A Non-local Approach to Tree Peak Detection in LIDAR Data

by

Sushant Joshi
B.Tech., Banaras Hindu University, 2005

Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of

Master of Science

in the School of Computing Science
Faculty of Applied Sciences

© Sushant Joshi 2014
SIMON FRASER UNIVERSITY
Fall 2014

All rights reserved.
However, in accordance with the Copyright Act of Canada, this work may be reproduced without authorization under the conditions for "Fair Dealing". Therefore, limited reproduction of this work for the purposes of private study, research, criticism, review and news reporting is likely to be in accordance with the law, particularly if cited appropriately.
Name: Sushant Joshi
Degree: Master of Science
Title: A Non-local Approach to Tree Peak Detection in LIDAR Data

Examining Committee: Chair: Dr. Ghassan Hamarneh
Professor

Dr. Hao Zhang
Senior Supervisor
Professor

Dr. Ping Tan
Supervisor
Assistant Professor

Dr. Parvaneh Saeedi
Internal/External Examiner
Associate Professor
School of Engineering Science

Date Approved: November 28th, 2014
Partial Copyright Licence

The author, whose copyright is declared on the title page of this work, has granted to Simon Fraser University the non-exclusive, royalty-free right to include a digital copy of this thesis, project or extended essay[s] and associated supplemental files ("Work") (title[s] below) in Summit, the Institutional Research Repository at SFU. SFU may also make copies of the Work for purposes of a scholarly or research nature; for users of the SFU Library; or in response to a request from another library, or educational institution, on SFU’s own behalf or for one of its users. Distribution may be in any form.

The author has further agreed that SFU may keep more than one copy of the Work for purposes of back-up and security; and that SFU may, without changing the content, translate, if technically possible, the Work to any medium or format for the purpose of preserving the Work and facilitating the exercise of SFU’s rights under this licence.

It is understood that copying, publication, or public performance of the Work for commercial purposes shall not be allowed without the author’s written permission.

While granting the above uses to SFU, the author retains copyright ownership and moral rights in the Work, and may deal with the copyright in the Work in any way consistent with the terms of this licence, including the right to change the Work for subsequent purposes, including editing and publishing the Work in whole or in part, and licensing the content to other parties as the author may desire.

The author represents and warrants that he/she has the right to grant the rights contained in this licence and that the Work does not, to the best of the author’s knowledge, infringe upon anyone’s copyright. The author has obtained written copyright permission, where required, for the use of any third-party copyrighted material contained in the Work. The author represents and warrants that the Work is his/her own original work and that he/she has not previously assigned or relinquished the rights conferred in this licence.

Simon Fraser University Library
Burnaby, British Columbia, Canada

revised Fall 2013
Abstract

The majority of existing methods for delineating trees from LIDAR point cloud data use a region growing approach. Seed points which are meant to represent the highest point of the trees (or simply tree peak) are detected in the input point cloud. The remaining points are iteratively assigned to one of the seed points, thus growing the region representing individual trees. Hence accurate detection of the tree peaks plays a crucial role in tree delineation. Current methodologies for tree peak detection are all based on local geometry analysis, identifying locally highest points within some appropriately sized neighborhood as tree peaks, where the neighborhood size is often chosen based on some empirical observations. Such local approaches often result in false positives over a tree with a large crown possessing multiple local peaks and false negatives when multiple trees are in close proximity. In this work, we advocate the use of non-local approaches to tree peak detection which look beyond height values of individual points. Specifically, we propose a method which takes into account point distribution, sidedness, along with height values and also analyzes mutual relation between points in the input point cloud. Our method starts with a conservative local analysis to extract a superset of local peaks. Then it executes a series of filters and attempts to eliminate false positives, namely branch tips and false central peaks. Eventually, we show superior results obtained by our method over the state of the art.

**Keywords:** point cloud processing; geometrical structure
To my family
“Does the walker choose the path, or the path the walker?”

—GARTH NIX, 1995
Acknowledgments

I am grateful to my entire family for being supportive and encouraging me to pursue my higher studies after a hiatus of almost 7 years; without them it would not have been possible for me to come back to school after years of being away from textbooks, term papers and tests.

I would also like to express my appreciation towards my senior supervisor, Dr. Hao Zhang, not just for the technical guidance he provided throughout this work, but also motivating everyone in our lab to try out something novel and challenging.

