Spring 2015

Automated segmentation, detection and fitting of piping elements from terrestrial LIDAR data

Yun-Ting Su
Purdue University

Follow this and additional works at: https://docs.lib.purdue.edu/open_access_dissertations
Part of the Civil Engineering Commons, and the Remote Sensing Commons

Recommended Citation
https://docs.lib.purdue.edu/open_access_dissertations/562

This document has been made available through Purdue e-Pubs, a service of the Purdue University Libraries. Please contact epubs@purdue.edu for additional information.
This is to certify that the thesis/dissertation prepared

By Yun-Ting Su

Entitled
AUTOMATED SEGMENTATION, DETECTION AND FITTING OF PIPING ELEMENTS FROM TERRESTRIAL LIDAR DATA

For the degree of Doctor of Philosophy

Is approved by the final examining committee:

James S. Bethel
Chair

Mireille Boutin

Jie Shan

Ayman F. Habib

To the best of my knowledge and as understood by the student in the Thesis/Dissertation Agreement, Publication Delay, and Certification Disclaimer (Graduate School Form 32), this thesis/dissertation adheres to the provisions of Purdue University’s “Policy of Integrity in Research” and the use of copyright material.

Approved by Major Professor(s): James S. Bethel

Approved by: Dulcy M. Abraham 4/17/2015

Head of the Departmental Graduate Program Date
AUTOMATED SEGMENTATION, DETECTION AND FITTING OF PIPING ELEMENTS FROM TERRESTRIAL LIDAR DATA

A Dissertation
Submitted to the Faculty
of
Purdue University
by
Yun-Ting Su

In Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy

May 2015
Purdue University
West Lafayette, Indiana
To my family
ACKNOWLEDGEMENTS

My years as a graduate student at Purdue University has been an exciting journey, filled with opportunities and wonderful memories, but also with challenges. I would like to thank my advisor, Professor James Bethel, for giving me the opportunities to develop as a researcher, and guiding me to overcome the obstacles and challenges along the way. He has inspired me to think creatively and innovatively, and taught me the value of patience and perseverance in the pursuit of knowledge. I would also like to thank the members of my Advisory Committee, Professors Jie Shan, Mireille Boutin, and Ayman Habib for their guidance and advice throughout the years.

To my husband, Shuowen Hu, I would like to express my sincere gratitude. His love and support along the way has strengthened my resolve in difficult times. And of course, special thanks to my parents, who gave me the opportunity to study overseas and have always believed in me. It has been a long journey, but one which has been worth traveling.
TABLE OF CONTENTS

<p>| LIST OF TABLES | vi |
| LIST OF FIGURES | vii |
| ABSTRACT | x |
| CHAPTER 1. INTRODUCTION AND PROBLEM STATEMENT | 1 |
| 1.1 Research Objectives | 3 |
| 1.2 LIDAR Principles | 5 |
| CHAPTER 2. OCTREE-BASED SEGMENTATION | 8 |
| 2.1 Introduction | 8 |
| 2.2 Background | 9 |
| 2.3 Methodology | 13 |
| 2.3.1 Octree Decomposition of LIDAR Point Cloud | 14 |
| 2.3.2 Graph Theory Based Splitting | 15 |
| 2.3.3 Merging Frame Work | 17 |
| 2.3.4 Connectivity Criteria for Merging Components | 21 |
| 2.3.4.1 Proximity Connectivity Criteria | 22 |
| 2.3.4.2 Orientation Connectivity Criteria | 23 |
| 2.3.4.3 Curvature Connectivity Criteria | 27 |
| 2.3.5 Sequential and Parallel Implementations | 31 |
| 2.4 Results | 32 |
| 2.4.1 Results from Subset of Steel Mill Scan | 33 |
| 2.4.2 Results from Whole Steel Mill Scene | 40 |
| 2.4.3 Results from Combined Scans of Steel Mill Scene | 41 |
| 2.4.4 Results from Subset of Chemical Plant Scan | 44 |
| 2.4.5 Results from Chemical Plant Scene | 47 |
| 2.4.6 Sequential and Parallel Execution Runtime | 48 |
| 2.5 Conclusions | 49 |
| CHAPTER 3. ESTIMATION OF CYLINDER ORIENTATION | 53 |
| 3.1 Introduction | 53 |
| 3.2 Background | 55 |
| 3.3 Methodology | 57 |
| 3.4 Results | 61 |</p>
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.4.1 Synthetic Experiments</td>
<td>62</td>
</tr>
<tr>
<td>3.4.2 Results using Terrestrial LIDAR Point Cloud Data</td>
<td>65</td>
</tr>
<tr>
<td>3.5 Conclusion</td>
<td>69</td>
</tr>
<tr>
<td>CHAPTER 4. CYLINDER PARAMETER ESTIMATION AND GRAPHIC REPRESENTATION</td>
<td>70</td>
</tr>
<tr>
<td>CHAPTER 5. CONCLUSIONS AND FUTURE WORK</td>
<td>77</td>
</tr>
<tr>
<td>5.1 Conclusions</td>
<td>77</td>
</tr>
<tr>
<td>5.2 Future Work</td>
<td>78</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>81</td>
</tr>
<tr>
<td>APPENDIX</td>
<td>89</td>
</tr>
<tr>
<td>A.1 General Least Squares Adjustment</td>
<td>89</td>
</tr>
<tr>
<td>A.2 Fitting a 3D Line by Least Squares</td>
<td>93</td>
</tr>
<tr>
<td>A.3 Fitting a Cylinder by Least Squares</td>
<td>96</td>
</tr>
<tr>
<td>VITA</td>
<td>100</td>
</tr>
</tbody>
</table>
LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1 Number of segments and number of pipes for steel mill subset determined by human subjects and the proposed segmentation algorithm</td>
<td>39</td>
</tr>
<tr>
<td>2.2 Number of segments and number of pipes for chemical plant subset determined by human subjects and the proposed segmentation algorithm</td>
<td>47</td>
</tr>
<tr>
<td>2.3 Runtimes for the split and merge procedures using sequential and parallel implementations</td>
<td>49</td>
</tr>
<tr>
<td>2.4 List of parameters and their descriptions for the proposed octree-based algorithm, along with general guidelines for setting these parameters</td>
<td>51</td>
</tr>
<tr>
<td>3.1 Circle fitting residual in projected 2D space using the axis orientation estimated by the Hough transform method and the proposed method</td>
<td>69</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1.1 Example of piping systems in a power plant</td>
<td>2</td>
</tr>
<tr>
<td>Figure 1.2 Flow chart illustrating organization of this dissertation</td>
<td>4</td>
</tr>
<tr>
<td>Figure 1.3 The measurement principle of time-of-flight scanner</td>
<td>6</td>
</tr>
<tr>
<td>Figure 1.4 The measurement principle of phase-based scanner</td>
<td>7</td>
</tr>
<tr>
<td>Figure 2.1 Illustration of an octree with a depth of two showing both its geometric and graph representations</td>
<td>14</td>
</tr>
<tr>
<td>Figure 2.2 Illustration of points within an octant and its graph representation</td>
<td>16</td>
</tr>
<tr>
<td>Figure 2.3 Illustration of the merging process for a group of eight adjacent octants, showing recursive pairwise merging into a single combined node/octant</td>
<td>19</td>
</tr>
<tr>
<td>Figure 2.4 Illustration of component merging procedure when combining two bins</td>
<td>20</td>
</tr>
<tr>
<td>Figure 2.5 Flow chart for the three merging criteria</td>
<td>21</td>
</tr>
<tr>
<td>Figure 2.6 Illustration of proximity connectivity analysis for component merging across bins, showing two bins with a shared face in the yz-plane</td>
<td>23</td>
</tr>
<tr>
<td>Figure 2.7 Illustration of two “long” components and their eigenvectors, whose lengths are scaled by their respective eigenvalues. Components A and B are located in two separate, but adjacent, octants</td>
<td>24</td>
</tr>
<tr>
<td>Figure 2.8 Illustration of a “long” and a “short” component and their eigenvectors, whose lengths are scaled proportionally with respect to the respective eigenvalues</td>
<td>26</td>
</tr>
<tr>
<td>Figure 2.9 Illustration of surface fitting procedure</td>
<td>28</td>
</tr>
<tr>
<td>Figure 2.10 Illustration of fitting procedure for pipes with &gt;180° coverage</td>
<td>29</td>
</tr>
<tr>
<td>Figure 2.11 (a) Point cloud of a region within the steel mill consisting of 552,368 points with an average point spacing of 0.0011 m, and (b) octants following octree decomposition</td>
<td>33</td>
</tr>
</tbody>
</table>
Figure 2.12 Three example octants with points displayed before and after graph theory based splitting into spatially unconnected components................................................................. 34

Figure 2.13 Examples of groups of 8 adjacent octants at each tree level for the steel mill region, before and after merging. ........................................................................................................ 36

Figure 2.14 Components of octants at Level 1 are merged to generate final segmentation results for the steel mill region................................................................. 37

Figure 2.15 Final segments which are identified as pipes ($\kappa_1>5$, $\kappa_2<1$, $R^2>0.6$) for steel mill region................................................................................................................. 38

Figure 2.16 Original point cloud of steel mill subset and rotated point cloud.................. 39

Figure 2.17 Segmentation results using rotated steel mill subset........................................ 40

Figure 2.18 (a) Single scan of complete steel mill, (b) final segmentation results, and (c) potential pipes ($\kappa_1>5$, $\kappa_2<1$, $R^2>0.3$).................................................................................. 41

Figure 2.19 (a) Combined point cloud of steel mill scene across six scans, and (b) final segmentation results............................................................................................ 42

Figure 2.20 (a) 74 pipe candidates identified by the algorithm using $\kappa_1>5$, $\kappa_2<1$, and $R^2>0.5$, and (b) 56 of the pipe candidates are actual piping elements.............................. 43

Figure 2.21 Examples of segments falsely identified as pipe candidates........................ 44

Figure 2.22 (a) Point cloud of a subset of the chemical plant scan, (b) final segmentation results, and (c) potential pipes ($\kappa_1>5$, $\kappa_2<1$, $R^2>0.1$)............................................. 45

Figure 2.23 Segmentation results of chemical plant subset using (a) proposed octree-based segmentation algorithm, and (b) Rabbani’s region growing based approach........ 46

Figure 2.24 (a) Point cloud of chemical plant scene, (b) final segmentation results, and (c) potential pipes ($\kappa_1>5$, $\kappa_2<1$, $R^2>0.1$).................................................................................. 48

Figure 3.1 Cylinder parameterization with $\theta$, $\varphi$, $\mathbf{u}$, $\mathbf{v}$, and $\mathbf{r}$............................. 55

Figure 3.2 Illustration of the spherical coordinate system, where $\theta$ is the inclination angle and $\varphi$ is the azimuth. ........................................................................................................ 58

Figure 3.3 Two perspectives of a synthetic cylinder point cloud with radius of 1.5, length of 10, containing 1,000 points................................................................. 62
<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 3.4</td>
<td>63</td>
</tr>
<tr>
<td>Figure 3.4</td>
<td>Axis orientation estimates shown in red at each iteration of the gradient descent optimization. Actual axis orientation shown in green.</td>
</tr>
<tr>
<td>Figure 3.5</td>
<td>64</td>
</tr>
<tr>
<td>Figure 3.5</td>
<td>Angular error in the estimated axis orientation for the Hough transformed based technique as a function of Hough Gaussian sphere sampling density.</td>
</tr>
<tr>
<td>Figure 3.6</td>
<td>65</td>
</tr>
<tr>
<td>Figure 3.6</td>
<td>Error histograms, or the empirical error distribution, of the proposed and Hough transform based techniques over 500 trials.</td>
</tr>
<tr>
<td>Figure 3.7</td>
<td>66</td>
</tr>
<tr>
<td>Figure 3.7</td>
<td>(a) LIDAR point cloud of a piping system acquired at a steel mill, (b) segmentation results using the octree-based segmentation technique, and (c) pipe segments identified by the segmentation technique using principal curvatures.</td>
</tr>
<tr>
<td>Figure 3.8</td>
<td>67</td>
</tr>
<tr>
<td>Figure 3.8</td>
<td>Axis orientations estimated using the proposed method (green) and using the Hough transform based approach (red) for the two example pipe segments.</td>
</tr>
<tr>
<td>Figure 3.9</td>
<td>68</td>
</tr>
<tr>
<td>Figure 3.9</td>
<td>Projections of the two pipe point clouds onto the plane perpendicular to the estimated axis orientations estimated.</td>
</tr>
<tr>
<td>Figure 4.1</td>
<td>70</td>
</tr>
<tr>
<td>Figure 4.1</td>
<td>Flow chart showing steps needed to generate a CAD model.</td>
</tr>
<tr>
<td>Figure 4.2</td>
<td>72</td>
</tr>
<tr>
<td>Figure 4.2</td>
<td>Illustration of position and radius estimation.</td>
</tr>
<tr>
<td>Figure 4.3</td>
<td>73</td>
</tr>
<tr>
<td>Figure 4.3</td>
<td>Geometry for cylinder min and max determination.</td>
</tr>
<tr>
<td>Figure 4.4</td>
<td>74</td>
</tr>
<tr>
<td>Figure 4.4</td>
<td>(a) “Projection” of points in cylinder point cloud onto the line represented by the cylinder axis, and (b) computed starting and ending points for the cylinder.</td>
</tr>
<tr>
<td>Figure 4.5</td>
<td>76</td>
</tr>
<tr>
<td>Figure 4.5</td>
<td>(a) Original cylinder point cloud, and (b) CAD model generated using cylinder parameter estimates, visualized in MuPAD.</td>
</tr>
<tr>
<td>Figure 4.6</td>
<td>76</td>
</tr>
<tr>
<td>Figure 4.6</td>
<td>(a) Correctly identified piping candidates from the combined point cloud across six scans of the steel mill scene, and (b) piping system CAD model.</td>
</tr>
<tr>
<td>Appendix Figure</td>
<td></td>
</tr>
<tr>
<td>Figure A.1</td>
<td>93</td>
</tr>
<tr>
<td>Figure A.1</td>
<td>Illustration of 3D line fitting parameters.</td>
</tr>
<tr>
<td>Figure A.2</td>
<td>94</td>
</tr>
<tr>
<td>Figure A.2</td>
<td>Illustration of counting $n_0$ for the case of 3D line fitting with four points.</td>
</tr>
<tr>
<td>Figure A.3</td>
<td>96</td>
</tr>
<tr>
<td>Figure A.3</td>
<td>Illustration of counting $n_0$ for the case of cylinder fitting with ten points.</td>
</tr>
<tr>
<td>Figure A.4</td>
<td>97</td>
</tr>
<tr>
<td>Figure A.4</td>
<td>Illustration of fitting observations to a cylinder model.</td>
</tr>
<tr>
<td>Figure A.5</td>
<td>98</td>
</tr>
<tr>
<td>Figure A.5</td>
<td>Illustration of projection from observations $S$ onto $u$.</td>
</tr>
</tbody>
</table>
ABSTRACT

Su, Yun-Ting. Ph.D., Purdue University, May 2015. Automated Segmentation, Detection and Fitting of Piping Elements from Terrestrial LIDAR Data. Major Professor: James S. Bethel.

Since the invention of light detection and ranging (LIDAR) in the early 1960s, it has been adopted for use in numerous applications, from topographical mapping with airborne LIDAR platforms to surveying of urban sites with terrestrial LIDAR systems. Static terrestrial LIDAR has become an especially effective tool for surveying, in some cases replacing traditional techniques such as electronic total stations and GPS methods. Current state-of-the-art LIDAR scanners have very fine spatial resolution, generating precise 3D point cloud data with millimeter accuracy. Therefore, LIDAR data can provide 3D details of a scene with an unprecedented level of details. However, automated exploitation of LIDAR data is challenging, due to the non-uniform spatial sampling of the point clouds as well as to the massive volumes of data, which may range from a few million points to hundreds of millions of points depending on the size and complexity of the scene being scanned.

This dissertation focuses on addressing these challenges to automatically exploit large LIDAR point clouds of piping systems in industrial sites, such as chemical plants, oil refineries, and steel mills. A complete processing chain is proposed in this work, using raw LIDAR point clouds as input and generating cylinder parameter estimates for
pipe segments as the output, which could then be used to produce computer aided design (CAD) models of pipes. The processing chain consists of three stages: (1) segmentation of LIDAR point clouds, (2) detection and identification of piping elements, and (3) cylinder fitting and parameter estimation. The final output of the cylinder fitting stage gives the estimated orientation, position, and radius of each detected pipe element.

