Edge Detection in Unorganized 3D Point Cloud

by

Razia Mahmood

A thesis submitted in partial fulfillment
of the requirements for the degree of
MSc Computational Sciences

The Faculty of Graduate Studies
Laurentian University
Sudbury, Ontario, Canada

Abstract

The application of 3D laser scanning in the mining industry is increasing progressively over the years. This presents an opportunity to visualize and analyze the underground world and potentially save countless man- hours and exposure to safety incidents.

This thesis envisions to detect the “Edges of the Rocks” in the 3D point cloud collected via scanner, although edge detection in point cloud is considered as a difficult but meaningful problem.

As a solution to noisy and unorganized 3D point cloud, a new method, EdgeScan method, has been proposed and implemented to detect fast and accurate edges from the 3D point cloud for real time systems. EdgeScan method is aimed to make use of 2D edge processing techniques to represent the edge characteristics in 3D point cloud with better accuracy. A comparisons of EdgeScan method with other common edge detection methods for 3D point cloud is administered, eventually, results suggest that the stated EdgeScan method furnishes a better speed and accuracy especially for large dataset in real time systems.

Keywords: *Scanning, Point Cloud, Edge Detection,*

Acknowledgments

First and foremost, I'd like to express my deepest gratitude and appreciations to my supervisor, Dr. Kalpdrum Passi, who gave me an opportunity, support, knowledge and dealt my queries with prodigious patience throughout the study period. I believe no supervisor could be better than who God gifted to me.

I'd also like to recognized remarkable assistance of Penguin Automated Systems Inc. and grateful to Dr. Greg Baiden, the owner and CEO of the company, specifically, for the data assistance and guidance. A very special thanks goes out to Mr. William Zeng for the supervision, without his guidance this thesis would not exist in its present form. I would also like to thanks Mr. Sean Brennan for all his support.

Last but not the least, I would like to acknowledge the encouragement of rest of my thesis committee: Dr. Ratvinder Grewal and Dr. Julia Johnson for their never-ending motivation.

Finally, and the most significantly, I would like to acknowledge persistent cooperation of my beloved family.

Table of Contents

Abstract iii
Acknowledgments iv
Table of contents v
List of Figures viii
List of Tables x

1. Introduction 1

1.1 Problem Statement 3
1.2 Overview 6
2. Point Cloud Data Acquisition 8

2.1. Point Cloud 8
2.2. Point Cloud Types 10
2.3. Data Acquisition 11
2.3.1 Laser Scans 12
2.3.2 Range Image 13
2.4 Measurement Techniques 14
2.4.1 Time of Flight 14
2.4.2 Phase Shift 16
2.4.3 Triangulation 16
2.4.4 Stereoscopic Vision 17
2.4.5 Structured Light 18

3. Edge Detection 20
3.1. Edge Detection 20
3.2. Steps for Edge Detection 23
3.3 Edge Detection Techniques 24
3.3.1. Robert Edge Detection 25
3.3.2. Sobel Edge Detection 26
3.3.3. Prewitt Edge Detection 27
3.3.4. Kirsch Edge Detection 27
3.3.5. Robinson Edge Detection 28
3.3.6. Marr-Hildreth Edge Detection 29
3.3.7. LoG Edge Detection 30
3.3.8. Canny Edge Detection 31
3.4 Analysis of Edge Detection 36
4. Related Work 39
4.1 Normal Estimation 42
4.2 3D Range Scan 45
4.3 RGB-Edge Detection 48
5. Methodology 50
5.1. Hang-up Robot 50
5.2. Point Cloud Data 53
5.3 EdgeScan Method 54
5.3.1. Scan Data Acquiring 56
5.3.2. Polar Coordinates 57
5.3.3. Convert Scanned data into 2D Image 57
5.3.4. Convert Scanned data into 3D Point Cloud 60
5.4 Edge Extraction 62

5.5 Mapping 2D Image into 3D Point Cloud 64
5.4 Software Implementation 66
6. Results and Analysis 67

6.1 Analysis on Edge Detection Methods for 2D Image 67
6.2 Results and Evaluations 68
6.2.1 Results 71
6.2.2 Running Time 75
6.3 Comparative Studies 77
6.4 Discussion 81
7. Conclusion and Future Work 82

7.1 Conclusion 82
7.2 Future Work 83

Bibliography 85

List of Figure

FIGURE 1.1 Hang-up draw bell example ……..3

FIGURE 2.1 3D Point Cloud examples ……9

FIGURE 2.2 Point Cloud data examples, XYZ format, PCD format……...…….…... 10

FIGURE 2.3 SICK Laser scanner (left), Velodyne Lidar (right)…….…….…… 12

FIGURE 2.4 SR 4000 TOF Camera (left), Microsoft Kinect (right)…….….....13

FIGURE 2.5 Time of Flight technique……..…….……..… 14

FIGURE 2.7 Point cloud captured with SR4000 camera ……16

FIGURE 2.7 Scene captured with Microsoft Kinect…….……19

FIGURE 3.1 Edge Detection example ……. 21

FIGURE 3.2 One-dimension Edge profiles ……. 22

FIGURE 3.3 Convolution example ……. 23

FIGURE 3.4 Simple model of non-maximum suppression…….. 35

FIGURE 3.5 Different Edge detection methods……37

FIGURE 41 Normal estimation (left), surface curvature examples……..45

FIGURE 4.2 Range image example ……46

FIGURE 4.3 Border extraction in Range image example ……47

FIGURE 4.4 RGB-D Edge Detection example……49

FIGURE 5.1 Hang-up Robot ……51

FIGURE 5.2 NORCAT Scan Data, Original ……...53

FIGURE 5.3 Laboratory Scan Data, Original……...54

FIGURE 5.4 Workflow of the algorithm, EdgeScan Method……..……55
FIGURE 5.5 Scanner Operating range diagram……...56
FIGURE 5.6 Polar coordinates diagram ……..57
FIGURE 5.7 Scanned data, Gray level data ……. ……..59
FIGURE 5.8 NORCAT scan data, 2D range image in Polar Coordinates ……..60
FIGURE 5.9 Laboratory scan data, 2D range image in Polar Coordinates……..60
FIGURE 5.10 TALIN device (left), World coordinates example (right)…….61
FIGURE 5.11 NORCAT scan data edge detection in 2D image ……63
FIGURE 5.12 Laboratory scan edge detection in 2D image……64
FIGURE 5.13 NORCAT scan data edge detection in 3D Point cloud…….65
FIGURE 5.14 Laboratory scan data edge detection in 3DPoint cloud …….……...65
FIGURE 6.1: Robert edge detection on 2D image……...68
FIGURE 6.2: Sobel edge detection on 2D image……68
FIGURE 6.3: Prewitt edge detection on 2D image……...69
FIGURE 6.4: LoG edge detection on 2D image……...69
FIGURE 6.5: Canny edge detection on 2D image……....70
GURE 6.6: Results of site1, (a) edge detection results overlaid on original, (b) edges, (c-f) details to edges ……73
FIGURE 6.7: Results of site1, (a) edge detection results overlaid on original, (b) edges, (c-f) details to edges ……...75
FIGURE 6.8: Boundary estimation results of site1, edge detection results overlaid on original ……....78
FIGURE 6.9 Boundary estimation results of site2, edge detection results overlaid on original …….79
FIGURE 6.10: Range Image method results of site2, (a) Range Image (b) edge detection results Overlaid original …..…….80

List of Tables

TABLE 6.1 Time consumed by different steps involved in algorithm 76

TABLE 6.2 Time consumed by different edge detection methods 81

Chapter 1

Introduction

The scanning technologies over past years have been rendered to be more accurate, reliable and affordable, thus it enhanced their usage in different segments commercially, however, highly dense point clouds, collected via scanning, have been widely used in applications not limited to such as making digital elevation models of the terrain, reverse engineering of industrial sites, tree reconstruction, medical imaginary and creating 3D models of urban environment.

Likewise, mining industry is also attaining this technology to visualize and analyze the underground world. This offers an opportunity where the virtual world of underground can be traversed with relative ease and many aspects of measurements could be obtained without hassle of visiting the working place or site recurrently, hence it potentially saving countless man-hours and still delivering precision work with significant savings in downtime, stoppages, and exposure to safety incidents.

Research is being continuously in progress to acquire improved point clouds. Point clouds contain the information to define and measure the world and the things that it includes,

nevertheless, the most common information that exist in point clouds that depict the 3D space, which surrounds us is the XYZ coordinates of each individual point. The exploitation of these three parameters leads to the knowledge of the dimensions and properties of natural and man-made entities. Also, many point cloud operations exploit the RGB and intensity information that is being provided by most measuring devices. Large point clouds requires the appropriate tools of elaboration and extraction of the significant information. Whereas, the extraction of geometric information of objects require the isolation of the points that describe the object itself by the rest of the point cloud. Henceforth, to achieve that, points should be detected that represent the boundaries or edges of the object.

Edges have been extensively used since the early age of computer vision research. Nonetheless, edges features have been used in a variety of applications, such as tracking, object recognition, pose estimation, visual odometry, and SLAM. There are several types of edges that occur in data. Depth discontinuities in the 3D data produce two related types of edges: occluding and occluded. Another type of 3D edge occurs at areas of high-curvature where surface normals change rapidly, which is call high curvature edges.

The Canny edge detector is one of the most broadly employed methods to find edges from 2D images due to its good localization and high recall.

This chapter commence with an introduction and motive of this research. Section 1.1 elaborates the problem statement along with research question, main objective and scope. Finally, an overview of algorithms and the thesis structure follows in Section 1.2.

1.1 Problem Statement

One of the issues faced by mining industry is hang-ups, which is a significant safety and production challenge for all the mines in the world. Hang-ups occurs in draw-bells, funnel-shaped cavities designed for ore collection during the process of "Block Caving". Block caving is an underground hard rock mining method in which a large section of rock is undercut, creating an artificial cavern that fills with its own rubble as it collapses. This broken ore falls into pre-constructed series of funnels and access tunnels underneath the broken ore mass. The miners extract it continuously from here. If the material isn't properly sized, it can get arches that results hang-up. To dislodge the hang-ups, "Key stone" is identified and blast. Key stone is a stone in a natural arch that's give its bearing capacity. If this stone is moved the arch will collapse. The miners are currently shoving bamboo poles up through the throat with explosives duck taped to them, and exposing themselves to hundreds of tons of rock that could give way without warning.

Figure 1.1: Hang-up draw bell example, Source: AZO Mining [43]

Figure 1.1 illustrates an example of blockage in the draw-bell of a mine. Of the estimated 200,000 draw-bells world -wide, up to10 percent of them are blocked at any one time. It creates dangerous situation for personnel and equipment to remove the blockage. Aside from being a safety issue, hang-ups also impact on productivity because ore is not available for collection and haulage from a blocked draw-bell.

Penguin Automated Systems Inc. has developed a hang-up assessment and removal robotic system. The robotic system features an arm that extends 15 feet horizontally, with a scanner mounted at the end for assessment, and 30 feet vertically through the throat of a draw-bell in a blocked cave mine. At the other end of the arm are a 3D camera, an infrared lighting system, a drill and an explosives loader.

The robot uses an INS positioning system (TALIN) that allows for the adjustment of the pitch, roll and yaw of the machine. The robot scans the inside of the draw-bell and with help of TALIN data, creates the Point cloud data. This point cloud data is then transferred to Unity 3D (gaming engine) , where the 3D Robot model simulates the original one, that allows an operator in a specially adapted Normet RBO personnel carrier a safe distance away to optimize the position of the arm, drill a hole, load an explosives charge and remotely clear the blockage.

Here comes the significance of the point cloud as it is the only source to estimate the situation of the blockage and identify the key stone to insert the explosive in order to clear the blockage. For this reason, point cloud should be more accurate and edges of rocks in draw-bell must be clearly visible as it is the first and most important step of the rock detection.

Many of the algorithms are designed for processing point clouds, but are slow and memory-inefficient to process large-size point cloud. Most of work for 3D edges detection has found edges or lines from polygonal mesh models or point clouds. Several approaches found crease edges directly from a point cloud. Although these approaches could detect sharp 3D edges from 3D models, they were computationally time-consuming because they relied on expensive curvature calculation and neighbor searching in 3D. Hence, these approaches may not be an ideal solution for this kind of real time robotic system.

To better describe an object visually, it is ideal to take the combined advantages of edge extraction approaches in 2D and 3D. In 2D images, physical edges are represented in the image by changes in the intensity function because image intensity is often proportional to scene radiance. In order to provide significant information about the 2D image, derivatives are computed by the edge detector which accepts digitized images as input and produces an edge map as output, which includes explicit information about the position, orientation, scale and strength of edges. However, since image deviates are sensitive to noise, smoothing operations, in most cases, are required to process the images aiming at reducing noise and regularizing the numerical differentiation, thus ensuring robust edge detection.

Based on the motivation mentioned above, the main research question of this thesis is:

- *Which algorithm is most appropriate to detect the edges of the rocks automatically and accurately in near real time in a dense and large unorganized 3D point cloud?*

In order to answer the research question, the main objective of this thesis is:

- *Design and implement an algorithm that enables to efficiently extract edges in 3D point cloud collected via a LASER scanner automatically for real time robotic system.*

1.2 Overview

This thesis presents a novel method, EdgeScan method, to extract edges in 3D unorganized point cloud. Detecting edges is the milestone which can eventually lead to detect Keystone. The main impression is extracting the edges data from a 2D image and mapping it into the corresponding 3D point cloud's points. In general, the overview of the EdgeScan method is primarily separated into three phases as shown in Figure 5.4. In first phase, scanned data is converted into 2D image. In the second phase, image processing part, edge extraction is applied on the image by using the Canny edge detection algorithm after the raw image data pre-processing. An easily-operating pixel data mapping mechanism is applied for corresponding 2D image pixels with 3D point cloud points in third phase. By referring to the correspondence map, 2D edge data are merged into 3D point cloud.

The structure of this thesis is organized as follows:

Chapter 2 first discusses the point clouds and its types and then some different point cloud data acquisition methods. Chapter 3 presents details about edge detection in image processing and discuss different well known methods for edge detection. Canny edge detection is discussed in detail and compared with the other methods as well. A comprehensive overview on different edge detection in 3D point cloud approaches over past decades are discussed Chapter 4.

Chapter 5 introduces the EdgeScan method in detail. Later in this chapter, implementation details about different point clouds preparation, software prototypes are discussed. Chapter 6 provides Analysis and results prepared in Chapter 5 and validates them both qualitatively and quantitatively. A comparative study is also conducted to validate the results. The limitations of proposed method are discussed afterwards. Chapter 7 concludes this thesis and gives recommendations for future improvement.

Chapter 2

Point Cloud Data Acquisition

This chapter illustrates a comprehensive overview of 3D point clouds and its types. Initially, the possibilities for data acquisition on-site is described like as laser scans, and range cameras, moreover, the acquisition techniques either generate a set of registered photographs or a sparse or dense point set. The following section presents some general concepts, which are often used for 3D point clouds.

2.1. Point Cloud

A point cloud is a collection of points with 3D coordinates $p(x; y; z)$ that represents the point's position. Alternatively, the point might have many other characteristics, such as color information (RGB), opacity or alpha (A), normal vector at that particular point ($N(x; y; z)$) and many more.

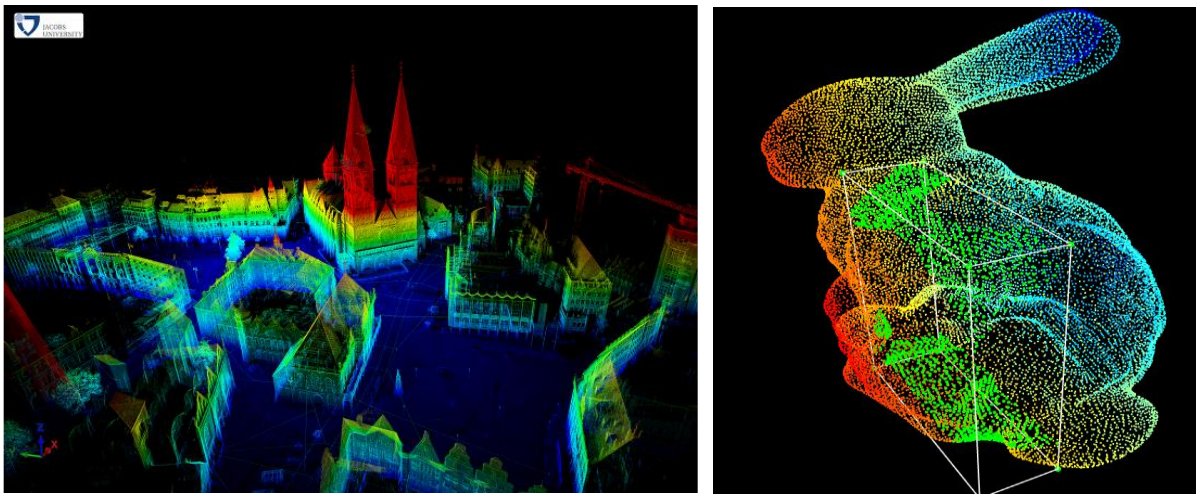
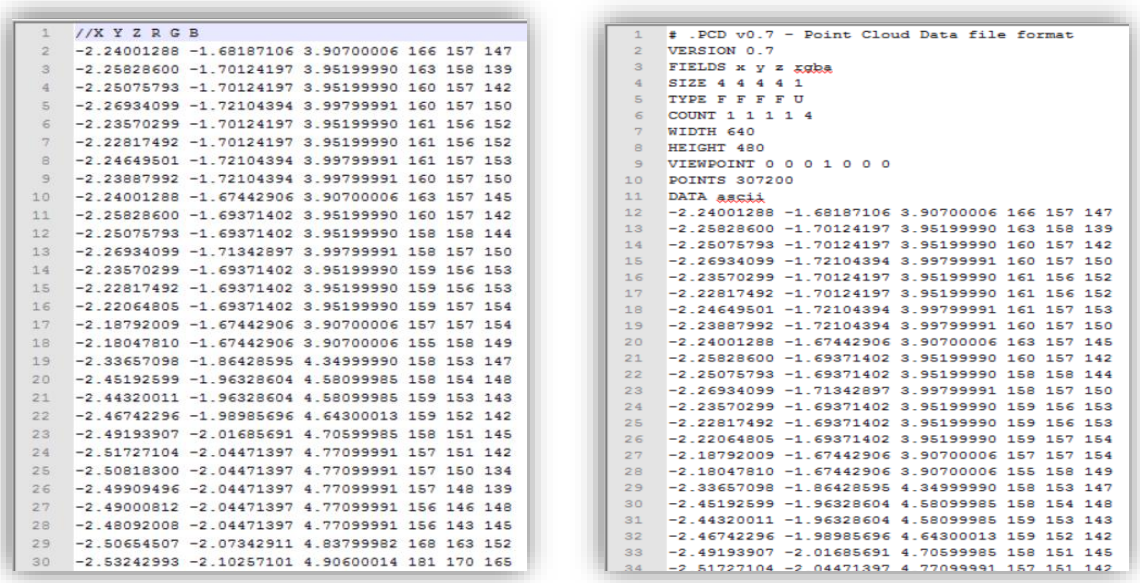


Figure 2.1: 3D Point Cloud examples, Source: Robotic3D scan repository, 3D Point cloud editor

Furthermore, Point clouds are considered as an effective and famous depiction of tangible geographical information. Digital elevation models of the terrain, reverse engineering of industrial sites, tree reconstruction and creating 3D models of urban environment are few examples of applications, where highly dense point clouds are extensively in use. Figure 2.1 demonstrate examples of the point cloud.