I am also grateful to all my colleagues at the GRaphics, Usability, and VIsualization (GrUVi) lab; this dissertation would not have been possible without the fruitful discussion that I had with them. Special thanks to Shuyang for his valuable insights, and Ibraheem - the go-to guy for all software related issues in our lab.

Lastly, I extend my gratitude to the Almighty for all the good fortune He has bestowed upon me.
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Approval</td>
<td>ii</td>
</tr>
<tr>
<td>Partial Copyright License</td>
<td>iii</td>
</tr>
<tr>
<td>Abstract</td>
<td>iv</td>
</tr>
<tr>
<td>Dedication</td>
<td>v</td>
</tr>
<tr>
<td>Quotation</td>
<td>vi</td>
</tr>
<tr>
<td>Acknowledgments</td>
<td>vii</td>
</tr>
<tr>
<td>Contents</td>
<td>viii</td>
</tr>
<tr>
<td>List of Tables</td>
<td>x</td>
</tr>
<tr>
<td>List of Figures</td>
<td>xi</td>
</tr>
</tbody>
</table>

1 Introduction
   1.1 Working principles of LIDAR               2
   1.2 Motivation                                5
   1.3 Approach                                  7
   1.4 Thesis organization                       8

2 Related work
   2.1 Delineation of rasterized point cloud     10
   2.2 Delineation of raw point cloud            13
   2.3 Urban modeling                            16

3 Data and preprocessing
   3.1 Input data                                19
List of Tables

6.1 First comparative study of our proposed algorithm vis-à-vis other state of the art approaches ........................................ 57
6.2 Second comparative study of our proposed algorithm vis-à-vis other state of the art approaches ........................................ 58
6.3 Third comparative study of our proposed algorithm vis-à-vis other state of the art approaches ........................................ 58
6.4 Time taken by our algorithm on files of various size ......................................................... 59
6.5 Possible improvements in running time of stage 2 of our algorithm ................................. 61
# List of Figures

1.1 Use of LIDAR in meteorology ........................................... 2
1.2 Use of LIDAR in forestry ............................................. 3
1.3 The concept of multiple returns ...................................... 4
1.4 Use of structural information for peak identification ........... 6
1.5 The pipeline of our proposed approach ............................. 7

2.1 CHM derived from point cloud ....................................... 10
2.2 Detection of tree peaks using a $3 \times 3$ window .................... 11
2.3 The inverted watershed approach .................................... 12
2.4 Detecting locally highest seed points ............................... 12
2.5 Visual comparison between data sets used by others and us ....... 13
2.6 Adaptive clustering .................................................... 14
2.7 The normalized cut approach ....................................... 15
2.8 Use of geometrical structure in urban modeling .................. 16
2.9 3D buildings reconstruction by assembling simple urban structures extracted from a predefine library of 3D parametric blocks .......................... 17
2.10 3D buildings reconstruction by assembling simple urban structures extracted from a predefine library of 3D parametric blocks .......................... 18

3.1 The shape file and its usage ........................................ 20
3.2 Isometric view of shape file placed over a LAS file ............... 21

4.1 Visual illustration of branch tips ................................. 26
4.2 Visual illustration of false central peaks ......................... 27
4.3 The intuition behind false center ................................. 29

5.1 Detection of all locally highest points in the point cloud ....... 31
5.2 Detailed explanation of branch tip detection ....................... 32
5.3 Observation used for identification of false centers ............ 34
5.4 Detailed explanation for creating the 2.5 D forest surface ................................. 36
5.5 Simplicial complex ................................................................................................. 38
5.6 The 3D alpha shape ............................................................................................... 38
5.7 Alpha shape of LIDAR point cloud ........................................................................ 39
5.8 Variance of path over 2.5 D surface plotted against the height of the higher peak. . 41
5.9 Variance of path over 2.5 D surface plotted against the height of the lower peak. . 42
5.10 Range of path over 2.5 D surface plotted against the height of the higher peak. . 42
5.11 Range of path over 2.5 D surface plotted against the height of the lower peak. . 43
5.12 Normalized dip of path over 2.5 D surface plotted against the height of the higher peak. 43
5.13 Normalized dip of path over 2.5 D surface plotted against the height of the lower peak. 44
5.14 Variance of path over $\alpha$-shape plotted against the height of the higher peak. . . 44
5.15 Variance of path over $\alpha$-shape plotted against the height of the lower peak. . . 45
5.16 Range of path over $\alpha$-shape plotted against the height of the higher peak. . . . 45
5.17 Range of path over $\alpha$-shape plotted against the height of the lower peak. . . . 45
5.18 Normalized dip of path over $\alpha$-shape plotted against the height of the higher peak. . 46
5.19 Normalized dip of path over $\alpha$-shape plotted against the height of the lower peak. . 47
5.20 Range of path over 2.5 D surface plotted against the height of the lower peak. The green line and the black curve are few of the possible demarcators. ................... 48
5.21 The black dotted curve is the demarcator calculated for this specific plot. .......... 49