A robust octree-based split and merge segmentation algorithm is proposed in this dissertation that can efficiently process LIDAR data. Following octree decomposition of the point cloud, graph theory analysis is used during the splitting process to separate points within each octant into components based on spatial connectivity. A series of connectivity criteria (proximity, orientation, and curvature) are developed for the merging process, which exploits contextual information to effectively merge cylindrical segments into complete pipes and planar segments into complete walls. Furthermore, by conducting surface fitting of segments and analyzing their principal curvatures, the proposed segmentation approach is capable of detecting and identifying the piping segments.

A novel cylinder fitting technique is proposed to accurately estimate the cylinder parameters for each detected piping segment from the terrestrial LIDAR point cloud. Specifically, the orientation, radius, and position of each piping element must be robustly estimated in the presence of noise. An original formulation has been developed to estimate the cylinder axis orientation using gradient descent optimization of an angular distance cost function. The cost function is based on the concept that surface normals of points in a cylinder point cloud are perpendicular to the cylinder axis. The key contribution of this algorithm is its capability to accurately estimate the cylinder orientation in the presence of noise without requiring a good initial starting point. After estimation of the cylinder’s axis
orientation, the radius and position are then estimated in the 2D space formed from the projection of the 3D cylinder point cloud onto the plane perpendicular to the cylinder’s axis. With these high quality approximations, a least squares estimation in 3D is made for the final cylinder parameters.

Following cylinder fitting, the estimated parameters of each detected piping segment are used to generate a CAD model of the piping system. The algorithms and techniques in this dissertation form a complete processing chain that can automatically exploit large LIDAR point cloud of piping systems and generate CAD models.
Light detection and ranging (LIDAR) originated in the early 1960s, with its first applications in atmospheric science. As LIDAR technology matured, improving in accuracy, speed, and spatial resolution, it has been adopted for use in numerous applications, from topographical mapping with airborne LIDAR platforms to surveying of urban sites with terrestrial LIDAR systems. Terrestrial LIDAR has become an especially effective tool for surveying, in some cases replacing traditional techniques such as electronic total stations and GPS methods. With terrestrial LIDAR, surveyors can scan an entire site at a stand-off distance without needing to have a person occupy points on the site (Lee, 2011).

Current generation LIDAR scanners have very fine spatial resolution, providing precise 3D point cloud data with millimeter accuracy (Soulard and Bogle, 2011). Typically, a site is surveyed from multiple perspectives to obtain more complete coverage, with the resulting scans aligned registered and combined to form a single high density point cloud. Consequently, LIDAR point clouds usually contain hundreds of thousands to tens of millions of individual points or more, depending on the size of the site being surveyed. Furthermore, LIDAR data is unstructured, since its point distribution is spatially heterogeneous (i.e. scanned points do not lie on a uniform spatial grid). The ultimate goal in most applications involving LIDAR is to process this high density unstructured
point cloud data and reconstruct a 3D computer-aided design (CAD) model of the scene, first by segmenting the point cloud data into appropriate segments and then recognizing primitives from the segments to generate 3D models. The focus of this dissertation is on automated exploitation of terrestrial LIDAR data, specifically for point clouds of piping systems. Piping systems are ubiquitous in industrial sites such as oil refineries, power plants, chemical facilities, and heating, ventilation, and air conditioning (HVAC) facilities. See the example in Figure 1.1.

![Figure 1.1 Example of piping systems in a power plant](image)

Up-to-date 3D schematics/models of the piping systems are critical for planning and documentation (as-built) purposes. Although research on exploitation of LIDAR data has made substantial progress over the past decade, accurate and automatic extraction of primitives (e.g. walls as planes, pipes as cylinders) remains challenging. Considering the fact that these primitives typically compose up to 85% of all objects in industrial sites
(Petitjean, 2002), accurate and efficient extraction of these primitives is important. Section 1.1 first introduces the research objectives for this dissertation, followed by an overview of the fundamental principles of LIDAR in Section 1.2. Chapters 2-4 then present in detail the author’s core contributions, specifically a novel octree-based segmentation technique as well as an original mathematical formulation for cylinder parameter estimation of piping segments.

1.1 Research Objectives

The research presented in this dissertation has two main objectives: automated segmentation of large scale LIDAR point clouds, and robust estimation of cylinder parameters (i.e. orientation, radius, and position) of the extracted cylinder segments. Segmentation and cylinder parameter estimation are the two most important steps that enable CAD modeling of LIDAR piping point clouds acquired at sites such as oil refineries and steel mills. Given an entire unstructured point cloud of a scene, the first objective, addressed in Chapter 2 is to perform segmentation and extract individual segments, determining which segments are likely to be pipes or walls. A robust octree-based split and merge segmentation algorithm is proposed that can efficiently process large scale LIDAR data. The novel use of graph analysis in the splitting process and the introduction of merging based on a series of connectivity criteria (proximity, orientation, and curvature) are the key contributions of this work. The proposed segmentation algorithm for terrestrial LIDAR data is a fundamentally novel bottom-up approach that is highly scalable and parallelizable.

The second objective, addressed in Chapters 3 and 4, is to accurately estimate the cylinder parameters from the individual piping segments extracted from the terrestrial
LIDAR point cloud in the first step. Specifically, the orientation, radius, and position of each piping element must be robustly estimated in the presence of noise. An original mathematical formulation is proposed in Chapter 3 to estimate the cylinder axis orientation using gradient descent optimization of an angular distance cost function. The cost function is based on the concept that surface normals of points on a cylinder point cloud are perpendicular to the cylinder axis. After estimation of the cylinder’s axis orientation, the radius and position are then estimated in the 2D space formed from the projection of the 3D cylinder point cloud onto the plane perpendicular to the cylinder’s axis. With these cylinder parameter estimates, a final estimate can be made via least squares, and a CAD model can then be generated. Figure 1.2 presents a flow chart depicting the organization of this dissertation.

Figure 1.2 Flow chart illustrating organization of this dissertation.
1.2 LIDAR Principles

LIDAR sensors are typically mounted on aircraft platforms, mobile and vehicle platforms, or on a static terrestrial fixture such as a tripod. Airborne LIDAR is typically used to obtain detailed elevation mapping data of urban and natural landscapes for civilian and military applications, while terrestrial LIDAR is commonly employed to acquire detailed range data in urban settings such as factories and industrial sites, though it has been used for geological surveying applications as well. The focus of this dissertation is on static terrestrial LIDAR. Today, most terrestrial laser scanners depend on oscillating mirrors to deflect the laser beam across the scene in a systematic pattern. The angles are defined by the rotation of the azimuth axis and the position of an oscillating or rotating mirror with motion about the elevation. A scanner typically records range, intensity and azimuth/elevation angle for each point. If the scanner has a built in digital camera, RGB imagery data may also be recorded. Optionally the RGB data may be used to “colorize” the point cloud. There are two types of range sensing within terrestrial laser scanners having differing range and accuracy characteristics, depending on the principle employed for range measurement.

Time-of-flight (TOF) scanners, also referred to as pulsed based scanners, are most prevalently used because these systems are capable of measurement at longer distances of up to several hundred meters, and have scan rates of 500,000 points per second, or more (California Department of Transportation, 2011). TOF scanners emit laser pulses and use a mirror to deflect the beam in a scanning pattern. The scanner’s optical receiver inside the system records and times the return pulse, using the time-of-flight of principle to compute the distance to the object. Specifically, the distance to a surface point is
calculated by dividing the two-way travel time by two and multiplying by the speed of light. Figure 1.3 illustrates the measurement principle of a time-of-flight scanner.

![Time-of-flight scanner diagram]

**Figure 1.3** The measurement principle of time-of-flight scanner

Besides the time-of-flight principle, phase-based measurement represents the other common technique. The principle of phase-based or phase-shift measurement is based on phase comparison. A phase-based scanner emits predefined sinusoids modulated onto the carrier and then measures the phase difference between transmitted and received signals. The distance measurement is determined by the phase difference expressed in time units multiplied by the speed of light (taking into account the round trip path). In contrast to time-of-flight systems, phase-based scanners have a higher scan rate but a shorter effective range, typically between 25-75m (California Department of Transportation, 2011). The range precision of both type of instrument are comparable. In recent years the range for phase based system is approaching that for TOF systems. Figure 1.4 illustrates the measurement principle of phase-based scanner.
While both time-of-flight and phase-shift technologies are used for static terrestrial laser scanners, each has its own advantages and disadvantages. As discussed previously, phase-based scanners typically have higher scan rates approaching 1 million points per second, while time-of-flight scanners usually have scan rates up to 500,000 points per second. Time-of-flight scanners offer the benefits of lower noise levels as well as more consistent noise patterns compared to phase-based scanners (California Department of Transportation, 2011). As this work focuses on scenes of building interiors, specifically piping systems, the primary datasets used in this dissertation has been acquired using a phase-based scanner. The high density point clouds offered by phase-based scanners are ideal for segmentation, primitive fitting, and CAD modeling.
CHAPTER 2. OCTREE-BASED SEGMENTATION

2.1 Introduction

This chapter focuses on efficient segmentation of terrestrial LIDAR data of piping systems in industrial sites (e.g. chemical plants, oil refineries, steel mills). Given an entire unstructured point cloud of a scene, the objective is to perform segmentation and extract individual segments, determining which segments are likely to be pipes, vessels, or walls. We propose a robust octree-based split and merge segmentation algorithm that can efficiently process large LIDAR data. Following initially splitting the dataset into octants based on point density using octree decomposition, the points in each octant are further split into spatially unconnected components using graph theory based analysis. Following splitting, the merging process is based on a series of connectivity criteria (proximity, orientation, and curvature). The novel split and merge procedures are the key contributions of this work. This proposed segmentation algorithm is a novel, bottom-up approach that is highly scalable and parallelizable.

The organization of this chapter is as follows. Section 2.2 describes prior, related work on the segmentation of terrestrial LIDAR data. Section 2.3 provides a detailed description of the proposed octree-based split and merge segmentation algorithm. Section 2.4 shows the results of the proposed algorithm on several LIDAR datasets, and Section 2.5 presents the conclusions of this work.
2.2 Background

The earliest studies on the segmentation of range data used sensors that acquired 2.5D range images, also referred to as depth maps, that lie on a uniform spatial grid – each regularly spaced point (x,y) on a rectangular grid has a range measurement. Henderson and Bhanu (1982) developed a planar region growing algorithm for range images using a spatial proximity graph. Hebert and Ponce (1982) proposed a method to segment depth maps into primitives such as planes and cylinders by mapping point surface normals to the Gaussian sphere. An edge-region segmentation ring operator was developed by Inokuchi et al. (1982). Besl and Jain (1985) provides an excellent literature review on the studies involving range image analysis. Unlike range images, point clouds from LIDAR sensors are 3D data that do not lie on a uniform spatial grid. While many concepts can be adopted from the earlier work with range images, there has been substantial progress recently in the segmentation of LIDAR point cloud data.

As mentioned in the introduction, LIDAR sensors are categorized as either airborne or terrestrial. Many techniques have been developed for segmentation of airborne LIDAR data. Arefi and Hahn, 2005; Zhao et al., 2011; Li et al., 2013; Yan et al., 2015; Poullis and S. You, 2009; R.A. McLaughlin, 2006 are a few representative papers describing these techniques. Terrestrial LIDAR can be subdivided into mobile platforms or static systems, which is the focus of this work. However, segmentation of terrestrial LIDAR data, especially scenes of piping systems, has received relatively less attention. The remainder of this section is devoted to discussing the relevant works in this area. Rabbani (2006) introduced a smoothness constraint based segmentation technique that is one of the most widely cited works on segmentation of terrestrial LIDAR data. Rabbani’s technique is a
bottom-up approach with two main steps: normal vector estimation and region growing. In the first step, the surface normal for each point is estimated by fitting a plane to its neighbors, found through the $k$-nearest neighbors method. The residual of the plane fitting to a point’s neighbors is used by Rabbani (2006) to approximate the local surface curvature. A small residual indicates that the neighbors lie on a planar surface, while a large residual indicates a more curved surface. However, a large residual may also be due to noise. Following computation of every point’s surface normal, the second step of region growing is initiated with a seed point that has the smallest residual from the first step. The neighboring points of this seed point with residual below a set threshold is added to the list of available points for consideration, and a surface smoothness constraint determines whether these available points are added to the current region. The surface smoothness constraint is implemented by considering the angle between the seed point’s surface normal and its neighbor’s surface normal. If this angle is below a certain threshold, typically set at 15º (Rabbani, 2006), this point is added to the region and updated to be the current seed point. The process continues iteratively until the list of available points is exhausted, and then a new region is started using the point with the smallest residual from the remaining point cloud.

Rabbani’s segmentation technique uses the smoothness constraint and has two limitations, as we observed through experimentation using our Matlab implementation of Rabbani’s algorithm. First, regions linked together by a smooth connector are segmented as a single region. For example, a vertical pipe connected to a horizontal pipe via a long radius elbow joint would exhibit smoothly varying surface normal from one end to the other, and would be segmented as a single region instead of three separate regions, as
typically would be desired. Rabbani (2006) also recognized this concern, but reasoned that this under-segmentation is more preferable to over-segmentation. The second limitation is the computational complexity of the algorithm, which requires the $k$-nearest neighbors (KNN) for every point in the dataset to be computed. The linear search solution for KNN has a running time of $O(Nd)$ where $N$ is the number of points and $d$ is the dimensionality of the data. For typical LIDAR point clouds that contain hundreds of thousands to tens of millions of points, computing the KNN of every point is computationally prohibitive. Space partitioning methods such as $k$-$d$ trees have been applied to KNN search (Freidman et al., 1977), reducing the search complexity to $O(\log N)$, but involves an offline phase to construct the $k$-$d$ tree.

Wang and Tseng (2010) introduced an incremental segmentation technique using an octree-structured voxel space. Their octree based split and merge segmentation algorithm first divides the input point cloud into octree subspaces (i.e. octants) until each octant only contains coplanar points during the splitting process. Coplanarity is measured by computing the residuals of plane fitting in an octant, similar to the plane fitting procedure of Rabbani (2006). If the variance of the residuals exceeds a user defined threshold, indicating that the points do not form a coplanar surface, the node is further subdivided into octants. Following the splitting procedure, Wang and Tseng (2010) perform a merging procedure, checking whether adjacent planes have similar surface normal orientations and are sufficiently proximate to be merged into a single plane. Since this technique only focuses on coplanarity during the split and merge steps, it is more suited for segmentation of airborne LIDAR data than for industrial scenes. Wang and Tseng (2011) extended this technique, incorporating the concept of co-surface during the merging
procedure to combine coplanar segments that lie along a smoothly curved surface. The co-
surface criterion is computed using the angle variation of the directions of the fitted planes
for adjacent coplanar segments. This extended technique can be used to process both
airborne and terrestrial LIDAR point clouds. However, for terrestrial LIDAR data of
scenes containing objects with complex shapes, as occurs frequently in industrial sites, the
co-surface criterion is not likely to be successful.

Liu et al. (2013) proposed a hierarchical structure detection and decomposition
method, specifically for massive point clouds with many pipe objects, such as LIDAR data
of oil refineries. Their proposed technique is based on the assumption that pipes are either
perpendicular or parallel to the ground. After finding the ground plane using the Gauss
map, which maps a point in $\mathbb{R}^3$ to a point on the unit sphere, Liu et al. (2013) projects the
point cloud onto the ground plane. Therefore, the task of finding pipes in $\mathbb{R}^3$ is reduced to
finding circles in $\mathbb{R}^2$, for which Liu et al. (2013) used a random sample consensus
(RANSAC) based method. After removing the detected pipes perpendicular to the ground,
the remaining points are projected onto several planes perpendicular to the computed
ground plane, followed by circle fitting to find pipes parallel to the ground. This method,
in some sense, can be viewed as a joint cylinder segmentation and cylinder fitting approach.
The technique is computationally efficient, since the pipeline reconstruction problem in $\mathbb{R}^3$
has been converted to a set of circle finding problems in $\mathbb{R}^2$. However, the main limitation
of this approach is the assumption that pipes are either perpendicular or parallel to the
ground. While this assumption may hold for the majority of pipes in an industrial setting,
there will be a number of pipes at most sites with axis orientations oblique to the ground
plane that violate this assumption.
Most recently, Lari and Habib (2014) proposed an adaptive segmentation and extraction approach for planar and cylindrical features in terrestrial LIDAR data. Eigenvectors and eigenvalues computed from each point’s local neighborhood are used to identify the points that belong to planar or cylindrical features. Note that these characteristic attributes are determined while taking the local density variation and noise level into consideration for robustness. The directional and positional parameters of the cylindrical features are used to generate final segmentation results through clustering. Lari et al. (2014) proposed a novel quality measure, the surface roughness factor, to evaluate the segmentation quality. Furthermore, Lari et al. (2014) defined quantitative measures of non-segmented points, over-segmentation, and under-segmentation.