OpenGL, Cloud Compare, PCL are some graphic rendering engine, which can effortlessly visualized a point cloud, in fact it can also be render in more advanced tools such as gaming engines. Point clouds are usually stored as ASCII, LAS, XYZ, PLY, PCD format.



```
//X Y Z R G B
-2.24001288 -1.68187106 3.90700006 166 157 147
-2.25828600 -1.70124197 3.95199990 163 158 139
-2.25075793 -1.70124197 3.95199990 160 157 142
-2.26934099 -1.72104394 3.99799991 160 157 150
-2.23570299 -1.70124197 3.95199990 161 156 152
-2.22817492 -1.70124197 3.95199990 161 156 152
-2.24649501 -1.72104394 3.99799991 161 157 153
-2.23887992 -1.72104394 3.99799991 160 157 150
-2.24001288 -1.67442906 3.90700006 163 157 145
-2.25828600 -1.69371402 3.95199990 160 157 142
-2.25075793 -1.69371402 3.95199990 158 158 144
-2.26934099 -1.71342897 3.99799991 158 157 150
-2.23570299 -1.69371402 3.95199990 159 156 153
-2.22817492 -1.69371402 3.95199990 159 156 153
-2.22064805 -1.69371402 3.95199990 159 157 154
-2.18792009 -1.67442906 3.90700006 157 157 154
-2.18047810 -1.67442906 3.90700006 155 158 149
-2.33657098 -1.86428595 4.34999990 158 153 147
-2.45192599 -1.96328604 4.58099985 158 154 148
-2.44320011 -1.96328604 4.58099985 159 153 143
-2.46742296 -1.98985696 4.64300013 159 152 142
-2.49193907 -2.01685691 4.70599985 158 151 145
-2.51727104 -2.04471397 4.77099991 157 151 142
-2.50818300 -2.04471397 4.77099991 157 150 134
-2.49909496 -2.04471397 4.77099991 157 148 139
-2.49000812 -2.04471397 4.77099991 156 146 148
-2.48092008 -2.04471397 4.77099991 156 143 145
-2.50654507 -2.07342911 4.83799982 168 163 152
-2.53242993 -2.10257101 4.90600014 181 170 165
```

```
# .PCD v0.7 - Point Cloud Data file format
VERSION 0.7
FIELDS x y z rgba
SIZE 4 4 4 4
TYPE F F F U
COUNT 1 1 1 4
WIDTH 640
HEIGHT 480
VIEWPOINT 0 0 0 1 0 0 0
POINTS 307200
DATA ascii
-2.24001288 -1.68187106 3.90700006 166 157 147
-2.25828600 -1.70124197 3.95199990 163 158 139
-2.25075793 -1.70124197 3.95199990 160 157 142
-2.26934099 -1.72104394 3.99799991 160 157 150
-2.23570299 -1.70124197 3.95199990 161 156 152
-2.22817492 -1.70124197 3.95199990 161 156 152
-2.24649501 -1.72104394 3.99799991 161 157 153
-2.23887992 -1.72104394 3.99799991 160 157 150
-2.24001288 -1.67442906 3.90700006 163 157 145
-2.25828600 -1.69371402 3.95199990 160 157 142
-2.25075793 -1.69371402 3.95199990 158 158 144
-2.26934099 -1.71342897 3.99799991 158 157 150
-2.23570299 -1.69371402 3.95199990 159 156 153
-2.22817492 -1.69371402 3.95199990 159 156 153
-2.22064805 -1.69371402 3.95199990 159 157 154
-2.18792009 -1.67442906 3.90700006 157 157 154
-2.18047810 -1.67442906 3.90700006 155 158 149
-2.33657098 -1.86428595 4.34999990 158 153 147
-2.45192599 -1.96328604 4.58099985 158 154 148
-2.44320011 -1.96328604 4.58099985 159 153 143
-2.46742296 -1.98985696 4.64300013 159 152 142
-2.49193907 -2.01685691 4.70599985 158 151 145
-2.51727104 -2.04471397 4.77099991 157 151 142
```

Figure 2.2: Point Cloud data examples, XYZ format (Left), PCD format (Right)

In this thesis, Point Cloud Library (PCL) [41] is used to operate and envisage point clouds. PCL is describes as substantial scale, open project for 2D/3D image and point cloud processing (in C++, w/ new python bindings), nevertheless it's framework consists of various sophisticated algorithms including filtering, feature estimation, surface reconstruction, registration, model fitting and segmentation.

2.2 Point Cloud Type

2.2.1 Organized Point Cloud Data

An organized point cloud dataset is elaborated as the point clouds that is similar to an organized image (or matrix) like structure, where the data is divided into rows and columns.

Stereo cameras or Time of Flight cameras are good examples of organized point cloud data. The major advantages of an organized dataset is that it's cost effective by knowing the relationship between adjacent points (e.g. pixels), nearest neighbor operations, which ultimate make it structured, systematic, logical and productive with the ease of speeding computation of certain algorithms.

2.2.2 Unorganized Point Cloud Data

Unlike organized point cloud data, unorganized point cloud is dataset acquire through spherical projection devices, LIDAR sensor is a good example of unorganized point cloud.

2.3 Data Acquisition

In this section various possibilities for measuring real-world scenes has been described, such as laser scanning and range cameras. Aforesaid techniques can generate multiple registered images and/or a point cloud of the measured scene. However, laser scanners and range cameras directly generate dense point sets of a scene, moreover few of the range cameras can also produce grayscale or color images. Selection of an appropriate acquisition technique depends on provision of scale and type of the scene along with enough fund support. It has been observed that laser scans and active range cameras can produce significant results for homogeneous surfaces. Eventually, numerous approaches rely on emission of a signal i-e radio, light etc., that are used for range sensing, however, reading its response from its reflection on the scene and analyzing it against the

original signal. This is the reason which makes these approaches susceptible to reflective surfaces, since for specular surfaces the majority of the original signal is reflected in a direction other than the sensor, which is usually located close to the emitter; with the exception of where the specular surface is perpendicular to the direction of the signal itself. Furthermore, in case of diffuse surfaces, the signal is disperse back in various ways, which enhances the possibility of a sensor to pick up its response.

2.3.1 Laser Scans

A famous technique for acquiring objects from real-world scenes is using laser scanners, also referred to as LiDAR (light detection and radar). Laser scanners with the help of triangular or with a time-of-flight (TOF) techniques can develop dense unstructured point clouds of a scene. Based on the point of acquisition, LiDAR data is divided into two main categories: known as Terrestrial LiDAR and Airborne LiDAR, the first one is obtain by ground-based devices and is mostly utilize in reconstructing the facades of buildings, however, the later one is taken from the air and generally manipulate for producing building footprints, reconstructing roofs and 2.5D building models [1].

Figure 2.3: SICK Laser scanner (left), Velodyne Lidar (right), Source: SICK AG, Velodyne lidar

Last but not the least, Hybrid approaches combine data from both types of scans [2], [3].

2.3.2 Range Images

Range cameras illustrate a comprehend way to demonstrate real-word scenes by creating depth maps, however, comparing to laser scanners their acquisition is faster thus that enable them to capture moving objects efficiently. A range image is like a conventional camera image, except that each pixel stores a depth rather than a color. Range images encode the position of surface directly. Therefore, the shape can be computed reasonably easy. Additionally, range cameras typically have lower resolution and precision along with an appropriate distance measurement range [4]. Microsoft Kinect provides a very cost-effective and efficient way for digitizing 3D scenes, which additionally includes color information (see Figure 2.4). Moreover, the combination of deepness and cool image from the same viewpoint is called RGB-D image.

Figure 2.4: SR400 TOF Camera (left), MS Kinect (right), Source: Swiss Ranger, Microsoft

2.4 Measurement Techniques

There are different techniques to estimate distance based on the characteristics of the returning reflection: Time of Flight, Phase Shift and Triangulation etc. which are discussed independently in the following sections.

2.4.1 Time of Flight

Time-of-Flight (TOF) systems basically measure the delay until an emitted signal strikes a surface and arrive back to the receiver, thus it measures a true distance from the sensor to the surface. The sensing devices of this variety are such as a Laser Measurement System (LMS) or LIDAR, radars, Time-of-Flight (TOF) cameras, or sonar sensors that send "rays" of light (e.g. laser) or sound (e.g. sonar) in the world, which will reflect and return to the sensor. Figure 2.5 illustrates example of Time-of-Flight technique.

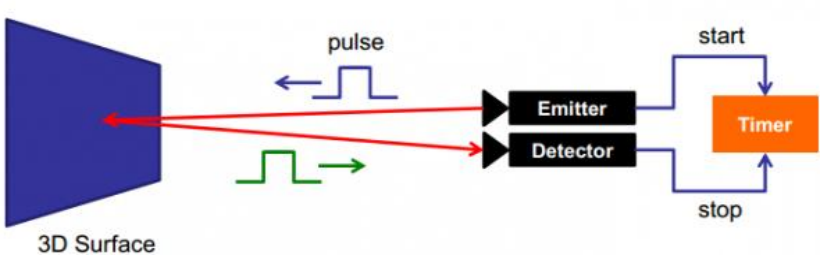


Figure 2.5: Time-of-Flight technique, Source: www.bashny.net

The distance *d* can be estimated, if the speed with which a ray propagates is known and using the precise path to measure the exact time when the signal was emitted and returned,

$$d = \frac{ct}{2}$$

where *c* represents the speed of the signal (e.g. speed of light for laser sensors), and *t* is the total time spent from when the ray was emitted and was received back [5].

Some laser measurement system inherently are 2D, hence, in the sense that they combine a laser range finder with a rotating mirror to measure distances on a plane. Additionally, to obtain a 3D point cloud, they are placed on rotating units such as pan/tilt or robot arms. By using the kinematics of the above mentioned units, one can obtain a multitude of such 2D measurement planes, which can be ultimately transformed into a steady 3D representations.

Hence, the resultant point cloud is quiet efficient to be used for modeling applications, however, systems using a rotating 2D laser are still encountering few drawbacks, out of all, important one is the speed of data acquisition. Nevertheless, to achieve high speed dense 3D data, Time-of-Flight camera systems, are used for instance Swiss Ranger 4000. These systems can provide 3D data representing a certain part of the world at frequencies up to 30Hz, thus enabling their usage in applications with fast reactive constraints. The disadvantage of resultant data is that it is noisier than the one acquired using laser sensors, not as dense (as the resolution of most TOF cameras is very low), and can suffer from big veiling effects. Figure 2.6 illustrates the example of such point cloud data, acquired using a SR 4000 TOF camera [5].

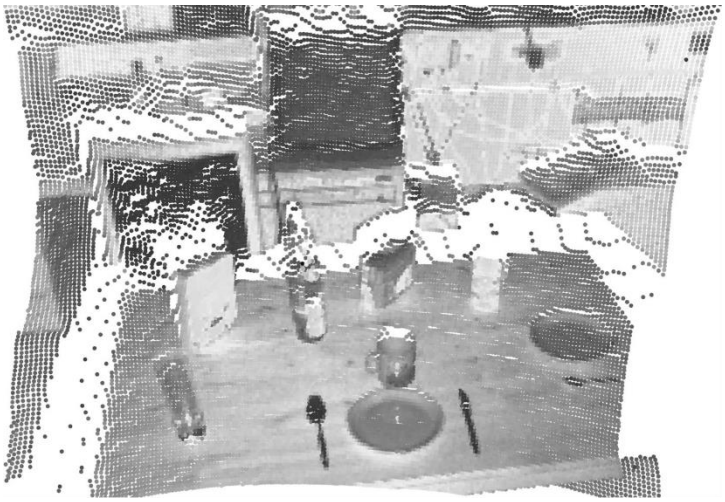


Figure 2.6: Point cloud acquired by SR 4000 TOF Camera, Source: R. Rusu, 2009, Dissertation [5]

2.4.2 Phase Shift

When the wavelength and the frequency of the signal are known, the shift in the phase of the emitted signal can specify the distance of the object. This method is used in scanners such as SICK LMS 200, an array of infrared LEDs is used to illuminate the scene with different modulations. On the other hand, the disadvantage of Phase Shift calculation is the short range of the infrared beam (typically less than 10 meters). Another limitation of the this method in terms of range coverage is that for a given signal of wavelength w , a reading of range s is not distinguishable from reading ranges $w + s$, $2w + s$ and so on. All of these distances have same phase shift.

2.4.3 Triangulation

In this method, the angle of the reflected laser ray is measured at which it enters the sensor. With

the help of this angle measurement, the distance of the emitter to the sensor and the size of the sensor, distance of the object to the emitter can be estimated.

This method calculates distances usually by using the following equation,

$$d = \frac{fT}{\|x_1 - x_2\|}$$

where f denotes the focal distance of both sensors, T the distance between the sensors, and x_1 and x_2 are the corresponding points in the two sensors [5]. Usage of stereo camera in mobile robotics application is the most admired system of triangular system despite many other applications are also in existence. Alike phase shift, the major drawback of triangulation based range sensing is its short maximum range for light beams [6].

2.4.4 Stereoscopic Vision

Stereoscopic is one of the famous techniques that is utilized by the human visual system to evaluate depth of objects. Since human eyes are positioned at slightly different places on the head, they provide the brain with two slightly different views of the scene they look at. The amount of displacement of a particular element between the images of the two eyes varies depending on the distance of the object. This helps the brain to recognize the element's depth. Position displacement depends upon the distance of an object, those which are closer to the viewer have greater position displacement, while distance objects tend to have less displacement.

Stereo Vision techniques apprehend the same technique used in RGB cameras. Two digital cameras can be used to capture two slightly different images if they are positioned close together. Corresponding pixels or patches between the two images can be identified using the color characteristics of the pixel and its neighbors, and the displacement in the projected images can be calculated in terms of pixels. Knowing the configuration of the camera, such as focal length, at the time of image capturing it is possible to determine the distance of each pixel/patch based on its displacement. Stereoscopic vision has been extensively discussed in literature and also commercially used in devices such as Sony HDR-TD10 for producing 3D videos.

2.4.5 Structured Light

In structured light application, light beams obtaining from the light source are projected onto the scene in a structured manner (e.g. stripes or a matrix of dots) and a sensor would capture how the structure is distorted when it is reflected back from the objects in the scene, hence, an easy example is Flashlight; as its projection would appear slightly bigger on further objects, on contrary smaller on the closer objects. This method is used solely such as in [7] and [8], or in conjunction with other methods such as in Microsoft Kinect and [8], [9]. Figure 2.7 shows capturing an interior scene with Microsoft Kinect. From left to right: Color image, depth image and 3D Point cloud respectively.

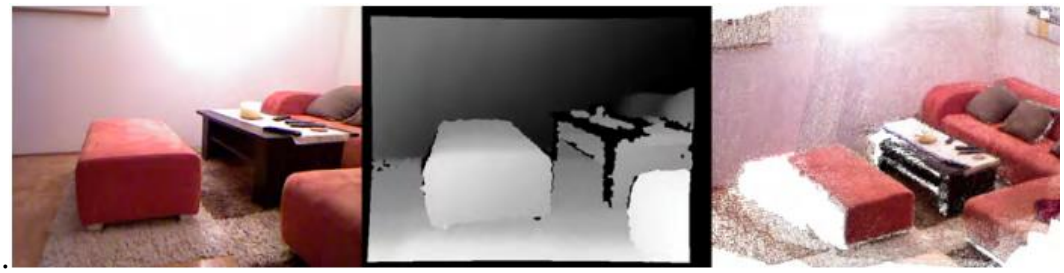


Figure 2.7: Scene captured with Microsoft Kinect (left to right), color image, depth image, point cloud, Source: Irene Reisner-Kollmann, 2013, Dissertation [10].

One major drawback of this method is the choice of pattern which shows the resolution of the sensing, nonetheless, in case of stripes, only the distance of points can be measured, where color is changed. Another approach would be to use a colored gradient so that every pixel can be measured, but this only works for scenes with minimal texture and changes in color. Another disadvantage of structured light methods is the interference between multiple light sources, can be mitigated in some cases.

Chapter 3

Edge Detection

In this chapter, Edge detections and its different algorithms will be under discussion. As mentioned in the introduction, 2D images are used in the thesis, therefore 2D edge detection and its techniques will be elaborated.

3.1 Edge Detection

In image analysis, edge detection is one of the most famous and frequently used operations. In image processing pertaining to computer vision, the edge detection utilizes the localization of important variations of a gray level image by the detection of the physical and geometrical characteristics of objects of the scene. Hence, it is a basic process that identifies and figure out an article and boundaries among objects and the background in the image. Edge detection commonly used in various application such as medical image processing, biometrics and many more by using object detection. Edge detection is an active area of research as it facilitates higher level image analysis.

Edges played a significant role in local changes in the image and that's the reason are important tool for analyzing images. Edges fundamentally appears on the boundary between two different sources in an image, nevertheless, an edge is defined by a discontinuity in gray level values. Figure 3.1 shows the discontinuity in gray level values. An edge can also be elaborated as a boundary between an object and the background. The structure of edges in images rely on numerous aspects: The geometrical and optical properties of the object, the illumination conditions, and the noise level in the images [11].

Figure 3.1: Edge intensity example, Source: Gonzalez & Woods, Digital Image Processing (2002)

Discontinuities in the image intensity can be demonstrate in the form of (1) step discontinuities and (2) line discontinuities. However, in first one the image intensity abruptly changes from one value on one side of the discontinuity to a different value on the opposite side, where as in the later one, the image intensity abruptly changes value but then returns to the starting value within some short distance, hence, step and line edges are few in real images. Sharp discontinuities are rarely exist in real world signals due to the low-frequency components or the smoothing introduced by most sensing devices. Step edges converted to ramp edges, whereas line edges become roof edges, where intensity changes are not instantaneous but occur over a finite

distance. Occasionally, edges could have features of both step and line. Additionally, spatial masks can be used to detect all types of discontinuities in an image. Figure 3.2 illustrates different types of edge discontinuities.

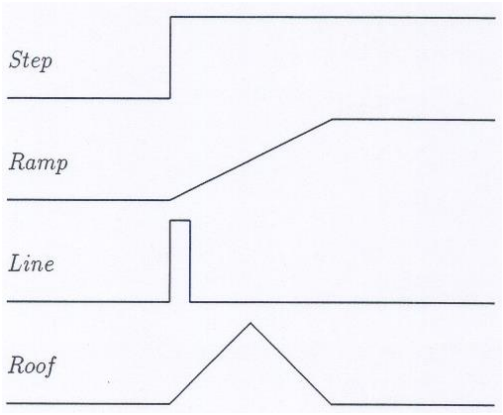


Figure 3.2: One-dimension Edge profiles, Source: Jain et al, *Machine Vision, chapter- 5 [13]*

Edge detecting in an image considerably reduces the amount of data and filters out unusable information, whereas preserve the vital structural properties in an image.

3.1.1 Convolution

Filtering is a common process in digital images to remove the noise. Convolution is a general filter effect for image and modifies the spatial characteristics of the image. It is done by a simple mathematical operation by multiplying a pixel's and its neighboring pixel values by a matrix called Kernel. Kernel or Mask is a small integer matrix, usually of size 3x3, used for convolution.

Convolution is done by moving the kernel on the image, starting from the top left corner

of the image to the bottom right within the boundaries. With each pixel of the image, the kernel cell value is multiplied and added together. Mathematically it can be written as:

$$P(i,j) = \sum_{k=1}^{m} \sum_{l=1}^{n} I(i+k-1, j+l-1)K(k,l)$$

As a result, the output is a new improved filtered image. The choice of the kernel affects the results accordingly. Figure 3.3 shows the process of convolution using 3x3 kernel.

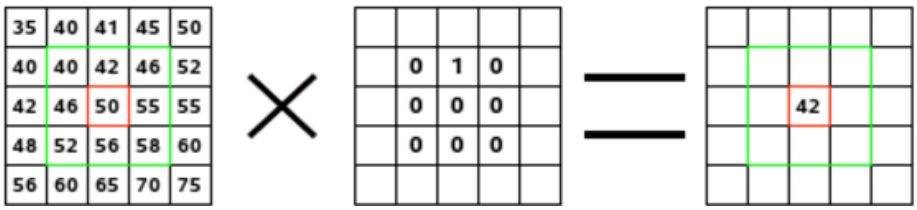


Figure 3.3: Convolution example, Image (left) Kernel (middle) output Image (right), Source: docs.gimp.org

3.2 Steps for Edge Detection

Edge detection's algorithms consist of steps as under;

- **Filtering**

Filtering is a process in which noise is minimized, however, noise is reduce as much as further do not affect the meaningful edges, hence, increasing filtering further will lead to loss of edge strength.

- **Enhancement**

To determine a change in intensity, the neighborhood of a point is vital, nevertheless, enhancement focuses on pixels, where a significant change in localized intensity values are noticed, which is commonly executed by computing the gradient magnitude.

- **Detection**

Detection is used to identified the point belongs to edge points as the points with only strong edge content are needed in the process, however, many points in an image have a non-zero value for the gradient, and not all of these points are edges for a particular application. Frequently, thresholding provides the criterion used for detection.

- **Localization**

The location of the edge can be estimated with sub-pixel resolution. It can be decided which of the local maxima output by the filter are meaningful edges and which are caused by noise.

3.3 Edge Detection Techniques

Over the last three decades edge detection techniques have been studied thoroughly and various literature is available on these techniques. The prominent characteristic of edge detection technique is its capability to extract the accurate edge line with an appropriate alignment. Moreover, the limitation has been noticed is that there is not yet any common performance directory to evaluate

the functioning of the edge detection techniques, whereas the performance of an edge detection techniques are mostly required a personal analysis and separately dependent to its application.

