc. Therefore, the features within

these environments become important to detect and extract for modelling. However, there are also several drawbacks inherent to feature detection in point clouds. The most common problem with the detection is that laser scanners cannot offer a full-scale representation of an object (360 degrees in 3D) unless the scanner has been positioned in multiple locations in order to view all objects from all angles, which is not usually cost-effective. Therefore, the detection algorithms must compensate for the fact that only half of an object may exist in the point cloud (such as a tree trunk or lamppost).

Another common problem is the presence of gaps, blank spaces and missing data. These gaps are due to laser beams not penetrating objects because obstacles in the line of sight create a shadow (gap) in point clouds. An intention to overcome these issues led to the fourth and fifth research objectives presented in Chapters 5 and 6.

Several important features are present within the point cloud, but edges are most important for surveyors and civil engineers. Edges of various features like buildings, kerbs, roofs etc., are useful. The second feature is trunks and poles, which play an important role in city planning and management. Therefore, this led to feature recognition algorithms presented in Chapters 5 and 6. The geometrical and non-geometrical shapes in point clouds are mostly user-controlled settings for detection algorithms that are intuitive as they are directly related to the feature's geometric properties, like distance and orientation. This is the key to successfully implementing an efficient and user-friendly system. The user settings and parameters also provide the flexibility to implement the proposed algorithm in a wide variety of situations, including the built environment, construction projects, quarry operations, forestry and the undeveloped natural environment. As a result, the proposed methods and algorithms are able to deliver great completeness and correct feature detection with high accuracy. In addition to the advantages, the proposed method works directly on the point clouds that are real-world scans; no data conversion or manipulation is required compared to many existing methods that were tested on synthetic data. Furthermore, the academic research in the industry for feature detection algorithms provides theoretical implementations, whereas the proposed methods and algorithms are implemented practically on commercial software used by large UK-based organisations.

8.2 Contribution to New Knowledge Generation

In summary, there are eight key contributions in this thesis for point cloud processing, including filtration, segmentation, edge detection and cylindrical feature detection, as follows:

- A tool called **Search Sphere** is introduced for the first time for various applications in this thesis. The search sphere is used for the analysis of the inclusion points and also to sample the points.

- A tool called **OctBox** is introduced and used to analyse points which are derived from the Octree structure. This thesis presents the first-ever use of Octree to define the bounding of a 3D box to filter outliers.

- A 3D **Box** tool is introduced that can handle the points inside by changing dynamically to remove noise from the point clouds.

- A **PCA-based Algorithm for Edge Detection** has been introduced, discussed, analysed and implemented on the commercial software product 3D Vision. The algorithm is the first PCA application of the best-fit planes for obtaining an edge.

- The robust **Edge Stream Method (**extension of Edge detection) is introduced, analysed, and implemented on the commercial software 3D Vision. Edge stream allows the automatic extraction of several edges along a line of any feature.

- A gridding system called **Terrain Extraction** is introduced, which is used to separate the ground and non-ground points of point clouds. Terrain extraction sub-samples the points to achieve further analysis of points. Ground points act as a terrain for different types of point cloud data. On the other hand, non-ground points are used to implement feature detection algorithms that are not part of the terrain.

- A segmentation and clustering technique using voxels has been introduced for fast extracting and finding neighbouring points in the point clouds.

- A **Voxel-based Algorithm to find Cylindrical Objects** has been introduced, discussed, analysed and implemented on the commercial software product 3D Vision. The algorithm can identify trunks and pole-like objects in point clouds.

8.3 Limitations and Future Work

While this thesis demonstrated the excellence of point cloud processing methods and algorithms, there are still possibilities to carry out future research in this area. The biggest challenge is to process millions of points in point clouds captured by new scanning technologies. Point cloud processing steps address this, which converts the point clouds into a model.

The important point cloud processing step is the filtration of noise and outliers. Due to time constraints, the proposed method's limitation is that the tools are used manually. The points are analysed and categorised by users, which means that the proposed method is fast and flexible to various point clouds. However, due to the method being manual, it can be click-heavy to use. Therefore, the proposed methods can be further improved in order to detect them automatically by analysing the points and their neighbours using the proposed tool in Chapter 4. For example, the clustering techniques described in Chapter 6 can be implemented in Oct Boxes for a detailed analysis of the points cluster. Furthermore, the grouping of points belonging to the same object can solve the problem of noise detection (points that are close to the features to be extracted), such as vegetation near a building.

Another important step of point cloud processing is feature detection. Various steps are involved in detecting the features, such as outlier and noise filtration, segmentation of the points for classification, clustering of the points, and modelling. With respect to feature detection, there are several features of importance in a typical point cloud that can become vital in point cloud processing. The future work will involve the detection and extraction of other features like building floorplan, vehicles, road markings etc. Examples of a few future work items are listed below.

Firstly, the visualisation of the detected features can be improved to demonstrate to the users very clearly which points belong to which feature. The easiest way to categorise points is during

the acquisition process and to specify the points at that moment. This process can be repeated until all the points are classified properly.

Secondly, while the proposed algorithm in this thesis is very effective in finding the edges, in the future, edge detection can be extended to find corners with the detection of three planes. The current limitation of the proposed edge detection methods (edge detection and edge stream options) is that they cannot detect the points of the corner features because the method was designed to extract the edges based on two surfaces. The future work will be implementing corner detection (by extending the current work to detect three surfaces), which would also be relevant in the edge stream algorithm, potentially making it more efficient.

Thirdly, a large number of point clouds are captured in man-made environments like cities, industrial areas, development sites, etc. It is essential to capture the features in these areas. There are several options for future work on the cylindrical detection algorithm presented in Chapter 6. The trunks can be extended to detect the foliage of the tree, providing the users with the parameters of the whole tree, such as crown height and spread. The advantage of tree detection is that the users can either use it to map the trees in the area or to delete the trees that are classified as noise. As for the pole-like objects, the future prospect is to classify them further into individual profiles, including utility poles, streetlights, traffic lights, road signs, flag poles and parking meters. The profiling can be done by saving the templates of all possible structures in a typical point cloud. After that, these templates are compared against the detected structures and are then classified.

Finally, other areas of point cloud processing could benefit from feature detection. Segmentation and region growing methods can be used to solve real-world point cloud scenario problems such as filling the gaps, categorising the features and data thinning.

Bibliography

Abdi, H. and Williams, L.J. (2010) 'Principal Component Analysis', *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(4), pp. 433–459. doi: 10.1002/wics.101.

Achlioptas, P., Diamanti, O., Mitliagkas, I. and Guibas, L (2018) 'Learning Representations and Generative Models for 3D Point Clouds', in *International conference on machine learning*. PMLR, pp. 40–49. doi: 10.48550/arXiv.1707.02392.

Adamson, A., Alexa, M. and Berlin, T.U. (2006) 'Point-Sampled Cell Complexes', *ACM SIGGRAPH*, pp. 671–680. doi: 10.1145/1179352.1141940.

Alexa, M., Behr, J., Cohen-Or, D., Fleishman, S., Levin, D. and Silva, C.T. (2001) 'Point Set Surfaces Related Papers Computing and Rendering Point Set Surfaces Point Set Surfaces', in *IEEE Proceedings Visualization 2001. VIS '01*, pp. 21–29. doi: 10.1109/VISUAL.2001.964489.

Alexa, M., Behr, J., Cohen-Or, D., Fleishman, S., Levin, D. and Silva, C.T. (2003) 'Computing and Rendering Point Set Surfaces', *IEEE Transactions on Visualization and Computer Graphics*, 9(1), pp. 3–15. doi: 10.1109/TVCG.2003.1175093.

Alexa, M. and Adamson, A. (2004) 'On Normals and Projection Operators for Surfaces Defined by Point Sets', *Eurographics Symposium on Point-Based Graphics*, pp. 149–155.

Aljumaily H, Laefer D and Cuadra D (2017) 'Urban Point Cloud Mining Based on Density Clustering and MapReduce', *ASCE Journal of Computing in Civil Engineering*, 31(5) 04017021.

Aluja-Banet T, Morineau A and Sanchez G (2018) Formulas for PCA - PCA for Data Science, *Book: Principal Component Analysis for Data Science*. Available at: https://pca4ds.github.io/formulas-for-pca.html (Accessed: 21 July 2022).

Amenta, N. and Kil, Y.J. (2004) 'Defining Point-set Surfaces', in *ACM Transactions on Graphics*, pp. 264–270. doi: 10.1145/1015706.1015713.

Amiri, N., Polewski, P., Yao, W., Krzystek, P. and Skidmore, A.K. (2017) 'Detection of Single Tree Stems in Forested Areas from High Density ALS Point Clouds Using 3d Shape Descriptors', in *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*. Copernicus GmbH, pp. 35–42. doi: 10.5194/isprs-annals-IV-2-W4-35-2017.

Ando, S. (2000) 'Image Field Categorization and Edge/Corner Detection from Gradient Covariance', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(2), pp. 179–190 doi: 10.1109/34.825756.

Arvanitis, G., Lalos, A.S., Moustakas, K. and Fakotakis, N. (2018) 'Outliers Removal of Highly Dense and Unorganized Point Clouds Acquired by Laser Scanners in Urban Environments', in *Proceedings - 2018 International Conference on Cyberworlds, CW 2018*. Institute of Electrical and Electronics Engineers Inc., pp. 415–418. doi: 10.1109/CW.2018.00080.

Avram, D., Bratosin, I., Ilie, D. CALIN, L. (2016) 'Surveying Theodolite Between Past and Future', *Journal of Young Scientist*, 4, pp 129-134. ISSN 2284-8017.

Avron, H., Sharf, A., Greif, C. and Cohen-Or, D. (2010) 'ℓ1-Sparse Reconstruction of Sharp Point Set Surfaces', *ACM Transactions on Graphics*, 29(5), pp 135:1-135:12. doi: 10.1145/1857907.1857911.

Balado, J., Martínez-Sánchez, J., Arias, P. and Novo, A. (2019) 'Road Environment Semantic Segmentation with Deep Learning From MLS Point Cloud Data', *Sensors (Switzerland)*, 19(16), 3466. doi:10.3390/s19163466.

Balta, H., Velagic, J., Bosschaerts, W., de Cubber, G. and Siciliano, B. (2018) 'Fast Statistical Outlier Removal Based Method for Large 3D Point Clouds of Outdoor Environments', in *IFAC-Papers On Line*. Elsevier B.V., pp. 348–353. doi: 10.1016/j.ifacol.2018.11.566.

Barnett T. P and Preisendorfer, R. (1987) 'Origins and Levels of Monthly and Seasonal Forecast Skill for United States Surface Air Temperatures Determined by Canonical Correlation Analysis', *Monthly Weather Review*, 115(9), pp. 1825–1850. doi: 10.1175/2010JCLI3527.1

Barnett Vic and Lewis Toby (1994) *Outliers in Statistical Data*. 3rd edn. New York: Wiley.

Bazazian, D., Casas, J.R. and Ruiz-Hidalgo, J. (2015) 'Fast and Robust Edge Extraction in Unorganized Point Clouds', in *2015 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*. Adelaide, Australia: IEEE, pp. 1–8. doi: 10.1109/DICTA.2015.7371262.

Bazazian, D. and Parés, M.E. (2021) 'EDC-net: Edge Detection Capsule Network for 3D Point Clouds', *Applied Sciences (Switzerland)*, 11(4), pp. 1–16. doi: 10.3390/app11041833.

Becker, C., Rosinskaya, E., Häni, N., d'Angelo, E. and Strecha, C. (2018) 'Classification of Aerial Photogrammetric 3D Point Clouds', *Photogrammetric Engineering and Remote Sensing*, 84(5), pp. 287–295. doi: 10.14358/PERS.84.5.287.

Behley, J., Steinhage, V. and Cremers, A.B. (2015) 'Efficient Radius Neighbor Search in Three-dimensional Point Clouds', *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, pp. 3625–3630. doi: 10.1109/ICRA.2015.7139702

Belton, D. and Kwang-Ho, B. (2009) 'Tracking Roadside Kerbs in Terrestrial Laser Scanner Point Clouds Using Principal Component Analysis', in *Proceedings of the Surveying & Spatial Sciences Institute Biennial International Conference*. The Institute, pp. 219–229.