6.1 Evaluation of algorithm with the help of shape files ............................................. 52
6.2 Evaluation of algorithm with the help of shape files ............................................. 54
6.3 Failure cases of the normalized cut approach ....................................................... 56

7.1 Failure to detect branch tips .................................................................................... 63
7.2 Failure to detect branch tips .................................................................................... 64
7.3 Failure to detect false central peaks ....................................................................... 65
7.4 Failure to detect false central peaks ....................................................................... 66
7.5 Alternate methods for branch detection ............................................................... 68
Chapter 1

Introduction

LIDAR is a type of RADAR system which uses laser waves instead of radio waves to gather information about the surrounding. In this system, a source emits rays of laser which are reflected by objects present in the environment. By studying various properties of the reflected rays, such as intensity and direction, one can gather information about the location, shape and size of those objects. This process of acquiring information about objects without making actual physical contact with them is termed as remote sensing. The ability of LIDAR to gather information about the surrounding is widely being used for creating high resolution maps of tangible objects such as geographical landscapes, archaeological sites, forests or even entire cities.

However, in the beginning LIDAR was used to gather information regarding something which was rather intangible; phenomena which were nearly impossible for humans to touch, or even to look at. As early as mid 1960’s, the ability of LIDAR to study various meteorological phenomena had been acknowledged [5]. Researchers discovered that whenever a laser pulse was transmitted into the atmosphere, some back scattering of the radiation took place. The nature of the back scattered laser could give information about the presence and location of cloud and smog layer, location of temperature inversion and haze layers, and information about the ceiling height through rain or low level cloud structure (Figure 1.1). This new type of RADAR system which used laser instead of the traditional radio waves, was aptly named as Light Detection And Ranging, or more commonly, LIDAR.

The modern day LIDAR system has a variety of applications in fields such as archeology, geology and soil science, military, mining, robotics and forestry. The use of LIDAR for forest inventory collection and management has grown rapidly over the last decade. Delineation of individual trees in LIDAR point cloud data is often the starting point for estimating important forest inventory parameters such as biomass, yield of timber or even estimating the dominant tree species in the forest. Many of the existing tree delineation methods have a similar work-flow and the accuracy of delineation is dependent on how accurately the initial seed points (refer Section 1.2) are selected. In this
dissertation, we ask the interesting question “What does an ideal seed point look like?” and then develop a heuristic to identify a suitable set of seed points for an input point cloud data.

1.1 Working principles of LIDAR

The exact working principles of a LIDAR system depend upon the precise nature of use. However, the core idea remains the same. In this section, we focus on understanding the working principles of LIDAR as applied to forestry. Figure 1.2 shows schematically how LIDAR data for forestry is captured.

A LIDAR system is mounted underneath a small aircraft and flown over a forest region along a straight line. At quick successions, the system emits laser pulses which travel in a plane parallel to the one formed by the pitch and yaw axis of the aircraft. The angle between the path traversed by the laser and the yaw axis is known as the scan angle.

Laser pulses that are emitted from the system get reflected by various structures present in the forest. By measuring the time delay between transmission of a pulse and detection of its reflection, the distance of an object from the aircraft can be calculated. If the altitude at which the aircraft is flying and the scan angle of the laser pulse is taken into account, the exact location of the target object with respect to that of the aircraft can be found. A high-precision global positioning system (GPS) is used to keep a track of the actual location of the aircraft. Thus for each reflection detected by the
Figure 1.2: Schematic diagram showing the use of LIDAR in forestry. An aircraft with a LIDAR system mounted underneath is flown along a straight line known as flight-line. The LIDAR system emits laser pulses which travel in a plane parallel to the one formed by the pitch and yaw axis of the aircraft. A narrow strip of forest is scanned using these laser pulses each time the aircraft flies over a flight-line. Data corresponding to multiple flight-lines is merged together to get the overall view of the entire forest (image courtesy [9]).
LIDAR system, a set of $x$, $y$ and $z$ coordinates can be calculated in a global frame of reference.