Edge detection methods could convert original images into edge images benefits from the changes of grey tones in the image and it has capability to review image, nonetheless been used by advanced computer vision algorithms.

Various literature on edge detection techniques are available, hence, the most commonly used discontinuity based edge detection techniques are reviewed in this section some of those techniques are Roberts 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, are discussed in detail;

3.3.1 Roberts Edge Detection

The Roberts edge detection was introduced by Lawrence Roberts (1965) [35]. This method performs a simple and fast 2D spatial gradient measurement on an image. This technique focuses on high spatial frequency regions as they often correspond to edges. A grayscale image is the input to the operator and the same as to the output. In the output, pixel values in every point represents the estimated complete magnitude of the spatial gradient at that point for the input image.

G_x and G_y are calculated using mask shows in Table 3.1

Table 3.1: Masks used in Robert's operator

-1	0
0	1

G_x

0	-1
+1	0

Gy

3.3.2 Sobel Edge Detection

The Sobel edge detection method was presented by Sobel in 1970 [14]. This method of edge detection for image segmentation finds edges using the Sobel approximation to the derivative. It starts looking for the edges at those points where the gradient is highest. The Sobel method performs a 2D spatial gradient quantity on an image and so emphasizes the areas of high spatial frequency that correspond to edges. This method is used to find the estimated absolute gradient magnitude in an input grayscale image at each point. In inference at least the operator consists of a pair of 3x3 complication kernels as shown in under Table 3.2. One kernel is simple and the other rotated by 90^o. It is very similar to the Roberts Cross operator.

Table 3.2: Masks used in Sobel's operator

-1	0	+1
-2	0	2
-1	0	+1

G_x

-1	-2	-1
0	0	0
+1	+2	+1

G_y

3.3.3 Prewitt Edge Detection

The Prewitt edge detection was represented by Prewitt in 1970 [14]. Prewitt method is a correct way to estimate the magnitude and orientation of an edge. To estimate the direction from the magnitudes in the x and y-directions different gradient edge detection needs a quiet time consuming calculation, the compass edge detection gets the direction directly from the kernel with the maximum rejoinder. It is limited to 8 possible directions; though knowledge demonstrates that most direct direction estimates are not too perfect. This edge detector is estimated in the 3x3 neighborhood for eight directions. All the eight convolution masks are calculated. One complication mask, with the largest module is then selected. Table 3.3 shows the masks used in Prewitt's operator.

Table 3.3: Masks used in Prewitt's operator

-1	-1	-1
0	0	0
+1	+1	+1

G_x

-1	0	+1
-1	0	+1
-1	0	+1

G_y

Based on the observation it has been identified that Prewitt detection is relevantly easier to apply computationally than the Sobel detection, but it is more prone to produce noisier outcome.

3.3.4 Kirsch Edge Detection

Kirsch edge detection was proposed by Kirsch (1971) [36]. In this technique the masks are defined

by considering a single mask and then rotating it to eight main compass directions: North, West, South, East, Northwest, Southwest, Southeast and Northeast. The masks are distinctive as follows N, W, S, E, NW, SW, SE and NE respectively.

$$\underset{l_1}{E=\begin{bmatrix}-1 & -1 & 3\\ -1 & 0 & 3\\ -1 & -1 & 3\end{bmatrix}} \quad \underset{l_2}{NE=\begin{bmatrix}-1 & 3 & 3\\ -1 & 0 & 3\\ -1 & -1 & -1\end{bmatrix}} \quad \underset{l_3}{N=\begin{bmatrix}3 & 3 & 3\\ -1 & 0 & -1\\ -1 & -1 & -1\end{bmatrix}} \quad \underset{l_4}{NW=\begin{bmatrix}3 & 3 & -1\\ 3 & 0 & -1\\ -1 & -1 & -1\end{bmatrix}}$$

$$\underset{l_5}{W=\begin{bmatrix}3 & -1 & -1\\ 3 & 0 & -1\\ 3 & -1 & -1\end{bmatrix}} \quad \underset{l_6}{SW=\begin{bmatrix}-1 & -1 & -1\\ 3 & 0 & -1\\ 3 & 3 & -1\end{bmatrix}} \quad \underset{l_7}{S=\begin{bmatrix}-1 & -1 & -1\\ -1 & 0 & -1\\ 3 & 3 & 3\end{bmatrix}} \quad \underset{l_8}{SE=\begin{bmatrix}-1 & -1 & 3\\ -1 & 0 & 3\\ -1 & 3 & 3\end{bmatrix}}$$

The maximum value found by convolution of each mask with the image is defined as edge magnitude. The mask that generates the maximum magnitude defines the direction. For example, mask l_6 corresponds to a diagonal edge whereas mask l_1 corresponds to a vertical edge. It can be notice that the last four masks are actually the same as the first four, but flipped about a central axis.

3.3.5 Robinson Edge Detection

The Robinson method was represent by Robinson in 1977, which is same as to Kirsch masks but is easier to implement as they depend only on coefficients of 0, 1 and 2 [37]. The masks are symmetrical about their directional axis, the axis with the zeros. Only four masks are computed and then the rest of the masks are computed by negating the result from the first four. The masks are as follows:

$$\underset{l_1}{E} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \underset{l_2}{NE} = \begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix} \underset{l_3}{N} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \underset{l_4}{NW} = \begin{bmatrix} 2 & 1 & 0 \\ 3 & 0 & -1 \\ -1 & -1 & -2 \end{bmatrix}$$

$$\underset{l_5}{W} = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \underset{l_6}{SW} = \begin{bmatrix} 0 & -1 & -2 \\ 1 & 0 & -1 \\ -2 & 1 & 0 \end{bmatrix} \underset{l_7}{S} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \underset{l_8}{SE} = \begin{bmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix}$$

The maximum value obtained from applying all eight masks to the pixel neighborhood is defined as magnitude of the gradient. The angle of the line of zeroes in the mask yielding the maximum response can be approximated as the angle of the gradient.

3.3.6 Marr-Hildreth Edge Detection

The Marr-Hildreth was proposed in 1980, which is a technique of detecting edges in digital images that is continuous curves where there are well-built and fast distinctions in image brightness [38]. It is easy and operates by convolving the image with the LoG function. Later, to find the edges, the zero-crossings are revealed in the filtered result. The LoG method, due to its image shape while turned up-side-down is also referred as the Mexican hat wavelet. The Marr-Hildreth edge detector algorithm is:

Smooth the image using a Gaussian

• A two-dimensional Laplacian is applied to the smoothed image (often the first two steps are combined into a single operation)

• Look for sign changes by applying loop through the result. If there is a sign change plus the slope across, mark as an edge if the sign change is greater than some threshold.

• It is possible to run the result of the Laplacian through a hysteresis alike to Canny's edge detection to get better results, although this is not how the edge detector was originally implemented.

3.3.7 LoG Edge Detection

The Laplacian of Gaussian (LoG) was introduced by Marr (1982). The LoG of an image f(x,y) is a second order derivative defined as,

$$\nabla^2 = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

This method has two effects, on one hand it smooths the image and on other hand computes the Laplacian, which produces a double edge image. Locating edges then consists of finding the zero crossings between the double edges. Generally mask as shown in Table 3.4 is used to implement Laplacian function,

Table 3.4: Masks used in LoG s operator

0	-1	0
-1	4	-1
0	-1	0

G_x

-1	-1	-1
-1	8	-1
-1	-1	-1

Gy

The Laplacian is generally used to found whether a pixel is on the dark or light side of an edge.

3.3.8 Canny Edge Detection

In commercial world, the Canny edge detection technique is considered as one of the standard edge detection techniques. Initially it was introduced by John Canny for his Master's thesis at MIT in 1983, and it is still excel many of the newer algorithms that have been developed over the years [12]. Canny is a highly used method to identify the edges by separating noise from the image before finding edges of image. The same theory is a better solution that without disturbing the features of the edges in the image afterwards it applying the tendency to find the edges and the serious value for threshold.

1. Detection: In detection the chances of identifying the real edge point should be increased, however, the chances of unreal and default non-edge points is decreased. This feature leads to get optimal signal-to-noise ratio.

2. Localization: The detected edges should be as nearer to the real edges.

3. Number of responses: One real edge should not produce more than one detected edge (it has been noticed that this is an absolute requirement).

The algorithm takes place in 5 separate steps:

- Smoothing: To remove noise, blurring of the image.

- Finding gradients: The gradients of the image having large magnitudes should be marked as edge.
- Non-maximum suppression: Only local maxima should be marked as edges.
- Double thresholding: Apply thresholding to determine potential edges.
- Edge tracking by hysteresis: All the other edges that are not connected to a very strong edge are suppressed to determine final edges.

All the steps are described in detail in the subsequent subsections.

3.3.8.1 Smoothing

It is predetermined that all images taken from a camera will encompass of some amount of noise, however, to prevent noise for edges it is necessary to reduce the noise. For that reason, a Gaussian filter is first applied to smooth the image. The kernel of a Gaussian filter with a standard deviation of $\sigma = 1.4$ is given in equation (1).

$$K = \frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} \quad (1)$$

3.3.8.2 Finding Gradients

One of the prominent feature of Canny algorithm is finding those edges, where the intensity of the grayscale images changes the most. With the help of gradients of the image these areas can be determine. Sobel-operator is applied on the smoothed image to determine the gradients at each pixel. In the first step, the kernels are applied to approximate the gradient in the x- and y-direction respectively, can see in equation (2).

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$
$$G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \quad (2)$$

By applying the law of Pythagoras, the gradient magnitudes can then be determined as a Euclidean distance measure as shown in equation (3). To reduce the computational complexity sometimes Manhattan distance measure also can be applied as shown in equation (4).

$$|G| = \sqrt{G_x^2 + G_y^2} \quad (3)$$

$$|G| = |G_x| + |G_y| \quad (4)$$

Gx and Gy are the gradients in the x- and y-directions respectively.

At this step, an image of the gradient magnitudes often indicate the edges prominently. However, the edges are eventually wide and couldn't stipulate the definite area of edges. Besides to make it

possible to evaluate this, the direction of the edges must be determined and gathered as indicated in equation (5).

$$\theta = arctan\left(\frac{G_x}{Gy}\right) \quad (5)$$

3.3.8.3 Non-Maximum Suppression

Non-maximum suppression is used to convert the "blurred" edges in the image of the gradient magnitudes to "sharp" edges. Basically the results have been determined by preserving all local maxima in the gradient image, whereas deleting the rest. The algorithm is for each pixel in the gradient image:

- Round the gradient direction θ to nearest 45°, corresponding to the use of an 8-connected neighborhood.
- Current pixel's edge strength is compared with the edge strength of the pixel in the positive and negative gradient direction. i.e. compare with the pixels to the north and south if the gradient direction is north (theta = 90°).
- Preserve the value of the edge strength if the edge strength of the current pixel is largest.

Suppress (i.e. remove) the value if it is not the largest.

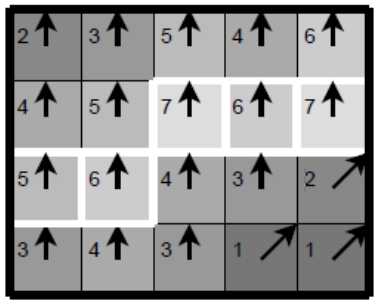


Figure 3.3: Simple model of non-maximum suppression

Over and above, Figure 3.3 illustrates a simple model of non-maximum suppression. Majority of the pixels have gradient directions pointing to the north, therefore compared with the pixels above and below. The white borders represents the pixels that turn out to be maximal in this comparison, whereas rest of the pixels will be suppressed.

3.3.8.4 Double Thresholding

Following the non-maximum suppression step, the remaining edge-pixel are still noticeable with their strength pixel-by-pixel. It is assumed that most of them are commonly the actual edge in the image but few could be created by noise and difference in color for example due to uneven surfaces. To overcome this issue, the easiest approach to recognize between these would be to use a threshold, so that only edges stronger that a certain value would be preserved. The Canny edge detection algorithm uses double thresholding. Edge pixels stronger than the high threshold are marked as strong; edge pixels weaker than the low threshold are suppressed and edge pixels between the two thresholds are marked as weak.

3.3.8.5 Edge Tracking by Hysteresis

The strong edges are explicated as “certain edges”, and can immediately be included in the final edge image. On contrary, the weak edges are included if and only if they are connected to strong edges. The phenomena behind is that noise and other minor elements are unlikely lead to strong edge (with proper adjustment of the threshold levels). Thus, in the original image, the strong edges will (almost) only be due to true edges, however, the weak edges can either be due to true edges or noise/color variations. The latter type will probably be distributed independently of edges on the entire image, and thus only a small amount will be located adjacent to strong edges. Weak edges due to true edges are much more likely to be connected directly to strong edges.

BLOB-analysis (Binary Large OBject) can be used for edge tracking to be implemented. The edge pixels are distributed into connected BLOB’s using 8-connected neighborhood. BLOB’s containing at least one strong edge pixel are then preserved, while rest of the BLOB’s are suppressed.

3.4 Analysis of Edge Detection Methods

Roberts edge detector, Sobel Edge Detector, Prewitt edge detector, Kirsch, Robinson, Marr-Hildreth edge detector, LoG edge detector and Canny Edge Detector are the few prominent relative performance edge detection techniques, which were under discussion in this chapter.

Based on the literature review, Roberts, Sobel, Prewitt, Kirsch and Robinson algorithms

Figure 3.4: Different Edge detection methods, Source: M.Radha2 et al, 2011, Edge detection techniques for Image segmentation [15]

are simple to implement and can detect edges and their orientations but have major drawbacks in sensitive to noise. As the dimension of the kernel filter and its coefficients are static and it cannot be adapted to a given image thus the results might be inaccurate. Marr-Hildreth and LoG algorithms find the correct edges as they test wider area around the pixel but they malfunction at corners, curves and where the gray level intensity fluctuates.

Figure 3.4 demonstrates that Roberts, Sobel and Prewitt outcomes are actually deviated from the others, in contrast, Marr-Hildreth, LoG and Canny develop almost the same edge map. In addition, Kirsch and Robinson edge maps look similar. Canny demonstrates the best results among all of them.

The performance of the Canny's algorithm mainly depends on the changing parameters which are standard deviation for the Gaussian filter, and its threshold values. Unlike Roberts and Sobel, the Canny's operation is not very susceptible to noise. By applying smoothing concept, the finding of errors is effective by using the probability. Improving signal with respect to noise ratio. Canny edge detection algorithm is more costly due to complex calculations in comparing to Sobel, Prewitt and Robert's operator. If the Canny detector worked well it would be superior.

Even though, so many edge detection techniques are available in the literature, it is a challenging task to the research communities to detect the exact image without noise from the original image.

Chapter 4

Related Work

From the beginning of computer vision research, edges have been used as the basis of different methods. Hence, sharp feature extraction is a vital concern in many scientific fields, such as computer graphics, medical imaging, computer vision and computational fluid dynamics. Moreover, it has been observed that few of the researches are concentrating on extracting sharp features on 3D point clouds. In fact, various researches have been found edges or lines from polygonal mesh models or point clouds.

A number of techniques for edge and sharp feature extraction in point cloud have been segmented into different classes: In [16], [17] and [18] the authors have discussed robust statistics to extract sharp features. Surface segmentation and line segmentation has been proposed to extract sharp features in [19], [20] and [21] respectively. Furthermore, a region growing method is

explored in [22], [23], [24] and [25] that segments the point cloud into clusters and based on the analysis of the normals of the points, identify the regions with sharp features.

Fleischman et al. [16] statistical techniques are used in order to identify sharp features. With the guidance of moving least squares (MLS) computation neighborhoods are created and then these neighborhoods of points are divided into regions corresponding to the same surface part. A further development of this work by Daniels et al. [17] extracts feature curves on the reconstructed MLS surface. The advantage of that points on the sharp feature can be identified in the case of noisy and rough input data. Later on, by following the same phenomenon, Oztireli et al. [18] embraced a robust implicit moving least-squares (RIMLS) method to locally approximate the scanned surface and to safe sharp features. Besides, they have also implemented kernel regression to extend the moving least squares (MLS) surface reconstruction with sharp features and their method maximized the arrangement of sharp features by combining the MLS and local kernel regression.

Further, Demarsin et al. [19] also worked on sharp features in point cloud data to obtain the closed sharp features and they apply the method of segmentation to recognize the regions of sharp features. Over and above, the output is basically a set of points with various points indicating the feature line. Therefore, they use a graph approach with a minimum spanning tree for closed feature lines. On the other hand, Xu et al. [20] introduced a method to segment surface and extract edge feature lines of rough fractured fragments, by merging faces based on face normal vector and roughness, an accurate surface segmentation is implemented whereas edge feature lines are detected based on the surface segmentation. Later in the years Gumhold et al. [24] proposed a

method that build the connectivity information in the point cloud by using the Riemannian tree. Then, principal component analysis (PCA) is use to analyze the neighborhood of each point. To determine a probability of a point belonging to a feature, or the kind of feature, the Eigen values of the correlation matrix are used. Line-type features, border and corner points can be distinguish easily by this method. The outcome is a pretty dense set of points including the feature, independently of whether the feature is sharp or not, as points with high curvature values are extracted.

Weber et al. [22], [23] proposed a method for extracting sharp features on an unstructured point cloud; Gauss map clustering on local neighborhoods is computed in this method in order to discard all points that are doubtful to belong to a sharp feature. Feng et al. [25] have introduced an algorithm for accurately detecting multiple planes from point clouds in real time. They have constructed a graph whose node represents a group of points and edge shows their neighborhood. To systematically merge nodes belonging to the same plane an agglomerative hierarchical clustering is performed on that graph.

It is assumed that noise reduction algorithms for example jump edge filtering is workable, especially for finding better boundaries [22] for each region. However, Lin et al. [2] presented a method that is capable of accurately extracting plane intersection line segments from large-scale raw scan points. The 3D line-support region, namely, a point set near a straight linear structure, is extracted simultaneously. Additionally, the 3D line-support region is fitted by a Line-Segment Half-Planes (LSHP) structure, which provides a geometric constraint for a line segment, making the line segment more reliable and accurate.

Moreover, Wang et al. [24] executed the majority voting scheme, in order to detect distinct geometric features which is implemented in sharp edges and outliers in a scanned point cloud. Eventually to achieve sharp edge features precisely, it is inevitable to analyze normal accurately using a convenient neighborhood points. In the following section, we will be summarize the normal estimation techniques on the sharp edge features.

4.1 Normal Estimation

The fundamental step in point cloud data processing is a reliable estimation of normal vectors at each point in a point cloud. However, in order to extract a sharp edge features from a 3D point cloud requires accurate normals as input to in to generate high quality surfaces. The performance of common point based rendering techniques is much dependent on the accuracy of the input normals. In this section, some researches on the computation of normals has been reviewed; particularly, some efforts on normal estimation to detect sharp features from point clouds.

Park et al. [26] proposed EGG (Elliptic Gabriel Graph) which is an intuitive extension of the Gabriel graph (GG), using an elliptic influence region. EGG provides balanced neighbors by considering both distance and directional spread and can be used for normal vector estimation. Later on, Holzer et al. [27] have introduced two methods for fast estimation of surface normals data using integral images from organized point cloud. It is possible to adapt the considered neighborhood size according to depth and object borders without any additional cost in terms of processing speed by using integral images.

Demarsin et al. [19] detects closed sharp feature lines to create a closed curve network. A first order segmentation is used to extract to be feature points and to recover the sharp feature lines process them as a graph. To estimate normal the Delaunay triangulation is considered and the normal of all sample point is estimated as the normal of the least squares plane through the neighbors.

On the other hand, Grim et al. [28] have focused on non-homogenously sampled and noisy point data. The outcomes of the algorithm is a surface normal for each data point, a local surface approximation in the form of a one-ring, the feature size, the local shape (flat, ridge, bowl, saddle, sharp edge, corner, boundary) and a definite value that can be used to determine areas where the sampling is poor or not surface-like. Whereas Li et al. [29] estimate normals on unorganized point clouds by using statistics methods to detect the local tangent plane for each point. This proposed algorithm is efficient for dealing with points located in high curvature regions or near/on complex sharp features.

Alternatively, Zhang et al. [30], for neighborhood clustering uses neighbor points normals as prior information. Subsequently, to represent the prior information as a guiding matrix an unsupervised learning process is design. Thereupon, the anisotropic neighborhood is segmented into several isotropic neighborhoods by low-rank subspace clustering with the guiding matrix. Therefore, the normal of the points near sharp features is estimated as the normal of a plane fitting the consistent sub-neighborhood. Even in noisy and anisotropic samplings, this method is can estimate normals whereas preserving sharp features within the original point data.