Belton, D., Moncrieff, S. and Chapman, J. (2013) 'Processing Tree Point Clouds using Gaussian Mixture Models', in *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*. Copernicus GmbH, pp. 43–48. doi: 10.5194/isprsannals-II-5-W2-43-2013.

Biosca, J.M. and Lerma, J.L. (2008) 'Unsupervised Robust Planar Segmentation of Terrestrial Laser Scanner Point Clouds Based on Fuzzy Clustering Methods', *ISPRS Journal of Photogrammetry and Remote Sensing*, 63(1), pp. 84–98. doi: 10.1016/j.isprsjprs.2007.07.010.

Boehler, W. and Marbs, A. (2004) '3D Scanning and Photogrammetry for Heritage Recording: A Comparison', in *Proc. 12th International Conference on Geoinformatics − Geospatial Information Research*, pp. 7–9.

Boehm, B.W. (1988) ‘A Spiral Model of Software Development and Enhancement’, *Computer*, 21(5), pp. 61–72. doi: 10.1109/2.59.

Borenstein, G. (2012) *Making Things See 3D Vision with Kinect, Processing, Arduino, and MakerBot*. 1st edn. Canada: Maker Media Inc.

Boster, M. (2016) *What is an Edge in Math? Study.com*. Available at: https://study.com/academy/lesson/what-is-an-edge-in-math.html (Accessed: 9 January 2019).

Boulic, R. and Renault, O. (1991) ‘3D Hierarchies for Animation’, in Magnenat-Thalmann Nadia and Thalmann Daniel (eds) *New Trends in Animation and Visualization*. England: John Wiley & Sons ltd, pp. 59–78.

Bremer, M., Wichmann, V. and Rutzinger, M. (2013) ‘Eigenvalue and Graph-based Object Extraction from Mobile Laser Scanning Point Clouds’, in *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*. Copernicus GmbH, pp. 55–60. doi: 10.5194/isprsannals-II-5-W2-55-2013.

Breunig, M.M., Kriegel, H.-P., Ng, R.T. and Sander, J. (2000) ‘LOF: Identifying Density-Based Local Outliers’, in *Proceedings of the 2000 ACM SIGMOD international conference on Management of data - SIGMOD ’00*. New York, New York, USA: ACM Press, pp. 93–104. doi: 10.1145/342009.335388.

C# - Basic Syntax (2022) *Tutorials point*. Available at: https://www.tutorialspoint.com/csharp/csharp_basic_syntax.htm (Accessed: 14 July 2022).

Cai, G., Jiang, Z., Wang, Z., Huang, S., Chen, K., Ge, X. and Wu, Y. (2019) ‘Spatial aggregation net: Point cloud semantic segmentation based on multi-directional convolution’, *Sensors (Switzerland)*, 19(19). doi: 10.3390/s19194329.

Cabo, C., Ordoñez, C., García-Cortés, S. and Martínez, J. (2014) ‘An Algorithm for Automatic Detection of Pole-like Street Furniture Objects from Mobile Laser Scanner Point Clouds’, *ISPRS Journal of Photogrammetry and Remote Sensing*, 87, pp. 47–56. doi: 10.1016/j.isprsjprs.2013.10.008.

Carr, J.C. and Slyder, J.B. (2018) 'Individual Tree Segmentation from a Leaf-off Photogrammetric Point Cloud', *International Journal of Remote Sensing*, 39(15–16), pp. 5195–5210. doi: 10.1080/01431161.2018.1434330.

Carrea, D. *et al.* (2021) 'Matlab virtual toolbox for retrospective rockfall source detection and volume estimation using 3d point clouds: A case study of a subalpine molasse cliff', *Geosciences (Switzerland)*, 11(2), pp. 1–19. doi: 10.3390/geosciences11020075.

Catalucci, S. *et al.* (2018) 'Comparison between point cloud processing techniques', *Measurement*, 127, pp. 221–226. doi: 10.1016/J.MEASUREMENT.2018.05.111.

Chatterjee, A. (2000) 'An Introduction to the Proper Orthogonal Decomposition', *Current Science*, 78(7), pp. 808–817.

Che, E., Jung, J. and Olsen, M.J. (2019) 'Object Recognition, Segmentation, and Classification of Mobile Laser Scanning Point Clouds: A State of the Art Review', *Sensors (Switzerland)*, 19(4), p. 810. doi: 10.3390/s19040810.

Chen, C., Li, X., Belkacem, A.N., Qiao, Z., Dong, E., Tan, W. and Shin, D. (2019) 'The Mixed Kernel Function SVM-Based Point Cloud Classification', *International Journal of Precision Engineering and Manufacturing*, 20(5), pp. 737–747. doi: 10.1007/s12541-019-00102-3.

Chen, M., Feng, A., McAlinden, R. and Soibelman, L. (2020) 'Photogrammetric Point Cloud Segmentation and Object Information Extraction for Creating Virtual Environments and Simulations', *Journal of Management in Engineering*, 36(2), p. 04019046. doi: 10.1061/(asce)me.1943-5479.0000737.

Chen, Y.H. and Liu, C.Y. (1997) 'Robust Segmentation of CMM Data Based on NURBS', *The International Journal of Advanced Manufacturing Technology*, 13(8), pp. 530–534. doi: 10.1007/BF01176296.

Cheng C. (2022) 'Principal Component Analysis (PCA) Explained Visually with Zero Math', *Towards Data Science*, 3 February. Available at: https://towardsdatascience.com/principal-component-analysis-pca-explained-visually-with-zero-math-1cbf392b9e7d (Accessed: 21 July 2022).

Chernov N. (2012) *Circular and Linear Regression: Fitting Circles and Lines by Least Squares: C++ Codes for Fitting Ellipses, Circles, Lines*. 1st edn. New York: Chapman & Hall/CRC.

Clarenz, U., Rumpf, M. and Telea, A. (2004) 'Fairing of point based surfaces', in *Proceedings of Computer Graphics International Conference, CGI*, pp. 600–603. doi: 10.1109/CGI.2004.1309272.

Collins, M., Dasgupta, S. and Schapire, R.E. (2001) 'A Generalization of Principal Component Analysis to the Exponential Family', *Advances in Neural Information Processing Systems*, 13(23). doi: 10.7551/mitpress/1120.003.0084.

Contreras, D. and Hitschfeld-Kahler, N. (2014) 'Generation of Polyhedral Delaunay Meshes', *23rd International Meshing Roundtable (IMR23)*, 82, pp. 291–300. doi: 10.1016/j.proeng.2014.10.391

Cropp C. (2021) 'The Most Popular Types of Point Cloud Processing Software', *Vercator Blog*. Available at: https://info.vercator.com/blog/popular-point-cloud-processing-software (Accessed: 14 July 2022).

Daniels, J., Tilo Ochotta, I.·, Ha, L.K., Cláudio, · and Silva, T. (2008) 'Spline-Based Feature Curves from Point-Sampled Geometry', *Visual Computer*, 24(6), pp. 449–462. doi: 10.1007/s00371-008-0223-2

Demantké, J., Mallet, C., David, N. and Vallet, B. (2011) 'Dimensionality Based Scale Selection in 3D LIDAR Point Clouds', *ISPRS - International Archives of the Photogrammetry Remote Sensing and Spatial Information Sciences*, 38(5), pp. 97–102. doi: 10.5194/isprsarchives-XXXVIII-5-W12-97-2011.

Demarsin, K., Vanderstraeten, D., Volodine, T. and Roose, D. (2007) 'Detection of Closed Sharp Edges in Point Clouds using Normal Estimation and Graph Theory', *CAD Computer Aided Design*, 39(4), pp. 276–283. doi: 10.1016/j.cad.2006.12.005.

DeVore, R., Petrova, G., Hielsberg, M., Owens, L., Clack, B. and Sood, A. (2013) 'Processing Terrain Point Cloud Data', *Society for Industrial and Applied Mathematics (SIAM) Journal on Imaging Science*, 6(1), pp. 1–31. doi: 10.1137/110856009.

Dey, T.K. and Sun, J. (2005) ‘An Adaptive MLS Surface for Reconstruction with Guarantees’, in *Eurographics Symposium on Geometry Processing*. Austria. doi: 10.2312:SGP:SGP05:043-052

Dolapsaki, M.M. and Georgopoulos, A. (2021) ‘Edge Detection in 3D Point Clouds Using Digital Images’, *ISPRS International Journal of Geo-Information*, 10(4). doi: 10.3390/ijgi10040229.

Dony, R.D. (2001) ‘Karhunen-Loève Transform’, *The transform and data compression handbook*, 1(34), p. 29.

Du, L. (2020) *Edge Detection in 3D Point Clouds for Industrial Applications*. University of Toronto.

Dubey A. (2018) *The Mathematics Behind Principal Component Analysis, Towards Data Science*. Available at: https://towardsdatascience.com/the-mathematics-behind-principal-component-analysis-fff2d7f4b643 (Accessed: 18 February 2022).

Eder J. (1992) ‘Octree’, *ACM Transactions on Graphical*, 11(3). Available at: https://www.cg.tuwien.ac.at/studentwork/VisFoSe98/eder/octree.htm (Accessed: 18 March 2022).

El-Halawany, S.I. and Lichti, D.D. (2011) ‘Detection of Road Poles from Mobile Terrestrial Laser Scanner Point Cloud’, in *2011 International Workshop on Multi-Platform/Multi-Sensor Remote Sensing and Mapping, M2RSM 2011*. doi: 10.1109/M2RSM.2011.5697364.

El-Sayed, E., Abdel-Kader, R.F., Nashaat, H. and Marei, M. (2018) ‘Plane Detection in 3D Point Cloud using Octree-balanced Density Down-sampling and Iterative Adaptive Plane Extraction’, *IET Image Processing*, 12(9), pp. 1595–1605. doi: 10.1049/iet-ipr.2017.1076.

Erdélyi, J. *et al.* (2017) ‘Automation of point cloud processing to increase the deformation monitoring accuracy’, *Applied Geomatics*, 9(2), pp. 105–113. doi: 10.1007/s12518-017-0186-y.

Esmeide, L.N. and Nallig Eduardo, L.N. (2006) ‘Point Cloud Denoising Using Robust Principal Component Analysis.’, in *Proceedings of the First International Conference on Computer Graphics Theory and Applications*. Setúbal, Portugal, pp. 51–58. doi: 10.5220/0001358900510058

Fabio R. (2004) ‘From Point Cloud to Surface: The Modeling and Visualization Problem’, in *ISPRS WG V/6 Workshop Visualization and Animation of Reality-based 3D Models*. doi: 10.3929/ethz-a-004655782.

Fan, H., Yao, W. and Tang, L. (2014) ‘Identifying Man-made Objects Along Urban Road Corridors from Mobile Lidar Data’, *IEEE Geoscience and Remote Sensing Letters*, 11(5), pp. 950–954. doi: 10.1109/LGRS.2013.2283090.

Farin, G., Hoschek, J. and Kim, M.-S. (2002) *Handbook of Computer Aided Geometric Design*. 1st edn. Amsterdam: Elsevier Science.

Feilzer, M.Y. (2010) ‘Doing Mixed Methods Research Pragmatically: Implications for the Rediscovery of Pragmatism as a Research Paradigm’, *Journal of Mixed Methods Research*, 4(1), pp. 6–16. doi: 10.1177/1558689809349691.

Feng, C., Taguchi, Y. and Kamat, V.R. (2014) ‘Fast Plane Extraction in Organized Point Clouds Using Agglomerative Hierarchical Clustering’, in *IEEE International Conference on Robotics and Automation (ICRA)*. Hong Kong, China: IEEE, pp. 6218–6225. doi: 10.13140/2.1.2125.1204

Fleishman, S., Cohen-Or, D. and Silva, Cláudio T (2005) ‘Robust Moving Least-squares Fitting with Sharp Features’, *ACM Transactions on Graphics*, 24(3), pp. 544–552. doi: 10.1145/1073204.1073227.

Fleishman, S., Drori, I. and Cohen-Or, D. (2003) ‘Bilateral Mesh Denoising’, *ACM SIGGRAPH*, 22(3), pp. 950–953. doi: 10.1145/1201775.882368

Focus (2016) *FARO® Knowledge Base*. Available at: https://knowledge.faro.com/Hardware/3D_Scanners/Focus (Accessed: 10 January 2022).