As is evident from the Figure 1.2, only a small portion of the forest underneath the aircraft can be scanned at a time. Since the aircraft is usually flown along a straight line, called the flight-line, this region is usually a narrow strip. The width of this strip is known as the scanning swath. In order to scan a large region, the aircraft is flown over multiple, usually parallel, flight-lines. Later, the data corresponding to individual flight-lines is merged together to produce the overall view of the forest.

Whenever a laser pulse emitted by the LIDAR system interacts with an object in the surrounding, a portion of it gets reflected back, while the rest is transmitted further as a pulse of considerably reduced strength. This means that for each laser pulse emitted, the LIDAR system could detect more than one reflections. These reflections are commonly known as returns. Figure 1.3 shows this concept of multiple returns for each emitted laser pulse. The number of returns obtained depends

Figure 1.3: Figure explaining the concept of multiple returns. The graphical plot towards the left schematically shows the strength of a reflected laser pulse as a function of height. Points corresponding to local maxima in this plot signify presence of some object. Since a laser beam can penetrate through the foliage, each laser pulse usually gives rise to more than one return (Image courtesy https://e-education.psu.edu/lidar/l8_p3.html).
on a variety of factors, such as nature of vegetation, strength of laser pulse etc.

Each return captured by the LIDAR system gives a data point with $x$, $y$ and $z$ coordinates in a global frame of reference. Thus when data corresponding to all flight-lines is merged together, a 3D point cloud is produced. This point cloud is usually saved in a LAS format which is a format developed primarily for exchange of LIDAR data.

1.2 Motivation

The majority of existing methods for delineating individual trees have a similar pipeline. Initially, a set of suitable seed points is detected in the input data. This is followed by a region growing step, during which rest of the points are iteratively assigned to one of the seed points, thus growing the region representing individual trees. This is true for methods which work directly on the LIDAR point cloud data, as well as for those which work on the Canopy Height Model (CHM) derived from the data.

These seed points are usually meant to represent the highest point in each of the tree, and hence can be called the true-peaks of the trees. Ideally each tree in the LIDAR point cloud data would possess exactly one true-peak. The accuracy of tree delineation is clearly linked to how accurately these true-peaks are identified. There are not many methods that do not utilize this concept of true-peak. However, almost all the existing methods study the distribution of the LIDAR points in a localized region to identify true-peaks; the highest point in a small, appropriate sized neighborhood is often selected as the true-peak.

At this point, it is worth emphasizing the difference between a local-peak (or simply peak) and the true-peak of the tree. A local-peak is a locally highest point within a tree, and hence is a candidate for the true-peak, whereas a true-peak is the highest point in the entire tree. A tree may have more than one candidates for the true-peak. One of these would actually be the true-peak of the tree, while the rest would be called the false-peaks. Figure 1.4 explains why some of the local-peaks should be regarded as false-peaks and not as the true-peak of the tree.

Hence, we could say that the existing methods use a localized approach towards identifying these peaks. The drawback with such approaches is that they fail to utilize any geometrical information related to the overall structure and shape of the tree. We believe that an approach that is non-local will give more reliable and accurate true-peaks.