Similarly, Wang et al. [31] have present a normal estimation method in order to establish effectually appropriate neighborhood for each point in the 3D point cloud. Precisely, an anisotropic neighborhood is formed, for a point near sharp features, to only encompass neighboring points located on the same surface patch as the point. Rest of the neighboring points on the other surface patches are rejected.

Estimating normals on the sharp feature points becomes more challenging as the neighborhood selected for the normal estimation would encompass points belonging to different surface areas across the sharp feature.

The problem of determining the normal to a point on the surface is approximated by the problem of estimating the normal of a plane tangent to the surface, which in turn becomes a least-square plane fitting estimation problem.

R.Rusu [5], has proposed normals estimating method in his dissertation. An open source implementation of normal estimation is also available in the Point Cloud Library [32]

The solution for estimating the surface normal is reduced to an analysis of the eigenvectors and eigenvalues (or PCA – Principal Component Analysis) of a covariance matrix created from the nearest neighbors of the query point. More specifically, for each point ***P***, the covariance matrix C is as follows:

$$C = \frac{1}{k}\sum_{i=1}^{k}.(p_i - p^-).(p_i - p^-)^T, C.\vec{v_j} = \lambda_j.\vec{v_j}, j \in \{0,1,2\}$$

Where k is the number of point neighbors considered in the neighborhood of p_i, p^- represents the 3D centroid of the nearest neighbors, λ_j is -the *j-th* eigenvalue of the covariance matrix, and $v_j^{\rightarrow}$ the *j-th* eigen vector.

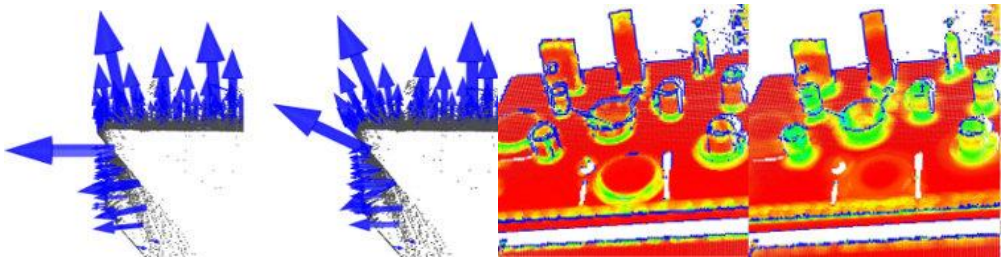


Figure 4.1: Normal estimation (left), surface curvature (right) examples, Source: PointClouds.org (PCL)

As mentioned earlier, a surface normal at a point needs to be estimated from the surrounding neighborhood support also called **k-neighborhood**. Therefore it is of great importance to select the right scale for **k** or **r** values which determine set of nearest neighbors of that point. The selection of the scale for estimation of neighborhood of point depends upon the magnitude of detail required for any particular application.

4.2 3D Range Scan

Range images are a special class of digital images. In a range image each pixel embody the distance between a known reference frame and a perceptible point in the scene. Hence, the 3D structure of a scene is reproduce by a range image. These images are also referred to as depth maps, xyz maps, depth images, surface profiles and 2.5D images.

Range images have two basic forms. One form is a list of 3D coordinates for cloud of points, order doesn't matters. The other one is a matrix of depth values of points along the directions of the x, y image axes, which makes spatial organization clear.

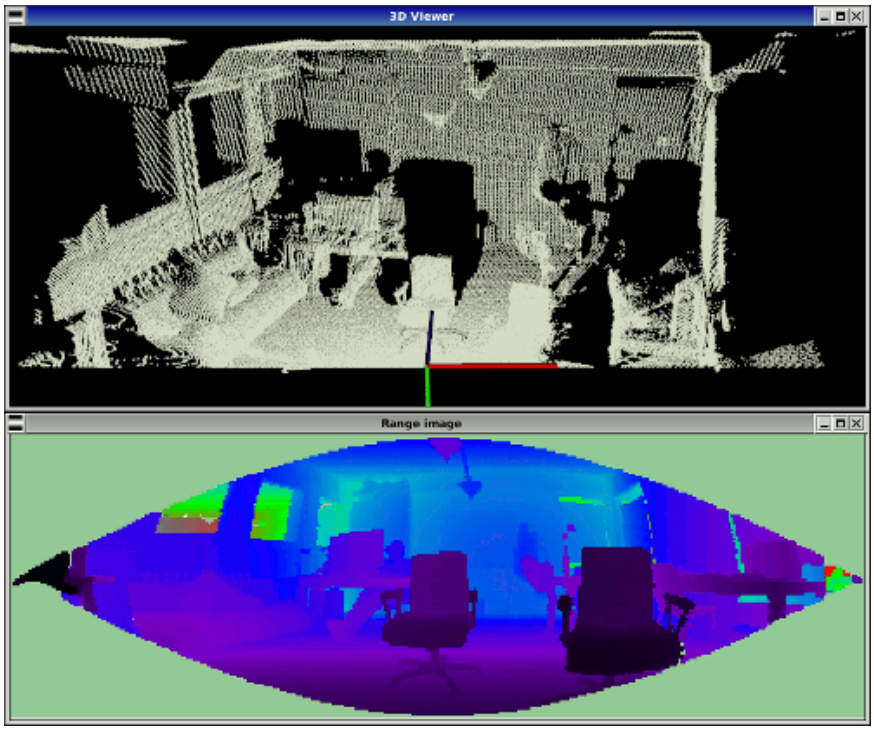


Figure 4.2: Range image example, Source: PointClouds.org (PCL)

B. Steder et al. [33] introduce a novel interest key point extraction method which operates on range images produced from unorganized 3D point clouds. Range image explicitly take in consideration the borders of the objects identified by transitions from foreground to background, which is also very robust against noise and changes in resolution. This is done by considering the change in the distance between neighboring points as a border indicator, all the point in the image are checked at its local neighborhood:

- To determine the 3D distance to neighboring points that are within the range defined by a border; a heuristic is adopted.
- a score of the possibility is calculated that a point belongs to a border by the maximum 3D distance obtained from the last step;
- deciding on the type of the border is decided for the point that falls into using a *Border Criteria*;
- To locate the border in the image a non-maximum suppression is performed.

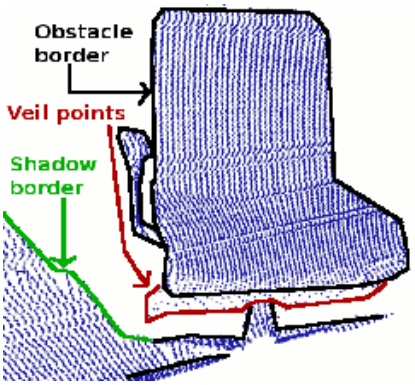


Figure 4.3: Border extraction in Range image example, Source: PointClouds.org (PCL)

Three kinds of border points are consider in a range image: object borders are the outermost visible points belonging to an object, shadow borders are points in the background that hook up occlusions, and veil points are interpolated points between the object border and the shadow border, which are a usually produced in 3D range data obtained by LIDAR's.

This method is also implemented in Point Cloud Library (PCL) [32].

4.3 RGB-D Edge Detection

To detect reliable 3D edges, geometric shape information from the depth channel and photometric texture information from the RGB channels are considered. Choi et al. [34] proposed a novel method to detect 3D edges for organized point cloud. Depth discontinuities and high curvature regions are exploit to search for prominent geometric edges, RGB image is used to detect 2D edges, which are back-project to get 3D points. As the point clouds from RGB-D devices are organized like images (rows and columns), search is done for neighbor with the row and column indices rather than carrying out a time-consuming 3D search, since is needed for general unorganized point clouds.

Occluding and occluded edges

Three types of edges are detected from the depth discontinuities: occluding, occluded, and boundary edges. In a given point cloud, occluding edges are depth discontinuous points on foreground objects while occluded edges are depth discontinuous points on the background. A local 8-neighbor search is performed to determine these edge points, and the maximum depth difference is calculated between the current location and local neighbors.

RGB and High Curvature Edges

As RGB-D cameras not only provide depth data but also united RGB data. Therefore 2D edges cab be detected from RGB data. Then Canny edge detector [1], due to its good localization and high recall, is applied to find the 2D edges.

When surface normals change abruptly, such as a corner where two partitions meet, high curvature edges occurs. A variant of Canny's edge detection [1] is applied to detect high curvature edge.

Figure 4.4 illustrates edges detected through RGB-D Method, occluding (green), occluded (red), boundary (blue) edges, RGB edges (cyan) and high curvature edges(yellow).

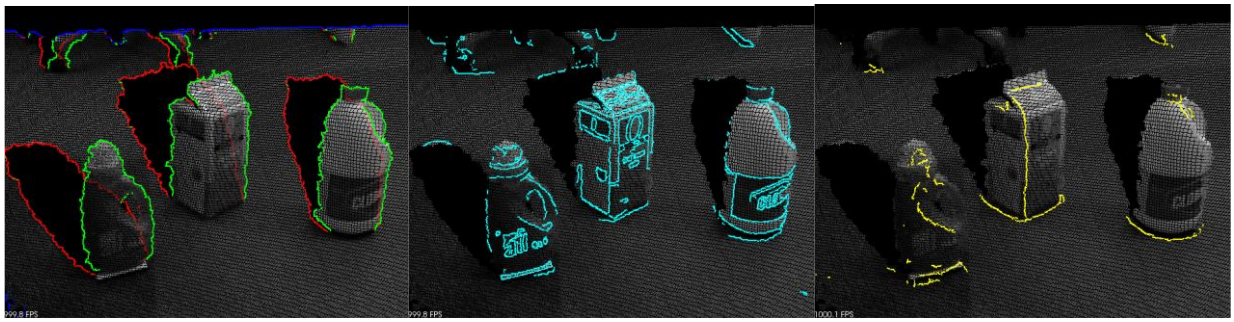


Figure 4.4: RGB-D Edge Detection example, Source:PointClouds.org (PCL)

Chapter 5

Methodology

In this chapter, an illustration is given regarding the workflow of the EdgeScan method (proposed algorithm) and then explanations is given of all the steps in algorithms for image pre-processing, edge detection, pixel mapping and data fusion from 2D images into the point cloud data.

5.1 Hang-up Robot

Penguin ASI Canada has a self-contained, tele robotic system for the clearing of hang-ups in their caving operations. This system will eliminate the manual placement of explosive charges and the associated hazard to personnel that currently exists with this operation.

5.1.1 Description

The system consists of a Transporter/Control Center, Robot and unique communications system. The Transporter is a Normet Ltd. manufactured RBO service vehicle with Penguin supplied control and communications systems for the Robot

Figure 5.1: Hang-up Robot, copyright, Penguin ASI

The Transporter with an attached trailer, used for the transfer of the Recovery Robot to the underground working area. Thereafter, the trailer is detached and the RBO Control Center is used to direct the Robot in its clearing of the hang ups. Both RBO and trailer have fully certified braking and operating systems for use underground.

5.1.2 Operation

After the unloading of the Robot from the trailer at the draw-bell access drift and the trailer removal, the operator in the RBO Control Center will position the communications devices such that the Control Center is remotely located relative to the draw-drift.

The Robot will then be navigated to the draw-bell where it will deploy booms and stabilizers, scan, image and view possible keystones. Upon transfer of this data to the Control Center, and to the mine engineering office should mine communications permit, a keystone is selected. The operator will then position the drill booms and an End Effector for drilling and placement of the explosive charge, or placement of an explosives pack, retreat the Robot, deploying the blasting wire back to the RBO Control Center area, and position the Robot and communications devices remotely from the access drift. The operator will then transfer the blasting wire into the safety of the Control Center where the charge will be detonated. The Robot will then be sent back into the draw-bell to review the effect of the blast and if necessary repeat the process until the clearance of the draw-bell is achieved.

5.1.3 Robot Operations and Communications

The Robot will be powered by two Lithium Iron Phosphate batteries, powering four electric motors two of which have independent failsafe disc brakes. As well there are three individual stingers to ground. The drive motors transmit power to four external, enclosed chain drive gearboxes each having two solid rubber tires. There are no hydraulic power systems required.

The Robot will be deployed and data transmitted via an optical communications system. Imaging of the draw-bell will be both laser and optical. Navigation of the Robot and boom positioning will be by means of an Inertial Navigational System. Two integral booms will be deployed for addressing both low and high reach drilling with an End Effector at the end of the

second boom. The End Effector will consist of a rotary drill, stinger, and an explosive placement device.

5.2 Point Cloud Data

The scanned data used in this thesis is acquired from NORCAT, Sudbury, ON. This data was captured by Penguin Automated Systems Inc. during the testing of their Hang-Up Robot, which has been discussed in section 5.1. It is a large point cloud having 1141,000 points and covers around 30 meters of the area of the mine. It only contains XYZ 3D data. No color or intensity information is present in the point cloud data.

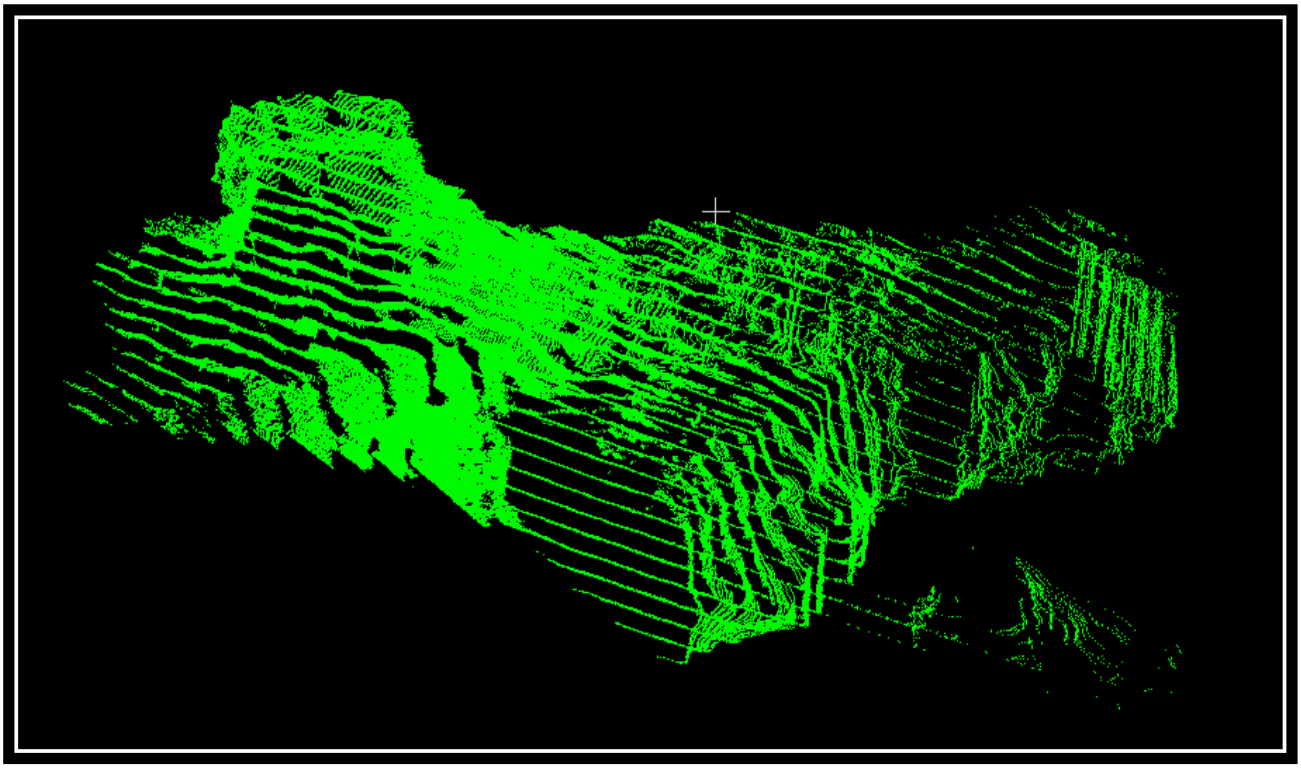


Figure 5.2: NORCAT Scan Data, Original, Copy right, Penguin ASI

Another set of data is taken from the laboratory of Penguin ASI, where the robot was build and tested. It is also large point cloud having 1141,000 points. It also contains XYZ data information.

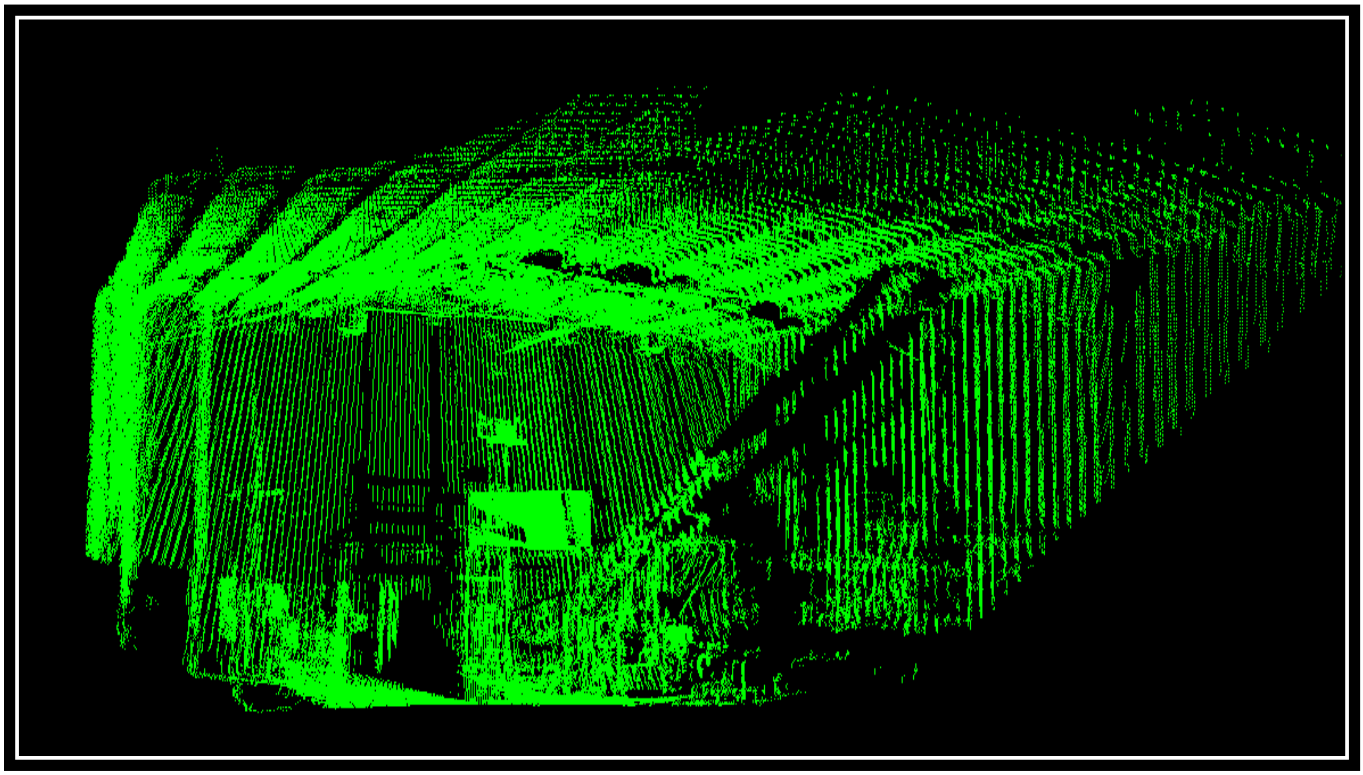


Figure 5.3: Laboratory Scan Data, Original, Copy right, Penguin ASI

5.3 EdgeScan Method (Proposed Method)

As mentioned in the introduction, this thesis introduces a novel real-time algorithm, EdgeScan method, to detect edges in an unorganized 3D point cloud, which leads to detect rocks in mine.