Fröhlich, C. and Mettenleiter, M. (2004) ‘Terrestrial Laser Scanning - New Perspective in 3D Surveying’, *International archives of photogrammetry, remote sensing and spatial information sciences*, 36(8), pp. 7–13.

Fua, P. and Sander, P. (1992) ‘Reconstructing Surfaces from Unstructured 3D Points’, in *Second European Conference on Computer Vision (ECCV’90)*.

Galantucci, L.M. and Percocol, G. (2005) ‘A Multilevel Approach to Edge Detection in Tessellated Point Clouds’, *CIRP Annals*, 54(1), pp. 127–130. doi: 10.1016/S0007-8506(07)60065-1.

Gao, Rui, Mengyu Li, Seung-Jun Y., and Kyungeun C. (2022) ‘Reflective Noise Filtering of Large-Scale Point Cloud Using Transformer’, *Remote Sensing*, 14(3) 577. doi: 10.3390/rs14030577.

Gargoum, S. and El-Basyouny, K. (2019) ‘Effects of LiDAR Point Density on Extraction of Traffic Signs: A Sensitivity Study’, *Transportation Research Record*, 2673(1), pp. 41–51. doi: 10.1177/0361198118822295.

Ge, L. and Feng, J. (2021) ‘Type-based Outlier Removal Framework for Point Clouds’, *Information Sciences*, 580, pp. 436–459. doi: 10.1016/j.ins.2021.08.090.

Gie Yong, A. and Pearce, S. (2013) ‘A Beginner’s Guide to Factor Analysis: Focusing on Exploratory Factor Analysis’, *Tutorials in Quantitative Methods for Psychology*, 9(2), pp. 79–94. doi: 10.20982/TQMP.09.2.P079

Gigli, G. and Casagli, N. (2011) ‘Semi-automatic extraction of rock mass structural data from high resolution LIDAR point clouds’, *International Journal of Rock Mechanics and Mining Sciences*, 48(2), pp. 187–198. doi: 10.1016/j.ijrmms.2010.11.009.

Gilani, S.A.N., Awrangjeb, M. and Lu, G. (2018) ‘Segmentation of Airborne Point Cloud Data for Automatic Building Roof Extraction’, *GIScience and Remote Sensing*, 55(1), pp. 63–89. doi: 10.1080/15481603.2017.1361509.

Golovinskiy, A., Kim, V.G. and Funkhouser, T. (2009) ‘Shape-based Recognition of 3D Point Clouds in Urban Environments’, in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2154–2161. doi: 10.1109/ICCV.2009.5459471.

Golub, G.H., Hoffman, A. and Stewart, G.W. (1987) ‘A Generalization of the Eckart-Young-Mirsky Matrix Approximation Theorem’, *Linear Algebra and its Applications*, 88, pp. 317–327.

Golub, G.H. and van Loan, C. (1983) *Matrix Computations*. 3rd edn. London: The John Hopkins Press.

Govorcin, M., Pribicevic, B. and Đapo, A. (2014) ‘Comparison and Analysis of Software Solutions for Creation of a Digital Terrain Model Using Unmanned Aerial Vehicles’, in *14th International Multidisciplinary Scientific GeoConference SGEM 2014*. doi: 10.13140/2.1.2352.4803.

Graham, B., Engelcke, M. and van der Maaten, L. (2018) ‘3D Semantic Segmentation with Submanifold Sparse Convolutional Networks’, in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, pp. 9224–9232. doi: 10.1109/CVPR.2018.00961.

Graham L. (2021) *Point Cloud Noise, GeoCue Group*. Available at: https://geocue.com/resources/article/point-cloud-noise/ (Accessed: 28 February 2022).

Gregorius B (2019) *LiDAR Intensity: What is it and What are it’s applications? Geodetics*. Available at: https://geodetics.com/lidar-intensity-applications/ (Accessed: 11 June 2022).

Grilli, E., Farella, E.M., Torresani, A. and Remondino, F. (2019) ‘Geometric Features Analysis for the Classification of Cultural Heritage Point Clouds’, in *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*. International Society for Photogrammetry and Remote Sensing, pp. 541–548. doi: 10.5194/isprs-archives-XLII-2-W15-541-2019.

Guan, H., Yu, Y., Li, J. and Liu, P. (2016) ‘Pole-Like Road Object Detection in Mobile LiDAR Data via Supervoxel and Bag-of-Contextual-Visual-Words Representation’, *IEEE Geoscience and Remote Sensing Letters*, 13(4), pp. 520–524. doi: 10.1109/LGRS.2016.2521684.

Guba, E.G. and Lincoln, Y. (1994) ‘Competing Paradigms in Qualitative Research’, in *Major paradigms and perspectives*, pp. 105–117.

Guislain, M. *et al.* (2016) ‘Detecting and Correcting Shadows in Urban Point Clouds and Image Collections’, in *2016 Fourth International Conference on 3D Vision (3DV)*. Stanford, USA: IEEE, pp. 537–545. doi: 10.1109/3DV.2016.63.

Gumhold, S., Wang Ý, X. and MacLeod Þ, R. (2001) ‘Feature Extraction from Point Clouds’, *In Proceedings of the 10th International Meshing Roundtable*, pp. 293–305.

Guo, Y., Wang, H., Hu, Q., Liu, H., Liu, L. and Bennamoun, M. (2021) ‘Deep Learning for 3D Point Clouds: A Survey’, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(12), pp. 4338–4364. doi: 10.1109/TPAMI.2020.3005434.

Gupta, A., Byrne, J., Moloney, D., Watson, S. and Yin, H. (2018) ‘Automatic Tree Annotation in LiDAR Data’, in *Proceedings of the 4th International Conference on Geographical Information Systems Theory, Applications and Management - GISTAM*. Scitepress, pp. 36–41. doi: 10.5220/0006668000360041.

Hackel, T., Wegner, J.D., Schindler, K., Hackel, T. and Wegner, J.D. (2016) ‘Fast Semantic Segmentation of 3D Point Clouds with Strongly Varying Density’, in *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*. Prague, Czech Republic, pp. 177–184. doi: 10.3929/ethz-b-000126659.

Hamilton C. (1995) *Lunar Image Map, Views of the Solar System*. Available at: https://solarviews.com/cap/moon/moonmap.htm (Accessed: 16 July 2022).

Hamilton N. (2008) *The A-Z of Programming Languages: C#, ComputerWorld*. Available at : https://www2.computerworld.com.au/article/261958/a-z_programming_languages_c_/ (Accessed: 26 June 2022).

Han, X.F., Jin, J.S., Wang, M.J., Jiang, W., Gao, L. and Xiao, L. (2017) ‘A Review of Algorithms for Filtering the 3D Point Cloud’, *Signal Processing: Image Communication*, 57, pp. 103–112. doi: 10.1016/j.image.2017.05.009.

Hautamäki, V., Kärkkäinen, I. and Fränti, P. (2004) 'Outlier Detection Using k-Nearest Neighbour Graph', in *Proceedings - International Conference on Pattern Recognition*, pp. 430–433. doi: 10.1109/ICPR.2004.1334558.

Hawkins M D (1980) *Identification of Outliers*. 1st edn. London: Chapman & Hall. doi: 10.1007/978-94-015-3994-4

He, Z., Xu, X. and Deng, S. (2003) 'Discovering Cluster-based Local Outliers', *Pattern Recognition Letters*, 24(9–10), pp. 1641–1650. doi: 10.1016/S0167-8655(03)00003-5.

Hejlsberg Anders, Torgersen Mads, Wiltamuth Scott and Golde Peter. (2011) *The C# Programming Language*. 4th edn. Boston: Pearson Education Inc.

Hernandez, M., Choi, J. and Medioni, G. (2015) 'Near Laser-scan Quality 3-D Face Reconstruction from a Low-quality Depth Stream', *Image and Vision Computing*, 36, pp. 61–69. doi: 10.1016/j.imavis.2014.12.004.

Higgins S. (2021) *Everything you need to know about point clouds, NavVis Blog*. Available at: https://www.navvis.com/blog/everything-you-need-to-know-about-point-clouds-navvis (Accessed: 15 May 2022).

Hildebrandt, K. and Polthier, K. (2004) 'Anisotropic Filtering of Non-linear Surface Features', *Computer Graphics Forum*, pp. 391–400.

Hodge, V.J. and Austin, J. (2004) 'A Survey of Outlier Detection Methodologies', *Artificial Intelligence Review*, 22(2), pp. 85–126. doi: 10.1023/B:AIRE.0000045502.10941.a9.

Hornung, A., Wurm, K.M., Bennewitz, M., Stachniss, C. and Burgard, W. (2013) 'OctoMap: An Efficient Probabilistic 3D Mapping Framework Based on Octrees', *Autonomous Robots*, 34(3), pp. 189–206. doi: 1007/s10514-012-9321-0.

Hotelling, H. (1933) 'Analysis of a Complex of Statistical Variables into Principal Components', *Journal of Educational Psychology*, 24(6), pp. 417–441. doi: 10.1037/H0071325.

Hu, Y., Yan, Z., Yin, Z. and Du, Z. (2020) ‘Collision Detection Based on Octree for Virtual Surgery System’, in *IOP Conference Series: Materials Science and Engineering*. Institute of Physics Publishing. doi: 10.1088/1757-899X/768/7/072107.

Huang, H., Wu, S., Gong, M., Cohen-Or, D., Ascher, U. and Zhang, H. (2013) ‘Edge-aware Point Set Resampling’, *ACM Transactions on Graphics*, 32(1), pp. 1–12. doi: 10.1145/2421636.2421645.

Huang, J. and You, S. (2015) ‘Pole-like Object Detection and Classification from Urban Point Clouds’, in *Proceedings - IEEE International Conference on Robotics and Automation*. Institute of Electrical and Electronics Engineers Inc., pp. 3032–3038. doi: 10.1109/ICRA.2015.7139615.

Hui, Z., Jin, S., Cheng, P., Ziggah, Y.Y., Wang, L., Wang, Y., Hu, H., Hu, Y. (2019) ‘An Active Learning Method for DEM Extraction from Airborne LiDAR Point Clouds’, *IEEE Access*, 7, pp. 89366–89378. doi: 10.1109/ACCESS.2019.2926497.

Hůlková, M., Pavelka, K. and Matoušková, E. (2018) ‘Automatic Classification of Point Clouds for Highway Documentation’, *Acta Polytechnica*, 58(3), pp. 165–170. doi: 10.14311/AP.2018.58.0165.

Ibrahim, S. and Lichti, D. (2012) ‘Curb-based Street Floor Extraction from Mobile Terrestrial Lidar Point Cloud’, *ISPRS - International Archives of the Photogrammetry Remote Sensing and Spatial Information Sciences*, XXXIX-B5, pp. 193–198. doi: 10.5194/isprsarchives-XXXIX-B5-193-2012.

Ioannou, Y., Taati, B., Harrap, R. and Greenspan, M. (2012) ‘Difference of Normals as a Multi-scale Operator in Unorganized Point Clouds’, in *Proceedings - 2nd Joint 3DIM/3DPVT Conference: 3D Imaging, Modeling, Processing, Visualization and Transmission, 3DIMPVT 2012*. IEEE Computer Society, pp. 501–508. doi: 10.1109/3DIMPVT.2012.12.

Jaadi Z. (2022) ‘A Step-by-Step Explanation of Principal Component Analysis (PCA)’, *Built In*. Available at: https://builtin.com/data-science/step-step-explanation-principal-component-analysis (Accessed: 19 January 2022).

Jain, A.K., Murty, M.N. and Flynn, P.J. (2000) 'Data Clustering: A Review', *ACM computing surveys (CSUR)*, 31(3), pp. 264–323. doi: 10.1145/331499.331504

Jaritz, M., Gu, J. and Su, H. (2019) 'Multi-view Pointnet for 3D Scene Understanding', in *Proceedings - 2019 International Conference on Computer Vision Workshop, ICCVW 2019*. Institute of Electrical and Electronics Engineers Inc., pp. 3995–4003. doi: 10.1109/ICCVW.2019.00494.

Javed, M., Meraz, M. and Chakraborty, P. (2020) 'A Quick Review on Recent Trends in 3D Point Cloud Data Compression Techniques and the Challenges of Direct Processing in 3D Compressed Domain', *ArXiv*, abs/2007.