The octree-based segmentation technique proposed in this work is focused on piping systems and seeks to intelligently segment cylindrical objects using a series of connectivity criteria. The higher level orientation connectivity criterion is similar conceptually to the classification step for plane and cylindrical features in Lari and Habib (2014). The proposed octree-based technique is a novel bottom-up approach that is computationally efficient, using graph theory based split and merge processing to effectively segment point clouds of piping systems.

2.3 Methodology

The proposed octree-based segmentation technique for terrestrial LIDAR data has three main steps: (1) octree decomposition of a point cloud based on point density into octree-style bins, also referred to as nodes or octants, (2) splitting of points within each octant into spatially connected components based on graph connectivity analysis, and (3) recursive merging of components across octants based on a set of connectivity criteria until
the root node is reached. In essence, the proposed technique performs an octree decomposition followed by a split-and-merge procedure. Section 2.3.1 describes the octree decomposition process, while Sections 2.3.2 and 2.3.3 presents the splitting and merging steps, respectively.

2.3.1 Octree Decomposition of LIDAR Point Cloud

An octree is a data structure represented by a tree in which each branch node has exactly eight children (Meagher, 1980), and therefore are commonly used to partition three dimensional space recursively into octants (Liu et al., 2008; M. Wang and Y-H. Tseng, 2010; Hornung et al., 2013). Figure 2.1 illustrates an example octree with a depth of two, with its corresponding spatial representation shown on the left.

![Octree Diagram](image)

Figure 2.1 Illustration of an octree with a depth of two showing both its geometric and graph representations.

In this work, a region is equally partitioned into eight octants, and this recursive subdivision proceeds until all octants contain less than a maximum number of points, referred to as the bin capacity and represented by $N_{\text{max}}$. Note that the terms bin, node, and octant have the same meaning in this work, and are used interchangeably. After octree decomposition, each leaf node (i.e. node without any children) is an octant containing less
than $N_{max}$ number of points. Therefore, the volume (i.e. spatial boundaries) occupied by a leaf node is different based on the level of the tree that the leaf node is located at – volumes will always be in multiples of eight with respect to the smallest octant at the deepest level. Note that the octree decomposition procedure used in this work is a standard technique (Meagher, 1980). The core contribution and novelty of this work is the proposed split and merge procedure that operates on the octants to perform a computationally efficient bottom-up segmentation of the terrestrial LIDAR point cloud.

2.3.2 Graph Theory Based Splitting

Following octree decomposition, each leaf node contains a group of points located within the rectangular boundaries of the octant. However, these points may not be spatially connected, in which case the group of points must be split or divided into separate components. In this work, a component is defined as a collection of points with the following two properties: (1) distance (i.e. edge length) between adjacent points are less than some given $\epsilon$, and (2) a path can be traced from any point in the component to any other point through adjacent edges. A graph theory based approach can be naturally applied to perform the splitting procedure and find the individual components within an octant.

Figure 2.2 shows an octant example containing 7 points ($p_1, \ldots, p_7$). Undirected edges are formed between point pairs ($p_1, p_2$), ($p_1, p_3$), ($p_2, p_3$), ($p_3, p_4$), ($p_5, p_6$), ($p_5, p_7$), and ($p_6, p_7$) because the distances (represented by edge $e_{ij}$) between these pairs of points are less than $\epsilon$. The corresponding graph representation, graph matrix $G$, is shown on the right in Figure 2.2. Tarjan’s algorithm (Tarjan, 1972) is used to find the connected components in the graph matrix $G$. While Tarjan’s algorithm is traditionally used to find
strongly connected components in a directed graph, it can also operate on an undirected graph by ignoring the upper triangle of the graph matrix $G$. In the example depicted in Figure 2, points $(p_1, p_2, p_3, p_4)$ form one component, while points $(p_5, p_6, p_7)$ form the other component in the octant. Even though $p_4$ has a single edge (with $p_3$), it is connected to $p_1$ and $p_2$ through $p_3$, and therefore is part of the component (i.e. there is a path from a point in the component to every other point in the component).

![Octant Diagram]

Figure 2.2 Illustration of points within an octant and its graph representation, matrix $G$, which is used by Tarjan’s algorithm to find connected components.

While the previous paragraph describes the general framework of the graph theory based splitting process, the following paragraphs will provide implementation details. To compute the graph matrix $G$, an intermediate distance matrix $D$ is first obtained by computing the Euclidean distance between every point in the octant. For an octant containing $N$ points, where $N$ is less than the bin capacity ($N_{max}$), the resulting matrix $D$ is an $N \times N$ matrix. Computing this Euclidean distance matrix is memory intensive – if $N=10000$, $D$ has 100 million elements, requiring 800 megabytes (MB) of system memory to be represented in double format. For a computer system with very limited memory, $D$
could also be created and populated in a sparse manner, which is more memory efficient but leads to longer execution time. For this work, execution time is prioritized over memory usage – \( D \) is computed directly in a non-sparse manner. Increasing \( N_{max} \) will quadratically increase the memory usage. For a typical computer with 4GB memory (part of the memory will already be utilized by the operating system and other processes), 10000 represents the memory constraint and is an approximate upper bound on \( N_{max} \). After computing \( D \), all entries with value greater than \( \varepsilon \) are set to zero, and the diagonal is forced to zero as well, forming the sparse graph matrix \( G \).

Choosing an appropriate \( \varepsilon \) is therefore important for the splitting process, and can be based on the point spacing of the point cloud by examining the distribution of closest neighbor distances and computing the average closest neighbor distance. Choosing \( \varepsilon \) to be five to ten times this estimated point spacing proves to be practical for the proposed technique. Note that for very dense point clouds generated by terrestrial LIDAR scanners, it is also desirable to define a minimum number of neighbors (within \( \varepsilon \) distance) a point must have before the point can be used to form connections – preventing situations where two large dense groups of points with only a few points between them form a single component. To enforce this scenario, the row and column entries of \( G \) corresponding to a point with less than \( K \) neighbors are set to 0. For this work, a minimum of 15 neighbors within \( \varepsilon \) is used to generate the final graph \( G \), which is used by Tarjan’s algorithm to split the points within an octant into individual components.

2.3.3 Merging Frame Work

The splitting procedure described previously is designed to produce an over-segmentation of the scene, recursively dividing the point cloud into octants and splitting
the points within each leaf node/octant into individual components. These small components must then be merged together across octant boundaries using a series of connectivity criteria, designed specifically to facilitate the merging of cylindrical components into pipes and planar components into floor, walls, etc. Cylinders and planes are the most prevalent geometric primitives in scans of industrial scenes in which liquid or gaseous matter must be moved about. In addition, they will be the primary objects of interest for CAD models. Therefore, accurate segmentation and identification of pipes and walls are critical. The overall merging framework is discussed in this section, while the connectivity criteria are described in detail in Section 2.3.4.

The merging stage of the proposed octree-based segmentation technique is a bottom-up procedure that starts at the deepest level of the octree, merging components across every eight adjacent octants in the octree structure. Here, adjacent octants are defined as the child nodes/octants belonging to a single branch node (also referred to as “the parent node”) from the level above. As each branch node has exactly eight children in the octree structure, the merging process occurs independently for every group of eight adjacent octants at a given level of the tree.

For a group of eight adjacent octants, the merging process occurs pairwise as illustrated in Figure 2.3. First, octants that shares a common face (i.e. plane) parallel to the $xy$ plane are combined pairwise, forming combined bins (1&2), (3&4), (5&6), and (7&8). The combined bins that share a common face parallel to the $xz$ plane are in turn combined, forming two bins $[(1&2)&(3&4)]$ and $[(5&6)&(7&8)]$. These two bins, which share a
common face parallel to the \(yz\) plane, are finally combined into a single bin \([(1&2)&(3&4)]\)
& \([(5&6)&(7&8)]\}, or octant, that resides at the tree level above.

![Diagram of merging process](image)

Figure 2.3 Illustration of the merging process for a group of eight adjacent octants, showing recursive pairwise merging into a single combined node/octant.

When combining two bins using the pairwise recursive procedure, components contained in the two bins are merged into a single component only if the connectivity criteria (detailed in Section 2.3.4) are satisfied, otherwise they remain separate components. However, merging components across bins is not a straightforward procedure, but requires application of graph analysis, as illustrated in Figure 2.4.
Figure 2.4 Illustration of component merging procedure when combining two bins.

As illustrated in Figure 2.4, each of the two bins have two components. Assuming that Component \( A_1 \) is connected to Component \( B_1 \) after conducting connectivity analysis and that Component \( A_2 \) also is connected to Component \( B_1 \), the component connectivity graph matrix \( G_c \) is generated by placing a value of 1 (representing a connection) in the appropriate matrix element. Tarjan’s algorithm also provides an elegant solution here in determining which interlinked components should be merged together. In Figure 2.4, Components \( A_1 \) and \( A_2 \), although unconnected in Bin \( A \), are merged together through the interlinking Component \( B_1 \). Component \( B_2 \), which is unconnected with any other component, remains an individual component in the combined Bin \( C \). Next, the details of the connectivity criteria used to merge components are described.
2.3.4 Connectivity Criteria for Merging Components

While the previous section describes the overall merging framework, this section presents the connectivity criteria used to determine whether two components located in two adjacent nodes should be merged together into a single component when the nodes are combined. Three specific merging criteria are developed in this research to determine component connectivity: proximity connectivity (Subsection 2.3.4.1), orientation connectivity (Subsection 2.3.4.2), and curvature connectivity (Subsection 2.3.4.3). See Figure 2.5 for the flow chart of merging criteria.

Figure 2.5 Flow chart for the three merging criteria
2.3.4.1 Proximity Connectivity Criteria

Proximity connectivity is the most critical merging criterion, assessed before considering either orientation or curvature connectivity – if a component in one node is not spatially connected to a component in an adjacent node across the shared face, then there is no need to examine their orientation connectivity or curvature connectivity. The proximity connectivity criterion is based on the same concept as the \( \epsilon \) distance threshold introduced in Section 2.3.2.

Figure 2.6 illustrates the proximity connectivity analysis, showing two components \((A_1 \text{ and } B_1)\) which have points within \( \epsilon \) distance of the shared face in the \( yz \)-plane. A Euclidean distance matrix \( D_c \) is computed between the points of Component \( A_1 \) that lie in the \( \epsilon \) region of Bin \( A \) and the points of Component \( B_1 \) that lie in the \( \epsilon \) region of Bin \( B \). The entries of \( D_c \) with values less than \( \epsilon \) are tallied, representing the total number of connecting points between the two components. To ensure that the two components are not spatially connected by only a few points, the average point density of the connecting points in the \( \epsilon \) region must be within an order of magnitude of the average point density of Components \( A_1 \) and \( B_1 \) along the direction perpendicular to the shared face (e.g. \( x \)-axis direction in Figure 5). Components that have been determined to be proximity connected across adjacent nodes must now be evaluated for the orientation connectivity and curvature connectivity.
Figure 2.6 Illustration of proximity connectivity analysis for component merging across bins, showing two bins with a shared face in the yz-plane.

2.3.4.2 Orientation Connectivity Criteria

The purpose of orientation connectivity is to facilitate the merging of cylindrical sections and planar sections into more complete pipes and walls. The orientation criterion can be viewed as based on higher level contextual information—specifically, that sections of the same cylinder/pipe must have similar axial orientation, and that sections of the same plane/wall must also have similar planar orientation. A common way to characterize points in a region is to assume they are random samples, and compute a 3D sample covariance matrix. The distribution in space of a component can be represented by the eigenvectors and eigenvalues, respectively, of its covariance matrix. Defining length and width as the two largest dimensions of a component, the contextual information used for the orientation criterion is based on first computing the ratio of the length of a component to its width.
Figure 2.7 Illustration of two “long” components and their eigenvectors, whose lengths are scaled by their respective eigenvalues. Components A and B are located in two separate, but adjacent, octants. The plane at x=5 denotes the shared face between the two octants. Magenta, green, and black represent the eigenvectors corresponding to the largest to smallest eigenvalues, respectively.

Figure 2.7 shows an example of two components spatially connected across two adjacent nodes/octants. The eigenvectors are overlaid –magenta, green, and black eigenvectors \((v_1, v_2, v_3)\) correspond to the largest, middle, and smallest eigenvalues \((\lambda_1, \lambda_2, \lambda_3)\) of each component. Note that the two cylindrical components illustrated in Figure 2.7 are parts of the same pipe, but are located in adjacent octants following octree decomposition. The concepts of “long” and “short” are introduced here to facilitate the merging of cylindrical and planar sections, and are defined by examining the ratio of the largest and middle eigenvectors. A component is considered “long” if \(\frac{\lambda_1}{\lambda_2} \geq r_{thr}\), else the component is “short”. Since the concepts “long” and “short” are abstract human concepts, setting the ratio threshold \(r_{thr}\) is therefore subjective – we define a long component as one whose length is at least five times its width (i.e. \(r_{thr}=5\)). This setting for \(r_{thr}\), chosen by experiences, has proven to be effective across multiple datasets, as demonstrated in the results section (Section 2.4).
After determining whether the pair of components under consideration are “long & long”, “long & short” (equivalently, “short & long”), or “short & short”, the next step in the orientation connectivity analysis seeks to assess whether the two components have similar orientation by examining the angle between the eigenvectors of the two components. If the two components are “long & long”, as illustrated in Figure 2.7, the angle between the eigenvectors corresponding to the largest eigenvalues (i.e. the angle $\theta_{11}^{AB}$ between $v_1^A$ from Component A in one Bin and $v_1^B$ from Component B in the other bin) is computed using the dot product relationship, where the vectors have unit length, as in Equations 2.1 and 2.2. If $\theta_{11}^{AB} \leq \theta_{thr}$, then the orientation connectivity criterion is satisfied. If this condition is not satisfied, the two components are not merged together even though they are spatially connected. The setting of $\theta_{thr}$=10° is used for this work, and has proven to be effective across multiple datasets, as demonstrated in the results section.

$$v_1^A \cdot v_1^B = v_{1x}^A v_{1x}^B + v_{1y}^A v_{1y}^B + v_{1z}^A v_{1z}^B = \|v_1^A\|\|v_1^B\|\cos(\theta_{11}^{AB}) \quad (2.1)$$

$$\theta_{11}^{AB} = \cos^{-1}(v_{1x}^A v_{1x}^B + v_{1y}^A v_{1y}^B + v_{1z}^A v_{1z}^B) \quad (2.2)$$

Figure 2.8 illustrates the case where two components are “long & short”. As in Figure 2.7, these two components are parts of the same pipe located in two adjacent octants. Unlike Figure 2.7, since Component B is “short”, its eigenvector corresponding to the largest eigenvalue is no longer along the true orientation of the pipe – however, its second eigenvector is along the orientation of the pipe. This example serves as the motivation behind our connectivity analysis for a “long & short” pair of components – we compute the angle between combinations of the first eigenvector from the “long” component and first two eigenvectors from the “short” component (i.e. $\theta_{11}^{AB}$, $\theta_{12}^{AB}$, where the first subscript
denotes the eigenvector from the “long” Component $A$ and the second subscript denotes the eigenvector from the “short” Component $B$). If either $\theta_{11}^{AB}$ or $\theta_{12}^{AB}$ is less than $\theta_{thr}$, the orientation criterion is considered satisfied.

For the remaining case where the two components are “short & short”, the orientation connectivity criterion is not evaluated and bypassed. The rationale behind this choice is that there is not sufficient contextual information available in two short segments, usually smaller components residing at the deepest levels of the tree, for orientation analysis to be meaningfully applied. This rationale was validated through experimentation, where we computed the angles between pairwise combinations of all eigenvectors in two “short” components, finding that orientation analysis of two “short” components does not facilitate the merging process for a piping system. Only in the case where both “short” components are determined to be planar through curvature analysis (Subsection 2.3.4.3) is orientation considered. The use of contextual information by categorizing components as “long” or “short” and examining their orientations are what improves the segmentation of
piping point clouds, and distinguishes the proposed technique from prior work (Rabbani, 2006; Wang and Tseng, 2010; Liu et al., 2013).

2.3.4.3 Curvature Connectivity Criteria

If both the spatial and orientation connectivity criteria are satisfied for the two components, then curvature connectivity is evaluated. Curvature connectivity, like orientation connectivity, is a higher level criterion using contextual information to facilitate the merging of cylindrical parts into pipes and planar parts into walls. In this work, the principal curvatures, denoted by κ₁ and κ₂, are used to evaluate the curvature connectivity criterion. Curvature has been examined extensively in earlier works for the segmentation and classification of objects in range images (Besl and Jain, 1986; Fan et al., 1986; Flynn and Jain, 1989; Vemuri et al., 1986). However, curvature estimates are highly sensitive to quantization noise, requiring multiple smoothings to obtain stable estimates (Flynn and Jain, 1989). Current generation LIDAR scanners exhibit less noise, but the scanned point cloud data is unstructured. In order for the principal curvatures to be computed, a surface must first be fitted to the points within a component. An example component is shown on the left in Figure 2.9, along with its three eigenvectors. The eigenvector shown in black is the third eigenvector corresponding to the smallest eigenvalue, and serves as an estimate of the surface normal of the component. The points in this component cannot be represented in the form \( z = f(x, y) \), as there may be multiple \( z \) values for a given \((x,y)\) coordinate. Therefore, the points within the component are first rotated so that its surface normal is aligned along the \( z \)-direction. The rotated points now have only a single \( z \) value for any given \((x,y)\) coordinate, and can be represented as a surface \( z = f(x, y) \).
Figure 2.9 Illustration of surface fitting procedure. A component is first rotated so that its surface normal (black vector) is pointing in the z-direction, then a 2nd order polynomial is fitted to the rotated points.