EdgeScan method works on scanner raw data. Three main steps are involved in this approach to identify the edges in a point cloud. Scan data acquiring, on one hand converting scan data into 2D image and on other hand converting scan data into 3D point cloud. Apply Canny edge detection method on 2D image for extracting the edges. Apply a mechanism for mapping 2D edges pixels into 3D point cloud and forming 3D edges. An overview of all the steps is illustrated in the Figure 5.4

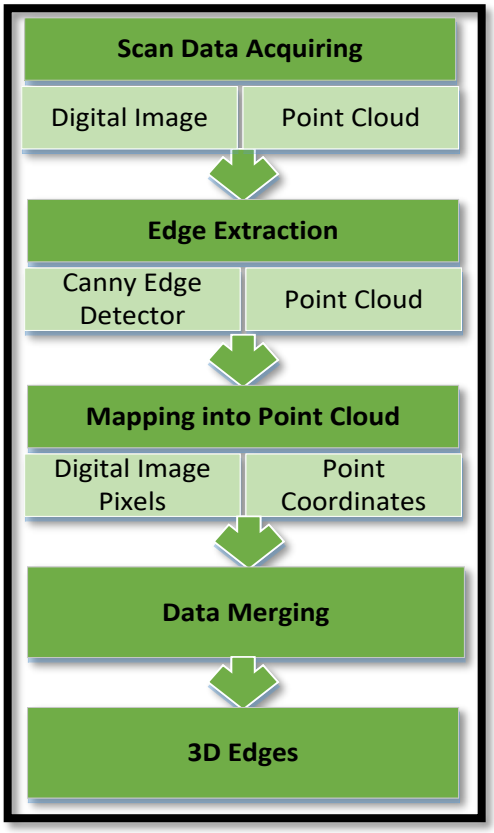


Figure 5.4: Workflow of the algorithm, Merging 3D Point Cloud and 2D Image

5.3.1 Scan Data Acquisition

The first step in this approach starts with scanned data. Scanned data is converted into Polar

Coordinates and then ultimately into gray level 2D image. On the other hand, scanned data is converted into Cartesian coordinates and then eventually into 3D Point cloud.

5.3.1.1 Scanner Working

The scanner used in this thesis is SICK LMS 511, spherical laser scanner, with acquisition frequencies of up to 100Hz and operation range to 0 - 40 meters. The scanner measurement generates a set of distance data from the sick center of rotation and the nearest object in XZ- plane. For example, a 0.5 degrees resolution of the SICK generates 1141 data points (0…1140) over 190 degrees (-5…95). The Stepper motor is used to rotate SICK scanner at 0.18 degrees per step to consider Y-plane to obtain 3D presentation. Figure 5.5 shows the range of SICK LMS scanner.

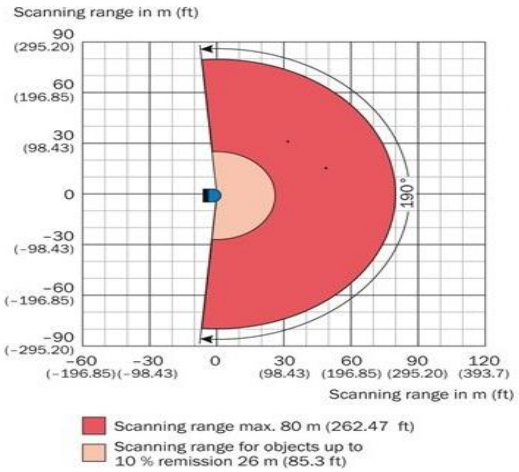


Figure 5.5: Scanner Operating range diagram, Source: SICK AG.

5.3.2 Polar Coordinate.

In two dimension, Polar coordinate system specify the position of a point *P* in a plane by distance *r* from the origin and θ angle made between the line segment from P to origin. Polar coordinates (r, θ) for point *P* has been elucidate in the Figure 5.6.

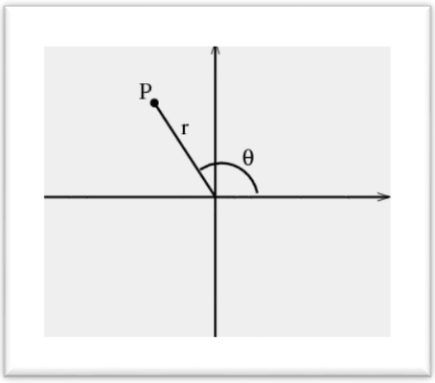


Figure 5.6: Polar coordinates diagram, Source: *mathinsight.org.*

Where r can range from 0 to infinity. θ ranges from 0 to 2π.

5.3.3 Convert Scanned data into 2D Image

In a black and white digital image, gray level is the color depth of a point / pixel which ranges from 0 to 255. 0 denotes black color whereas 255 represents white. The Mach bands effect states that edges are the areas corresponding to dramatic gray value changes in a range image [40].

Therefore, edge points are identified according to the intensity in gray value changes. The 2D gray scale image is generated from scanned data directly. A 2D image conversion includes the following steps:

- Determination of the size of 2D image.

  Scanner collects 1141 points over 190 degree and stepper rotate 180 degrees that is 0.18 per degree step generating 1000 steps. This determine the size of the image 1141 * 1000.

- Determination of the position of scanning point in the 2D image.

  Now each scan point holds the corresponding position in 2D image.

- Calculating pixel gray value of each scanning point corresponding to the 2D image.

$$c = 255 / (S_{max} - S_{min})$$

$$\text{Depth (i)} = Si * c$$

where Depth is gray value. Si is the distance from a point to the scanner center. S_{max} and S_{min} are the maximum and minimum distances from points to the instrument center. c is the constant.

After converting scanner data into gray level, image is saved in any image format .png, .bmp, .jpeg etc. Figure 5.7 illustrates scanned data and gray level data.

Figures 5.8 and 5.9 demonstrate the converted gray level 2D images of the scanned data of NORCAT and laboratory sites.

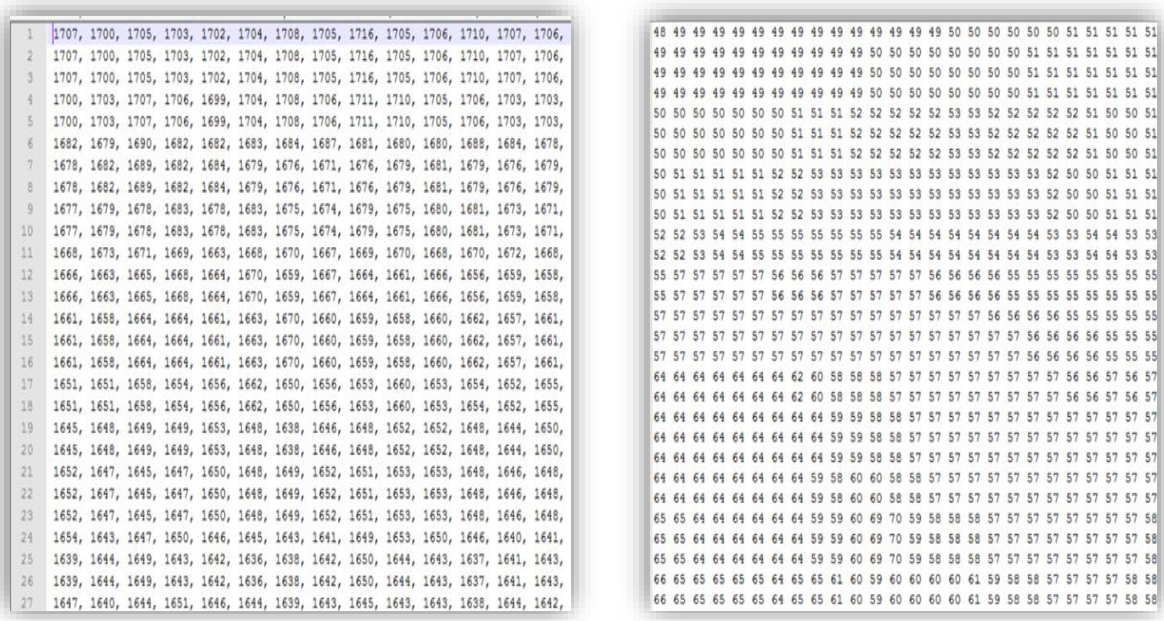


Figure 5.7: Scanned data (left), Gray level data (right)

Figure 5.8: NORCAT scan data, 2D range image in Polar Coordinates

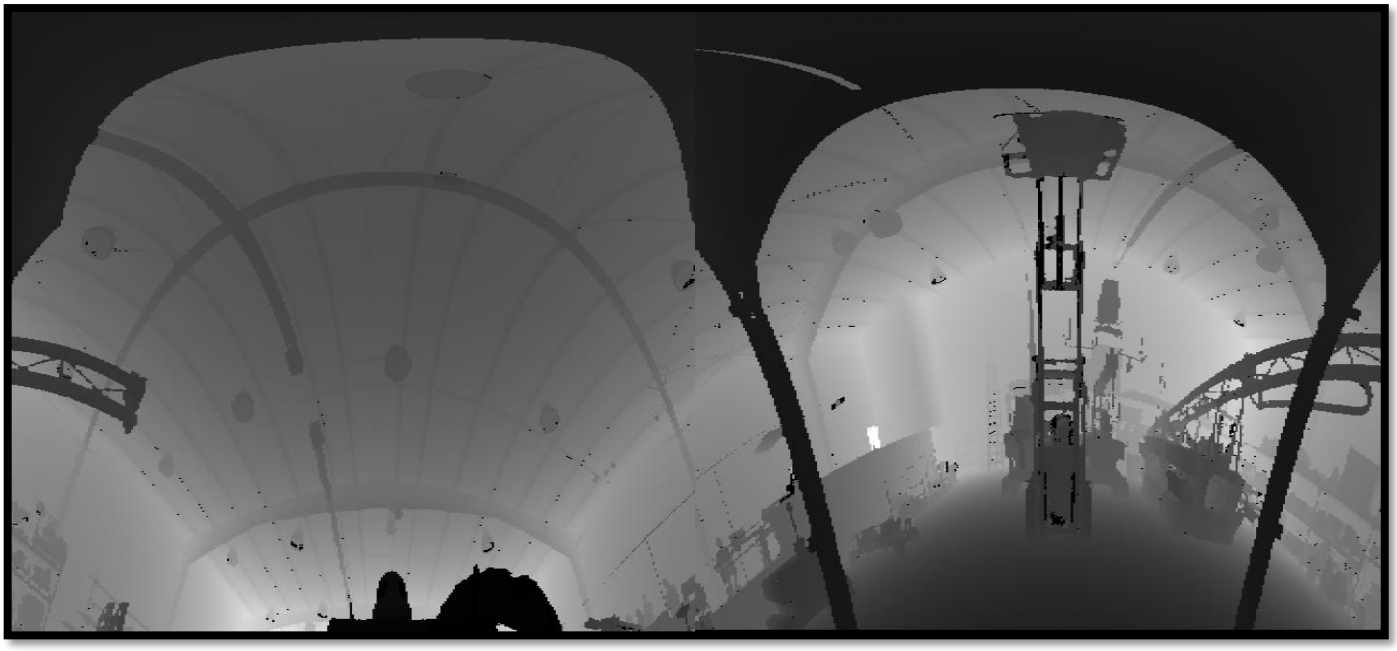


Figure 5.9: Laboratory scan data, 2D range image in Polar Coordinates

5.3.4 Convert Scanned data into 3D Point Cloud

A 3D point cloud data is generated by first converting scanner data into Polar coordinates and then finally into Cartesian coordinates XYZ. Precisely for Hang-up software, a 3D point cloud need to be generated from the scanner data, whose XYZ coordinates represent world coordinates. TALIN and Inclinometer are used to get the accurate GPS position. A relationship is built between scanner data and GPS to get a point cloud which embody the world coordinates.

TALIN

TALIN (Tactical Advance Land Inertia Navigator) is a flexible, reliable, best-value INS/GPS navigator for combat vehicles. It provides accuracy and precision navigation for GPS-denied

environment. Fundamentally, it is an inertia navigation system with high-accuracy ring laser-gyro and accelerometers for unparalleled performance. However, it also acted as a Zero Velocity Updated (ZUPT) Aided Dead Reckoning system, Dead reckoning is a form navigation whereby the current position of a moving vehicle is inferred by knowing speed and direction of travel since the last known position.

Figure 5.10: TALIN device (left), World coordinates example (right), Source: *Honeywell*

It has multiple accuracy configurations. It's thermal operating rate -46°C to 71°C. It has 3-axis inertial sensors (internal), VMS, DAGR, PLGR. Points presents world coordinates as Northing, Easting and Elevation. The TALIN (INS) measurement gives the orientation Θ, ψ, Ø (pitch, roll, yaw) and displacement (X, Y, Z) of the robot with respect to a fixed reference point (assuming it is earth center that it is referenced to). Figure 5.10 shows TALIN device.

Inclinometer

It is an instrument for measuring angles of slope and inclination of an object with respect to its gravity by creating an artificial horizon. It is also known as a tilt sensor, tilt indicator, slope meter, slope gauge, gradient meter, gradiometer, level gauge & level meter.

The 2 DOF (X-Y plane) inclinometer is placed at the end of the lower boom to measure tilt angle of the SICK scanner. The Stepper motor is to rotate SICK scanner at 0.18 degrees per step.

With all these entities together, set of equations are created that can transform the SICK scanner data points into useable 3 dimensional data.

- Convert scanner data into XYZ coordinates by converting polar coordinates into Cartesian coordinates.
- Convert Sick Scanner coordinate to Inclinometer Coordinate that with same origin but different orientation.
- Transform Position in Sick Coordinate origin to a new coordinate with identical orientation of Robot coordinate.
- First convert robot coordinate to TALIN vehicle Coordinate.
- Finally, convert Talin coordinate to world coordinate.

  Eventually, by applying all the aforementioned steps the 3D point cloud is generated.

5.4 Edge Extraction

In this step, 2D image is loaded and one of most popular, reliable and edge detection algorithm of image processing, “Canny Edge Detection” is performed on it. Canny edge detection is already discussed in detailed in section 3.3.8. The steps involved in this process are:

- Load 2D image.
- To reduce the noise, a Gaussian kernel of size 3x3 is applied.
- Established a ratio of 1:3 for lower: upper threshold values.
- Different lower threshold values are applied and examined and the most appropriate threshold values are selected.

After applying Canny's algorithm, edges are generated as shown in Figures 5.11 and 5.12. These images only show extracted edges on the black background of NORCAT and laboratory scanned data respectively.

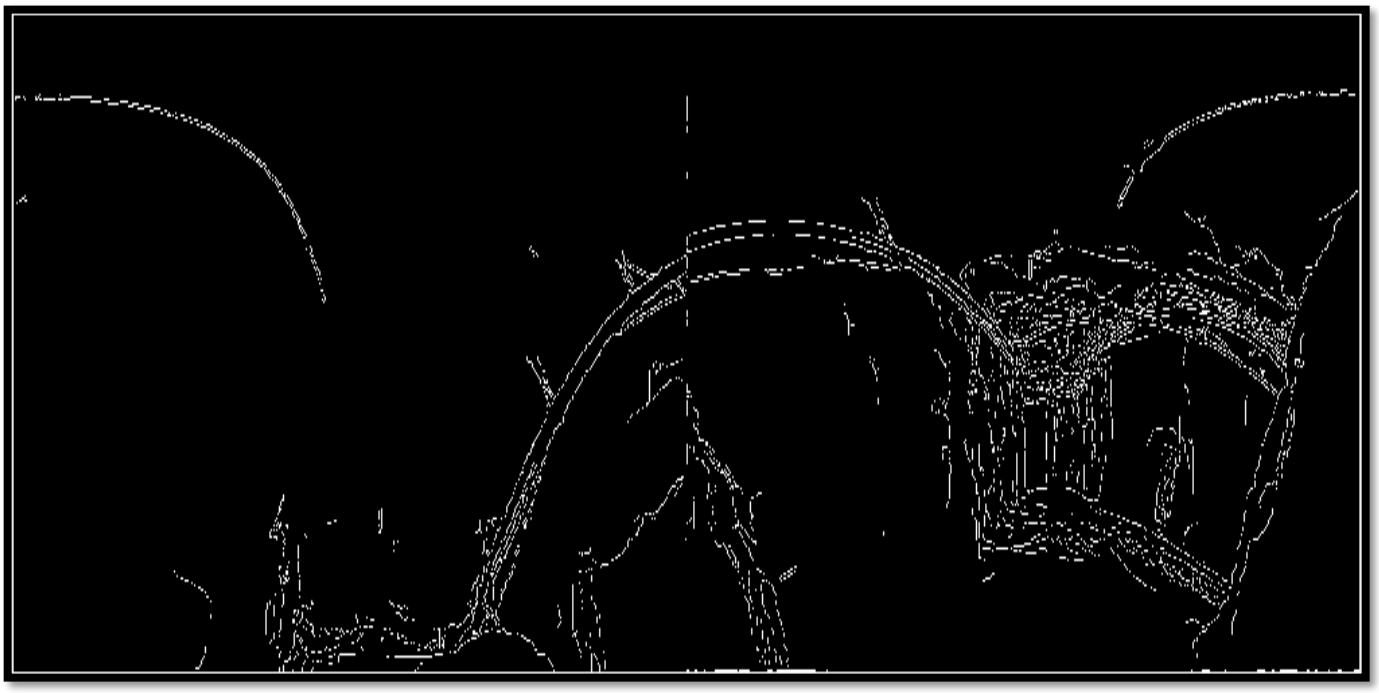


Figure 5.11: NORCAT scan data edge detection in 2D image

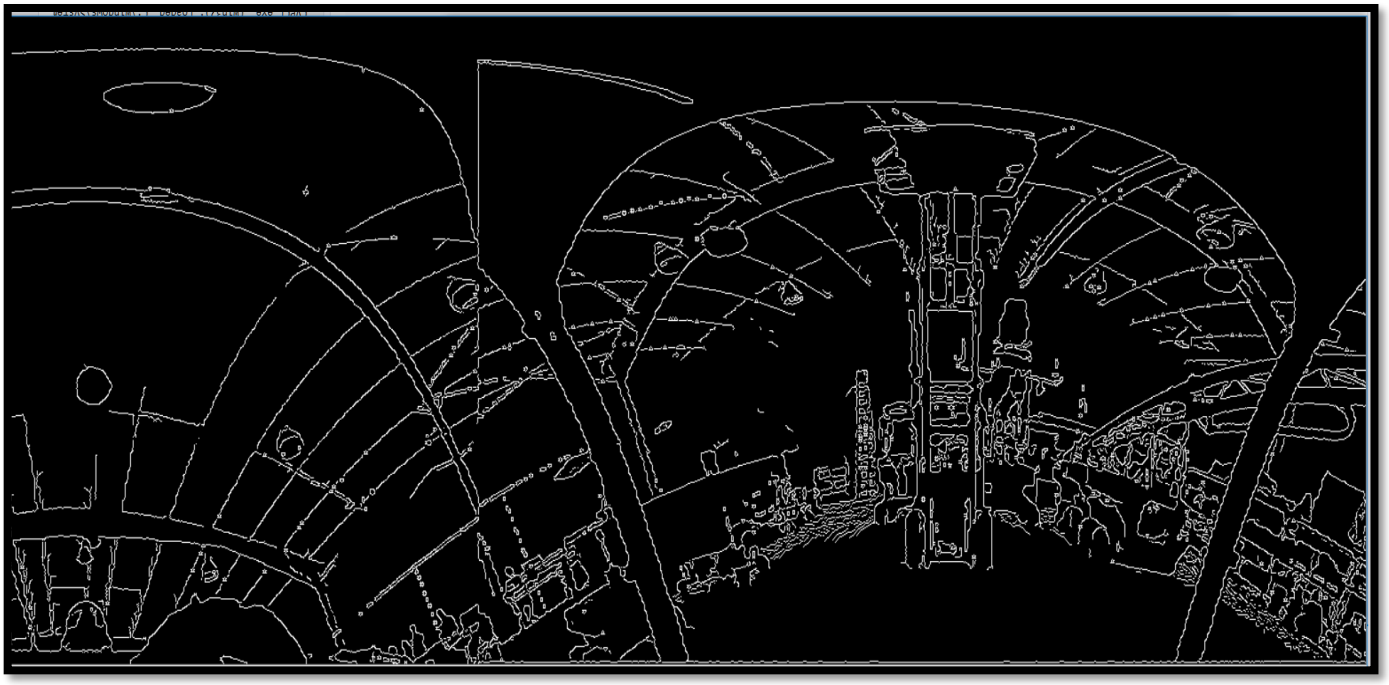


Figure 5.12: Laboratory scan edge detection in 2D image

5.5 Mapping 2D Image Edges into 3D Point Cloud

Third and important step is mapping the 2D image pixels and 3D point cloud coordinates. The main idea of correspondence between 2D image and 3D point cloud is that, as they are obtained from the same source, scanned data, so they share the identical indexing.

- Each pixel in image map is examine whether it is detected as “edge” or not. If edge pixel is found then its index is used to identify the same index to the corresponding position in the 3D point cloud.