Jenke, P., Wand, M., Bokeloh, M., Schilling, A. and Straßer, W. (2006) 'Bayesian Point Cloud Reconstruction', *Computer Graphics Forum*, 25(3), pp. 379–388. doi: 10.1111/j.1467-8659.2006.00957.x

Jevtic G. (2019) *What is SDLC? Phases of Software Development & Models, PhoenixNAP Blog*. Available at: https://phoenixnap.com/blog/software-development-life-cycle (Accessed: 25 June 2022).

Jia, C.C., Wang, C.J., Yang, T., Fan, B.H. and He, F.G. (2018) 'A 3D Point Cloud Filtering Algorithm based on Surface Variation Factor Classification', in *Procedia Computer Science*. Elsevier B.V., pp. 54–61. doi: 10.1016/j.procs.2019.06.010.

Jiang, L., Zhao, H., Liu, S., Shen, X., Fu, C.W. and Jia, J. (2019) 'Hierarchical Point-Edge Interaction Network for Point Cloud Semantic Segmentation', in *Proceedings of the IEEE International Conference on Computer Vision*. Institute of Electrical and Electronics Engineers Inc., pp. 10432–10440. doi: 10.1109/ICCV.2019.01053.

Johnson, T., Kwok, I. and Ng, R. (1998) 'Fast Computation of 2-Dimensional Depth Contours', *KDD*, pp. 244–228.

Jolliffe I.T (2002) *Principal Component Analysis*. 2nd edn. New York: Springer-Verlag.

Jones J. and Waddel S. (2019) *The Cascading Costs of Waterfall, Medium Operating Company*. Available at: https://medium.com/@joneswaddell/the-cascading-costs-of-waterfall-5c3b1b8beaec (Accessed: 19 June 2022).

Jones, T.R., Durand, F. and Desbrun, M. (2003) 'Non-Iterative, Feature-Preserving Mesh Smoothing', *ACM Transactions on Graphics*, 22(3), pp. 943–949. doi: 10.1145/882262.882367.

Kabacoff Robert (2019) 'Chapter 14. Principal components and factor analysis', in *R in Action*. 3rd edn. Shelter Island: Manning Publications.

Kadam, K.D. (2014) 'Face Recognition using Principal Component Analysis with DCT', *International Journal of Engineering Research and General Science*, 2(4), pp 276 -280.

Kalogerakis, E., Nowrouzezahrai, D., Simari, P. and Singh, K. (2009) 'Extracting Lines of Curvature from Noisy Point Clouds', *CAD Computer Aided Design*, 41(4), pp. 282–292. doi: 10.1016/j.cad.2008.12.004.

Kang, Z., Yang, J., Zhong, R., Wu, Y., Shi, Z. and Lindenbergh, R. (2018) 'Voxel-Based Extraction and Classification of 3-D Pole-Like Objects from Mobile LiDAR Point Cloud Data', *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 11(11), pp. 4287–4298. doi: 10.1109/JSTARS.2018.2869801.

Kaula W, Schubert G, Lingenfelter R, Sjogren W and Wollenhaupt W. (1973) 'Lunar Topography from Apollo 15 and 16 Laser Altimetry', *Proceedings of the Lunar Science Conference*, 3, pp. 2811–2819.

Knorr, E.M. and Ng, R.T. (1997) 'A Unified Notion of Outliers: Properties and Computation', *KDD*, 97, pp. 219–222.

Knorr, E.M., Ng, R.T. and Tucakov, V. (2000) 'Distance-based Outliers: Algorithms and Applications', *The VLDB Journal*, 8, pp. 237–253.

Kovacs J. (2007) *C#/.NET History Lesson, Weblog*. Available at: http://jameskovacs.com/2007/09/07/cnet-history-lesson/ (Accessed: 26 June 2022).

Kriegel Hans Peter, Kröger Peer, Schubert Erich and Zimek Arthur. (2009) 'LoOP: Local Outlier Probabilities', in *Proceedings of the 18th ACM conference on Information and knowledge management*. Association for Computing Machinery, pp. 1649–1652. doi: 10.1145/1645953.1646195

Kriegel, H.-P., Kröger Peer and Zimek, A. (2010) ‘Outlier Detection Techniques’, in *The 2010 SIAM International Conference on Data Mining*, pp. 1–76.

Ku, T., Veltkamp, R.C., Boom, B., Duque-Arias, D., Velasco-Forero, S., Deschaud, J.E., Goulette, F., *et al.* (2020) ‘SHREC 2020: 3D Point Cloud Semantic Segmentation for Street Scenes’, *Computers and Graphics (Pergamon)*, 93, pp. 13–24. doi: 10.1016/j.cag.2020.09.006.

Kumar, B., Pandey, G., Lohani, B. and Misra, S.C. (2019) ‘A Multi-faceted CNN Architecture for Automatic Classification of Mobile LiDAR Data and an Algorithm to Reproduce Point Cloud Samples for Enhanced Training’, *ISPRS Journal of Photogrammetry and Remote Sensing*, 147, pp. 80–89. doi: 10.1016/j.isprsjprs.2018.11.006.

Laine, S. and Karras, T. (2010) ‘Efficient Sparse Voxel Octrees’, in *ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games (I3D)*.

Lalonde, J.-F., Vandapel, N. and Hebert, M. (2006) *Automatic Three-Dimensional Point Cloud Processing for Forest Inventory*. The Robotics Institute, Carnegie Mellon University.

Lam, J., Kusevic, K., Mrstik, P., Harrap, R. and Greenspan, M. (2010) ‘Urban Scene Extraction from Mobile Ground Based LiDAR Data’, in *International Symposium on 3D Data Processing Visualization and Transmission*. IEEE, pp. 1–8.

Lan, Z., Yew, Z.J. and Lee, G.H. (2019) ‘Robust Point Cloud Based Reconstruction of Large-scale Outdoor Scenes’, in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, pp. 9682–9690. doi: 10.1109/CVPR.2019.00992.

Landa, J. and Ondroušek, V. (2016) ‘Detection of Pole-like Objects from LIDAR Data’, *Procedia - Social and Behavioral Sciences*, 220, pp. 226–235. doi: 10.1016/j.sbspro.2016.05.494.

Landa, J., Prochazka, D. and Štastny, J. (2013) 'Point Cloud Processing for Smart Systems', *Acta Universitatis Agriculturae et Silviculturae Mendelianae Brunensis*, 61(7), pp. 2415–2421. doi: 10.11118/actaun201361072415.

Landrieu, L. and Simonovsky, M. (2018) 'Large-Scale Point Cloud Semantic Segmentation with Superpoint Graphs', in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, pp. 4558–4567. doi: 10.1109/CVPR.2018.00479.

Lange, C., Polthier, K. and Berlin, Tu (2005) 'Anisotropic Smoothing of Point Sets', *Computer Aided Geometric Design*, 22(7), pp. 680–692. doi: 10.1016/j.cagd.2005.06.010

Lee, K.-W. and Wang, W.-P. (2005) 'Feature-Preserving Mesh Denoising via Bilateral Normal Filtering', in *Ninth International Conference on Computer Aided Design and Computer Graphics*, pp. 6–11. doi: 10.1016/j.patrec.2006.04.016

Lee, Y.S., Koo, H.S. and Jeong, C.S. (2006) 'A Straight Line Detection using Principal Component Analysis', *Pattern Recognition Letters*, 27(14), pp. 1744–1754. doi: 10.1016/j.patrec.2006.04.016.

Lefebvre, S., Hornus, S. and Neyret Fabrice (2005) 'Octree Textures on the GPU', in *GPU gems*, pp. 595–613.

Lehtomäki, M., Jaakkola, A., Hyyppä, J., Kukko, A. and Kaartinen, H. (2010) 'Detection of Vertical Pole-like Objects in a Road Environment using Vehicle-based Laser Scanning Data', *Remote Sensing*, 2(3), pp. 641–664. doi: 10.3390/rs2030641.

Lehtomäki, M., Jaakkola, A., Hyyppä, J., Kukko, A. and Kaartinen, H. (2012) 'Performance Analysis of a Pole and Tree Trunk Detection Method for Mobile Laser Scanning Data', *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XXXVIII-5/W12, pp. 197–202. doi: 10.5194/isprsarchives-xxxviii-5-w12-197-2011.

Leica RTC360 3D Laser Scanner (2018) *Leica Geosystems*. Available at: https://leica-geosystems.com/products/laser-scanners/scanners/leica-rtc360 (Accessed: 10 January 2022).

Levente, L.B. and Editors, T. (2015) *Handling Uncertainty and Networked Structure in Robot Control*. 1st edn. Springer Cham. doi: 10.1007/978-3-319-26327-4.

Levin, D. (1998) 'The Approximation Power of Moving Least-Squares', *MATHEMATICS OF COMPUTATION*, 67(224), pp. 1517–1531. doi: 10.1090/S0025-5718-98-00974-0

Li, G. (2014) *Automatic Detection of Temporary Objects in Mobile LiDar Point Clouds*. University of Twente.

Li, J., Chen, B.M. and Lee, G.H. (2018) 'SO-Net: Self-Organizing Network for Point Cloud Analysis', in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, pp. 9397–9406. doi: 10.1109/CVPR.2018.00979.

Li, J. and Cheng, X. (2022) 'Supervoxel-based Extraction and Classification of Pole-like Objects from MLS Point Cloud Data', *Optics and Laser Technology*, 146. doi: 10.1016/j.optlastec.2021.107562.

Li, L., Li, D., Zhu, H. and Li, Y. (2016) 'A Dual Growing Method for the Automatic Extraction of Individual Trees from Mobile Laser Scanning Data', *ISPRS Journal of Photogrammetry and Remote Sensing*, 120, pp. 37–52. doi: 10.1016/j.isprsjprs.2016.07.009.

Li, L., Li, Y. and Li, D. (2016) 'A Method Based on an Adaptive Radius Cylinder Model for Detecting Pole-like Objects in Mobile Laser Scanning Data', *Remote Sensing Letters*, 7(3), pp. 249–258. doi: 10.1080/2150704X.2015.1126377.

Li, M. and Sun, C. (2018) 'Refinement of LiDAR Point Clouds using a Super Voxel Based Approach', *ISPRS Journal of Photogrammetry and Remote Sensing*, 143, pp. 213–221. doi: 10.1016/j.isprsjprs.2018.03.010.

Li, Y., Wang, W., Li, X., Xie, L., Wang, Y., Guo, R., Xiu, W. and Tang S. (2019) 'Pole-like Street Furniture Segmentation and Classification in Mobile LiDAR Data by Integrating Multiple Shape-descriptor Constraints', *Remote Sensing*, 11(24). doi: 10.3390/rs11242920.

Li, Y. and Wei, L. (2021) ‘An Outlier Removal Method from UAV Point Cloud Data for Transmission Lines’, in *2021 Computing, Communications and IoT Applications (ComComAp)*. IEEE, pp. 238–241. doi: 10.1109/ComComAp53641.2021.9652941.

Lin Chien-Chou, Yen-Chou T., Jhong-Jin L. and Yong-Sheng C. (2017) ‘A Novel Point Cloud Registration using 2D Image Features’, *EURASIP Journal on Advances in Signal* Processing, 5, pp 1- 11. doi: 10.1186/s13634-016-0435-y

Lin, Y., Wang, C., Cheng, J., Chen, B., Jia, F., Chen, Z. and Li, J. (2015) ‘Line Segment Extraction for Large Scale Unorganized Point Clouds’, *ISPRS Journal of Photogrammetry and Remote Sensing*, 102, pp. 172–183. doi: 10.1016/j.isprsjprs.2014.12.027.

Liu, J., Skidmore, A.K., Jones, S., Wang, T., Heurich, M., Zhu, X. and Shi, Y. (2018) ‘Large off-nadir Scan Angle of Airborne LiDAR Can Severely Affect the Estimates of Forest Structure Metrics’, *ISPRS Journal of Photogrammetry and Remote Sensing*, 136, pp. 13–25. doi: 10.1016/j.isprsjprs.2017.12.004.

Lozes, F., Elmoataz, A. and Lezoray, O. (2014) ‘Partial Difference Operators on Weighted Graphs for Image Processing on Surfaces and Point Clouds’, *IEEE Transactions on Image Processing*, 23(9), pp. 3896–3909. doi: 10.1109/TIP.2014.2336548.

LSS (2020) *DTMSoftware.com*. Available at: https://www.dtmsoftware.com/ (Accessed: 18 June 2022).