Since the main primitives of interest in this work are cylinders and planes, which can be represented by a 2nd order polynomial, the points in each component are fitted to a quadratic equation (Equation 2.3).

\[ f(x, y) = ax^2 + bxy + cy^2 + d \]  

(2.3)

For components with less than 180° hemispheric coverage, the fitting approach described previously can be used. However, components with more than 180° hemispheric coverage cannot be represented in the form \( z = f(x, y) \). Note that a single LIDAR scan will never acquire more than 180° coverage of an object’s surface, while combined scans taken from multiple locations is expected to produce more than 180° coverage. For such scenarios, the fitting technique is extended to handle more than 180° coverage by dividing a component into an upper half surface and a lower half surface (each with no more than 180° coverage), prior to polynomial surface fitting. An illustration of this procedure is presented in Figure 2.10 using a synthetic cylinder with 300° of coverage. First, the component is projected along the first eigenvector (corresponding to the largest eigenvalue) onto a plane formed by the second and third eigenvectors. Circle fitting is performed in the projected 2D space. If the root mean square error (RMSE) of the circle fitting is less
than one, suggesting that the component is likely to be a cylinder, the points are split into upper and lower halves of the circle. The corresponding points in the original 3D space form two cylindrical surfaces. The surface with more points is chosen to undergo polynomial surface fitting. For a planar component, the RMSE of the circle fitting will be high and the planar component will not be divided into two surfaces. This is in fact desired, as multiple scans of a plane will still form a plane, which can be readily represented as a polynomial for curvature estimation. The focus of the curvature connectivity analysis is only on cylinders and planes, which will be explained in more detail in the next paragraphs.

Figure 2.10 Illustration of fitting procedure for cylinders/pipes with >180° coverage. Given a cylindrical component, it is projected onto the plane formed by the 2nd and 3rd eigenvectors. The 2D points are fitted to a circle, and the corresponding surface in 3D space with more points undergoes polynomial fitting.

Given the fitted polynomial, the principal curvatures $\kappa_1$ and $\kappa_2$ of the surface $f(x,y)$ can be computed from the polynomial coefficients as follows (Brown, 2014):

$$\kappa_1 = (a + c) + \sqrt{(c - a)^2 + b^2} \quad (2.4)$$

$$\kappa_2 = (a + c) - \sqrt{(c - a)^2 + b^2} \quad (2.5)$$

For the cylindrical primitive, one of its principal curvatures is zero (Flynn and Jain, 1989). For the planar primitive, both its principal curvatures are zero. Since LIDAR data contains noise, the point cloud of a pipe will not be a perfect cylinder and the point cloud
of a wall will not be a perfect plane – the principal curvature will not be exactly zero. Letting $\kappa_1$ be the larger of the two principal curvatures, we define a component to be cylindrical if $(\kappa_1>1, \kappa_2<1)$ and planar if $(\kappa_1<1, \kappa_2<1)$. Furthermore, the coefficient of determination ($R^2$), a value typically between 0 and 1 measuring the goodness of fit, is evaluated for each component’s fitted surface. If a pair of components A and B have satisfied the proximity and orientation connectivity criteria and $R^2>0.5$ for both their fitted surface, then each component is categorized as cylindrical, planar, or neither. If both components are cylindrical $(\kappa_1>1$ and $\kappa_2<1)$, then the curvature connectivity criterion is satisfied if $\kappa_1^A$ of component A and $\kappa_1^B$ of component B are within a factor of two of each other. Since the principal curvature of a cylinder is directly related to its radius, the conceptual rationale behind the curvature connectivity criterion is that two cylindrical components must have similar radii. If both fitted surfaces are planar $(\kappa_1<1$ and $\kappa_2<1)$, the two components have curvature connectivity if the angle between their surface normals is less than $\theta_{thr}$. Conceptually, this means that two planar surfaces are connected if they have similar planar orientation. For fitted surface with $R^2<0.5$, as well as for fitted surfaces that are neither cylinders nor planes, curvature connectivity analysis is bypassed, with only the proximity and orientation criteria considered during merging. The rationale is that the proposed technique is designed for segmentation of piping scenes, and that complex shapes which cannot be fitted by a quadratic polynomial should not undergo curvature connectivity analysis for merging.

The curvature connectivity criterion described above facilitates the merging of cylindrical components and planar components, especially at higher levels of the octree. In addition, this curvature analysis is important in identifying potential candidates for
pipes and walls from the scene to designate for later CAD modeling. When the bottom-up merging process is completed, the components of the root node represent the final segmentation results. The segments with \((\kappa_1 > 1, \kappa_2 < 1)\) can be labeled as pipes, while those with \((\kappa_1 < 1, \kappa_2 < 1)\) can be labeled as walls, floor, ceiling, etc. This capability also distinguishes the proposed segmentation technique from prior work, whose segmentation techniques do not identify potential piping segments.

**2.3.5 Sequential and Parallel Implementations**

In its sequential implementation, the proposed octree-based bottom-up segmentation algorithm is computationally efficient in terms of execution time. Unlike the region growing technique of Rabbani (2006), the proposed technique does not need to compute the \(k\)-nearest neighbors of each point, but rather processes blocks of data (i.e. octants) at a time using matrix operations and graph theory based analysis. The proposed technique is however more memory intensive compared to Rabbani’s technique, requiring approximately 4GB of memory (though this requirement can be reduced by adjusting the bin capacity of the octree decomposition). Current mid-range laptops and desktops typically have 8-16GB or more of memory. Therefore, runtime is a more critical consideration for segmentation algorithms than memory usage. Furthermore, typical processors are multi-core, enabling a computer to run multiple threads in parallel.

The proposed bottom-up segmentation algorithm is designed to be highly parallelizable, and is capable of exploiting multiple cores for parallel execution. Both the splitting process and the merging process can be parallelized. In its sequential form, the splitting procedure iterates over every leaf node/octant of the octree structure, performing graph connectivity analysis of the points in each octant. Since each iteration processes a
different leaf node and is therefore independent of any previous iteration, this loop can be readily parallelized using Matlab’s Parallel Computing Toolbox, specifically using its “parfor” functionality.

The merging process can also be parallelized. Unlike the splitting process, the merging process consists of nested for loops. The outer loop iterates over each tree level while each inner loop iterates over every group of eight adjacent octants at a given tree level. The iterations of the outer loop are dependent because of the bottom-up merging procedure, but at a given tree level, the iterations over every group of eight adjacent octants are independent – we therefore parallelize this inner loop. The speedup in execution time achieved with parallelization of the split and merge procedures are presented in Section 2.4.

2.4 Results

The proposed octree-based segmentation algorithm is evaluated on two terrestrial LIDAR datasets. The first is a high point density dataset of a steel mill acquired using a Faro Focus 3D S120 Laser Scanner, while the second is a dataset of a pilot chemical plant at Purdue University. The chemical plant scene is an older dataset acquired using a Cyra 2500 3D Laser Scanner – this dataset has lower point density as well as higher noise levels. The organization of this section is as follows. Section 2.4.1 presents results on a smaller region within the steel mill scene, with the purpose of clearly illustrating the steps of the proposed algorithm from octree decomposition to split and merge. Note that focusing on the smaller region enables the finer details to be visualized. Section 2.4.2 then presents results on the whole steel mill with one scan. Section 2.4.3 presents results on the whole steel mill with combined scans. Evaluation results on the chemical plant dataset are
similarly presented in Sections 2.4.4 and 2.4.5. In Section 2.4.4, we also compare the results of the proposed octree-based segmentation algorithm with the results of Rabbani’s region growing based segmentation method. Section 2.4.6 describes parallelization results with respect to runtime and memory usage.

2.4.1 Results from Subset of Steel Mill Scan

The point cloud of the steel mill subset is shown in Figure 2.11(a), consisting of a total of 552,368 points extracted from a single LIDAR scan and covering a volume of approximately 0.9m×3.7m×1.3m in size. Note that this is a subset of the entire scan. The average point spacing of this point cloud is very high at 0.0011 m.

Figure 2.11 (a) Point cloud of a region within the steel mill consisting of 552,368 points with an average point spacing of 0.0011 m, and (b) octants following octree decomposition.

The first step of the proposed octree-based segmentation algorithm performs octree decomposition, recursively dividing nodes/octants until the number of points in each octant is less than the user defined maximum bin capacity (set at 10000 here). For this point cloud, the octree has 5 levels (root node being level 0) containing a total of 345 nodes as shown in Figure 2.11(b), of which 302 are leaf nodes/octants.
Following octree decomposition, graph connectivity analysis based on the $\epsilon$ distance threshold is used to split the points within each octant into spatially unconnected components. In choosing the value for $\epsilon$, it is undesirable to set $\epsilon$ to be equal to the average point spacing, which would be too small and result in a large number of unconnected components. The value of $\epsilon$ is set to 0.01 m here, an order of magnitude greater than the average point spacing for the point cloud.

![Figure 2.12](image)

Figure 2.12 Three example octants with points displayed before and after graph theory based splitting into spatially unconnected components. Borders denote octant spatial boundaries.

Figure 2.12 shows several examples of octant points before and after the splitting procedure is applied. In Figure 2.12(a), all the points in the octant are connected, generating a single component after splitting. In Figure 2.12(b), two narrow pipes are split into two unconnected components. In Figure 2.12(c), the octant boundary occurs at the
middle section of a pipe, causing the resulting points to be split into two spatially unconnected components. Note that during the merging process, these two components will be combined together with the middle section, as conceptually illustrated in Figure 2.4 previously.

After all the leaf nodes undergo the splitting procedure, the bottom-up merging process commences at the lowest level of the octree, merging components connected across adjacent octants. Once all groups of eight adjacent octants have been combined at the given octree level, the process is repeated at the next higher up octree level until the root node is reached. Examples of adjacent octants at octrees level 5 to 2, before and after merging, are shown in Figure 2.13(a)-(d) for this steel mill subset. At the deepest tree level in Figure 2.13(a), smaller components that are part of a cylindrical surface are merged together primarily based on proximity connectivity. At higher octree levels shown in Figure 2.13(b)-(c), orientations of components also play a key role during merging. Note especially that the two components forming an elbow in Figure 2.13(b) are merged together, but remain separate from the horizontal piping segment because of the difference in orientation angles. As this region of the steel mill does not have a straight pipe composed of two segments with different radii, curvature connectivity analysis is not demonstrated here. However, the principal curvatures computed during curvature connectivity analysis enables the identification of components that are likely to be pipes.
Figure 2.13 Examples of groups of 8 adjacent octants at each tree level for the steel mill region, before and after merging.

Figure 2.14 shows the eight adjacent octants at level 1, and the final merging results on the right – these components represent the final segments that are the output of the complete segmentation algorithm.
Figure 2.14 Components of octants at Level 1 are merged to generate final segmentation results for the steel mill region.

Given the final segmentation results, those segments that are likely to be pipes can be identified based on their principal curvatures (Section 2.3.4.3). For a perfect cylinder, one of the principal curvatures is zero while the other principal curvature is positive. Since LIDAR data contains noise, we define a segment to be a pipe candidate if its $\kappa_1 > 5$ and $\kappa_2 < 1$, letting $\kappa_1$ be the larger of the two principal curvatures. The thresholds of 5 and 1 for $\kappa_1$ and $\kappa_2$, respectively, were chosen using a histogram based analysis of a population of cylindrical and planar components – these values generate the best separation between the two primitives. As the principal curvatures are computed from the fitted 2nd order polynomial surface, the goodness of fit ($R^2$) must also be considered. For this steel mill scene, a pipe candidate must have $R^2 > 0.6$ – these segments are displayed in Figure 2.15. The segments that are piping candidates can then be passed to cylinder parameter estimation and CAD model generation.
Choosing a setting for $R^2$ when identifying piping candidates should be dependent on LIDAR sensor characteristics as well as scene characteristics. High density scans of piping systems with large radius pipes will be relatively less noisy than lower density scans of piping systems dominated by smaller pipes (due to scattering/deflection of laser pulses by small radii pipes with highly curved surfaces). One suggested approach in setting $R^2$ is to first ground truth a representative subset of the LIDAR scan using human expert(s), and then vary the setting of $R^2$ until the pipes identified by the algorithm match well with the ground truth. This setting can then be used to identify LIDAR pipes from the whole LIDAR scan.

To provide a more quantitative evaluation of the performance of the proposed segmentation algorithm, we asked human subjects to determine the total number of segments in the steel mill subset as well as identify the segments which are pipes. Subjects were shown the raw point cloud, which they could rotate and visualize at different perspectives in Matlab. Table 2.1 tabulates the results from three subjects, as well as the results from the proposed algorithm. While there is some inter-subject variability for the
number segments, the number of pipes is fairly consistent. The proposed algorithm produces a slight over-segmentation, but the number of identified pipes matches well with the human results.

Table 2.1 Number of segments and number of pipes for steel mill subset determined by human subjects and the proposed segmentation algorithm.

<table>
<thead>
<tr>
<th>Subject</th>
<th>Number of Segments</th>
<th>Number of Pipes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subject 1</td>
<td>21</td>
<td>11</td>
</tr>
<tr>
<td>Subject 2</td>
<td>29</td>
<td>11</td>
</tr>
<tr>
<td>Subject 3</td>
<td>29</td>
<td>13</td>
</tr>
<tr>
<td>Proposed Algorithm</td>
<td>34</td>
<td>11</td>
</tr>
</tbody>
</table>

For the steel mill subset shown previously in Figure 2.11, the majority of the pipes are aligned either along the x-, y-, or z-axis directions. We demonstrate that the proposed octree-based segmentation algorithm is capable of handling obliquely oriented pipes using the following example. Given the steel mill subset, we first rotate the point cloud about the x-, y-, and z-axis, as shown in Figure 2.16.

Figure 2.16 Original point cloud of steel mill subset and rotated point cloud. Rotation parameters are 30°, 20°, and 60° about the x-, y-, and z-axis, respectively.
The rotation parameters used in Figure 2.16 are 30°, 20°, and 60° about the x-, y-, and z-axis, respectively, representing a highly rotated scenario. This large rotation will cause the cutting plane between the components to be oblique to the primary cylinder axis. The rotated point cloud is then segmented using the octree-based segmentation algorithm – the results are shown in Figure 2.17. As can be observed, the segmentation results are relatively unchanged, even with the large rotation. This demonstrates the robustness of the proposed approach for obliquely oriented pipes.

Figure 2.17 Segmentation results using rotated steel mill subset.

2.4.2 Results from Whole Steel Mill Scene

The previous section used a smaller region within the steel mill point cloud data to show the finer details of the proposed segmentation algorithm. In this section, segmentation results for the whole steel mill dataset are presented, with all algorithm parameter settings the same as before. Figure 2.18(a) shows the whole point cloud consisting of 12,462,216 points from a single scan, with spatial dimensions of approximately 7m×5m×4.75m. The final segmentation results are shown in Figure 2.18(b), demonstrating the robustness of the proposed bottom-up octree-based
segmentation technique. As can be observed, the proposed algorithm generally segments the objects correctly – neither over-segmenting nor under-segmenting the majority of the pipes and walls in the scene. Based on curvature analysis, Figure 2.18 (c) displays the segments automatically identified as pipe candidates, which can be then used to generate CAD models.

\[
\begin{align*}
\text{(a) Steel Mill Point Cloud} & \quad \text{(b) Segmentation Results} & \quad \text{(c) Pipe Candidates}
\end{align*}
\]

Figure 2.18 (a) Single scan of complete steel mill, (b) final segmentation results, and (c) potential pipes ($\kappa_1 > 5$, $\kappa_2 < 1$, $R^2 > 0.3$).

The proposed technique is highly scalable, generating accurate results in this large scale dataset of ~12 million points without the need to change any parameter settings from before. Furthermore, a laptop computer with 8GB of memory was able to run the proposed segmentation algorithm on this large point cloud dataset, demonstrating its practicality and computational efficiency.