In this work, we propose one such non-local approach for true-peak identification. The main contribution of this work is that it is possibly the first approach that attempts to exploit geometrical information related to the more global structure of trees by studying the way peaks are connected and arranged with respect to each other. We believe that there is an abundance of information related to geometrical structure concealed in the LIDAR point cloud data. In a related but different problem setting of modeling urban cities, extracting and utilizing this geometrical information is
Figure 1.4: A local-peak is a point which is locally highest in a small neighborhood (shown by arrows in these images) whereas a true-peak is defined as the highest point of the tree. Thus only a local-peak could be considered the true-peak of a tree. (a) Under ideal circumstances, a tree would have only one local-peak which would also be the true-peak of the tree. (b) Elongated branches growing in the upward direction may result in peaks near the periphery of the tree. Such peaks are known as \textit{branch tips}. These should not be identified as true-peaks. (c) If a group of peaks are found near the center of tree, then one of them would the true-peak of the tree, whereas rest should be identified as \textit{false central peaks}. 
relatively easy since buildings usually exhibit regularity in shape and structure. A plethora of work on 3D reconstruction of urban buildings from point cloud such as Elaksher et al. [8], You et al. [29], Verma et al. [26] and Lafarge et al. [15], all rely heavily on the regular geometric structure of buildings. These works mostly treat natural vegetation and trees as undesirable artifacts in the point cloud, and hence filter them out at the initial stage itself. The ideas presented in these works cannot be directly applied on the point cloud data of trees, since trees do not exhibit the kind of regularity that buildings often do. This would mean that extracting structural information from point cloud data of trees is not easy and this makes the true-peak identification a challenging problem.

### 1.3 Approach

Figure 1.5 gives an overview of our proposed multistage pipeline for detecting true-peaks in a given point cloud. We have designed a series of filters which are sequentially applied on the input point cloud data. To begin with, we can assume that each of the point present in the point cloud could possibly be the true-peak of some tree. Each subsequent filter applied on the point cloud identifies (and removes) points which are unlikely to be the true-peak of any of the trees. Our filters are conservative in the sense that they remove a point from the list of potential peaks only when there is a high level of confidence that the peak is a false one. Thus each filter reduces the possible candidates for true-peaks in a sequential manner. At the end of the pipeline, our approach is able to remove most of the false-peaks, while ensuring that most of the true-peaks are identified correctly.
1.4 Thesis organization

The remaining parts of this thesis are organized as follows. In Chapter 2, we summarize various approaches used for delineating trees from point cloud data. We also look briefly into ideas that are used in modeling urban cities from point clouds, which will enable us to better appreciate the complexity of our problem. In Chapter 3, we describe the nature of our input data, and the preprocessing performed on it. Chapter 4 gives the main intuition behind our approach and lays the groundwork for detailed discussion of our proposed method which is carried out in Chapter 5. Chapter 6 discusses our evaluation method and presents results obtained by our algorithm. Finally, in Chapter 7 we present our conclusions and propose ideas for possible future work.
Chapter 2

Related work

In this chapter, we provide an overview of previous works for delineation of trees from point cloud. We also look briefly into related ideas used in modeling urban cities from LIDAR point. Having an understanding of the mathematical ideas used in urban reconstruction would facilitate a better understanding of the complexity of our problem.

It should be noted that our work, in the current form, focuses on finding true-peaks of the trees (as explained in Section 1.2) rather than delineating them. However, our end goal would be to delineate the trees, and the number of peaks identified correctly by our algorithm can give us an intuition of how well our proposed method performs vis-à-vis other approaches for tree delineation.

One of the factors that governs the overall quality of tree delineation is the accurate identification of the true-peaks of the trees. However, to the best of our knowledge, there has been no previous work which focuses on detecting the correct set of true-peaks, although most of the delineation methods inherently try using the concept of a peak point.

Broadly speaking, the methods for delineating trees in LIDAR point cloud data could be divided into two categories. The first category comprises methods which rasterize the input point cloud data. The point cloud is converted into a Canopy Height Model (CHM), which is actually a height map representing the heights of the outer most layer of leaves (the ones immediately adjoining the atmosphere) of individual trees. Thus the CHM can be treated as a raster image. Meaningful segments representing trees are segmented out of this image. The second category comprises methods which take input data in a form that more closely resembles the original raw point cloud. In the next two sub-sections, we outline some of the prominent works. Later, in Section 2.3, we discuss related approaches developed for modeling urban cities.
2.1 Delineation of rasterized point cloud

A number of methods were proposed during the last decade (2001-2010) which worked on raster images (e.g., CHM) extracted from the original point cloud data and not on the point cloud itself. Conversion of point cloud into a CHM not only leads to loss of data but can bring in spatial errors depending upon the precise method used for creating the CHM (Guo et al. [10]). Hence it is unclear why researchers preferred working on a raster image rather than the original point cloud.