Lu, W., Wan, G., Zhou, Y., Fu, X., Yuan, P. and Song, S. (2019) ‘DeepVCP: An End-to-End Deep Neural Network for Point Cloud Registration’, in *Proceedings of the IEEE International Conference on Computer Vision*. Institute of Electrical and Electronics Engineers Inc., pp. 12–21. doi: 10.1109/ICCV.2019.00010.

Luo, D. and Liao, L. (2010) ‘Mining Outliers from Point Cloud by Data Slice’, in *Proceedings - 2010 International Conference on Artificial Intelligence and Education, ICAIE 2010*. IEEE, pp. 663–666. doi: 10.1109/ICAIE.2010.5641031.

Ma, L., Li, Y., Li, J., Wang, C., Wang, R. and Chapman, M.A. (2018) 'Mobile Laser Scanned Point-clouds for Road Object Detection and Extraction: A Review', *Remote Sensing*, 10(10), p. 1531. doi: 10.3390/rs10101531.

Maguya, A.S., Junttila, V. and Kauranne, T. (2014) 'Algorithm for Extracting Digital Terrain Models under Forest Canopy from Airborne LiDAR Data', *Remote Sensing*, 6(7), pp. 6524–6548. doi: 10.3390/rs6076524.

Mahmood, R. (2017) *Edge Detection in Unorganized 3D Point Cloud*. Laurentian University.

Mallet, C. and David, N. (2016) 'Digital Terrain Models Derived from Airborne LiDAR Data', *Optical Remote Sensing of Land Surface: Techniques and Methods*, pp. 299–319. doi: 10.1016/B978-1-78548-102-4.50007-7.

Mansur, M.O., Noor, M., Sap, M. and Malaysia, U.T. (2005) 'Outlier Detection Technique in Data Mining: A Research Perspective', in *Postgraduate Annual Research Seminar, CMS press*, pp. 23–31. doi: 10.1007/978-3-030-05127-3_2

MATLAB MathWorks, Available at: https://uk.mathworks.com/products/matlab.html (Accessed: 12 February 2023).

Measuring Trees · The Tree Register (2022) *The Tree Register of the British Isles*. Available at: https://www.treeregister.org/more/measuring-trees/ (Accessed: 29 March 2022).

Mixing Agile and Waterfall (2021) Adobe Experience Cloud.

Monahan, A.H., Fyfe, J.C., Ambaum, M.H.P., Stephenson, D.B. and North, G.R. (2009) 'Empirical Orthogonal Functions: The Medium is the Message', *Journal of Climate*, pp. 6501–6514. doi: 10.1175/2009JCLI3062.1.

Monnier, F., Vallet, B. and Soheilian, B. (2012) 'Trees Detection from Laser Point Clouds Acquired in Dense Urban Areas by a Mobile Mapping System', *ISPRS Annals Photogrammetry Remote Sensing and Spatial Information Sciences*, 1(3), pp. 245–250. doi: 10.5194/isprsannals-I-3-245-2012.

Nguyen, T.H., Daniel, S., Gueriot, D., Sintes, C. and Caillec, J.-M. le. (2019) 'Unsupervised Automatic Building Extraction Using Active Contour Model on Unregistered Optical

Imagery and Airborne LiDAR Data', *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII(2), pp. 181–188. doi: 10.5194/isprs-archives-XLII-2-W16-181-2019.

Nicole (2021) *What are Point Clouds?, PointCab Blog*. Available at: https://pointcab-software.com/en/2021/09/01/what_are_point_clouds/ (Accessed: 29 Dec 2021).

Nievergelt, Y. (1997) 'Schmidt-Mirsky Matrix Approximation With Linearly Constrained Singular Values', *Linear algebra and its applications*, 261(1–3), pp. 207–219. doi: 10.1016/S0024-3795(96)00403-X.

Ning, X., Li, F., Tian, G. and Wang, Y. (2018) 'An Efficient Outlier Removal Method for Scattered Point Cloud Data', *PLOS ONE*, 13(8). doi: 10.1371/journal.pone.0201280

Nurunnabi, A., Belton, D. and West, G. (2012) 'Robust Segmentation in Laser Scanning 3D Noisy Point Cloud Data', in *Proceedings of the International Conference on Digital Image Computing Techniques and Applications (DICTA)*. Fremantle, WA: IEEE, pp. 1–8. doi: 10.1109/DICTA.2012.6411672.

Nurunnabi, Abdul, West, G. and Belton, D. (2015a) 'Outlier Detection and Robust Normal-curvature Estimation in Mobile Laser Scanning 3D Point Cloud Data', *Pattern Recognition*, 48(4), pp. 1404–1419. doi: 10.1016/j.patcog.2014.10.014.

Nurunnabi, A, West, G. and Belton, D. (2015b) 'Robust Methods for Feature Extraction from Mobile Laser Scanning 3D Point Clouds', in *Research Locate*, pp. 109–120.

O'Day E. (2013) *3D Laser Scanning: Different Type of Scanners, Ideate*. Available at: https://www.ideateinc.com/blog/2013/07/3d-laser-scanning-different-type-of (Accessed: 30 April 2022).

Ogala, J., Ogala, B. and Onyarin, J. (2020) 'Comparative Analysis of C, C++, C# and JAVA Programming Languages', *Global Scientific Journals*, 8(5), pp. 1899–1913.

Oreifej, O. (2013) *Robust Subspace Estimation Using Low-Rank Optimization: Theory and Applications*. University of Central Florida.

Oztireli, C., Guennebaud, G. and Gross, M. (2009) 'Feature Preserving Point Set Surfaces based on Non-Linear Kernel Regression', *Computer Graphics Forum*, 28(2), pp. 493–501. doi: 10.1111/j.1467-8659.2009.01388.x

Papadimitriou, S., Kitagawa, H., Gibbons, P.B. and Faloutsos, C. (2003) 'LOCI: Fast Outlier Detection Using the Local Correlation Integral', in *Proceedings - International Conference on Data Engineering*, pp. 315–326. doi: 10.1109/ICDE.2003.1260802.

Papon, J., Abramov, A., Schoeler, M. and Worgotter, F (2013) 'Voxel Cloud Connectivity Segmentation - Supervoxels for Point Clouds', in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 2027–2034. doi: 10.1109/CVPR.2013.264.

Paris, S. and Durand, F. (2006) 'A Fast Approximation of the Bilateral Filter Using a Signal Processing Approach', in *European conference on computer vision*. Springer, Berlin, Heidelberg, pp. 568–580.

Park S and Jun Y (2002) 'Automated Segmentation of Point Data in a Feature-based Reverse Engineering System', *Proceedings of the Institution of Mechanical Engineers Part B Journal of Engineering Manufacture*, 216(3), pp. 445–461. doi: 10.1243/0954405021519951.

Parkhan M J (2019) *Combined use of Airborne Laser Scanning and Hyperspectral Imaging for Forest Inventories*. EPFL, Switzerland. doi: 10.5075/EPFL-THESIS-9033

Pauly, M., Mitra, N.J. and Guibas, L.J. (2004) 'Uncertainty and Variability in Point Cloud Surface Data', *Eurographics Symposium on Point-Based Graphics*, pp. 77–84. doi: 10.2312/SPBG/SPBG04/077-084

Pearson, K. (1901) 'On lines and Planes of Closest Fit to Systems of Points in Space', *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11), pp. 559–572. doi: 10.1080/14786440109462720.

Pepe, M. and Prezioso, G. (2015) 'A Matlab geodetic software for processing airborne LIDAR bathymetry data', in *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*. International Society for

Photogrammetry and Remote Sensing, pp. 167–170. doi: 10.5194/isprsarchives-XL-5-W5-167-2015.

Petras, V. *et al.* (2023) 'Point Density Variations in Airborne Lidar Point Clouds', *Sensors*, 23(3), p. 1593. doi: 10.3390/s23031593.

Petrova, E., Pauwels, P., Svidt, K. and Jensen, R.L. (2019) 'Towards Data-driven Sustainable Design: Decision Support Based on Knowledge Discovery in Disparate Building Data', *Architectural Engineering and Design Management*, 15(5), pp. 334–356. doi: 10.1080/17452007.2018.1530092.

Pfeifer, N., Wien, T.U., Fan, H., Dorninger, P. and Haring, A (2007) 'Investigating Terrestrial Laser Scanning Intensity Data: Quality and Functional Relations', *Researchgate*, pp. 328–337.

Pierce, R. (2018) *Vertices, Edges and Faces, Maath is Fun*. Available at: https://www.mathsisfun.com/geometry/vertices-faces-edges.html (Accessed: 20 November 2018).

Pirotti, F., Ravanelli, R., Fissore, F. and Masiero, A. (2018) 'Implementation and Assessment of Two Density-based Outlier Detection Methods Over Large Spatial Point Clouds', *Open Geospatial Data, Software and Standards*, 3(1). doi: 10.1186/s40965-018-0056-5.

Point Clouds for Beginners: Your Questions Answered (2022) *GeoSlam*. Available at: https://geoslam.com/point-clouds/ (Accessed: 20 May 2021).

Ponciano, J.J., Trémeau, A. and Boochs, F. (2019) 'Automatic Detection of Objects in 3D Point Clouds Based on Exclusively Semantic Guided Processes', *ISPRS International Journal of Geo-Information*, 8(10), 442. doi: 10.3390/ijgi8100442.

Poux, F. and Billen, R. (2019) 'Voxel-based 3D Point Cloud Semantic Segmentation: Unsupervised Geometric and Relationship Featuring vs Deep Learning Methods', *ISPRS International Journal of Geo-Information*, 8(5), 213. doi: 10.3390/ijgi8050213.

Poux F. (2020) *Fundamentals to Clustering High-Dimensional Data: 3D Point Clouds, Towards Data Science*. Available at: https://towardsdatascience.com/fundamentals-to-

clustering-high-dimensional-data-3d-point-clouds-3196ee56f5da (Accessed: 13 February 2022).

Pratt, V. (1987) 'Direct Least-Squares Fitting of Algebraic Surfaces', *Computer Graphics*, 21(4), pp. 145–152.

Pu, S., Rutzinger, M., Vosselman, G. and Oude Elberink, S. (2011) 'Recognizing Basic Structures from Mobile Laser Scanning Data for Road Inventory Studies', *ISPRS Journal of Photogrammetry and Remote Sensing*, 66(6 SUPPL.). doi: 10.1016/j.isprsjprs.2011.08.006.

Qi, C.R., Su, H., Mo, K. and Guibas, L.J. (2017) 'PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation', in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*. Institute of Electrical and Electronics Engineers Inc., pp. 77–85. doi: 10.1109/CVPR.2017.16.

Rakotosaona, M.-J., la Barbera, V., Guerrero, P., Mitra, N.J. and Ovsjanikov, M. (2020) 'PointCleanNet: Learning to Denoise and Remove Outliers from Dense Point Clouds', *Computer Graphics Forum*, 39(1), pp. 185–203. doi: 10.1111/cgf.13753.

Rastiveis, H., Shams, A., Sarasua, W.A. and Li, J. (2020) 'Automated Extraction of Lane Markings from Mobile LiDAR Point Clouds Based on Fuzzy Inference', *ISPRS Journal of Photogrammetry and Remote Sensing*, 160, pp. 149–166. doi: 10.1016/j.isprsjprs.2019.12.009.

Remondino, F. (2004) 'International Archives of the Photogrammetry', *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XXXIV-5/W10. doi: 10.3929/ethz-a-004655782.

Rodríguez, A.S., Rodríguez, B.R., Rodríguez, M.S. and Sánchez, P.A. (2018) 'Laser Scanning and its Applications to Damage Detection and Monitoring in Masonry Structures', in *Long-term Performance and Durability of Masonry Structures: Degradation Mechanisms, Health Monitoring and Service Life Design*. Woodhead Publishing, pp. 265–285. doi: 10.1016/B978-0-08-102110-1.00009-1.

Rodríguez-Cuenca, B., García-Cortés, S., Ordóñez, C. and Alonso, M.C. (2015) 'Automatic Detection and Classification of Pole-like Objects in Urban Point Cloud Data using an Anomaly Detection Algorithm', *Remote Sensing*, 7(10), pp. 12680–12703. doi: 10.3390/rs71012680.