  Point cloud [row(k)] = Pixel(row (i) * no of cols + col(j))

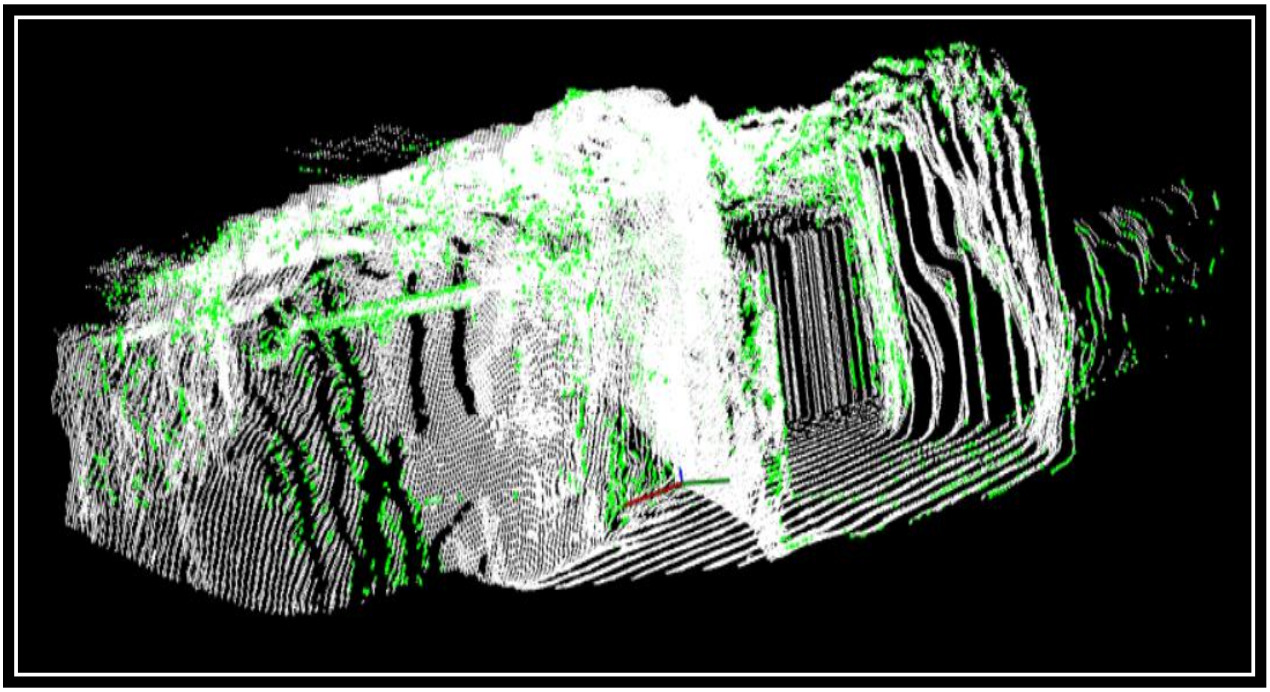


Figure 5.13: NORCAT scan data edge detection in 3D Point cloud

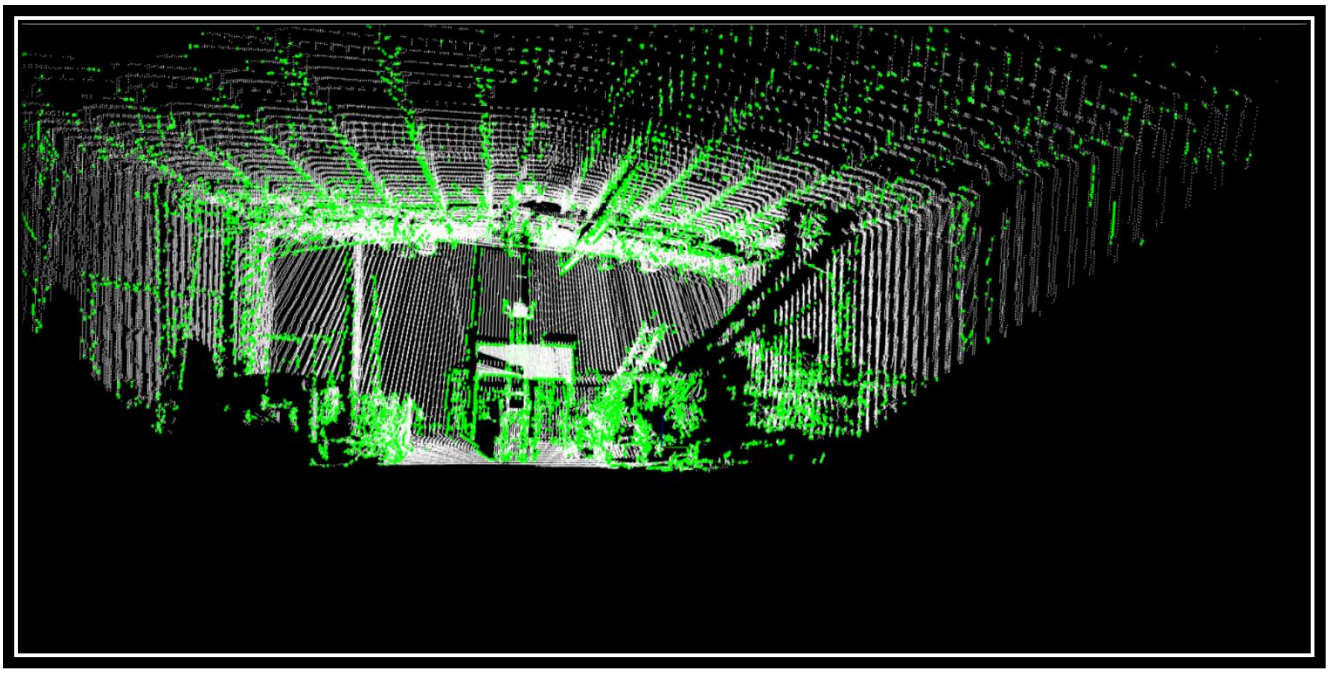


Figure 5.14 Laboratory scan data edge detection in 3DPoint cloud

Figures 5.13 and 5.14 illustrate the mapped edges in the 3D Point clouds of NORCAT and laboratory correspondingly.

5.6 Software Implementation

Algorithm is implemented in visual C++ using visual studio 2015. OpenCV [42] and Point cloud library [41] have been used for the functionality. OpenCV is used for image processing section, edge detection from 2D image whereas PCL is used for 3D point cloud operations on point cloud data and visualization. An integration of OpenCV and PCL libraries was made to implement the algorithm.

Chapter 6

Results and Analysis

In this chapter, results of the proposed method, using steps introduce in chapter 5, are presented and an analysis on them is performed. Different edge detection algorithm for image processing are also compared. A comparative study on two different methods discussed in chapter 4 is also conducted.

6.1 Analysis on Edge Detection Methods for 2D Image

In this thesis, edge detection for 3D point cloud is achieved by converting scanned data into 2D image and then applying Canny's algorithm to extract the edges. Although, different edge detection algorithm are elaborated in chapter 3 in detail, however some of the methods are applied on the dataset present for the thesis to analyze the results.

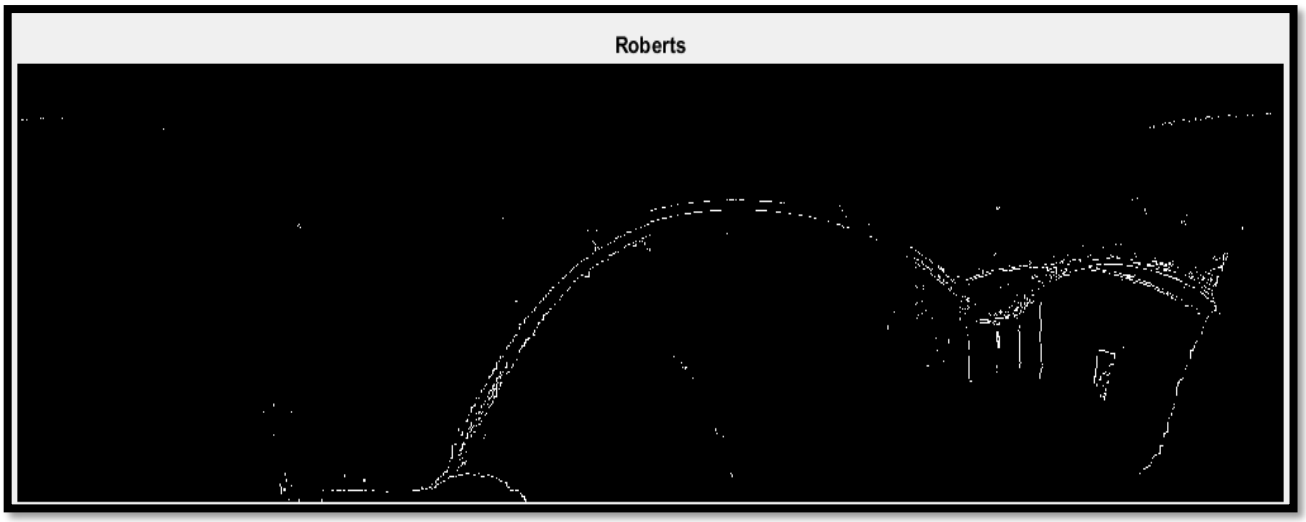


Figure 6.1: Robert edge detection on 2D image

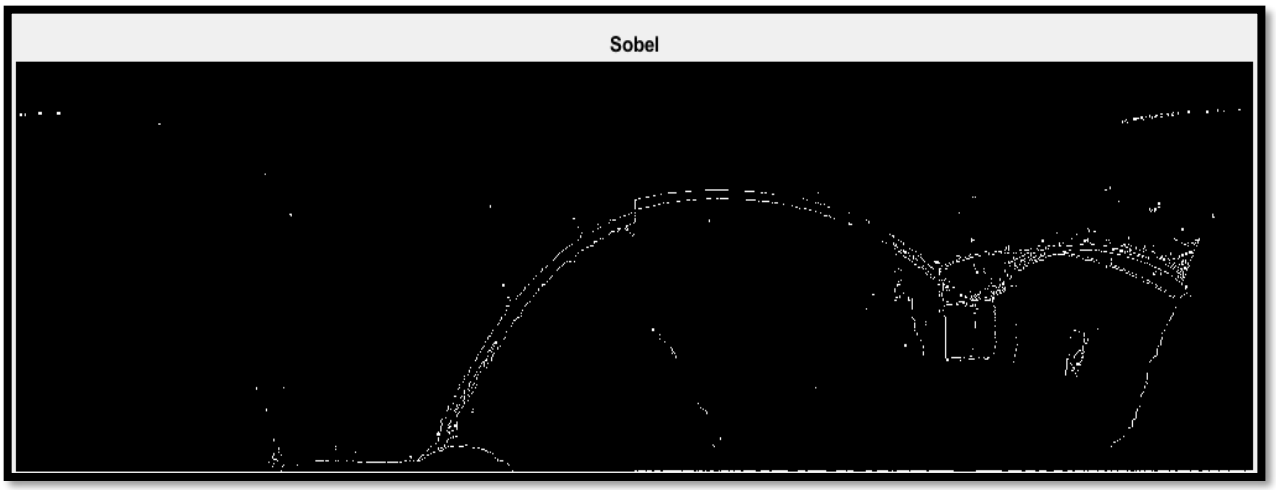


Figure 6.2: Sobel edge detection on 2D image

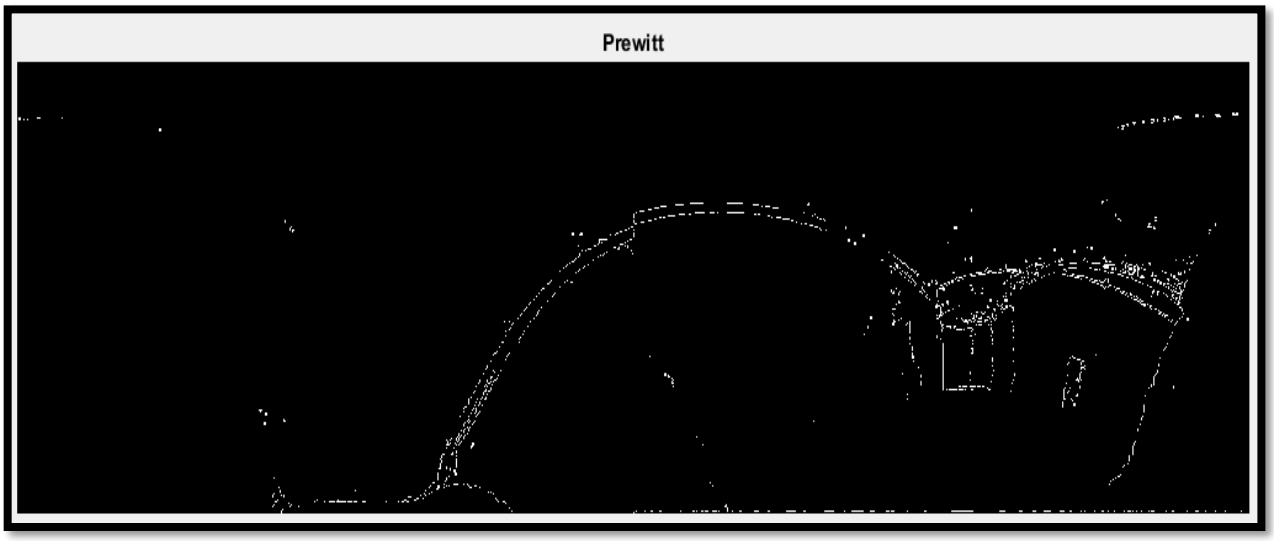


Figure 6.3: Prewitt edge detection on 2D image

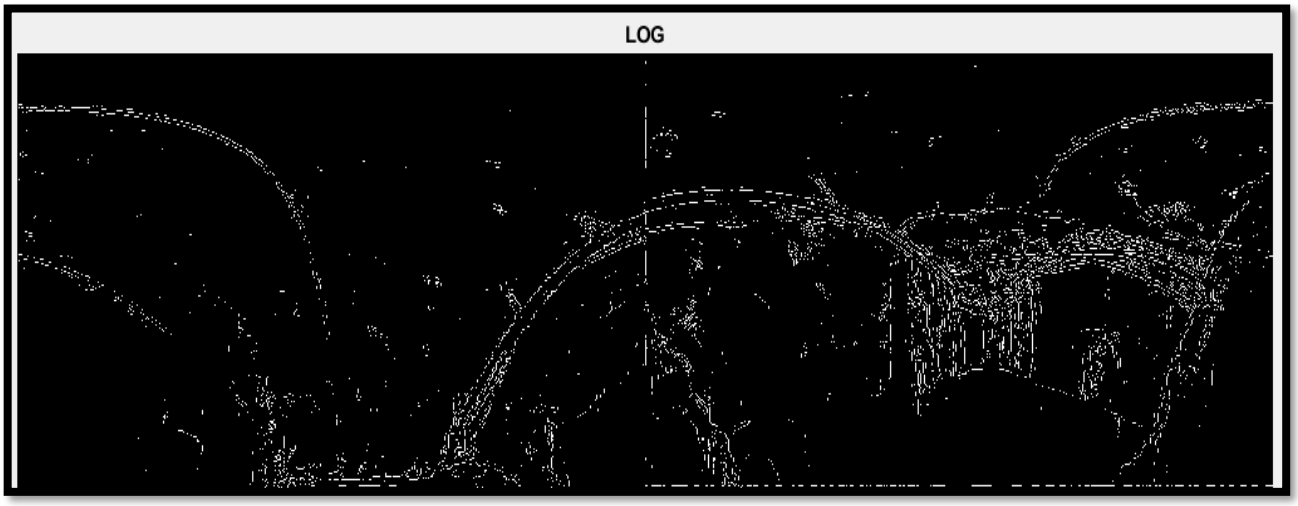


Figure 6.4: LoG edge detection on 2D image

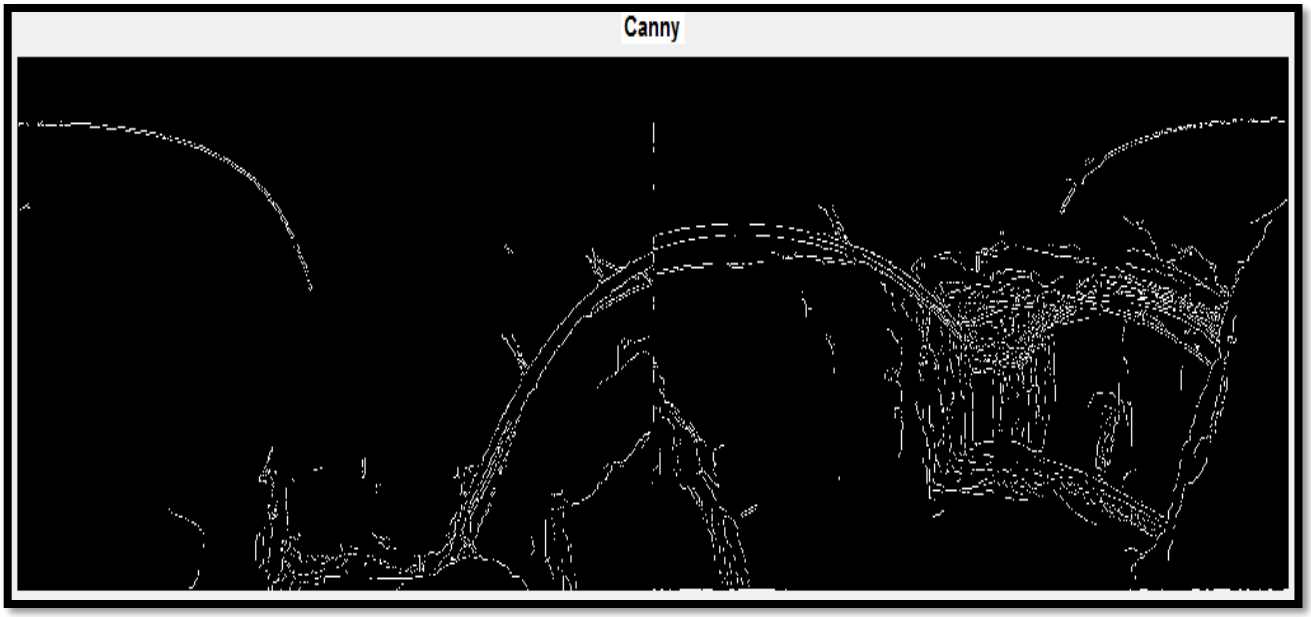


Figure 6.5: Canny edge detection on 2D image

Figures 6.1, 6.2, 6.3, 6.4 and 6.5 demonstrate the implementation of Roberts, Sobel, Prewitt, LoG and Canny algorithms respectively. The edge detection techniques were implemented using MATLAB R2009a, and tested with an image (NORCAT). The purpose is to develop a clear edge map/overview by extracting the principal edge characteristics of the image.

Visual comparison of these images gives help in qualitative evaluation of these edge detectors. Roberts, Sobel and Prewitt algorithms are sensitive to noise thus produce inaccurate results. LoG and Canny produced more or less alike edge map. since, different edge detections work better under different conditions, but Canny exhibits superior performance. Canny result is much useful for the contour calculation which is often used for pattern recognition. The figures indicates that, Canny result is superior by far to the results of other techniques.

6.2 Results and Evaluations

The EdgeScan method is implemented and tested on the two data sets mentioned in section 5.2.

The results are evaluated both qualitatively and quantitatively. For simplicity NORCAT dataset is refer as Site1 and laboratory as Site2.