We feel there could be two possible explanations for this. Firstly, the conversion from point cloud to raster image reduces the amount of data that needs to be subsequently handled. This drastically reduces the overall running time as well as the memory requirements for the algorithm. However, with the presence of enormous computing power even in the lower end consumer systems, the focus could now shift primarily on the quality of delineation produced by the algorithm, rather than speed and efficiency. In our opinion, the second reason rasterization based methods gained popularity was because the delineation of trees was primarily treated as a problem belonging to the field of Remote Sensing and Photogrammetry. Hence a more rigorous geometrical approach was not opted for and the geometrical information embedded in the point cloud often went unnoticed.

![Isometric view of a CHM, which is actually a height map, derived from a LIDAR point cloud. The pseudo colors are used to indicate height, red representing taller regions while blue the shorter ones. Figure taken from [11].](image)

Figure 2.1: Isometric view of a CHM, which is actually a height map, derived from a LIDAR point cloud. The pseudo colors are used to indicate height, red representing taller regions while blue the shorter ones. Figure taken from [11].

The first prominent idea for delineating trees from raster images was proposed by Hyypa et al. [11]. They first converted the point cloud into a raster image (CHM) (Figure 2.1). Then pixels which had highest intensity in a local neighborhood were selected as peak points of the trees. To achieve this goal, a 3 pixel × 3 pixel square window was placed over each pixel such that the center
of the window coincided with the pixel (Figure 2.2). If the pixel had higher intensity than all the eight neighboring pixels in that square window, it was selected as a tree peak. This was followed by a region growing algorithm (an inverted water shed algorithm, Figure 2.3) to delineate the trees.

Variations of the above stated method were used in many of the works that followed. For example, Popescu et al. [20] decided to use a variable window size rather than a fixed 3 pixel × 3 pixel. The intuition was that taller the tree, wider the crown. Hence a larger window should be selected for higher points whereas a smaller one for lower points (Figure 2.4). The relation between the windows size and height was derived empirically from real life data. A variation of the above method was used by Chen et al. [4]. They used a Gaussian smoothing kernel on the CHM prior to using the variable window concept. Their claim was that the Gaussian smoothing helped in suppressing mild peaks, which were not the true peaks of the tree.

The main assumption behind all the above mentioned methods was that the shape of the tree is more or less conical, and hence if a point is highest in a small appropriate sized neighborhood, it would be the highest point of the tree. Such methods make a very strong assumption regarding the shape of the tree (see Figure 2.5). Also, these methods are very sensitive to the choice of window size. Ideally the window size should be as big as the radius of the crown of the tree. Hence it is obvious that these methods work well only on data set which have predominantly cone shaped trees.

In 2003, Brandtberg et al. [3] applied the concept of scale space for delineating trees from
Figure 2.3: An illustration of watershed segmentation algorithm. (a) The CHM constructed from LIDAR point cloud. (b) The complement (inversion) of the CHM. (c) Dams built at the divide line to complete watershed. Figure taken from [4].

Figure 2.4: A portion of the CHM derived from LIDAR point cloud and the variable windows that successfully identified locally highest points (tree peaks). Height of the CHM in a region and the nature of vegetation (if known beforehand) is taken into account to decide the size of window. Image courtesy [20].
Figure 2.5: Most of the existing methods for tree delineation are capable of handling data sets where trees are predominantly conical shaped. (a) and (b) show the data sets that were used by Li et al. [17] and Rahman et al. [21]. Their data predominantly contained uniform conical shaped trees. In contrast, trees in our data set have more diversity; trees often show flat tops rather than conical as shown in (c). Consequently, traditional tree delineation methods are not able to handle such data sets.

the CHM. The concept of scale space states that “any structures in an image (CHM) exists as meaningful entity only over certain range of spatial scales”. For example, if a viewer wants to discern the structure of a tree present in a forest, he should ideally be viewing the tree from a distance of a few meters from the tree. If he attempts to view the tree from a distance of a few kilometers, he would see the structure of the entire forest but not that of the individual tree. On the other hand, if he views the same tree from a very close proximity, he would see the structure of branches or leaves, but not that of the tree. So in accordance, they performed the Gaussian smoothing of the CHM at different scales and tried detecting meaningful blobs.