Rooms F. (2019) *Point Clouds from the Clouds*, *Bricsys CAD Blog*. Available at: https://blog.bricsys.com/point-cloud-lidar-airborne-mapping/ (Accessed: 7 June 2022).

Rousell, A. (2014) 'Influence of point cloud density on the results of automated Object-Based building extraction from ALS data', in *AGILE conference Castellon, Spain*. Castellón.

Rousseeuw Peter and Leroy Annick (1987) *Robust Regression and Outlier Detection*. Third. Toronto, Canada: John Wiley and Sons Inc. doi: 10.1002/0471725382.

Rousseeuw, P.J. and Hubert, M. (2011) 'Robust Statistics for Outlier Detection', *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(1), pp. 73–79. doi: 10.1002/widm.2.

Rousseeuw, P.J. and Hubert, M. (2018) 'Anomaly Detection by Robust Statistics', *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(2). doi: 10.1002/widm.1236.

Roynard, X., Deschaud, J.-E. and Goulette, F. (2018) 'Classification of Point Cloud Scenes with Multiscale Voxel Deep Network', *arXiv:1804.03583*. doi: 10.48550/arXiv.1804.03583

Ruchay, A. N., Dorofeev, K.A. and Kalschikov, V. v. (2019) 'Accuracy Analysis of 3D Object Reconstruction using Point Cloud Filtering Algorithms', in *Proceedings of the 5th Information Technology and Nanotechnology (ITNT-2019)*, pp. 169–174. doi: 10.18287/1613-0073-2019-2391-169-174.

Ruparelia, N.B. (2010) 'Software Development Lifecycle Models', *ACM SIGSOFT Software Engineering Notes*, 35(3), pp. 8–13. doi: 10.1145/1764810.1764814.

Rusu, R.B., Marton, Z.C., Blodow, N., Dolha, M. and Beetz, M. (2008) 'Towards 3D Point Cloud Based Object Maps for Household Environments', *Robotics and Autonomous Systems*, 56(11), pp. 927–941. doi: 10.1016/j.robot.2008.08.005.

Rusu, R.B. and Cousins, S. (2011) '3D is Here: Point Cloud Library (PCL)', in *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, pp. 1-4, doi: 10.1109/ICRA.2011.5980567.

Ruwen Schnabel, D.-I., Klein, R. and Gumhold, S. (2010) *Efficient Point-Cloud Processing with Primitive Shapes*, University Bonn.

Safaie, A.H., Rastiveis, H., Shams, A., Sarasua, W.A. and Li, J. (2021) 'Automated Street Tree Inventory using Mobile LiDAR Point Clouds based on Hough Transform and Active Contours', *ISPRS Journal of Photogrammetry and Remote Sensing*, 174, pp. 19–34. doi: 10.1016/j.isprsjprs.2021.01.026.

Sahin, C., Alkis, A., Ergun, B., Kulur, S., Batuk, F. and Kilic, A. (2012) 'Producing 3D City Model with the Combined Photogrammetric and Laser Scanner Data in the Example of Taksim Cumhuriyet Square', *Optics and Lasers in Engineering*, 50(12), pp. 1844–1853. doi: 10.1016/j.optlaseng.2012.05.019.

Salman, N., Yvinec, M., Merigot, Q. (2010) 'Feature Preserving Mesh Generation from 3D Point Clouds', *Computer Graphics Forum*, 29(5), pp. 1623–1632. Oxford, UK: Blackwell Publishing Ltd.

Sampaio, J.H.B. (2006) 'An Iterative Procedure for Perpendicular Offsets Linear Least Squares Fitting with Extension to Multiple Linear Regression', *Applied Mathematics and Computation*, 176(1), pp. 91–98. doi: 10.1016/j.amc.2005.09.054.

Sankaranarayanan, J., Samet, H. and Varshney, A. (2007) 'A Fast all Nearest Neighbor Algorithm for Applications Involving Large Point-Clouds', *Computers and Graphics (Pergamon)*, 31(2), pp. 157–174. doi: 10.1016/j.cag.2006.11.011.

Schall, O., Belyaev, A. and Seidel, H.-P. (2005) 'Robust Filtering of Noisy Scattered Point Data', in *Proceedings Eurographics/IEEE VGTC Symposium Point-Based Graphics*, pp. 71–144. doi: 10.1109/PBG.2005.194067

Schall, O., Belyaev, A. and Seidel, H.P. (2008) 'Adaptive Feature-preserving Non-local Denoising of Static and Time-varying Range Data', *CAD Computer Aided Design*, 40(6), pp. 701–707. doi: 10.1016/j.cad.2008.01.011.

Schauer, J. and Nüchter, A. (2018) ‘Removing Non-static Objects from 3D Laser Scan Data’, *ISPRS Journal of Photogrammetry and Remote Sensing*, 143, pp. 15–38. doi: 10.1016/j.isprsjprs.2018.05.019.

Schneiders, R., Schindler, R. and Weiler, F. (1996) ‘Octree-based Generation of Hexahedral Element Meshes’, in *Proceedings of the 5th International Meshing Roundtable.* doi: 10.1142/S021819590000022X

Scheiner, N. *et al.* (2021) ‘Object detection for automotive radar point clouds – a comparison’, *AI Perspectives*, 3(1). doi: 10.1186/s42467-021-00012-z.

Scholkopf, B., Smola, A. and Muller Klaus (1997) ‘Kernel Principal Component Analysis’, in Gerstner W., Germond A., Hasler M. and Nicoud JD. (eds) *International conference on artificial neural networks*. Berlin Heidelberg: Springer, pp. 583–588. doi: 10.1007/BFb0020217.

Schön, B., Mosa, A.S.M., Laefer, D.F. and Bertolotto, M. (2013) ‘Octree-based Indexing for 3D Pointclouds within an Oracle Spatial DBMS’, *Computers and Geosciences*, 51, pp. 430–438. doi: 10.1016/j.cageo.2012.08.021.

Schwaber K. and Sutherland J. (2020) *What is Scrum? The Scrum Guide*. Available at: https://www.scrum.org/resources/what-is-scrum (Accessed: 19 June 2022).

Sengupta, A.M. and Mitra, P.P. (1997) ‘Distributions of Singular Values for Some Random Matrices’, *Physical review. E, Statistical physics, plasmas, fluids, and related interdisciplinary topics*, 60, p. 3389. doi: 10.1103/PhysRevE.60.3389.

Senior M. (2021) *The Future of Point Cloud Processing and 3D Models*, *Geo Insight - GEO BUSINESS*. Available at: https://www.geobusinessshow.com/the-future-of-point-cloud-processing-and-3d-models/ (Accessed: 7 June 2022).

Serifoglu Yilmaz, C., Yilmaz, V. and Güngör, O. (2018) ‘Investigating the Performances of Commercial and Non-commercial Software for Ground Filtering of UAV-based Point Clouds’, 39(15–16), pp. 5016–5042. doi: 10.1080/01431161.2017.1420942.

Shao, M., Ijiri, Y. and Hattori, K. (2015) ‘Grouped Outlier Removal for Robust Ellipse Fitting’, in *Proceedings of the 14th IAPR International Conference on Machine Vision*

Applications, MVA 2015. Institute of Electrical and Electronics Engineers Inc., pp. 138–141. doi: 10.1109/MVA.2015.7153152.

Shaw, P.J.A. (2003) *Multivariate statistics for the Environmental Sciences, New York*. John Wiley & Sons Inc.

Shen, J., Liu, J., Zhao, R. and Lin, X. (2011) 'A Kd-tree-based Outlier Detection Method for Airborne LiDAR Point Clouds', in *2011 International Symposium on Image and Data Fusion, ISIDF* Tengchong, China, 2011, pp. 1-4. doi: 10.1109/ISIDF.2011.6024307.

Shi, B.Q., Liang, J. and Liu, Q. (2011) 'Adaptive Simplification of Point Cloud Using k-Means Clustering', *CAD Computer Aided Design*, 43(8), pp. 910–922. doi: 10.1016/j.cad.2011.04.001.

Shi, Q. and Jaja, J. (2006) 'Isosurface Extraction and Spatial Filtering Using Persistent Octree (POT)', *IEEE Transactions on Visualization and Computer Graphics*, 12(5), pp 1283-1290. doi: 10.1109/TVCG.2006.157.

Shi, S. Wang Z., Shi J., Wang X., and Li H. (2021) 'From Points to Parts: 3D Object Detection from Point Cloud with Part-aware and Part-aggregation Network', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43, pp. 2647-2664. doi: 10.1109/TPAMI.2020.2977026.

Shi, S., Wang, X. and Li, H. (2019) 'PointRCNN: 3D Object Proposal Generation and Detection from Point Cloud', in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–779. doi: 10.48550/arXiv.1812.04244

Shi, Z., Kang, Z., Lin, Y., Liu, Y. and Chen, W. (2018) 'Automatic Recognition of Pole-like Objects from Mobile Laser Scanning Point Clouds', *Remote Sensing*, 10(12), pp. 1-23. doi: 10.3390/rs10121891.

Shirowzhan, S., Sepasgozar, S.M.E., Li, H., Trinder, J. and Tang, P. (2019) 'Comparative Analysis of Machine Learning and Point-based Algorithms for Detecting 3D Changes in Buildings Over Time Using Bi-temporal Lidar Data', *Automation in Construction*, 105, pp. 102841. doi: 10.1016/j.autcon.2019.102841.

Singh S. (2020) ‘What is Testing in Software? — The Three Main Types of Testing Explained in Simple English’, *Level Up Coding*. Available at: https://levelup.gitconnected.com/what-is-testing-in-software-the-three-main-types-of-testing-explained-in-simple-english-da0fec7ae5d6 (Accessed: 7 August 2022).

Soilán, M., Sánchez-Rodríguez, A., del Río-Barral, P., Perez-Collazo, C., Arias, P. and Riveiro, B. (2019) ‘Review of Laser Scanning Technologies and their Applications for Road and Railway Infrastructure Monitoring’, *Infrastructures*, 4(4), p. 58. doi: 10.3390/infrastructures4040058.

Sotoodeh, S. (2006) ‘Outlier Detection in Laser Scanner Point Clouds’, *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 36(5), pp. 297–305. doi: 10.3929/ETHZ-B-000037220

Sotoodeh, S. (2007) ‘Hierarchical Clustered Outlier Detection in Laser Scanner Point Clouds’, *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 36(3/W52), pp. 383–388. doi: 10.3929/ethz-b-000004210.

Stewart, G.W. (1993) ‘On early History of the Singular Value Decomposition’, *SIAM Review, Society for Industrial and Applied Mathematics*, 35(4), pp. 551–566. doi: 10.1137/1035134.

Stucker, C., Richard, A., Wegner, J.D. and Schindler Photogrammetry, K. (2018) ‘Supervised Outlier Detection in Large-scale MVS Point Clouds for 3D City Modeling Applications’, *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 4, pp. 263–273. doi: 10.3929/ethz-b-000271685.

Su, H., Jampani, V., Sun, D., Maji, S., Kalogerakis, E., Yang, M.H. and Kautz, J. (2018) ‘SPLATNet: Sparse Lattice Networks for Point Cloud Processing’, in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, pp. 2530–2539. doi: 10.1109/CVPR.2018.00268.

Su Zhonghua, Li Shihua, Liu Hanhu and Liu Yuhan. (2019) ‘Extracting Wood Point Cloud of Individual Trees Based on Geometric Features’, *IEEE Geoscience and Remote Sensing Letters*, 16(8), pp. 1294–1298. doi: 10.1109/LGRS.2019.2896613.

Sun, Y., Schaefer, S. and Wang, W. (2015) ‘Denoising Point Sets via L0 Minimization’, *Computer Aided Geometric Design*, 35–36, pp. 2–15. doi: 10.1016/j.cagd.2015.03.011.

Sunday, D. (2021) *Practical Geometry Algorithms with C++ Code*. 1st edn. Amazon KDP.

Suryanarayana, T.M.V. and Mistry, P.B. (2016) ‘Principal Component Analysis in Transfer Function’, in *SpringerBriefs in Applied Sciences and Technology*. Springer Verlag, pp. 17–25. doi: 10.1007/978-981-10-0663-0_2.

Ta, V.T., Elmoataz, A. and Lézoray, O. (2011) ‘Nonlocal PDEs-based Morphology on Weighted Graphs for Image and Data Processing’, *IEEE Transactions on Image Processing*, 20(6), pp. 1504–1516. doi: 10.1109/TIP.2010.2101610.