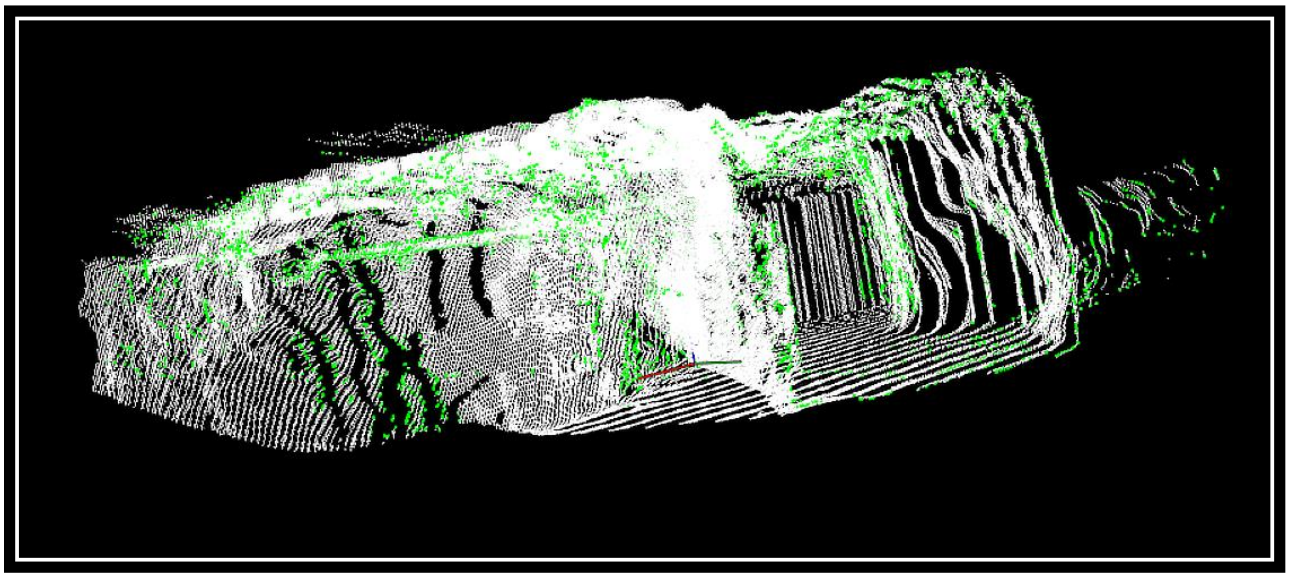
6.2.1 Results

Figures 6.6 and 6.7 demonstrate edge detection results of EdgeScan method for site1 and site2 respectively. In 6.6(a) and 6.7(a) edges overlaid on original point cloud are present. The point clouds are in white color and the detected edges in green color for the sake of clarity. It can be seen from figures 5.2 and 5.3, original point clouds show a rough out-line of the objects. Edges are not clear as noise is present in the point cloud. Elimination process of noise often loses some of edge information. Therefore an efficient and effective method, Canny algorithm is implemented which not only de-noises the image smoothly but also extracts the edges accurately.

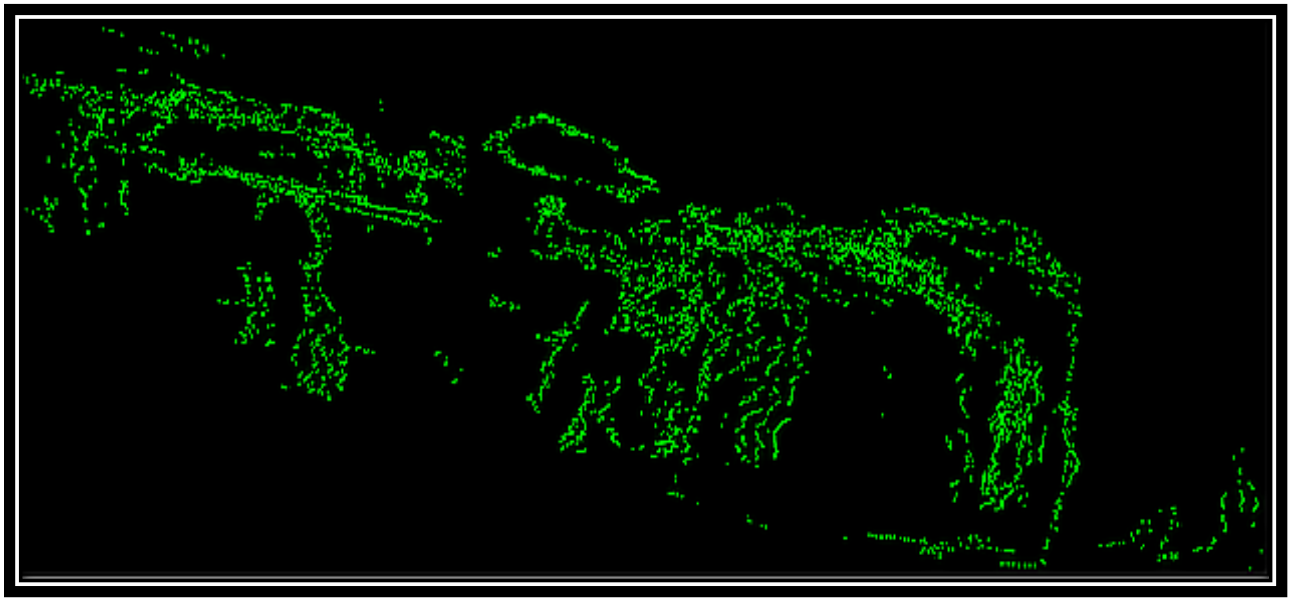
Although site1, which is the inner projection of a mine, does not contains too much pointed objects but one can easily detect the ventilation pipes, tunnel entrances and rocks. On the other hand, site2 encloses internal environment of the laboratory. It contains number of objects which have concrete shapes.

It can be clearly seen that almost all the edges are detected in figures 6.6 (b) and 6.7 (b).

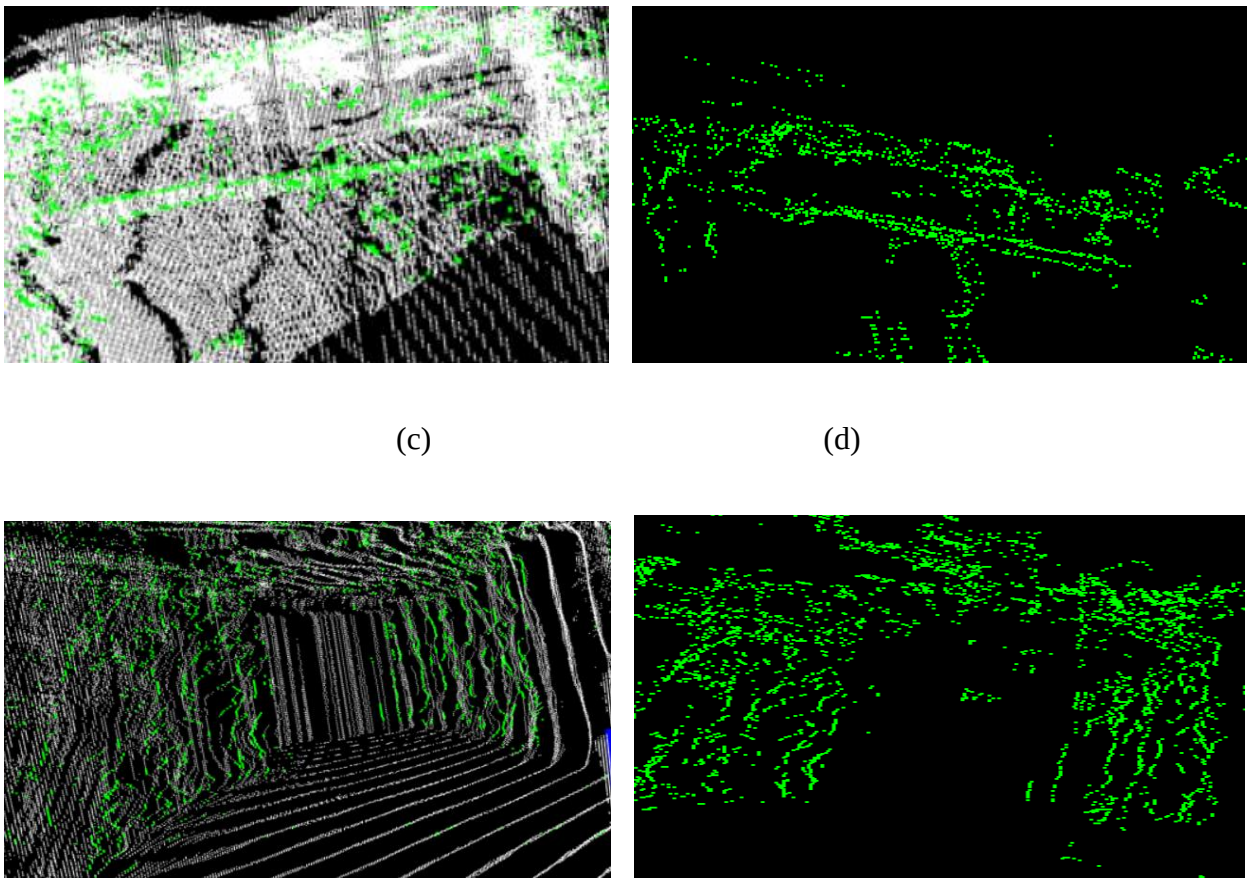
2D edge pixels are accurately mapped into the 3D Point cloud via mapping mechanism. Edges and non-edge points are well separated as shown in 6.6 (a), (b) and 6.7(a), (b). Figures 6.6 (c), (d), (e),(f) and 6.7(c), (d), (e), (f) shows some detail results of site1 and site2 correspondingly.



(a)



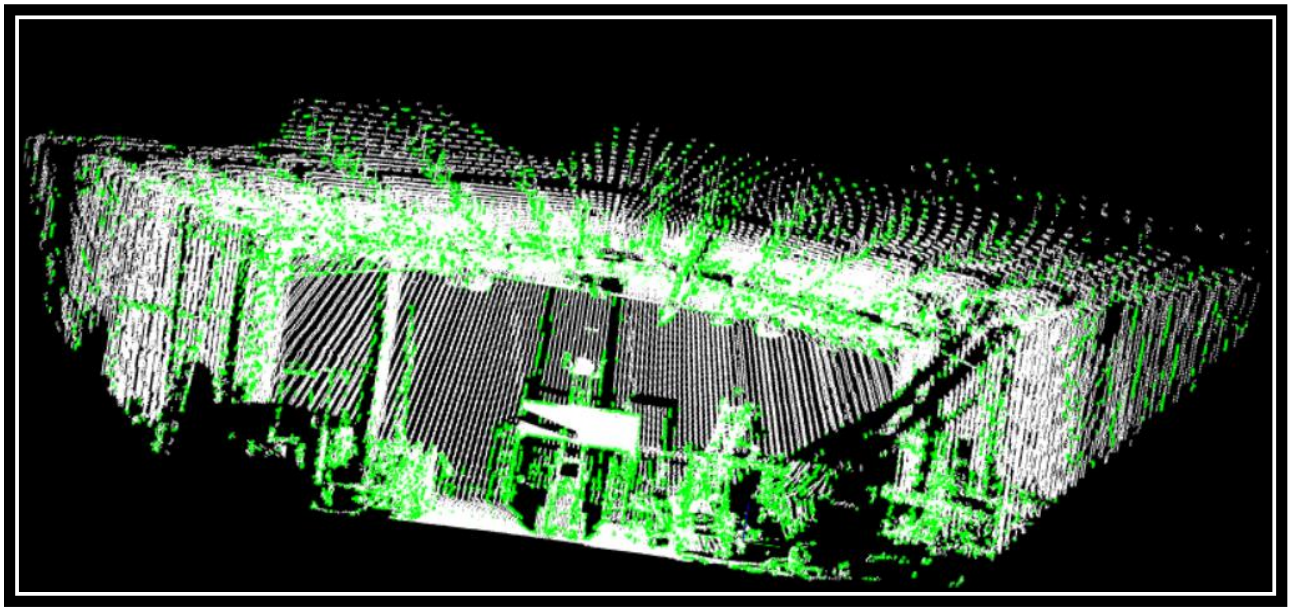
(b)



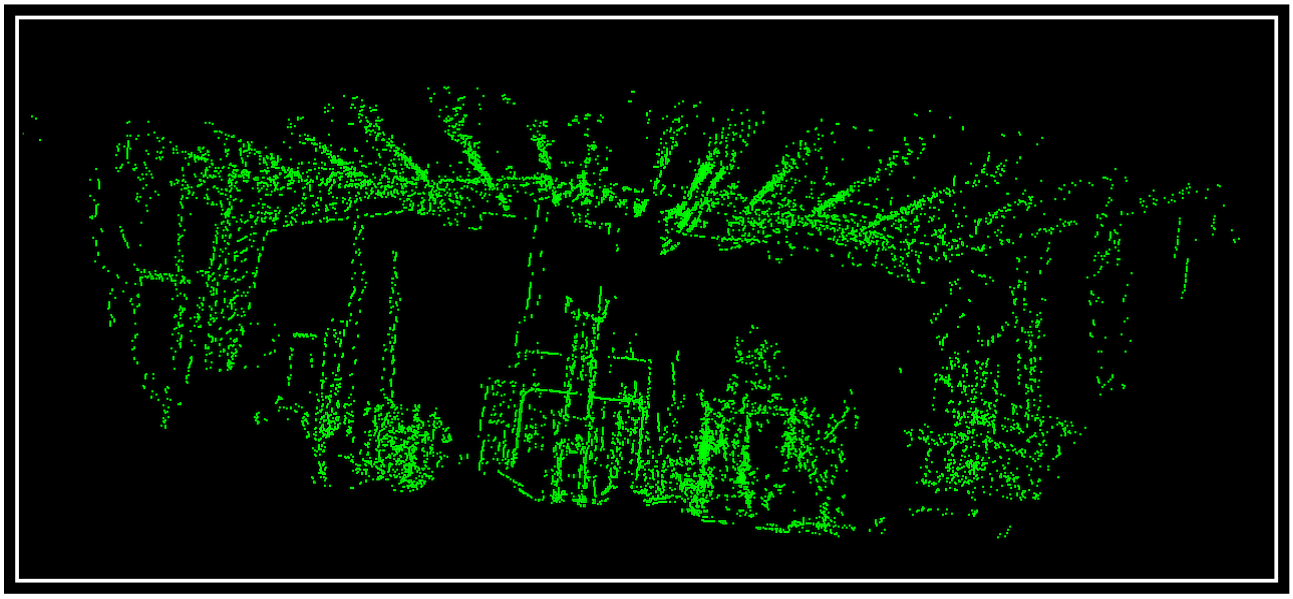
(c) (d)

(e) (f)

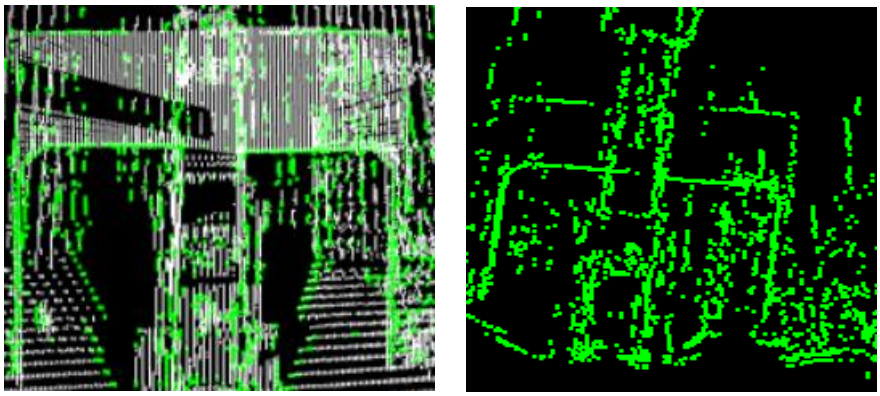
Figure 6.6: Results of site1, (a) edge detection results overlaid on original, (b) edges, (c-f) details to edges



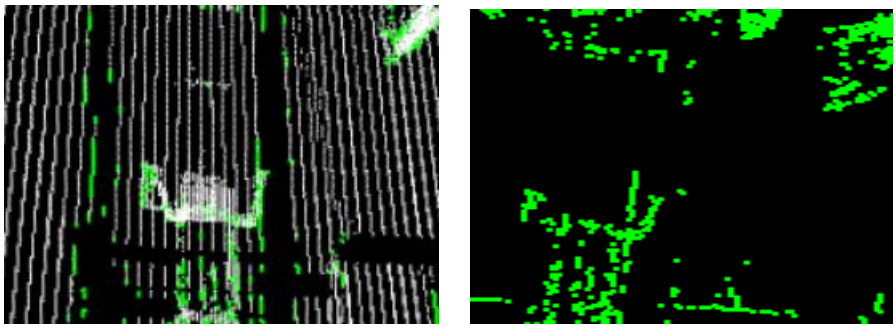
(a)



(b)



(c) (d)



(e) (f)

Figure 6.7: Results of site2, (a) edge detection results overlaid on original, (b) edges, (c-f) details to edges

6.2.2 Running Time

As mentioned earlier in chapter 1, the main purpose of the thesis is to introduce a method which can detect the edges accurately in near real-time. Therefore the time consumed for the process is very important. Table 6.1 shows the average computational time of edge detection for both datasets.

Table 6.1: Time consumed by different steps involved in EdgeScan method

	Site 1	Site2
Image Conversion	62 ms	62 ms
Noise Removal	90 ms	93 ms
Edge detection	16 ms	17 ms
Merge 2D into Point cloud	129 ms	135 ms
Total	297 ms	307 ms

The process is divided into sub steps to show the CPU runtime of each step in the method. According to the table, merging image into point cloud is the most time consuming step among all others. This is because to the size of the point cloud, 1141000 points which is quite dense. Noise removal, image conversion and edge detection takes less amount of time respectively.

Total CPU running time for the EdgeScan method is around 300 ms, which is quite satisfactory for the large unorganized 3D point cloud in real time systems.

The point cloud loading time is not included, since it may vary with respect to software in which it is implemented. The Processor used for the thesis is Intel CORE i7 and RAM is 12 GB.

6.3 Comparative Studies

In comparative studies, not all the methods discuss in chapter 4 are included for evaluation. Only two methods are to be compared to the EdgeScan method. Boundary estimation and Range image methods. Both of the methods are implemented in PCL.

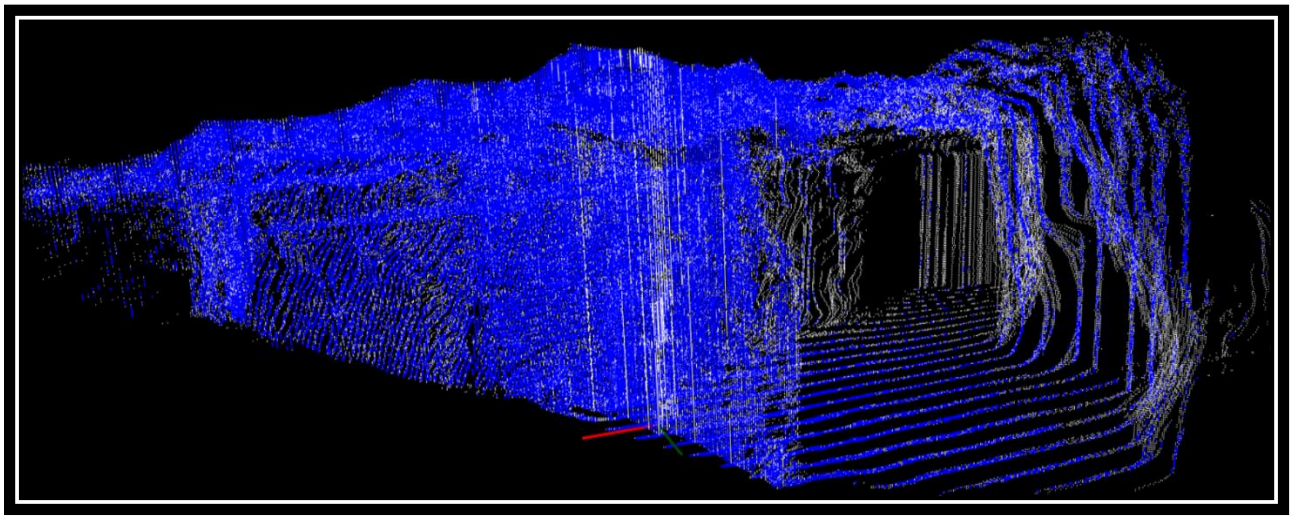
Boundary estimation present in PCL first estimates surface normal at each point of the point cloud and then estimates boundaries by using angle criterion which is discussed in section 4.1.

Range Image method is also present in PCL. The range image is generated from the 3D point cloud and the range border detector is applied to define the borders of the object by transitions from foreground to background as discussed in the section 4.2.

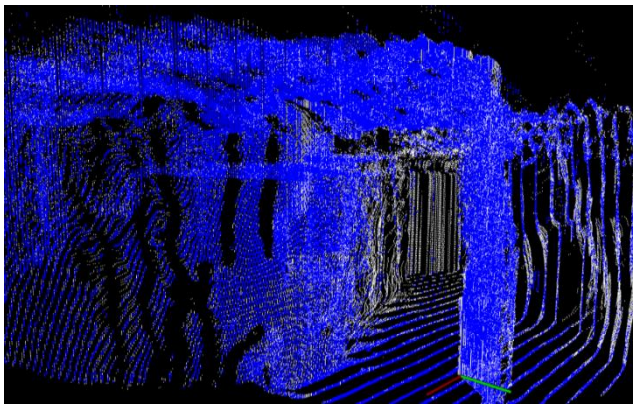
Comparative results of boundary estimation and range image are illustrated in Figures 6.8, 6.9 and 6.10. The point clouds are in white color and the detected edges in blue color for Boundary estimation method. Range image point cloud is in black color and border edges are in green color. The tested outcomes reveals that the EdgeScan method in this thesis can detect all types of edges and is fast and robust as compare to the methods aforementioned.