Another work worth mentioning would be, Kwak et al. [14]. They used a morphological image segmentation approach on the height maps. However, the results they achieved were at par with those obtained by various approaches based on region growing mentioned earlier. They also acknowledged that their method, like majority of others, was more suited for coniferous trees, which are predominantly conical in shape. Thus, we could say that most of the methods that work on raster images, give comparable results and work well only on data having conical shaped trees.

2.2 Delineation of raw point cloud

In 2008, Pang et al. [19] worked on the Digital Terrain Model (DTM), which is a height map similar to CHM. Their algorithm starts with locally highest points, followed by region growing for creating initial tree segments, but later on it attempts to merge over segmented trees. However, their end goal was to estimate the amount of timber in the forest and not the number of trees. Hence they did not mention the accuracy of their tree delineation algorithm. Although the idea they gave seems
promising, it also seems to be very much dependent on data quality and structure of the trees. A similar approach was proposed in 2010 by Lee et al. [16]. Their approach first over segments the point cloud and then uses an adaptive clustering to hierarchically merge smaller clusters into meaningful bigger clusters representing individual trees (Figure 2.6). However, they worked with managed forests which provided more regularity in the data.

One of the most promising works for tree delineation uses the normalized cut approach (Figure 2.7). The normalized cut is one of the prominent methods in the field of image segmentation. In this method, the image to be segmented is first represented as a graph and then the eigenvectors of the Laplacian matrix of this graph are utilized to partition the graph (and hence the corresponding image) into meaningful segments. Yao et al. [28] used the same algorithm to segment (delineate) trees from LIDAR point cloud. However the algorithm has a large memory footprint, which often makes it impractical for big data sets and requires fine tuning a large number of parameters to get an optimum result. Lu et al. [18] suggested a bottom-up approach for delineating trees. Their proposed method first identifies the tree trunk and uses it as the seed region for growing the tree. However, in dense forests, where laser rays cannot penetrate the foliage, identifying the trunk is usually not possible, and this limits the practicality of the proposed method.

The idea of fitting predefined geometrical shapes to the point cloud to delineate trees was explored by Tittmann et al. [25]. Thus we can say that they make a very strong assumption regarding
Figure 2.7: Yao et al. [28] used the concept of a normalized cut for tree segmentation. The input point cloud data is first voxelized (left). This reduces the amount of data to be subsequently processed and the memory footprint of the algorithm. The voxels are then converted into a weighted undirected graph $G = (V, E)$, where nodes of the graph are the 3D voxels, and an edge is formed between every pair of nodes. The weight on each edge, $w(i, j)$, represents the likelihood that the two voxels belong to the same tree. A normalized cut segments this graph into two subgraphs ($A$ and $B$ on the right). The subgraphs are repetitively subdivided unless individual trees are segmented out of the voxel.
the possible shape of the trees. Li et al. [17] also rely on finding locally highest point in the point cloud, followed by a region growing approach. Thus, their overall approach is similar to those used by researchers on CHM. A similar region growing approach was proposed by Rahman et al. [21]. However, instead of taking the locally highest points in the point cloud as the seed point, they identified regions with the highest point density and treated them as the seed regions. The understanding was that the density of points would usually be higher towards the center of the tree, and would gradually decrease as one would move outwards. However, they did not notice any major improvement over the methods that used locally highest points as the seed points.

### 2.3 Urban modeling

Urban modeling primarily involves 3D reconstruction of buildings from LIDAR point cloud data. Buildings usually exhibit geometrical regularity in shape and structure and this fact can often be used advantageously for 3D reconstruction. This makes the problem comparatively more tractable (Figure 2.8).

![Figure 2.8: A DTM of Monroe County, NY derived from a point cloud. Parts of DTM that correspond to buildings show more consistency and regularity in their geometrical structure than parts which correspond to vegetation. This regularity in geometric structure makes the task of reconstructing buildings more tractable than delineating trees (Image courtesy http://www.airborne1.com/.)](image)

For example, the method proposed by Elaksher et al. [8] uses a voting mechanism on the entire point set to identify potential planes in the point cloud data. Planes which get more than a threshold number of votes are accepted as valid planes that would comprise the walls/roofs of the building. You et al. [29] proposed a method that was inspired by the idea of Constructive Solid Geometry.
Their approach, which requires minor user assistance, agglomerates simple geometrical primitives such as planes, cubes, cylinders and spheres to approximate the shape of complex looking buildings (Figure 2.9).