Tang, P., Huber, D., Akinci, B., Lipman, R. and Lytle, A. (2010) ‘Automatic Reconstruction of as-built Building Information Models from Laser-scanned Point Clouds: A Review of Related Techniques’, *Automation in Construction*, 19(7), pp. 829–843. doi: 10.1016/j.autcon.2010.06.007.

Tapken P. (2019) ‘The Rising Demand for Total Stations and Terrestrial Laser Scanners’, *GIM International*, 5 February. Available at: https://www.gim-international.com/content/article/the-rising-demand-for-total-stations-and-terrestrial-laser-scanners (Accessed: 14 July 2022).

Tashakkori, A. and Teddlie, C. (2010) ‘The Past and Future of Mixed Methods Research: From Triangulation to Mixed Model Design’, in *Handbook of Mixed Methods in Social and Behavioural Research*. 2nd edn. CA: SAGE.

Taubin, G. (1991) ‘Estimation of Planar Curves, Surfaces, and Nonplanar Space Curves Defined by Implicit Equations with Applications to Edge and Range Image Segmentation’, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(11), pp. 1115–1138. doi: 10.1109/34.103273.

Taubin, G. (1995) ‘Estimating the Tensor of Curvature of a Surface from a Polyhedral Approximation’, in *IEEE International Conference on Computer Vision*. IEEE, pp. 902–907. doi: 10.1109/iccv.1995.466840.

Tazir, M.L., Checchin, P. and Trassoudaine, L. (2016) ‘Color-based 3D Point Cloud Reduction’, in *2016 14th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pp. 1–7. doi: 10.1109/ICARCV.2016.7838685.

Teng, M., Zhuangzhi, W., Lu, F., Pei, L. and Xiang, L. (2010) ‘Point Cloud Segmentation Through Spectral Clustering’, in *2nd International Conference on Information Science and Engineering, ICISE2010, China*. IEEE, pp. 1–4. doi: 10.1109/ICISE.2010.5690596.

Teo, T.A. and Chiu, C.M. (2015) ‘Pole-Like Road Object Detection from Mobile Lidar System Using a Coarse-to-Fine Approach’, *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 8(10), pp. 4805–4818. doi: 10.1109/JSTARS.2015.2467160.

Thank the Egyptians; The History of Surveying & Mapping (2019) *DRMPerspective*. Available at: https://drmp.com/drmperspective?id=895612/thank-the-egyptians-the-history-of-surveying-mapping (Accessed: 16 July 2022).

Thomson C. (2019) *6 industries that need to understand point clouds, VEERCATOR*. Available at: https://info.vercator.com/blog/6-industries-that-need-to-understand-point-clouds-in-2019 (Accessed: 29 April 2022).

Thrun, S., Burgard, W. and Fox, D. (1998) ‘A Probabilistic Approach to Concurrent Mapping and Localization for Mobile Robots’, *Autonomous Robots*, 5(3–4), pp. 253–271. doi: 10.1023/a:1008806205438.

Tipping, Michael E. and Bishop, C.M. (1999) ‘Probabilistic Principal Component Analysis’, *Journal of the Royal Statistical Society. Series B: Statistical Methodology*, 61(3), pp. 611–622. doi: 10.1111/1467-9868.00196.

Tomasi, C. and Manduchi, R. (1998) ‘Bilateral Filtering for Gray and Color Images’, in *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271), Bombay*, India, pp. 839–846. doi: 10.1109/ICCV.1998.710815

Tombari, F., Fioraio, N., Cavallari, T., Salti, S., Petrelli, A. and di Stefano, L. (2014) ‘Automatic Detection of Pole-like Structures in 3D Urban Environments’, in *IEEE*

International Conference on Intelligent Robots and Systems. Institute of Electrical and Electronics Engineers Inc., pp. 4922–4929. doi: 10.1109/IROS.2014.6943262.

Tombari, F., Cavallari, T. and Stefano, L. di (2016) 'Poles from Point Clouds', *GIM International*, pp. 1–13.

Tran, T.H.G., Ressl, C. and Pfeifer, N. (2018) 'Integrated Change Detection and Classification in Urban Areas Based on Airborne Laser Scanning Point Clouds', *Sensors (Switzerland)*, 18(2), 448. doi: 10.3390/s18020448.

Tucker A. (2021) 'Computer Science - Programming Languages', *Encyclopedia Britannica*. Available at: https://www.britannica.com/science/computer-science/Programming-languages (Accessed: 18 June 2022).

Tuley, J., Vandapel, N. and Hebert, M. (2005) 'Analysis and Removal of Artifacts in 3-D LADAR Data', in *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 2203–2210. doi: 10.1109/ROBOT.2005.1570440.

Tuominen, S., Näsi, R., Honkavaara, E., Balazs, A., Hakala, T., Viljanen, N. and Pölönen, I., *et al.* (2018) 'Assessment of Classifiers and Remote Sensing Features of Hyperspectral Imagery and Stereo-Photogrammetric Point Clouds for Recognition of Tree Species in a Forest Area of High Species Diversity', *Remote Sensing*, 10(5). doi: 10.3390/rs10050714.

Turner, M., Moxey, D. and Peiró, J. (2015) 'Automatic Mesh Sizing Specification of Complex Three Dimensional Domains using an Octree Structure', *24th International Meshing Roundtable (IMR24)*.

Unnikrishnan, R. (2008) 'Statistical Approaches to Multi-scale Point Cloud Processing', *Wwwoldricmuedu*, The Robotics Institute Carnegie Mellon University (May), pp. 1-146.

Upadhyay Raj K. (2020) 'Advantages and Disadvantages of using Spiral Model', *GeeksforGeeks*. Available at: https://www.geeksforgeeks.org/advantages-and-disadvantages-of-using-spiral-model/ (Accessed: 19 June 2022).

Uy, M.A., Pham, Q.-H., Hua, B.-S., Nguyen, D.T. and Yeung, S.K. (2019) 'Revisiting Point Cloud Classification: A New Benchmark Dataset and Classification Model on Real-

World Data', in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 1588–1597. doi:10.1109/ICCV.2019.00167.

Vidal, R., Ma, Y. and Sastry, S.S. (2005) 'Generalized Principal Component Analysis', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(12), pp. 1945–1959. doi: 10.1007/978-981-10-2915-8_7.

Vo, A.V., Truong-Hong, L., Laefer, D.F. and Bertolotto, M. (2015) 'Octree-based Region Growing for Point Cloud Segmentation', *ISPRS Journal of Photogrammetry and Remote Sensing*, 104, pp. 88–100. doi: 10.1016/j.isprsjprs.2015.01.011.

V.S. R. (2005) 'Working with Namespaces in C#', *C# Corner*. Available at: https://www.c-sharpcorner.com/article/working-with-namespaces-in-C-Sharp/ (Accessed: 14 July 2022).

Wang, J. and Shan, J. (2009) 'Segmentation of LiDAR Point Clouds for Building Extraction', in *American Society for Photogramm. Remote Sens. Annual Conference, Baltimore*, pp. 9–13.

Wang, J., Yu, Z., Zhu, W. and Cao, J. (2013) 'Feature-preserving Surface Reconstruction from Unoriented, Noisy Point Data', *Computer Graphics Forum*, 32(1), pp. 164–176. doi: 10.1111/cgf.12006.

Wang, J., L., R. and Menenti, M. (2017) 'SigVox – A 3D Feature Matching Algorithm for Automatic Street Object Recognition in Mobile Laser Scanning Point Clouds', *ISPRS Journal of Photogrammetry and Remote Sensing*, 128, pp. 111–129. doi: 10.1016/j.isprsjprs.2017.03.012.

Wang, P., Gan, Y., Shui, P., Yu, F., Zhang, Y., Chen, S. and Sun, Z. (2018) '3D shape Segmentation via Shape Fully Convolutional Networks', *Computers and Graphics (Pergamon)*, 70, pp. 128–139. doi: 10.1016/j.cag.2017.07.030.

Wang, W., Zhang, Y., Ge, G., Jiang, Q., Wang, Y. and Hu, L. (2021) 'A Hybrid Spatial Indexing Structure of Massive Point Cloud Based on Octree and 3D R*-Tree', *Applied Sciences (Switzerland)*, 11(20), pp. 9581. doi: 10.3390/app11209581.

Wang, X., Zhou K., Yang J., and Lu Y. (2011) ‘MATLAB tools for lidar data conversion, visualization, and processing’, in *International Symposium on Lidar and Radar Mapping 2011: Technologies and Applications*. SPIE, pp. 82860M. doi: 10.1117/12.912529.

Wang, X., He, J. and Ma, L. (2019) ‘Exploiting Local and Global Structure for Point Cloud Semantic Segmentation with Contextual Point Representations’, in *NeurIPS*, pp. 1–11. doi: 10.48550/arXiv.1911.05277.

Wang, Y. and Feng, H.Y. (2015) ‘Outlier Detection for Scanned Point Clouds using Majority Voting’, *CAD Computer Aided Design*, 62, pp. 31–43. doi: 10.1016/j.cad.2014.11.004.

Wang, Y., Cheng, L., Chen, Y., Wu, Y. and Li, M. (2016) ‘Building Point Detection from Vehicle-borne LiDAR Data Based on Voxel Group and Horizontal Hollow Analysis’, *Remote Sensing*, 8(5). doi: 10.3390/rs8050419.

Wang, Y. and Solomon, J. (2019) ‘Deep Closest Point: Learning Representations for Point Cloud Registration’, in *Proceedings of the IEEE International Conference on Computer Vision*. IEEE, pp. 3522–3531. doi: 10.1109/ICCV.2019.00362.

Weber, C., Hahmann, S. and Hagen, H. (2010a) ‘Methods for Feature Detection in Point Clouds’, in *Visualization of Large and Unstructured Data Sets - Applications in Geospatial Planning, Modeling and Engineering (IRTG 1131 Workshop), VLUDS 2010*, pp. 90–99. doi: 10.4230/OASIcs.VLUDS.2010.90.

Weber, C., Hahmann, S. and Hagen, H. (2010b) ‘Sharp Feature Detection in Point Clouds’, in *SMI 2010 - International Conference on Shape Modeling and Applications, Proceedings*. IEEE Computer Society, pp. 175–186. doi: 10.1109/SMI.2010.32.

Weber, C., Hahmann, S., Hagen, H. and Bonneau, G.P. (2012) ‘Sharp Feature Preserving MLS Surface Reconstruction Based on Local Feature Line Approximations’, *Graphical Models*, 74(2). doi: 10.1016/j.gmod.2012.04.012ï.

Weinmann, M., Jutzi, B., Mallet, C. and Weinmann, M. (2017) ‘Geometric Features and Their Relevance for 3D Point Cloud Classification’, in *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*. Copernicus GmbH, pp. 157–164. doi: 10.5194/isprs-annals-IV-1-W1-157-2017.

Wen, X., Han, Z., Liu, X. and Liu, Y.S. (2019) 'Point2SpatialCapsule: Aggregating Features and Spatial Relationships of Local Regions on Point Clouds using Spatial-aware Capsules', *IEEE Transactions on Image Processing*, 29, pp. 8855–8869. doi: 10.1109/TIP.2020.3019925.

Wengefeld, T., Lewandowski, B., Seichter, D., Pfennig, L. and Gross, H.-M. (2019) 'Real-time Person Orientation Estimation using Colored Pointclouds', in *2019 European Conference on Mobile Robots (ECMR)*, pp. 1–7. doi: 10.1109/ECMR.2019.8870914.

What are Point Clouds ? (2018) *Tech 27 Stay connected with the latest in Industrial AI, Smart Engineering & IoT*. Available at: https://tech27.com/resources/point-clouds/ (Accessed: 11 June 2022).

What is a Point Cloud Survey? (2021) *SkyKam*. Available at: https://skykam.co.uk/what-is-a-point-cloud/ (Accessed: 14 July 2022).

What Is Laser Scanning and How Can It Be Used? (2020) *TopoDot blog*. Available at: https://blog.topodot.com/what-is-laser-scanning-and-how-can-it-be-used/ (Accessed: 29 May 2022).

What Is Point Cloud Processing and Why Is It Important? (2019) *TopoDot Blog*. Available at: https://blog.topodot.com/what-is-point-cloud-processing-and-why-is-it-important/ (Accessed: 7 June 2022).