2.4.3 Results from Combined Scans of Steel Mill Scene

In this section, segmentation results on the combined scans of the steel mill scene are presented, using the same algorithm parameters as before. The combined point cloud across the six scans contains a total of 24,138,521 points and occupies a volume of approximately 8.5m×5m×4.75m. Figure 2.19(a) shows the combined point cloud and Figure 2.19(b) shows the segmentation results of the scene.
From all the segments, 74 pipe candidates are identified by the algorithm, shown in Figure 2.20(a) using the parameter settings $\kappa_1 > 5$, $\kappa_2 < 1$, and $R^2 > 0.5$. To assess the accuracy of the identified pipe candidates, each pipe candidate is examined manually and labeled as a true cylinder or a false identification. As can be observed in Figure 2.20(a), the majority of the larger pipe segments have been correctly identified. Point clouds of smaller pipes with high curvature are inherently noisy (due to the scattering/deflection effect of laser pulses on highly curved surfaces), and therefore are difficult to be identified as cylinders (e.g. railings of the stairs). Of the 74 pipe candidates, 56 are true pipe elements, displayed in Figure 2.20(b), resulting in an identification accuracy of 75.7%.
Figure 2.20 (a) 74 pipe candidates identified by the algorithm using $k_1 > 5$, $k_2 < 1$, and $R^2 > 0.5$, and (b) 56 of the pipe candidates are actual piping elements, determined through manual assessment.

Some examples of segments falsely identified as pipe candidates are shown in Figure 2.21. These falsely identified segments generally consist of two intersecting planes that had not been divided into separate components during the splitting process. Recall that during the splitting process, points within each octant are split into separate components based on spatial connectivity. Therefore, two intersecting planes that are located within a single octant would not be split. Furthermore, the intersecting planes form an L-shaped segment that produces principal curvature estimates similar to that of a cylinder, as well as a relatively high goodness of fit ($R^2$) value from the 2nd order degree polynomial fitting, resulting in these segments being falsely identified as pipe candidates. This suggests that the segmentation technique can be extended and improved by enhancing the complexity of the splitting process. Instead of only relying on spatial connectivity during splitting, surface normal orientations can be also considered. Given the points within an octant generated by octree decomposition, first compute the surface normal of
each point. Then a clustering technique can be applied to group the points into cluster(s) based on similarity of surface normal orientation. The challenge lies in developing a split algorithm which can automatically determine the number of clusters from the surface normal distribution, and then perform graph analysis on the cluster to separate the points into spatially unconnected components. Though difficult, this approach is expected to improve overall segmentation results.

Figure 2.21 Examples of segments falsely identified as pipe candidates.

2.4.4 Results from Subset of Chemical Plant Scan and Comparison with Prior Work

For the chemical plant scene, we first focus on a subset of approximately 0.8m × 0.5m × 0.5m containing 42,903 total points, to allow for a detailed visualization of the segmentation results. The chemical plant dataset is an older dataset, with higher noise characteristics as well as a substantially higher point spacing of 0.0023m (compared to the 0.0011m point spacing of the steel mill dataset). Consequently, the ε distance threshold must be increased – set at 0.015m for this scene. With the larger point spacing, the bin capacity for the octree decomposition must also be reduced (set at 2,500 here) in order to create an initial over-segmentation. Recall the proposed algorithm is designed to intelligently merge the over-segmented components from the octree decomposition and graph theory based splitting. All other parameters for the proposed segmentation
algorithm remain the same as for the steel mill scene. Figure 2.22 shows the raw point cloud of the chemical plant subset, the segmentation results, and the final segments identified as piping candidates. As can be observed, the proposed octree-based segmentation technique effectively separates piping segments – spatially connected but differently oriented pipes are separated into individual segments, as typically desired. However, due to a combination of factors (lower point spacing, higher noise characteristics, and pipe junctions that frequently occur in this piping system), only four pipe candidates are identified in the region, even with a lowered $R^2$ threshold.

![Figure 2.22](image)

Figure 2.22 (a) Point cloud of a subset of the chemical plant scan, (b) final segmentation results, and (c) potential pipes ($\kappa_1 > 5$, $\kappa_2 < 1$, $R^2 > 0.1$).

To demonstrate the effectiveness of the proposed octree-based segmentation algorithm, we compare with the recent and widely cited region growing based segmentation algorithm of Rabbani et al. (2006). Figure 2.21 shows the segmentation results from both the proposed approach and Rabbani’s region growing technique on this chemical plant subset. For the region growing technique, the key smoothness constraint parameter for surface normals is set at $15^\circ$, as in Rabbani et al. (2006). While the proposed octree-based segmentation technique is able to separate spatially connected pipes which have distinct axial orientations, Rabbani’s region growing technique produces a single
segment (red segment in Figure 2.23(b)), since these spatially connected pipe elements have smooth surface normal transitions via the connecting pipe elbows. Furthermore, whereas the proposed algorithm took minutes to run, Rabbani’s technique (our Matlab implementation) took hours to process this chemical plant point cloud. Although Rabbani’s region growing technique is very memory efficient, this is less of a concern for algorithm development with the constant advances in size of random access memory. However, because of its intractable runtime for larger point clouds, we only tested Rabbani’s technique in this section for this smaller point cloud.

As in Section 2.4.1, human subjects are also asked to count the total number of segments in the chemical plant subset as well as determine which segments are pipes. The results are tabulated in Table 2.2. For this older dataset with higher noise, there is large inter-subject variability, suggesting that it is challenging for even human subjects to count the total number of segments and identify the pipes. The number of segments generated
by the proposed algorithm falls within the range of human results, but identifies fewer pipe segments, due to the adverse impact of the high noise level.

Table 2.2 Number of segments and number of pipes for chemical plant subset determined by human subjects and the proposed segmentation algorithm.

<table>
<thead>
<tr>
<th></th>
<th>Number of Segments</th>
<th>Number of Pipes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subject 1</td>
<td>13</td>
<td>9</td>
</tr>
<tr>
<td>Subject 2</td>
<td>27</td>
<td>14</td>
</tr>
<tr>
<td>Subject 3</td>
<td>19</td>
<td>8</td>
</tr>
<tr>
<td>Rabbani’s Algorithm</td>
<td>26</td>
<td>N/A</td>
</tr>
<tr>
<td>Proposed Algorithm</td>
<td>22</td>
<td>4</td>
</tr>
</tbody>
</table>

2.4.5 Results from Chemical Plant Scene

The entire chemical plant scan from the Chemical Engineering Building at Purdue University covers a region approximately $2.8m \times 3.2m \times 2.8m$ in dimension. The point cloud contains a total of 641,777 points and has a point spacing of 0.0023m. The parameter values used here remain the same as for the chemical plant subset in the previous section—the bin capacity is set at 2,500 and $\varepsilon=0.015m$. The chemical plant scene in Figure 2.24(a) is highly complicated, containing many short pipes as well as other objects. Figure 2.24(b) displays the segmentation results and Figure 2.24(c) shows the segments that are identified as pipe segments. While most of the objects are properly segmented from a visual examination, some over-segmentation is produced using the proposed algorithm—noticeably the large cylindrical vessel in the lower right region is separated into two segments (blue and red). While most of the major pipes appear to be properly identified in Figure 2.24(c), some of the pipe segments were missed and some objects were misidentified as pipe candidates. This lower performance (compared to the results using
the newer steel mill dataset) can be expected, due to the lower point spacing and higher noise characteristics of the chemical plant dataset.

![Figure 2.24](image)

**Figure 2.24** (a) Point cloud of chemical plant scene, (b) final segmentation results, and (c) potential pipes ($\kappa_1>5$, $\kappa_2<1$, $R_2>0.1$).

### 2.4.6 Sequential and Parallel Execution Runtime

In this section, we present runtimes of the proposed segmentation approach in both its sequential and parallel forms. Both implementations are programmed in Matlab, and the parallel version uses the Parallel Computing Toolbox. The dataset used for runtime analysis is the steel mill region containing 552,368 points (Section 2.4.1, Figure 2.11). Note that the bin capacity is set to 5,000 here so that the memory consumption of all the worker processes does not exceed the total available memory. Since the core contribution of this work is the novel split and merge procedures following octree decomposition, only runtimes for the split procedure and for the merge procedure are presented in Table 2.3. The octree decomposition part of the algorithm is not parallelized, but is already efficient, with a runtime of 1.60 s for this point cloud. The computer used is a 2010 Dell XPS laptop with a Quad Core i7 2.20GHz processor and 8GB of memory.
Table 2.3 Runtimes for the split and merge procedures using sequential and parallel (2 & 4 workers) implementations on a 2010 Dell XPS laptop with a Quad Core i7 2.20GHz processor and 8GB of memory.

<table>
<thead>
<tr>
<th>Procedure</th>
<th>Sequential Runtime</th>
<th>Parallel Runtime 2 Workers</th>
<th>Parallel Runtime 4 Workers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Split Procedure</td>
<td>77.77 s</td>
<td>54.33 s</td>
<td>36.68 s</td>
</tr>
<tr>
<td>Merge Procedure</td>
<td>17.20 s</td>
<td>17.15 s</td>
<td>15.29 s</td>
</tr>
</tbody>
</table>

As discussed in Section 2.3.5, the split procedure consists of a single loop that is highly parallelizable, resulting in substantial reduction in runtime from sequential to parallel implementation, especially with 4 workers. Since the processor used here has 4 cores, using more than 4 workers is not practical. The merge procedure, consisting of multiple nested loops, is not as inherently parallelizable – a slight reduction in runtime of approximately 10% is achieved with the parallel merge implementation using 4 workers. However, a more substantial reduction in runtime for the merge procedure is expected with larger scale point clouds and additional available cores. Overall, the complete sequential algorithm takes a total of 96.57s to run, and the parallel version using 4 workers takes a total of 53.07s to run on this point cloud of half a million points, demonstrating that parallelization of the proposed octree-based segmentation algorithm achieves further computational efficiency.

2.5 Conclusions

Efficient segmentation algorithms for large scale terrestrial LIDAR data are critical for automated exploitation of 3D point clouds, with the ultimate goal of CAD model generation. In this work, the octree-based segmentation approach is designed for terrestrial LIDAR scans of industrial sites containing piping systems, such as steel mills, oil refineries, and chemical plants. The proposed approach uses octree decomposition to recursively
divide the scene into octants until each octant contains no more than a certain number of points, referred to as the bin capacity. The intent for the octree decomposition is to produce an over-segmentation of the terrestrial LIDAR scene. The core contributions of this work are the split and merge procedures that first separate octant points into spatially unconnected components, and then intelligently merges these components using a series of connectivity criteria across octants, proceeding from the deepest level of the tree and progressing recursively up the tree until the root node is reached. Part of what distinguishes it from prior work is that the proposed technique does not seek to find properties of each point individually, such as the surface normal through k-nearest neighbors. Rather, the proposed technique is based on the octant structure, processing blocks of data at a time. Although memory intensive, the octree-based segmentation approach is highly computationally efficient in terms of runtime as well as being inherently parallelizable. During the merging procedure, the proposed technique uses a series of connectivity criteria (proximity, orientation, and curvature) to intelligently merge and label components across octants. The orientation analysis exploits higher level contextual information, distinguishing between “long” and “short” components. Furthermore, the curvature analysis identifies potential planar and cylindrical surfaces through polynomial surface fitting, facilitating merging of these elements into walls and pipes. These higher level connectivity criteria facilitates the segmentation and labeling of point clouds of piping scenes. Compared to Rabbani’s region growing based segmentation algorithm, which is also designed to process point clouds of piping systems, the proposed algorithm has some advantages due to these higher level connectivity criteria. However, the more complex connectivity analysis also leads to a higher number of parameters for the proposed
algorithm. Table 4 provides a full list of algorithm parameters, along with their description and guidelines for setting the values.

Table 2.4 List of parameters and their descriptions for the proposed octree-based segmentation algorithm, along with general guidelines for setting these parameters.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Guidelines for Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\varepsilon$</td>
<td>Distance threshold for spatial connectivity used in both split and merge procedures</td>
<td>Depends on point spacing of point cloud. Set at 5-10 times the point spacing.</td>
</tr>
<tr>
<td>Bin Capacity</td>
<td>Maximum number of points per octant in octree decomposition</td>
<td>Depends on point spacing of point cloud and available system memory. Maximum of 10,000 for 8GB system.</td>
</tr>
<tr>
<td>$\theta_{thr}$</td>
<td>Angle threshold for orientation connectivity analysis</td>
<td>Set at 10º. Does not require fine tuning across datasets.</td>
</tr>
<tr>
<td>$r_{thr}$</td>
<td>Ratio threshold of first and second eigenvalues to determine “long” vs “short” components during merging procedure</td>
<td>Ratio set at 5. Does not require fine tuning across datasets.</td>
</tr>
<tr>
<td>$\kappa_1$</td>
<td>Larger of two principal curvatures. Used in curvature connectivity analysis.</td>
<td>Segment is cylindrical if ($\kappa_1 &gt; 1$, $\kappa_2 &lt; 1$) Segment is planar if ($\kappa_1 &lt; 1$, $\kappa_2 &lt; 1$)</td>
</tr>
<tr>
<td>$\kappa_2$</td>
<td>Smaller of two principal curvatures. Used in curvature connectivity analysis.</td>
<td>Do not require fine tuning across datasets.</td>
</tr>
<tr>
<td>$R^2$</td>
<td>Goodness of fit for polynomial surface fitting. Used to identify pipe candidates.</td>
<td>Depends on noise characteristics for data. Value between [0,1]. Select threshold based on analysis of previous processed data, as described in Sections 4.1.</td>
</tr>
</tbody>
</table>

A total of seven parameters are utilized by the octree-based segmentation technique. The most critical parameter is $\varepsilon$, the distance threshold used during both the split and merge procedures. From experiments conducted in this work, setting $\varepsilon$ to be five to ten times the average point spacing of the dataset works well. The second parameter that must be adjusted is the bin capacity, which depends on both the point spacing of the dataset as well as memory constraints of the system. Most importantly, setting the bin capacity to produce an initial over-segmentation is desirable. A setting of 10,000 for a point spacing of ~0.001m (e.g. steel mill scene) works well for a system with 8GB of memory. The other parameters ($\theta_{thr}$, $r_{thr}$, $\kappa_1$, $\kappa_2$) do not have to be fine-tuned across datasets, as demonstrated using the two very different datasets. The $R^2$ parameter used to identify pipe candidates...
can be set as described in Sections 2.4.1. Therefore, of the seven parameters used by the proposed segmentation technique, only two critical parameters, \( \epsilon \) and bin capacity, that depend on the point spacing of the terrestrial LIDAR data have to be adjusted for each dataset. Guidance is provided on how to assign values to these parameters.

In conclusion, an octree-based segmentation technique is presented in this work, with extensive evaluation performed on two very different scenes from a steel mill and a chemical plant. Given the octree decomposition of a scene, the split and merge procedures use graph theory to split the octant points into spatially unconnected components, and then intelligently merge components across octants recursively from the deepest tree level upwards. This technique is fundamentally a bottom-up design that is highly computationally efficient, utilizing a series of connectivity criteria to accurately segment terrestrial LIDAR scans of piping systems at industrial sites. Furthermore, the proposed technique is capable of identifying potential pipe candidates from the segmentation results, facilitating the ultimate objective of CAD modeling of terrestrial LIDAR data.
CHAPTER 3. ESTIMATION OF CYLINDER ORIENTATION

3.1 Introduction

Several previous approaches have been proposed in literature to estimate cylinder parameters in 3D point clouds. The work by Bolles and Fischler (1981) proposed a random sample consensus (RANSAC; Fischler and Bolles, 1981) based method for cylinder detection in range data. Chaperon and Goulette (2001) also used RANSAC in conjunction with the Gaussian image for cylinder extraction. Lukas et al. (1998) and Marshall et al. (2001) developed methods based on non-linear least squares, minimizing the distance of the point cloud from the fitted cylinder. However, geometric fitting using least squares has been shown to be sensitive to outliers (Bjorck, 1996). Furthermore, the iterative non-linear least squares methods may stop converge to local minima if the initialization estimates are poor. Mulat et al. (2008) proposed cylinder orientation estimators based on finite impulse response filters, introducing gradient masks for orientation estimation in 3D space. However, the accuracy of the technique suffers in the case of noisy data and oblique cylinders. For robustness in the presence of noise and outliers, the Hough transform has been an effective technique, first proposed by Hough (1960) to recognize complex patterns and later extended to find lines and curves (Duda and Hart, 1972) as well as circles (Kimme, 1975). For simple primitives such as lines, circles, and planes, the parameterization can be achieved with two or three parameters,
resulting in a two or three dimensional Hough parameter space. However, cylinders have five degrees of freedom, requiring a 5D Hough parameter space, for which the computation complexity in terms of both runtime and memory become prohibitive. Rabbani (2005) proposed an efficient two stage process to estimate cylinder parameters using the Hough transform, consisting of a novel first stage to estimate the axis orientation using surface normals from points in the 3D data, followed by projection onto a 2D space and estimation of the cylinder position and radius using the standard circle finding technique of Kimme et al. (1975). Rabbani's technique (Rabbani, 2005) is a recent and widely cited work in literature for cylinder parameter estimation and still represents the state-of-the-art. However, the Hough transform approach is a sampling based method that is fundamentally limited by the cell size used. The most recent work on cylinder detection by Liu et al. (2013) focuses on large-scale point cloud data of pipeline plants, first estimating the major pipeline directions and then projecting the point cloud onto planes normal to these directions - cylinders are then detected as circles in the 2D space. The work of Liu et al. (2013) does not focus on the accuracy of axis orientation estimation of individual cylinders, which is the motivation behind this work.