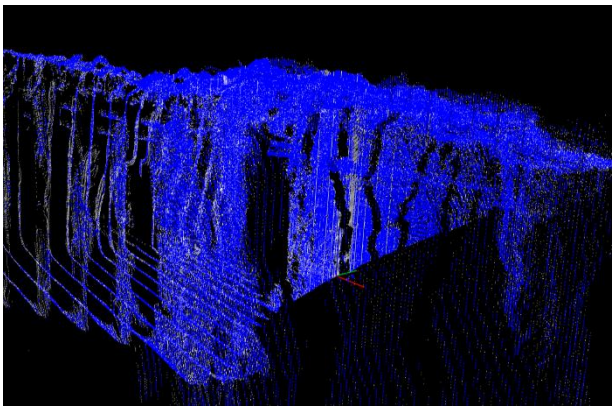
The boundary estimation method is not capable of detecting all edges in the point cloud, the reason behind is that it uses only angle criterion for detecting edges which is not adequate. Furthermore, noise and outliers are also not considered as it uses PCA-Normal for surface edge detection.

In Addition, as demonstrated in the Table 6.2 the CPU runtime for the boundary estimation is extremely high as compared to EdgeScan method. The method estimates surface normal and uses Kd Tree search for neighborhood point search which are time consuming process. Furthermore, runtime also varies with respect to different nature of dataset. Consequently, Boundary estimation method is not appropriate for the complex vicinity and real time systems.



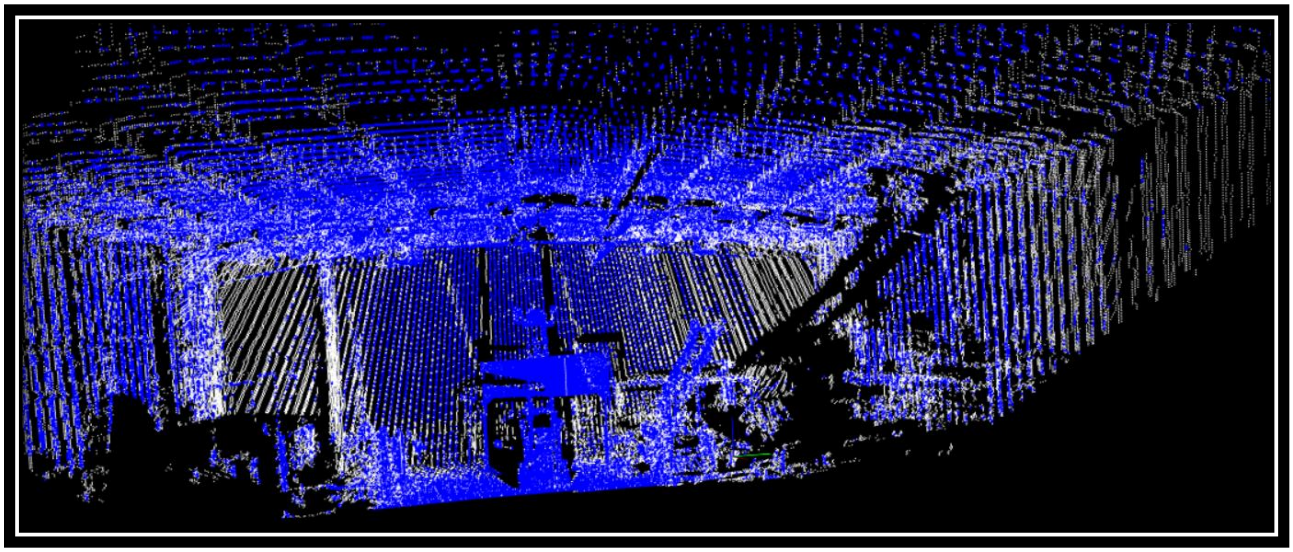
(a)



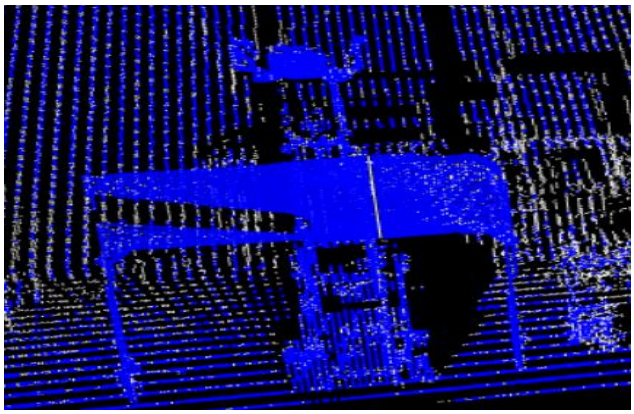


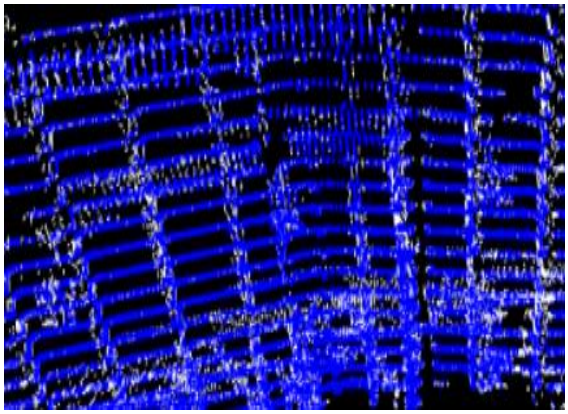
(b) (c)

Figure 6.8: Boundary estimation results of site1, edge detection results overlaid on original



(a)



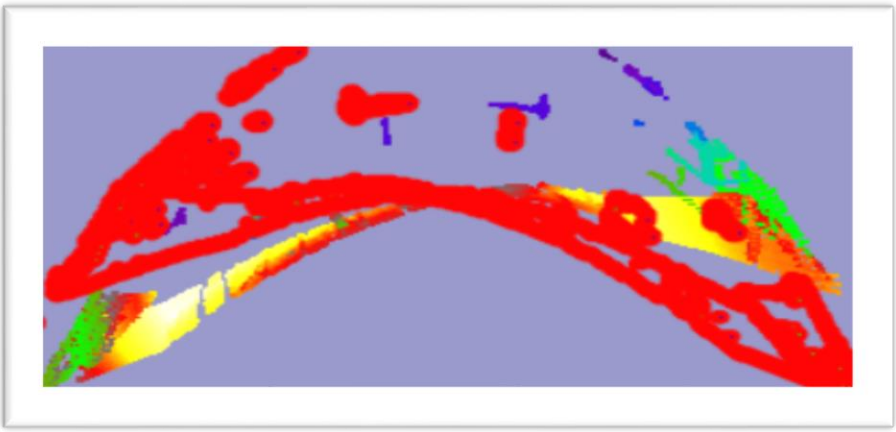


(b) (c)

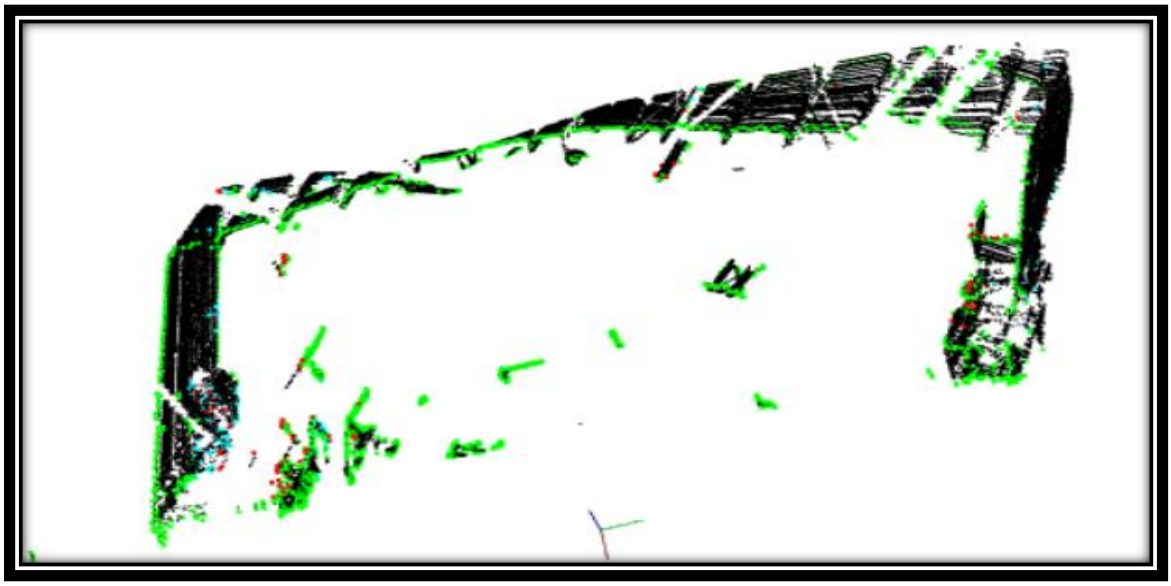
Figure 6.9: Boundary estimation results of site2, edge detection results overlaid on original

Alternatively, Range image method shows better performance and its processing time consumption is also fast. But drawback of this method is that it does not support large dataset. It didn't work for the whole point cloud dataset used in the thesis. The Range image method has been applied on a subset of the point cloud data having 83,000 points to evaluate the results. Moreover the range image can only be generated for data having small range. It is also not suitable large

scene. It is more appropriate for point clouds captured from RGB-D cameras having low resolution.



(a)



(b)

Figure 6.10: Range Image method results of site2, (a) Range Image (b) edge detection results overlaid on original

Table 6.2 Time consumed by different Methods

	Site 1	Site2
Boundary Estimation	1852880 ms	4762470 ms
Range Image	4510ms	4650 ms
EdgeScan Method	297 ms	307 ms

6.4 Discussion

It can be drive from the results of the demonstrated point clouds that the EdgeScan method can efficiently detect the edges in unorganized 3D large size point cloud. The CPU runtime consumption is also near real time, 300 ms which is significantly faster as compared to other methods. It can increase as the point cloud increases in size.

Inherently, the fundamental limitation of this technique is its dependency on scanned information. The implementation of the EdgeScan method, other than the point cloud, needs having scanned raw data to convert it into 2D image.

Chapter 7

Conclusions and Future Work

In this chapter, conclusion of the thesis is provided. Furthermore, recommendations for the future work are also present.

7.1 Conclusion

Edge detection is being a vital concern since the early age of computer vision, using dataset of 2D images or 3D point clouds. In this thesis, a novel method, EdgeScan method, is proposed and implemented to detect the edges in an unorganized 3D point cloud. The conclusion is made with respect to the research question:

- *Which algorithm is most appropriate to detect the edges of the rocks automatically and accurately in near real time in a dense and large unorganized 3D point cloud?*

The answer to the research question as follow:

The EdgeScan method, proposed and implemented in this thesis, is capable to detect all the edges in the 3D point cloud with accuracy and near real time. The main work comprises of converting scan data into 2D image, extracting the edges data from the 2D image and mapping it into the corresponding 3D point cloud's points.

To accomplish the tasks, firstly the scanned data is directly transformed into 2D image, then the Canny edge detection algorithm is applied on the 2D image to extract the edges. Finally, pixel mapping mechanism is applied to map 2D image pixel to the corresponding 3D point cloud points.

The outcomes of the aforementioned approach clearly favors it, both qualitatively and quantitatively.

The comparative study of the EdgeScan method with other common edge detection methods for 3D point cloud reveals that the stated EdgeScan method is much faster in speed and the edges are clear and accurate especially for large dataset in real time systems. Therefore it can be an efficient, accurate and fast solution to detect the edges of rocks in unorganized 3D point cloud particularly.

7.2 Future Work

According to the limitations and issues presently faced during the implementations, some recommendation for the future work can be proposed to enhance the performance of the EdgeScan method.

Gaps and holes present in the original point cloud can be reduced either by improving scanner’s calibration or applying filling gaps in point cloud by interpolation method. This technique can be used to calibrate the scanner's assembly error by matching the edges in the middle line of the images

In particularly, to further focus on rocks in a point cloud, convex hull algorithm can be applied.

Bibliography

[1] Vivek Verma, Rakesh Kumar, and Stephen Hsu. “3D Building Detection and Modeling from Aerial LIDAR Data”. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 2213–2220. IEEE, 2006.-[152]

[2] Jan Böhm and Norbert Haala. “Efficient integration of aerial and terrestrial laser data for virtual city modeling using lasermaps”. In ISPRS workshop laser scanning, pages 192– 197, 2005.

[3] Norbert Haala, Susanne Becker, and Martin Kada. “Cell decomposition for the generation of building models at multiple scales”. In IAPRS Symposium Photogrammetric Computer Vision, pages 19–24, 2006.

[4] Norbert Pfeifer, Camillo Ressl, and Wilfried Karel. “Range calibration for terrestrial laser scanners and range cameras”. Proceedings of SPIE, 7447:1–15, 2009.

[5] Rusu, R.B., “Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments”, PhD dissertation, Institute for Informatics der Technischen Universitt Mnchen, 2009.

[6] M. Magnusson, A. Lilienthal, and T. Duckett.” Scan registration for autonomous mining vehicles using 3d-ndt”. Journal of Field Robotics, pages 803{827, 2007.

[7] K. Wang and Q. Yu. “Accurate 3d object measurement and inspection using structured light systems”. In Proceedings of the 12th International Conference on Computer Systems and Technologies, CompSysTech '11, pages 221{227, New York, NY, USA, 2011. ACM.

[8] F. Sadlo, T. Weyrich, R. Peikert, and M. Gross.” A practical structured light acquisition system for point-based geometry and texture”. In Proceedings of the Second

Eurographics / IEEE VGTC Conference on Point-Based Graphics, SPBG'05, pages 89{98, Aire-la-Ville, Switzerland, Switzerland, 2005. Eurographics Association.

[9] T. Botterill, R. Green, and S. Mills.” Reconstructing partially visible models using stereo vision, structured light, and the g2o framework”. In Proceedings of the 27th Conference on Image and Vision Computing New Zealand, IVCNZ '12, pages 370{375, New York, NY, USA, 2012. ACM.

[10] Irene Reisner-Kollmann, “Reconstruction of 3D Models from Images and Point Clouds with Shape Primitives”, Dissertation, Vienna University of Technology, 2013.

[11] Canny, J. F, “Finding edges and lines in images”, Master's thesis, MIT. AI Lab. TR-720, 1983.

[12] Canny, J. F, “A computational approach to edge detection”, IEEE Transaction on Pattern Analysis and Machine Intelligence, 8, 679-714, 1986.

[13] Ramesh Jain, Rangachar Kasturi, Brian G. Schunck, “Machine Vision”, chapter 5, pages 140-185, 1995.

[14] Rafael C. Gonzalez, Richard E. Woods & Steven L. Eddins,” Digital Image Processing Using MATLAB”, Pearson Education Ptd. Ltd, Singapore, 2004.

[15] Muthukrishnan.R and M.Radha,,” Edge detection techniques for Image segmentation”, International Journal of Computer Science & Information Technology (IJCSIT) Vol 3, No 6, Dec 2011

[16] Fleischman, S., Cohenor, D., Silva, T. “Robust moving least-squares fitting with sharp features”. ACM Trans Graph, pp. 37-49, 2005.

[17] Daniels, J., Ochotta, T., Ha, L. K.,” Spline-based feature curves from point sampled geometry”. Vis. Comput, 24(6), pp. 449-462, 2008.

[18] Oztireli, C., Guennebaud, G., Gross, M, “Feature preserving point set surfaces based on non-linear kernel regression”. Computer Graphics Forum, 28(2), 2009.

[19] Demarsin, K., Vanderstraeten, D., Volodine, T., Roose, D. “Detection of closed sharp edges in point clouds using normal estimation and graph theory”. Computer-Aided Design, 39(4), pp. 276-283, 2007.

[20] Xu, J., Zhou, M., Wu, Z., Shui, W., Ali, S. “Robust surface segmentation and edge feature lines extraction from fractured fragments of relics”, Journal of Computational Design and Engineering, 2(2), pp. 79-87, 2015.

[21] Lin, Y., Wang, C., Cheng, J., Chen, B., Jia, C., Chen, Z., Li, J. “Line segment extraction for large scale unorganized point clouds”. ISPRS Journal of Photogrammetry and Remote Sensing, 102, pp. 172-183, 2015.

[22] Weber, C., Hahmann, S., Hagen, H. “Sharp feature detection in point clouds”. Shape modelling international conference, pp. 175-186, 2010.

[23] Weber, C., Hahmann, S., Hagen, H. “Methods for feature detection in point clouds. Visualization of Large and Unstructured Data Sets” –IRTG Workshop, pp. 90-99, 2010.

[24] Gumhold, S., Wang, X., Mcleod, R.” Feature extraction from point clouds”. Proceedings of 10th International Meshing Roundtable, 2001.

[25] Feng, C., Taguchi, Y., Kamat, V. “Fast plane extraction in organized point clouds using agglomerative hierarchical clustering”. IEEE International Conference on Robotics and Automation (ICRA), pp. 6218-6225, 2014.

[26] Park, J., Shin, H., Choi, B.” Elliptic gabriel graph for finding neighbors in a point set and its application to normal vector estimation”. Computer Aided Design, 38(6), pp. 619-626, 2006.

[27] Holzer, S., Rusu, R.B., Dixon, M., Gedikli, S., Navab, N. “Adaptive neighborhood selection for real-time surface normal estimation from organized point cloud data using integral images”. IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2684-2689, 2012.

[28] Grim, C., Smart, W. “Shape classification and normal estimation for non-uniformly sampled, noisy point data”. Computers Graphics, 35(4), pp. 904-915, 2011.

[29] Li, B., Schnabel, R., Klein, R., Cheng, Z., Dang, G., Jin, S. “Robust normal estimation for point clouds with sharp features”. Computers Graphics, 34(2), pp. 94-106, 2010.

[30] Zhang, J., Cao, J., Liu, X., Wang, J., Liu, J., Shi, X. “Point cloud normal estimation via low-rank subspace clustering”. Shape Modeling International (SMI) Conference, 37(6), pp. 697-706, 2013.

[31] Wang, Y., Feng, H., Yung, D., Felix, E., Engin, S. “An adaptive normal estimation method for scanned point clouds with sharp features”. Computer-Aided Design, 45 (11), pp. 1333-1348, 2013.

[32] Rusu, R.B., Cousins, S. “3D is here: Point Cloud Library (PCL)”. In IEEE International Conference on Robotics and Automation (ICRA), pp. 1-4, 2011.

[33] B. Steder, R. Rusu, K. Konolige, and W. Burgard, “Point feature extraction on 3D range scans taking into account object boundaries,” in Proc. IEEE Int’l Conf. Robotics Automation (ICRA), 2011, pp. 2601–2608.

[34] Changhyun Choi, Alexander J. B. Trevor, and Henrik I. Christensen, “RGB-D Edge Detection and Edge-based Registration”, Center for Robotics & Intelligent Machines College of Computing Georgia Institute of Technology Atlanta, GA 30332, USA

[35] Roberts, L, “Machine Perception of 3-D Solids”, Optical and Electro-optical Information Processing, MIT Press,1965

[36] Kirsch, R., “Computer determination of the constituent structure of biological images”, Computers and Biomedical Research, 4, 315–328, 1971.

[37] Robinson. G, “Edge detection by compass gradient masks”, Computer graphics and image processing, 6, 492-501, 1977.

[38] Marr, D & E. Hildreth, “Theory of edge detection”, Proc. Royal Society of London, B, 207, 187–217, 1980.

[39] Marr, D, “Vision”, Freeman Publishers, 1982.

[40] Ji-fen Wang, Xin-rong Zhang, Computer Image Recognition, China Railway Publishing, Beijing, 1988.

[41] Rusu, R.B., Cousins, S. “Point Cloud Downloads”, http://pointclouds.org/, Willow Garage, 2011, Accessed Feb 2016.

[42] Intel, Willow Garage, “OpenCV Downloads”, http://opencv.org/, Itseez, 2000, Accessed Aug 2016.

[43] An AZO Network Site, “Education Underground Mining”, AZO Mining, http://www.azomining.com/video-details.aspx?VidID=23, Accessed OCT 2015.