What is Rapid Application Development?, *The Economic Times*. Available at: https://economictimes.indiatimes.com/definition/rapid-application-development (Accessed: 19 June 2022).

Widyaningrum, E., Gorte, B. and Lindenbergh, R. (2019) 'Automatic Building Outline Extraction from ALS Point Clouds by Ordered Points Aided Hough Transform', *Remote Sensing*, 11(14), 1727. doi: 10.3390/rs11141727.

Wilhelms Jane and Gelder Allen (2000) 'Octree for Faster Isosurface Generation', *IEEE Transactions on Medical Imaging*, 19, pp. 739–758. doi: 10.1145/130881.130882.

Williams, R.M. and Ilieş, H.T. (2018) ‘Practical Shape Analysis and Segmentation Methods for Point Cloud Models’, *Computer Aided Geometric Design*, 67, pp. 97–120. doi: 10.1016/j.cagd.2018.10.003.

Wolff, K., Kim, C., Zimmer, H., Schroers, C., Botsch, M., Sorkine-Hornung, O. and Sorkine-Hornung, A. (2016) ‘Point Cloud Noise and Outlier Removal for Image-Based 3D Reconstruction’, in *2016 Fourth International Conference on 3D Vision (3DV)*. IEEE, pp. 118–127. doi: 10.1109/3DV.2016.20.

Wood L. (2022a) *European 3D Scanner Market - Forecasts from 2022 to 2027*, *Researchandmarkets.com*. Available at: https://www.researchandmarkets.com/reports/5576399/european-3d-scanner-market-forecasts-from-2022 (Accessed: 14 July 2022).

Wood L. (2022b) *The Global 3D Scanning Market Will Grow to USD 16.66 Billion by 2030, at a CAGR of 16.3%, BusinessWire*. Available at: https://www.businesswire.com/news/home/20220509005444/en/The-Global-3D-Scanning-Market-Will-Grow-to-USD-16.66-Billion-by-2030-at-a-CAGR-of-16.3---ResearchAndMarkets.com (Accessed: 14 July 2022).

Woz U. (2020) What is Syntax in Computer Programming?, *Woz U*. Available at: https://woz-u.com/blog/what-is-syntax-in-computer-programming/ (Accessed: 29 July 2022).

Wu, B., Yu, B., Yue, W., Shu, S., Tan, W., Hu, C. and Huang, Y., *et al.* (2013) ‘A Voxel-based Method for Automated Identification and Morphological Parameters Estimation of Individual Street Trees from Mobile Laser Scanning Data’, *Remote Sensing*, 5(2), pp. 584–611. doi: 10.3390/rs5020584.

Wu, F., Wen, C., Guo, Y., Wang, J., Yu, Y., Wang, C. and Li, J. (2017) ‘Rapid Localization and Extraction of Street Light Poles in Mobile LiDAR Point Clouds: A Supervoxel-based Approach’, *IEEE Transactions on Intelligent Transportation Systems*, 18(2), pp. 292–305. doi: 10.1109/TITS.2016.2565698.

Wu Rongren, Yiping Chen, Wang Cheng and Li Jonathan. (2018) ‘Estimation of Forest Trees Diameter from Terrestrial Laser Scanning Point Clouds Based on a Circle Fitting

Method', in *IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium*. IEEE, pp. 2813–2816. doi: 10.1109/IGARSS.2018.8517303.

Wu, W., Qi, Z. and Fuxin, L. (2019) 'PointCONV: Deep Convolutional Networks on 3D Point Clouds', in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, pp. 9613–9622. doi: 10.1109/CVPR.2019.00985.

Xiang, C., Qi, C.R. and Li, B. (2019) 'Generating 3D Adversarial Point Clouds', in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, pp. 9128–9136. doi: 10.1109/CVPR.2019.00935.

Xiao, W., Vallet, B., Schindler, K. and Paparoditis, N. (2016) 'Street-side Vehicle Detection, Classification and Change Detection using Mobile Laser Scanning Data', *ISPRS Journal of Photogrammetry and Remote Sensing*, 114, pp. 166–178. doi: 10.1016/j.isprsjprs.2016.02.007.

Xie, Y., Tian, J. and Zhu, X.X. (2020) 'Linking Points with Labels in 3D: A Review of Point Cloud Semantic Segmentation', *IEEE Geoscience and Remote Sensing Magazine*, 1 December, pp. 38–59. doi: 10.1109/MGRS.2019.2937630.

Xin, S., Nousias, S., Kutulakos, K.N., Sankaranarayanan, A.C., Narasimhan, S.G. and Gkioulekas, I. (2019) 'A Theory of Fermat Paths for Non-Line-of-Sight Shape Reconstruction', in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6800–6809. doi: 10.1109/CVPR.2019.00696.

Xu, C., Wu, B., Wang, Z., Zhan, W., Vajda, P., Keutzer, K. and Tomizuka, M. (2020) 'SqueezeSegV3: Spatially-Adaptive Convolution for Efficient Point-Cloud Segmentation', in *European Conference on Computer Vision*. Springer, Cham, pp. 1–19. doi: 10.48550/arXiv.2004.01803.

Xu, J., Zhou, M., Wu, Z., Shui, W. and Ali, S. (2015) 'Robust Surface Segmentation and Edge Feature Lines Extraction from Fractured Fragments of Relics', *Journal of Computational Design and Engineering*, 2(2), pp. 79–87. doi: 10.1016/j.jcde.2014.12.002.

Xu, Y., Boerner, R., Yao, W., Hoegner, L. and Stilla, U. (2019) ‘Pairwise Coarse Registration of Point Clouds in Urban Scenes using Voxel-based 4-Planes Congruent Sets’, *ISPRS Journal of Photogrammetry and Remote Sensing*, 151, pp. 106–123. doi: 10.1016/j.isprsjprs.2019.02.015.

Xu, Y., Tuttas, S., Hoegner, L. and Stilla, U. (2021) ‘Voxel-based Segmentation of 3D Point Clouds from Construction Sites using a Probabilistic Connectivity Model’, *Pattern Recognition Letters*, 102, pp. 67–74. doi: 10.1016/j.patrec.2017.12.016.

Xu, Y., Tong, X. and Stilla, U. (2021) ‘Voxel-based Representation of 3D Point Clouds: Methods, Applications, and its Potential use in the Construction Industry’, *Automation in Construction*, 126. doi: 10.1016/j.autcon.2021.103675.

Yadav, M., Husain, A., Singh, A.K. and Lohani, B. (2015) ‘Pole-shaped Object Detection using Mobile Lidar data in Rural Road Environments’, in *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*. Copernicus GmbH, pp. 11–16. doi: 10.5194/isprsannals-II-3-W5-11-2015.

Yan, L., Li, Z., Liu, H., Tan, J., Zhao, S. and Chen, C.(2017) ‘Detection and Classification of Pole-like Road Objects from Mobile LiDAR Data in Motorway Environment’, *Optics and Laser Technology*, 97, pp. 272–283. doi: 10.1016/j.optlastec.2017.06.015.

Yan, W.Y., Morsy, S., Shaker, A. and Tulloch, M. (2016) ‘Automatic Extraction of Highway Light Poles and Towers from Mobile LiDAR Data’, *Optics and Laser Technology*, 77, pp. 162–168. doi: 10.1016/j.optlastec.2015.09.017.

Yang, B. and Dong, Z. (2013) ‘A Shape-based Segmentation Method for Mobile Laser Scanning Point Clouds’, *ISPRS Journal of Photogrammetry and Remote Sensing*, 81, pp. 19–30. doi: 10.1016/j.isprsjprs.2013.04.002.

Yang, B., Dong, Z., Zhao, G. and Dai, W. (2015) ‘Hierarchical Extraction of Urban Objects from Mobile Laser Scanning Data’, *ISPRS Journal of Photogrammetry and Remote Sensing*, 99, pp. 45–57. doi: 10.1016/j.isprsjprs.2014.10.005.

Yang, Z.X., Tang, L., Zhang, K. and Wong, P.K. (2018) ‘Multi-View CNN Feature Aggregation with ELM Auto-Encoder for 3D Shape Recognition’, *Cognitive Computation*, 10(6), pp. 908–921. doi: 10.1007/s12559-018-9598-1.

Yin, Y., Wan, W. and Liu, R. (2013) ‘Filtering Outliers using Statistical Analysis on Neighbors Distances’, in *IET International Conference on Smart and Sustainable City 2013 (ICSSC 2013)*. IEEE, pp. 149–152. doi: 10.1049/cp.2013.1993.

Yokoyama, H., Date, H., Kanai, S. and Takeda, H. (2011) ‘Pole-like Objects Recognition from Mobile Laser Scanning Data using Smoothing and Principal Component Analysis’, *ISPRS - International Archives of the Photogrammetry Remote Sensing and Spatial Information Sciences*, 38(5), pp. 115–120. doi: 10.5194/isprsarchives-XXXVIII-5-W12-115-2011.

Yokoyama, H., Date, H., Kanai, S. and Takeda, H. (2013) ‘Detection and Classification of Pole-like Objects from Mobile Laser Scanning Data of Urban Environments’, *International Journal of CAD/CAM*, 13(1), pp. 1–10. doi: 10.1016/j.optlastec.2017.06.015

You, H., Ji, R., Feng, Y. and Gao, Y. (2018) ‘PVNet: A Joint Convolutional Network of Point Cloud and Multi-view for 3D Shape Recognition’, in *MM 2018 - Proceedings of the 2018 ACM Multimedia Conference*. Association for Computing Machinery, Inc, pp. 1310–1318. doi: 10.1145/3240508.3240702.

Young, S.I., Lindell, D.B., Girod, B., Taubman, D. and Wetzstein, G. (2020) ‘Non-line-of-sight Surface Reconstruction using the Directional Light-cone Transform’, in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1404–1413. doi: 10.1109/CVPR42600.2020.00148.

Zaganidis, A., Sun, L., Duckett, T. and Cielniak, G. (2018) ‘Integrating Deep Semantic Segmentation into 3-D Point Cloud Registration’, *IEEE Robotics and Automation Letters*, 3(4), pp. 2942–2949. doi: 10.1109/LRA.2018.2848308.

Zai, D., Li, J., Guo, Y., Cheng, M., Lin, Y., Luo, H. and Wang, C. (2018) ‘3-D Road Boundary Extraction from Mobile Laser Scanning Data via Supervoxels and Graph

Cuts', *IEEE Transactions on Intelligent Transportation Systems*, 19(3), pp. 802–813. doi: 10.1109/TITS.2017.2701403.

Zegaoui Younes (2018) *LIRMM-BL 3D Urban Object Scan Dataset, LIRMM.fr*. Available at: http://www.lirmm.fr/~zegaoui/#download (Accessed: 20 February 2022).

Zeybek, M. (2021a) 'Extraction of Road Lane Markings from Mobile Lidar Data', in *Transportation Research Record*. SAGE Publications Ltd, pp. 30–47. doi: 10.1177/0361198120981948.

Zeybek, M. (2021b) 'Inlier Point Preservation in Outlier Points Removed from the ALS Point Cloud', *Journal of the Indian Society of Remote Sensing*, 49(10), pp. 2347–2363. doi: 10.1007/s12524-021-01397-4.

Zeybek, M. and Şanlıoğlu, İ. (2019) 'Point Cloud Filtering on UAV based Point Cloud', *Measurement: Journal of the International Measurement Confederation*, 133, pp. 99–111. doi: 10.1016/j.measurement.2018.10.013

Zhang Bibo, Xiang Bin and Zhang Lin (2017) 'Parameter-Free Outlier Removal of 3D Point Clouds with Large-Scale Noises', in *17th International Symposium on Communications and Information Technologies (ISCIT)*, Cairns, QLD, Australia, 2017, pp. 1-6, doi: 10.1109/ISCIT.2017.8261207.

Zhang, Y. *et al.* (2019) 'Data-driven point cloud objects completion', *Sensors (Switzerland)*, 19(7), 1514. doi: 10.3390/s19071514.

Zhang, Y., Liang, X. and Xu, G. (2013) 'A Robust 2-Refinement Algorithm in Octree and Rhombic Dodecahedral Tree Based All-Hexahedral Mesh Generation', *Computer Methods in Applied Mechanics and Engineering*, 256, pp. 88–100. doi: 10.1016/j.cma.2012.12.020