We propose a novel mathematical formulation to accurately and efficiently estimate the cylinder axis orientation using gradient descent optimization of an angular distance cost function. The cost function is based on the concept that surface normals of points in a cylinder point cloud are perpendicular to the cylinder axis. We demonstrate the accuracy of the proposed technique, and compare with the state-of-the-art approach of Rabbani (2005). Section 3.2 gives a brief overview of Rabbani’s two stage Hough transform based approach. Section 3.3 describes the formulation of the proposed technique, and Section
3.4 demonstrates its accuracy on both synthetic and real point cloud data. Section 3.5 presents the conclusions.

3.2 Background

In this section, we present a brief overview of the two stage Hough transform approach for efficient cylinder parameter estimation (Rabbani, 2005), which we use to compare with our proposed method. Rabbani (2005) utilizes a five parameter parameterization of the cylinder. As illustrated in Figure 3.1, \( a(\theta, \varphi) \) is the axis (defined in terms of the inclination angle \( \theta \) and azimuth \( \varphi \)), \( r \) is the radius of the cylinder, and \( Q \) is a point on the axis following projection along the axis orientation. Under this representation, the azimuth \( \varphi \) is ambiguous for a cylinder that is oriented perfectly along the \( z \)-axis (i.e. when \( \theta = 0 \)). However, the proposed solution will not be singular in such a scenario, since no division by zero will occur in the proposed mathematical formulation (Section 3.3).

Also illustrated on Figure 3.1 is a point on the cylinder, \( P_i \), and its surface normal, \( n_i \). The surface normal can be computed using eigen-analysis of the spatial “covariance matrix” of the \( k \)-nearest neighbors of each point.

Figure 3.1 Cylinder parameterization with \( \theta, \varphi, u, v, \) and \( r \). In the diagram, \( a \) is the axis (defined in terms of azimuth and elevation), \( r \) is the radius of the cylinder, and \( Q \) is a point on the axis following projection along the axis orientation.
The first and most critical stage of the two stage Hough transform approach (Rabbani, 2005) estimates the cylinder orientation. Given a 3D cylinder point cloud, the surface normal of each point is first computed. Each surface normal traces a great circle in the Gaussian sphere, as formulated by Carmo (1976). Note that a great circle is formed from the intersection of the unit sphere with a plane perpendicular to a given point's surface normal. Consequently, the intersection of the great circles traced by all the surface normals estimates the orientation of the cylinder axis (Rabbani, 2005). Rabbani's approach is a discrete space implementation of this concept, requiring a sampled Hough space to represent the Hough Gaussian sphere. Each point votes into the cells of the Hough Gaussian sphere that its great circle passes through. The cell with the highest accumulator (i.e. votes) then represents the estimated cylinder orientation. This voting procedure is highly dependent on the sampling density of the Hough space - the accuracy of the estimated orientation is therefore fundamentally limited by the cell size.

Following estimation of the axis orientation, the second stage estimates the cylinder radius $r$ and position $Q(u,v)$. Using the estimated axis orientation, the 3D cylinder point cloud is projected onto the plane perpendicular to the cylinder axis. The radius and position of the cylinder are then computed using circle fitting on the projected points. Rabbani (2005) employed the traditional Hough transform circle fitting procedure of Kimme et al. (1975) to estimate the radius and position. Note that the accuracy of the second stage is highly dependent on the accuracy of the estimated cylinder axis orientation in the first stage. Therefore, accurate estimation of the cylinder axis orientation is critical for the two stage Hough transform method.
3.3 Methodology

We propose a mathematical formulation for the accurate estimation of cylinder axis orientation. The proposed approach is inspired by the two stage Hough transformed based method (Rabbani, 2005) - we also use the five parameter parameterization of the cylinder, as well as employ surface normals of points in the 3D point cloud to estimate the cylinder axis orientation. However, instead of the voting based Hough transform procedure that is dependent on the cell size in the sampled Hough Gaussian sphere, we propose a cost function based on the aggregate angular distance between each point's surface normal and the estimated axis orientation.

For a given point \( p_i \), we first compute its surface normal \( \mathbf{n}_i = (x_i, y_i, z_i) \) using eigen-analysis of the covariance matrix of its \( k \)-nearest neighbors. Let \( \mathbf{a} = (x_a, y_a, z_a) \) denote the axis orientation vector, then the angle \( \omega_i \) between the vectors \( \mathbf{n}_i \) and \( \mathbf{a} \) can be computed using the dot product as in Equation 3.1.

\[
\omega = \cos^{-1}\left( \mathbf{n}_i \cdot \mathbf{a} \right) = \cos^{-1}\left( \frac{x_n x_a + y_n y_a + z_n z_a}{\sqrt{x_n^2 + y_n^2 + z_n^2} \sqrt{x_a^2 + y_a^2 + z_a^2}} \right) \tag{3.1}
\]

The surface normal \( \mathbf{n}_i \) is already a unit vector with a norm of 1. The axis orientation vector \( \mathbf{a} \) can be constrained to be a unit vector as well, as only the orientation matters. At this point, a transformation to spherical coordinates is conducive to reducing the number of parameters from three to two in the cost function to be derived shortly. Let \( r \) denote the radial distance, \( \theta \) denote the inclination angle, and \( \varphi \) denote the azimuth (spherical coordinate system definition commonly used in mathematics, as shown in Figure 3.2), then

\[
x = r \sin(\theta) \cos(\varphi), \quad y = r \sin(\theta) \sin(\varphi), \quad \text{and} \quad z = r \cos(\theta).
\]
The cylinder axis \( \mathbf{a} = (x_a, y_a, z_a) \) in Cartesian coordinates maps to \( \mathbf{a} = (r_a, \theta_a, \varphi_a) \) in spherical coordinates. Since the axis orientation vector is constrained to be a unit vector, then \( r_a = 1 \). Using this coordinate conversion and under the unit norm constraint for \( \mathbf{a} \), Equation 3.1 becomes Equation 3.2:

\[
\omega_i = \cos^{-1}\left\{ x_n \sin(\theta_a) \cos(\varphi_a) + y_n \sin(\theta_a) \sin(\varphi_a) + z_n \cos(\theta_a) \right\}
\]

(3.2)

If the point \( \mathbf{p}_i \) lies exactly on the surface of the cylinder with axis \( \mathbf{a} \), and if \( \mathbf{p}_i \)'s surface normal \( \mathbf{n}_i \) can be perfectly computed, then \( \omega_i = \pi/2 \). However, a 3D point cloud of a pipe/cylinder acquired by a terrestrial LIDAR scanner will contain measurement noise, and furthermore, a point's surface normal cannot be perfectly computed, but only estimated using its \( k \)-nearest neighbors. Therefore, we propose a function \( f_i(\theta_a, \varphi_a) \) to compute how far away from being perpendicular is a point’s surface normal from the cylinder axis orientation in Equation 3.3. The concept "how far from perpendicular" is represented by \(|\omega_i - \pi/2|\). The absolute value is taken, as the angular distance is important, not the \( \pm \) sign.

\[
f_i(\theta_a, \varphi_a) = \left| \omega_i - \frac{\pi}{2} \right| = \left| \cos^{-1}\left\{ x_n \sin(\theta_a) \cos(\varphi_a) + y_n \sin(\theta_a) \sin(\varphi_a) + z_n \cos(\theta_a) \right\} - \frac{\pi}{2} \right| \]

(3.3)
Conceptually, each estimated surface normal of a cylinder point cloud should be nearly perpendicular to the cylinder axis orientation. Therefore, we aggregate over all $f_i$ and arrive at the cost function $F$ in Equation 4, which reaches a minimum when the optimal $\theta_a$ and $\varphi_a$ have been attained (i.e. when the estimated axis is as close to perpendicular to all the surface normals from the point cloud, in an aggregate sense).

$F(\theta_a, \varphi_a) = \sum_{i=1}^{N} f_i(\theta_a, \varphi_a)$

$$= \sum_{i=1}^{N} \left| \cos^{-1}\left(\frac{x_n \sin(\theta_a) \cos(\varphi_a) + y_n \sin(\theta_a) \sin(\varphi_a) + z_n \cos(\theta_a)}{\frac{\pi}{2}}\right) \right|$$  

(3.4)

In Equation 3.4, $N$ is the number of points in the cylinder point cloud. This formulation is robust to outliers, as the cost function is based on the angular distance between point surface normals and the axis orientation estimate - the impact of any outlier is bounded between 0º and 180º. For a point cloud of a single cylinder (assuming prior segmentation), the cost function $F$ has two equivalent minima: a minimum in the upper hemisphere $(\theta_a, \varphi_a)$, and a corresponding minimum in the lower hemisphere $(\theta_a + \pi, \varphi_a)$. The solution to Equation 4 can be obtained iteratively through gradient descent optimization of the variables $\theta_a$ and $\varphi_a$, as in Equation 3.5.

$$\alpha_a(t+1) = \alpha_a(t) - \gamma(\nabla F) = \alpha_a(t) - \gamma \left\{ \nabla \left( \sum_{i=1}^{N} f_i(\alpha_a) \right) \right\} = \alpha_a(t) - \gamma \left\{ \sum_{i=1}^{N} \nabla f_i(\alpha_a) \right\}$$  

(3.5)

where $\alpha_a = \begin{bmatrix} \theta_a \\ \varphi_a \end{bmatrix}$, $\nabla f_i(\alpha_a) = \begin{bmatrix} \frac{\partial f_i}{\partial \theta_a} \\ \frac{\partial f_i}{\partial \varphi_a} \end{bmatrix}$, and $t$ denotes time (i.e. iteration).

Equation 3.5 exploits the linearity property of the gradient operator. Next, we compute the partial derivatives of $f_i$ with respect to $\theta_a$ and $\varphi_a$. We first define a sequence
of intermediate variables (Equations 3.6-3.8) to facilitate usage of the chain rule in computing the partial derivatives.

\[ f = \left| g - \frac{\pi}{2} \right| \quad (3.6) \]

\[ g = \cos^{-1}(h) \quad (3.7) \]

\[ h = x_n \sin(\theta_a)\cos(\varphi_a) + y_n \sin(\theta_a)\sin(\varphi_a) + z_n \cos(\theta_a) \quad (3.8) \]

The chain rules for the partial derivatives of \( f_i \) with respect to \( \theta_a \) and \( \varphi_a \) are defined in Equations 3.9 and 3.10.

\[ \frac{\partial f_i}{\partial \theta_a} = \frac{df}{dg} \frac{\partial h}{\partial \theta_a} \quad (3.9) \]

\[ \frac{\partial f_i}{\partial \varphi_a} = \frac{df}{dg} \frac{\partial h}{\partial \varphi_a} \quad (3.10) \]

Using the definitions of the derivative for absolute value function and the derivative for inverse cosine, the derivatives are given in Equations 3.11-3.14. Note that the derivative of \(|\chi|\) is \( \text{sgn}(\chi) \), which is defined for \( \chi \neq 0 \) and indeterminate at 0. However, \((g - \pi/2)\) is never exactly 0 during actual computations, so the signum (\( \text{sgn} \)) function is used.

\[ \frac{df}{dg} = \text{sgn}(g - \frac{\pi}{2}) \quad (3.11) \]

\[ \frac{dg}{dh} = -\frac{1}{\sqrt{1 - h^2}} \quad (3.12) \]

\[ \frac{\partial h}{\partial \theta_a} = x_n \cos(\theta_a)\cos(\varphi_a) + y_n \cos(\theta_a)\sin(\varphi_a) + z_n (\sin(\theta_a)) \quad (3.13) \]

\[ \frac{\partial h}{\partial \varphi_a} = x_n \sin(\theta_a) (\sin(\varphi_a)) + y_n \sin(\theta_a)\cos(\varphi_a) \quad (3.14) \]
The final form of the partial derivatives of $f_i$ with respect to $\theta_a$ and $\varphi_a$ are given in Equations 3.15 and 3.16, where the intermediate variables $g$ and $h$ were defined in Equations 3.7 and 3.8.

$$\frac{\partial f}{\partial \theta_a} = \frac{df}{dg} \frac{dh}{d\theta_a} = \frac{d}{dx} \left( g - \frac{\pi}{2} \right) \frac{-1}{\sqrt{1-h^2}} \left[ x_n \cos(\theta_a) \cos(\varphi_a) + y_n \cos(\theta_a) \sin(\varphi_a) + z_n (-\sin(\theta_a)) \right]$$  (3.15)

$$\frac{\partial f}{\partial \varphi_a} = \frac{df}{dg} \frac{dh}{d\varphi_a} = \frac{d}{dx} \left( g - \frac{\pi}{2} \right) \frac{-1}{\sqrt{1-h^2}} \left[ x_n \sin(\theta_a) (-\sin(\varphi_a)) + y_n \sin(\theta_a) \cos(\varphi_a) \right]$$  (3.16)

3.4 Results

The parameter values used in the gradient descent optimization of the proposed angular distance based cost function for cylinder orientation estimation are given as follows. First, the number of neighbors used to estimate each surface normal is set at $k = 50$. The choice of $k$ depends on the point density of the 3D data as well as on the cylinder radius, and should be set by considering the tradeoff between noise and goodness of the local surface normal estimate. An overly small choice of $k$ would generate a noisy estimate for the surface normal, while an overly large choice of $k$ would not generate a good estimate for the local surface orientation. The setting $k=50$ has proven to be a practical choice for this work. For gradient descent optimization, the parameter $\alpha$ controls the step size in changing $\theta_a$ and $\varphi_a$ during each iteration, and is set at $\gamma = 0.05$. Both $\theta_a$ and $\varphi_a$ are initialized to be 0 (i.e. along the $z$-axis). Note that the initialization values for $\theta_a$ and $\varphi_a$ do not affect the convergence of the proposed approach, as the cost function has a single minimum for the upper hemisphere and an equivalent minimum in the lower hemisphere. The gradient descent optimization stops once $\theta_a$ and $\varphi_a$ converges (i.e. $\Delta \theta_a$ and $\Delta \varphi_a$ falls below a given threshold, which is set at $0.0001^\circ$ or $0.00000087266$ radians). The maximum number of iterations is set at 500. Next, we demonstrate the accuracy of the method using synthetic
cylinder data in Section 3.4.1, as well as on real 3D terrestrial LIDAR point cloud data in Section 3.4.2, and compare against the approach of Rabbani (2005).

3.4.1 Synthetic Experiments

Experiments using synthetically generated 3D cylinder data, where the actual cylinder axis orientation is known, enables a quantitative assessment of the accuracy of the proposed algorithm. Figure 3.3 shows an example synthetic cylinder consisting of 1,000 points, generated with an axis orientation of $\theta = 1.103488$ and $\phi = 0.120824$ (in radians). The points lie along the surface of the cylinder primitive, as no noise has been added for this example. Using the proposed gradient descent optimization of the cost function $F$, the algorithm converges after 44 iterations. Figure 3.4 shows the estimated cylinder axis orientation during each iteration. The actual cylinder orientation is displayed in green. The estimated orientation using the proposed technique is $\theta_a^* = 1.10443$ and $\phi_a^* = 0.119177$ (compare with actual orientation of $\theta = 1.103488$ and $\phi = 0.120824$), resulting in an angular error of 0.0018 radians or 0.1016º. Note that this estimate will be further refined by least squares, as described in Chapter 4.
Figure 3.4 Axis orientation estimates shown in red at each iteration of the gradient descent optimization. Actual axis orientation shown in green. Estimated orientation from the proposed technique is $\theta_a^* = 1.103488$ and $\varphi_a^* = 0.120824$.

Since the Hough transform based technique of Rabbani (2005) is highly dependent on the cell size (proportional to number of samples on the Hough Gaussian sphere), we compare our approach with Rabbani’s over a range of sampling densities (Figure 3.5). As the number of samples on the Hough Gaussian sphere increase, the accuracy of the Hough transform based orientation estimates tends to increase. However, the accuracy is partially dependent on chance - if the cylinder axis orientation happens to occur near a cell center in the approximately uniform sampled Hough Gaussian sphere, a higher accuracy will result. If the axis orientation happens to occur at the edge of a cell, the accuracy will decrease. This phenomenon is reflected in Figure 5, as the error of the estimated axis orientation is not monotonically decreasing as the sampling rate increases. The accuracy of the Hough Transform based orientation estimation is fundamentally limited by the cell size - even for a high sampling density of 5,000 samples for the Hough Gaussian sphere,
the cell length is approximately \(2.0311^\circ\) in terms of arc angle. The proposed mathematical formulation outperforms the Hough transform method at all examined sampling densities.

![Figure 3.5 Angular error in the estimated axis orientation for the Hough transformed based technique as a function of Hough Gaussian sphere sampling density, for the synthetic cylinder point cloud of Figure 3.3. Accuracy of the proposed technique is shown in green.](image)

The proposed technique does not require good initialization and achieves significantly better accuracy than the Hough transform based technique. Furthermore, the proposed technique is very computationally efficient. As an example, for the synthetic cylinder point cloud of 1,000 points shown previously in Figure 3.3, the proposed method converges in 44 iterations and 0.26 seconds, while the Hough transform approach using 5,345 sampled points on the Hough Gaussian sphere requires 12.15 seconds on the same Intel Core-i7 2.2GHz processor with 8GB of system memory. These execution times are only for the iteration or voting part – the time taken for surface normal vector computations is excluded. Therefore, not only is the proposed method more accurate, it is more computationally efficient than the Hough transform approach of Rabbani (2005).

To further assess the accuracy of the proposed method, 500 trials were conducted with randomly generated cylinder axis orientations as well as randomly generated cylinder
radius $\epsilon [1, 5]$. The distribution of the angular errors over the 500 trials for the proposed technique (using the same parameters as described previously) is compared with the Hough transform based approach (Rabbani, 2005) in Figure 3.6.

As can be observed, the distribution of the errors for the proposed technique lies to the left (i.e. lower average error) and is more compact (i.e. smaller variance) than the distribution of the errors for the Hough transform based approach. Over the 500 trials, the mean and standard deviation of the errors are 0.0544 and 0.0363 in degrees for the proposed method, while for the Hough transform based approach, the mean and standard deviation of the errors are 0.9547 and 0.6597.

### 3.4.2 Results using Terrestrial LIDAR Point Cloud Data

Next we evaluate the proposed approach using real point cloud data of a steel mill scene, shown in Figure 3.7(a), acquired using a terrestrial LIDAR scanner. The octree-based segmentation approach of Su et al. (2015) is used to separate the raw point cloud into individual segments shown in Figure 3.7(b). The segmentation technique of Su et al.
(2015) also identifies potential piping elements based on the segment’s principal curvatures obtained through polynomial fitting. In this scene, 12 segments are identified as piping elements, and are used to evaluate the proposed algorithm.

Figure 3.7 (a) LIDAR point cloud of a piping system acquired at a steel mill, (b) segmentation results using the octree-based segmentation technique of Su et al. (2015), and (c) pipe segments identified by the segmentation technique using principal curvatures.

Note that the pipe segments do not have full 360° coverage, presenting a more challenging evaluation than the synthetic cylinder data. The axis orientation of two pipe segments (the orange and red pipes from Figure 3.7) are estimated using the proposed method and the Hough transform based approach, and the axis orientation estimates are overlaid on the centroid of the pipe segments in Figure 3.8. For both pipes, the proposed method's orientation estimates (shown in green) appears to align visually better to the respective pipe point cloud than the orientation estimates from the Hough transform based approach (shown in blue). This visual examination provides a qualitative assessment of the accuracy of the estimated pipe segment orientations.
Figure 3.8 Axis orientations estimated using the proposed method (green) and using the Hough transform based approach (red) for the two example pipe segments.

Although the true pipe axis orientations are unknown, a quantitative accuracy assessment can be conducted by projecting each pipe segment onto the \((u,v)\) plane that is perpendicular to the estimated axis orientation, and then examining the residual of circle fitting. Note that for a perfect cylinder point cloud, this projection would generate a circle in 2D space. However, since LIDAR data contains noise, the projection along the axis orientation would not form a perfect circular arc. Figure 3.9 shows the projected points on the right using the axis orientation estimates obtained from the proposed method, and on the left, using the axis orientation estimates obtained from the Hough transform approach. The proposed approach yields projected points that are more compact in the 2D \((u,v)\) plane.
Figure 3.9 Projections of the two pipe point clouds onto the plane perpendicular to the axis orientations estimated using the proposed method (in green) and using the Hough transform approach (in red).

The compactness of the points in the projected 2D space can be characterized by the residual through circle fitting. A least squares based technique, described in Chapter 4, is used for circle fitting to find the radius, center position, and compute the residual. For each of the 12 identified pipe segments in Figure 3.7(c), this residual is computed for the proposed approach and for the Hough transform approach of Rabbani (2005), and tabulated in Table 3.1. The orange and red pipes of Figure 3.8 are Segment 1 and Segment 4, respectively. For all pipe segments (except for Segment 11), the circle fitting residual in the 2D space formed by projecting along the pipe axis orientation estimated using the proposed method is lower, often times substantially smaller than that of the Hough transform method, signifying that the pipe orientations estimated with the proposed technique are more accurate.
Table 3.1 Circle fitting residual of each pipe segment in projected 2D space using the axis orientation estimated by the Hough transform method and the proposed method.

<table>
<thead>
<tr>
<th>Pipe Segment</th>
<th># Points in Segment</th>
<th>Hough Method</th>
<th>Proposed Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pipe Segment 1</td>
<td>8822</td>
<td>0.7981</td>
<td>0.3743</td>
</tr>
<tr>
<td>Pipe Segment 2</td>
<td>48349</td>
<td>2.5715</td>
<td>1.1650</td>
</tr>
<tr>
<td>Pipe Segment 3</td>
<td>15226</td>
<td>0.4697</td>
<td>0.3983</td>
</tr>
<tr>
<td>Pipe Segment 4</td>
<td>32645</td>
<td>1.0266</td>
<td>0.2373</td>
</tr>
<tr>
<td>Pipe Segment 5</td>
<td>47146</td>
<td>1.5263</td>
<td>0.3895</td>
</tr>
<tr>
<td>Pipe Segment 6</td>
<td>1547</td>
<td>0.0460</td>
<td>0.0432</td>
</tr>
<tr>
<td>Pipe Segment 7</td>
<td>46343</td>
<td>0.7513</td>
<td>0.6638</td>
</tr>
<tr>
<td>Pipe Segment 8</td>
<td>2725</td>
<td>0.0582</td>
<td>0.0510</td>
</tr>
<tr>
<td>Pipe Segment 9</td>
<td>35845</td>
<td>1.1088</td>
<td>1.0904</td>
</tr>
<tr>
<td>Pipe Segment 10</td>
<td>48839</td>
<td>1.6065</td>
<td>0.2055</td>
</tr>
<tr>
<td>Pipe Segment 11</td>
<td>4201</td>
<td>0.0629</td>
<td>0.0638</td>
</tr>
<tr>
<td>Pipe Segment 12</td>
<td>55129</td>
<td>4.0205</td>
<td>3.2264</td>
</tr>
</tbody>
</table>

3.5 Conclusion

In this work, we proposed a new approach for cylinder axis orientation estimation in 3D point cloud data based on gradient descent optimization of an angular distance cost function using point surface normal vectors. The proposed method results in higher accuracy than the state-of-the-art Hough transform approach, and furthermore, is more computationally efficient. We demonstrated the robust performance of the proposed technique on both synthetic cylinder point clouds as well as on real LIDAR scans of piping systems, showing its robustness in the presence of noise and outliers.
CHAPTER 4. CYLINDER PARAMETER ESTIMATION AND GRAPHIC REPRESENTATION

The flow chart of this chapter is given in Figure 4.1, showing the sequence of steps needed to generate a CAD model. After estimating the orientation of the cylinder axis using the proposed angular distance based approach described in Chapter 3, the cylinder position (partial) and radius are then estimated in a projected 2D space using traditional least squares circle fitting.

Figure 4.1 Flow chart showing steps needed to generate a CAD model.
The first step in finding the cylinder’s position and radius is to project the 3D cylinder point cloud onto the plane perpendicular to the cylinder axis. This plane can be determined by fixing a Cartesian coordinate system with the cylinder orientation vector as one of its axes. Let the cylinder axis orientation vector be represented in Cartesian coordinates by unit vector \( \hat{n} = (n_x, n_y, n_z) \). The other two axes are added arbitrarily, then the Gram-Schmidt process is used to generate the orthonormal basis \((\hat{u} \ \hat{v} \ \hat{n})\), illustrated in Figure 4.2. The vectors \( \hat{u} \) and \( \hat{v} \) form a transformation matrix \( T = (\hat{u} \ \hat{v}) \), which projects the 3D points from the \((x, y, z)\) coordinate system to 2D points in the \((u \ v)\) coordinate system, as in Equation 4.1. If the orientation of the cylinder axis has been estimated accurately, then the 3D cylinder point cloud forms a circle in the projected 2D space.

\[
\begin{bmatrix}
u \\
v
\end{bmatrix} =
\begin{bmatrix}
u_x & \nu_y & \nu_z \\
v_x & v_y & v_z
\end{bmatrix}
\begin{bmatrix}x \\
y
\end{bmatrix} = T
\begin{bmatrix}x \\
y
\end{bmatrix}
\tag{4.1}
\]

Following this projection, the radius \( R \) and center of the circle \( c = (u_c, v_c) \) can be estimated using least squares based fitting techniques in two dimensions. In this work, the circle fitting is done by conventional least squares. The condition equation is

\[
R = \sqrt{[(u - u_c)^2 + (v - v_c)^2]}
\tag{4.2}
\]

Equation 4.2 is recast as

\[
F = R - \sqrt{[(u - u_c)^2 + (v - v_c)^2]} = 0
\tag{4.3}
\]

and becomes a nonlinear least squares problem. Initial approximations are needed for the parameters \( u_c, v_c, R \), and a Newton iteration method is used to refine the parameter estimates, while minimizing the objective function

\[
\varphi = v^T \mathbf{W} v
\tag{4.4}
\]
where $\mathbf{v}$ is the residual vector and $W$ is a weight matrix. This problem is solved by using "general least squares" (mixed model)

$$A\mathbf{v} + B\Delta = f$$

since we have both parameters and multiple observations in each condition equation.

Figure 4.2 Illustration of position (partial) and radius estimation by first projecting the cylinder point cloud onto the plane perpendicular to its axis orientation, followed by least squares circle fitting.

Once estimates for the (partial) position has been estimated in the 2D $(u,v)$ space, as illustrated in Figure 4.2, it must be transformed back into the 3D $(x,y,z)$ coordinate system that the cylinder resides in. Note that a point in the 2D coordinate system represents a line in 3D space – the estimated circle center transforms into the cylinder axis. First, the inverse rotation matrix is formed:

$$R = \begin{bmatrix} u_x & u_y & u_z \\ v_x & v_y & v_z \\ n_x & n_y & n_z \end{bmatrix}, \begin{bmatrix} u \\ v \\ n \end{bmatrix} = R \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \begin{bmatrix} x \\ y \\ z \end{bmatrix} = R^T \begin{bmatrix} u \\ v \\ n \end{bmatrix}$$

An arbitrary point on the cylinder axis $(u_c, v_c, n_0)$ is selected and rotated back into the $(x, y, z)$ coordinate system,

$$\begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} = R^T \begin{bmatrix} u_c \\ v_c \\ n_0 \end{bmatrix}$$
to produce the reference point in the \((x, y, z)\) coordinate system. Then the cylinder axis (line in 3D space) can be expressed as Equation 4.8

\[
(x_a, y_a, z_a) = (x_0, y_0, z_0) + t(n_x, n_y, n_z)
\]

(4.8)

with \( \mathbf{n} = (n_x, n_y, n_z) \) a unit vector. Equation 4.8 is the standard parameterized equation of a line in 3D space, using the point \((x_0, y_0, z_0)\) on the (unit) axis and the axis orientation vector, \( \mathbf{n} = (n_x, n_y, n_z) \), which represents the direction of the line. Before the cylinder can be rendered as a CAD model, the starting and ending positions of the cylinder must be determined. Specifically, a starting point \((x_s, y_s, z_s)\) lying on the cylinder axis as well as an ending point \((x_e, y_e, z_e)\) lying on the cylinder axis must be determined. Conceptually, these two points can be determined by first “projecting” each point in the cylinder point cloud onto the cylinder axis, and then finding the two extrema axis points. “Projecting onto the axis” can be implemented by geometric projection onto the cylinder axis, or by finding the closest corresponding point on the axis for each point in the cylinder point cloud. The geometry for this is shown in Figure 4.3.
The vector, $\vec{q}$, from the reference point to a point, $P$, on the cylinder is

$$\vec{q} = \begin{bmatrix} p_x - x_0 \\
p_y - y_0 \\
p_z - z_0 \end{bmatrix}$$

(4.9)

The cylinder axis is represented by the expression

$$\begin{bmatrix} x_0 + tn_x \\
y_0 + tn_y \\
z_0 + tn_z \end{bmatrix}$$

(4.10)

t can be determined by projecting $\vec{q}$ onto $\hat{n}$

$$t = \vec{q} \cdot \hat{n} = n_x p_x - n_x x_0 + n_y p_y - n_y y_0 + n_z p_z - n_z z_0$$

(4.11)

Then point $A$ is just given by Equation 4.10.

After computing $t$ for each point in the cylinder point cloud, the starting and ending points for the cylinder are $(x_s, y_s, z_s) = (x_0 + t_{min}n_x, y_0 + t_{min}n_y, z_0 + t_{min}n_z)$ and $(x_e, y_e, z_e) = (x_0 + t_{max}n_x, y_0 + t_{max}n_y, z_0 + t_{max}n_z)$. Figure 4.4(a) shows the “projection” of the cylinder points onto the axis, and Figure 4.4(b) shows the computed starting and ending points for the cylinder.

![Figure 4.4 (a) “Projection” of points in cylinder point cloud onto the line represented by the cylinder axis, and (b) computed starting and ending points for the cylinder.](image)
The computed starting and ending points of the cylinder along with the orientation and position parameters and the estimated radius, complete its initial determination. Recall that the cylinder parameter estimation technique developed in this work is a two-step procedure: (1) angular distance based orientation estimation, followed by (2) position and radius estimation via circle fitting in projected 2D space. The key benefits of this two-step procedure are that it produces accurate estimates for the cylinder parameters without requiring a good initial estimate (unlike least squares fitting), and is robust in the presence of noise. However, the accuracy of the estimated cylinder parameters can be further refined by simultaneous parameter estimation (orientation, position, and radius) with a least squares based approach, using the computed parameter values from the two-step procedure as good initial estimates for least squares fitting. The new least squares formulation has been developed to perform simultaneous cylinder parameter estimation, as shown in Appendix A. Note that another significant advantage of the simultaneous least squares estimation approach is that the covariance matrix can be obtained for all estimated quantities. Thus rigorous error propagation can be extended to provide confidence statements about the uncertainty of the estimates.

Using the refined starting and ending positions of the cylinder along with the refined estimate of the cylinder radius, a CAD model for the cylinder can be generated in MuPAD, as shown in Figure 4.5. MuPAD is part of Matlab’s Symbolic Math Toolbox, and has an interactive graphic system that supports animations in 3D.
Using the procedure described above, each piping candidate can be displayed as a CAD model, and all the elements can be displayed together to form a piping system CAD model for a given scene. Figure 4.6 shows such a piping system CAD model, generated from the correctly identified piping elements of the combined point cloud of the steel mill scene.
CHAPTER 5. CONCLUSIONS AND FUTURE WORK

5.1 Conclusions

The increasing adoption of LIDAR for use in applications ranging from topographical mapping with airborne platforms to surveying of urban sites with terrestrial systems necessitates the development of techniques for the automatic exploitation of 3D LIDAR point clouds. These high density point clouds are spatially non-uniform, and are typically massive in size, presenting significant challenges to the development of effective processing techniques that are also computationally efficient. In this dissertation, a complete processing chain is developed to process point cloud data of piping systems from industrial sites such as chemical plants, steel mills, and oil refineries. The processing chain consists of three stages: (1) octree-based split and merge segmentation of LIDAR point clouds, (2) detection and identification of piping candidates through surface fitting and principal curvature analysis, and (3) cylinder fitting using first an angular distance based cost function and then least squares for the parameter estimation. Each of these three stages contributes novel concepts and formulations, adding needed tools for LIDAR point cloud processing. The key contributions of the proposed segmentation technique are a series of connectivity criteria that utilizes higher level contextual information of components to intelligently merge cylindrical segments into whole pipes and planar segments into whole walls. Furthermore, the octree based approach is a bottom-up approach that is inherently
computationally efficient. Instead of point by point processing adopted by region growing based techniques, the proposed techniques process the point cloud in a block by block basis. Furthermore, the block based processing is iteratively independent, and can be conducted in parallel on multiple cores or in a cluster to further reduce the execution runtime of the segmentation algorithm. Following segmentation, a novel cylinder detection technique is developed based on surface fitting to identify those segments that are pipes by analyzing their principal curvatures. The detected pipes must then undergo cylinder fitting to estimate their parameters (orientation, position, and radius). The key contribution of the cylinder fitting technique proposed in this work is a novel mathematical formulation for orientation estimation using an angular distance cost function of point surface normal orientations. The proposed orientation estimation is robust to noise and does not require an initial starting point, which traditional nonlinear least squares based cylinder fitting techniques require. This step is followed by conventional least squares which has the advantage of providing rigorous error propagation for all estimated quantities. Furthermore, the proposed continuous-space formulation is significantly more accurate compared to the recent two-step Hough transform based method, which is a voting based procedure in discrete space.

5.2 Future Work

Though the techniques developed in this dissertation form an effective and complete processing chain, it can certainly be improved and extended. Within the proposed segmentation algorithm, the splitting process can be enhanced by considering surface normal orientations, instead of only relying on spatial connectivity during splitting. Relying only on spatial connectivity for point splitting within each octant results in L-
shaped components (e.g. section consisting of two intersecting walls as described in Section 2.4.3) whose planes should ideally be separated into different planes. The splitting process can be enhanced as follows. Given the points within an octant generated by octree decomposition, analyze surface normals as well as spatial proximity to influence the segment decisions. A clustering analysis might be beneficial to avoid combining multiple primitives into a single segment. The challenge lies in developing a split algorithm which can automatically determine the number of clusters based on the number of modes observed in the surface normal distribution. Subsequently, graph analysis must be performed utilizing both spatial connectivity information and surface orientation information to separate the points into unconnected components. This approach is expected to improve overall segmentation results.

Though the developed processing chain is capable of generating CAD models of pipes in a piping system, it does not output pipe connection information, indicating which pipes are connected together. A simple graph theory based approach could be implemented by examining the spatial locations of starting and ending positions of each pipe and determination pipe linkage. However, the ideal method is to be able to segment and model pipe connectors such as elbows, flanges, T-junctions, and other types of junctions. However, these pipe connectors are typically smaller elements within a piping system – current LIDAR scanners may not acquire sufficient number of points on these elements for accurate segmentation and modeling. The continual advancement in LIDAR technology, producing increasingly accurate and dense point clouds, will enable segmentation and modeling of pipe connectors. The proposed segmentation technique establishes a
foundation, from which more advanced contextual connectivity criteria can be developed for these more complex shapes to facilitate their segmentation and subsequent modeling.

Additionally, an approach combining laser scanning with imagery, through its ability to capture edges, colors, and textures, would likely prove to be a powerful combination. Continuing advances in laser point cloud accuracy, density, multi-spectral capability, and registration quality will also contribute to the success of exploitation techniques.
REFERENCES


Brown, K.S. (n.d). Old notes on curvature. Retrieved from


A general least squares formulation is developed to perform simultaneous cylinder parameter estimation. A key feature of general least squares is that combined observations and parameters can be handled in the condition equations without restricting each equation to having only a single observation. Appendix A is organized as follows. Section A.1 describes the derivation of general least squares. Section A.2 discusses line fitting in 3D space under the least squares framework. Section A.3 presents least squares fitting of cylinder, which is an extension of the line fitting principles discussed in Section A.2.

A.1 General Least Squares Adjustment

When starting an adjustment, the minimum number of distinct variables \( n_0 \) needed to determine the unique mathematical model must be defined first. This value, \( n_0 \), must allow construction of the geometry of the problem together with the location of all observations. In the general case when performing adjustment with observations and independent parameters, given \( n \) observations, the redundancy \( r = n - n_0 \) is computed. This redundancy represents the principle that for the \( n \) observations, \( r \) conditions must be satisfied. Note that the observations fit the model perfectly when \( r = 0 \). Now, we consider parameters in addition to observations. For \( u \) unknown independent parameters, the number of conditions needed to retain the same number of redundancy \( r \) will be given by \( c = r + u \). With \( u \) parameters, a total of \( c = r + u \) independent condition equations must
be created from the \( n \) observations and \( u \) parameters. For line and cylinder fitting, the initial component of \( n_0 \) specifies the geometric object, line or cylinder, respectively 4 and 5. For the line we must designate a reference point and a direction. But the reference point could be anywhere on the line so to guarantee unique parameters, we “pre-designate” one component of the reference point, leaving 2 independent coordinate components. Likewise the direction vector could have an infinite number of correct values so we force it to be a unit vector. This comes only 2 of its components to be independent. So for the line, the initial component of \( n_0 \) is \( 2 + 2 = 4 \). For the cylinder the initial component of \( n_0 \) includes an extra parameter, the radius, \( R \), so the initial \( n_0 \) is 5.

But as stated earlier, \( n_0 \) must permit construction of the full geometry of the problem including locations of all of the observations. In the case of a line, each \( X, Y, Z \) observation requires one coordinate component, in addition to the line itself, to locate that observation. So for \( n \) points, \( n_0 \) for the least squares model would be \( 4 + n \). For the cylinder, in addition to the cylinder model itself, we need two coordinate components per \( X, Y, Z \) observation in order to locate the point. These two coordinate components effectively determine a line and the intersection of that line with the cylinder determines the point location. The apparent ambiguity of two points of intersection with the cylinder is not a problem since the observation itself resolves the ambiguity. Some examples will be provided later to illustrate these concepts.

The nonlinear condition equations \( F(l, x) = 0 \) are linearized via Taylor Series approximation, discarding derivatives higher than first. This leads to

\[
F(l, x) \approx F(l^0, x^0) + \frac{\partial F}{\partial l} \Delta l + \frac{\partial F}{\partial x} \Delta x = 0 \quad (A.1)
\]
The relationship between $\Delta l$ and the residual, $v$, is

$$l^0 + \Delta l = l + v \quad (A.2)$$

And we make substitutions $\frac{\partial F}{\partial l} = A \frac{\partial F}{\partial x} = B$ and shorten $\Delta x$ to $\Delta$, leading to,

$$F(l^0, x^0) + A(l - l^0 + v) + B\Delta = 0 \quad (A.3)$$

$$Av + B\Delta = -F(l^0, x^0) - A(l - l^0)$$

$$A_{c,n}v_{n,1} + B_{c,u}\Delta_{u,1} = f_{c,1}$$

This is the fundamental form of the condition equations for adjustment of observations and independent parameters. It represents $c$ condition equations with $u$ unknown parameters and $n$ unknown residuals. The unique least squares solution is obtained by

$$\Phi = \nu^t W \nu \rightarrow \text{minimum}, \text{ where } W \text{ is the weight matrix of the observations.}$$

To enforce this criterion and at the same time have a solution for $Av + B\Delta = f$, the method of constrained minima by Lagrange multipliers is used. Thus if $k_{c,1}$ represents the yet unknown Lagrange multiplier vector, then the least squares solution is obtained by minimizing the following equation:

$$\Phi' = \nu^t W \nu - 2k^t (Av + B\Delta - f) \quad (A.4)$$

where $k$ are the Lagrange multipliers

To minimize $\Phi'$, (Mikhail and Ackermann, 1976), its partial derivatives with respect to $\nu, \Delta$ and $k$ are set equal to zero,

$$\frac{\partial \Phi'}{\partial \nu} = 2\nu^t W - 2k^t A = 0^t \quad (A.5)$$

$$\frac{\partial \Phi'}{\partial \Delta} = -2k^t B = 0^t \quad (A.6)$$
\[
\frac{\partial \varphi'}{\partial k} = -2(Av + B\Delta - f)^t = 0^t \tag{A.7}
\]

These represent the normal equations. Transposing and rearranging gives

\[-Wv + A^t k = 0 \tag{A.8}\]

\[B^t k = 0 \tag{A.9}\]

\[Av + B\Delta = f \tag{A.10}\]

There these equations can be solved simultaneously or by elimination. Using the elimination method where \(Q\) and \(W\) are inverses,

\[v = W^{-1}A^t k = QA^t k \tag{A.11}\]

Substituting equation (A.11) into (A.10):

\[AQ A^t k + B\Delta = f \tag{A.12}\]

Let \(Q_e = QA^t\), and \(W_e = Q_e^{-1}\),

\[k = W_e(f - B\Delta) \tag{A.13}\]

Substituting this into (A.9) gives,

\[(B^t W_e B)\Delta = (B^t W_e f) \tag{A.14}\]

\[N\Delta = t\]

These are the reduced normal equations, solving these yields,

\[\Delta = N^{-1} t \tag{A.15}\]

Then proceeding by back substitution, first into (A.13)

\[k = W_e(f - BN^{-1} t) \tag{A.16}\]

Finally substituting this into (A.11)

\[v = QA^t W_e(f - BN^{-1} t) \tag{A.17}\]
A.2 Fitting a 3D Line by Least Squares

We first consider the simple problem of 3D line fitting using least squares. Let \( O \) be a reference point on the line, and \( u \) be a unit vector parallel to the line. We wish to apply least squares to observation points such as \( L \) so we can fit a line, as illustrated in Figure A.1.

![Figure A.1 Illustration of 3D line fitting parameters](image)

First, \( n_0 \) must be defined as described in Section A.1. Four independent parameters \( u \) are needed in this case to specify the model. Two parameters are taken from \((X_O, Y_O, Z_O)\), as one of these variables can be chosen arbitrarily since \( O \) can lie anywhere on the line. The remaining two parameters are taken from the unit vector \((u_X, u_Y, u_Z)\). Only two of the components are independent since it has unit length. Note that when choosing two components from \((X_O, Y_O, Z_O)\) we should eliminate the coordinate corresponding to the longest component of \( u \).

For example, if we want to fit a line in 3D by four points (See Figure A.2 for illustration):

\[\text{The number of observations (n)} = 4 \times 3 = 12\]
The number of independent parameters to determine the line $(u) = 4$

The minimum variables to fix the line and all observations $(n_0) = 4 + 4 \times 1 = 8$

The redundancy $(r) = n - n_0 = 12 - 8 = 4$

The number of condition equations $(c) = r + u = 4 + 4 = 8$

(two equations per point)

![Figure A.2 Illustration of counting $n_0$ for the case of 3D line fitting with four points](image)

The details for the derivation of the condition equations are presented as follows:

For any point $L$ on the line, there is a related parameter, $t$, such that

$$
\begin{bmatrix}
X_L \\
Y_L \\
Z_L
\end{bmatrix} =
\begin{bmatrix}
X_O \\
Y_O \\
Z_O
\end{bmatrix} +
t
\begin{bmatrix}
u_x \\
u_y \\
u_z
\end{bmatrix}
$$

(A.18)

Rewriting (A.18)

$$
L = O + tu =
\begin{bmatrix}
X_O + tu_x \\
Y_O + tu_y \\
Z_O + tu_z
\end{bmatrix}
$$

(A.19)
If \( \mathbf{u} \) has unit length then \( t \) is just the length from \( O \) to \( L \). Equation (A.18) can be solved for \( t \),

\[
t = \frac{x_L - x_0}{u_X} = \frac{y_L - y_0}{u_Y} = \frac{z_L - z_0}{u_Z}
\]  
(A.20)

and the two condition equations are

\[
\frac{x_L - x_0}{u_X} = \frac{y_L - y_0}{u_Y}
\]
\[
\frac{x_L - x_0}{u_X} = \frac{z_L - z_0}{u_Z}
\]  
(A.21)

If we assume that \( u_X \) is the longest component of \( \mathbf{u} \), we can rearrange these to avoid singularities as follows

\[
u_Y(X_L - X_0) = u_X(Y_L - Y_0)
\]
\[
u_Z(X_L - X_0) = u_X(Z_L - Z_0)
\]  
(A.22)

further,

\[
\left(\frac{u_Y}{u_X}\right)(X_L - X_0) = Y_L - Y_0
\]
\[
\left(\frac{u_Z}{u_X}\right)(X_L - X_0) = Z_L - Z_0
\]  
(A.23)

finally,

\[
F_1 = \left(\frac{u_Y}{u_X}\right)(X_L - X_0) - Y_L + Y_0 = 0
\]
\[
F_2 = \left(\frac{u_Z}{u_X}\right)(X_L - X_0) - Z_L + Z_0 = 0
\]  
(A.24)

In this case \( X_0 \) and \( u_X \) would be considered constants with \( u_X \) recomputed each iteration to maintain unit length. The four known parameters would be \( Y_0, Z_0, u_Y, u_Z \). The condition equations are linearized in the usual way. In case \( u_Y \) or \( u_Z \) are the “long” axis, the appropriate two condition equations would be chosen from (A.21) and the unknowns would be \( X_0, Z_0, u_X, u_Z \) and \( X_0, Y_0, u_X, u_Y \) respectively.
A.3 Fitting a Cylinder by Least Squares

Fitting a cylinder by least squares is closely related to the line fitting problem, with an additional radius parameter $R$. Again, $n_0$ needs to be defined first. There are five independent parameters $u$ needed to specify the model: two parameters from $(X_O, Y_O, Z_O)$, two parameters from $(u_X, u_Y, u_Z)$ like the line fit problem, and the fifth parameter $R$. Also, by choosing two components from $L$, all the points on the geometric model can be located via the two planes just chosen. The intersection of these two planes will form a line that intersects the cylindrical surface twice. This defines the location of the observed point. The two intersections do not present an ambiguity since the given observed coordinates will resolve which of the two are correct.

For example, if we want to fit a cylinder in 3D by ten points (as illustrated in Figure A.3):

![Diagram of cylinder fitting](image)

Figure A.3 Illustration of counting $n_0$ for the case of cylinder fitting with ten points
The number of observations \((n) = 10 \times 3 = 30\)

The number of independent parameters \((u) = 5\)

The minimum variables to fix the cylinder and all observations \((n_0)\)

\[= 5 + 10 \times 2 = 25\]

The redundancy \((r) = n - n_0 = 30 - 25 = 5\)

The number of condition equations \((c) = r + u = 5 + 5 = 10\)

The derivation of the condition equations are presented as follows:

![Diagram of fitting observations to a cylinder model](image)

Figure A.4 Illustration of fitting observations to a cylinder model

As in the case of the line in 3D we will select parameters based on the orientation of the cylinder axis. The choice of which components to fix will change with the orientation of the cylinder. Since the cylinder illustrated in Figure A.4 is aligned along the y-axis, we
fix $Y_o, u_Y$, and furthermore constrain $u$ to be unit length. From the observation $S$ (point lying on the surface of the cylinder), we can compute $t$ by projection (i.e. dot product) as illustrated in Figure A.5:

$$t = (S - O) \cdot \hat{u}$$

$$t = (X_S - X_o)u_x + (Y_S - Y_o)u_y + (Z_S - Z_o)u_z$$  \hspace{1cm} (A.25)$$

Obtain $L$ by equation (A.19). Then form vector $\vec{SL}$:

$$\vec{SL} = \vec{L} - \vec{S} = \begin{bmatrix} X_o + tu_x - X_S \\ Y_o + tu_y - Y_S \\ Z_o + tu_z - Z_S \end{bmatrix}$$

The cylinder condition equation can be derived as follows:

$$|\vec{SL}| = R$$

$$\left(\vec{SL}^T \vec{SL}\right)^{1/2} = R$$

The condition equation can then be expressed as,

$$F = R - \left(\vec{SL}^T \vec{SL}\right)^{1/2} = 0$$

expanding,

$$F = R - \left[ (X_o + t\mu_x - X_S)^2 + (Y_o + t\mu_y - Y_S)^2 + (Z_o + t\mu_z - Z_S)^2 \right]^{1/2} = 0$$  \hspace{1cm} (A.27)$$
Therefore, each point in the cylinder point cloud gives one condition equation. General least squares as defined in Equation (A.3) can be used to solve this cylinder fitting problem. Note that for the example above, we fixed $Y_0$ and $u_y$. Similarly, $X_0$ and $u_x$ are fixed if the orientation of the cylinder is along the x-axis, and $Z_0$ and $u_z$ are fixed if the orientation of the cylinder is along the z-axis. The condition equation for each of these cases can be derived using the same procedure as described above. The equation is linearized in the usual manner and the rotation is iterated until convergence.
VITA

Yun-Ting Su graduated with a bachelor’s degree in Surveying Engineering from National Cheng Kung University in 2003. She obtained a master of science in Civil Engineering from Purdue University in 2006, and subsequently enrolled in the doctoral program, also in Civil Engineering at Purdue University. Her research interests are in geomatics, computer vision, and LIDAR processing. She is a member of Professor Bethel’s research group. Her research is on automatic exploitation of LIDAR point cloud data of industrial scenes, specifically focusing on segmentation and cylinder detection and modeling of piping scenes.