Figure 2.9: The approach proposed by You et al. [29] combines simple geometry primitives to approximate shapes of complex looking building structures, as shown in the left image. The image on right shows a building model constructed by their approach (cyan colored) overlaid over the mesh data representing the LIDAR point cloud (grey colored). The model in cyan provides a good approximation of the building present in the LIDAR input data.

Since the LIDAR scanned data of buildings is by nature 2.5 dimensional, the most predominant feature in the point cloud of an urban city would be the roof of the building. Verma et al. [26] used this fact to their advantage and made an attempt to represent complex roof topology using simpler parametric shapes. Lafarge et al. [15] attempted to reconstruct 3D buildings from a single Digital Elevation Model (DEM) by assembling simple urban structures extracted from a predefine library of 3D parametric blocks (Figure 2.10).

The commonality between all these methods is that they rely heavily on the regular geometric structure of buildings. They mostly treat natural vegetation as undesirable artifacts in the point cloud, and hence these are filtered out at the initial stage itself. For example, [30] uses local features such as regularity, horizontality, flatness and normal distribution in the vicinity of a point to detect if the point belongs to a building or to a tree.

Hence we can say that majority of the works for modeling urban cities attempt to utilize geometrical information encapsulated in the point cloud. However, the ideas presented in these works can not be directly applied on point cloud data of trees, since trees do not exhibit the type of regularity that buildings often do. This also means that extracting structural information from point cloud data of trees is not easy and this makes the problem of identifying true-peaks challenging.
Figure 2.10: Lafarge et al. [15] approach attempts to reconstruct 3D buildings by assembling simple urban structures extracted from a predefined library of 3D parametric blocks. The image on the left shows a part of their library. The image on the right shows complex building structures reconstructed by fitting appropriate building blocks to the DEM.
Chapter 3

Data and preprocessing

This project was carried out in conjunction with our industry partner, which specializes in providing a variety of geospatial solutions such as urban terrain visualization, geo-referenced 3D visualization and timber species identification. The delineation of trees in point cloud data is of special interest to our industry partner since it is expected to improve the accuracy of timber species identification. In this chapter, we describe the nature of our input point cloud data and discuss the steps involved in its preprocessing.

3.1 Input data

The data, which was provided by our industry partner in the form of standard LAS files, comprises a wide variety of trees in their natural habitat. The land topography in our data set was highly irregular and at places had steep elevation drops. The data was captured using Terra Remote MKIII sensor over an area of 150 kilo-hectare during July 2010 through October 2011. The flying height of the LIDAR sensor was 600 meters. The data was captured in a leaf-on condition, i.e., before the trees started shedding their leaves for the winter season. The LIDAR sensor used provides a minimum of 8 points/meter$^2$ on the barren ground.

All the points in the LAS files had already been classified as either belonging to the ground (henceforth referred to as ground point) or to the trees (henceforth referred to as tree point). Throughout this work, the vertical direction, which is the usual direction of tree growth, is referred to as the $Z$ direction. Hence, the $X - Y$ plane would be a horizontal one perpendicular to the $Z$ axis. The $X$ and $Y$ axis are orthogonal to each other, however, their exact orientation does not affect the working of our algorithm.

For each LAS file, we were also provided the corresponding shape file. The shape file contained two dimensional polygons which outline the shape of trees as viewed from top. These polygons
Figure 3.1: The shape file can be viewed as a stencil with ground truth polygons cut out from it. (a) The ground truth polygons are polygons outlining the boundary of individual trees as viewed from top. (b) When placed over the corresponding LAS file, the polygons segregate the points in LAS file that belong to the same tree.
Figure 3.2: Isometric view of shape file from Figure 3.1 placed over the corresponding LAS file. The vertical lines in red are drawn to emphasize the idea that the polygons in the shape file define the boundary of trees as viewed from top.
were manually created by experts in the forestry industry and henceforth will be referred to as ground truth polygon of individual trees. The shape file can be viewed as a stencil with ground truth polygons cut out from it. When placed over the corresponding LAS file, this stencil segregates the points that belong to the same tree, and thus acts as the ground truth for our data (Figure 3.1 and 3.2).

In total, we had been provided 125 LAS files. Since these files contained both coniferous and deciduous trees, they can be considered a suitable representation of a typical forest that grows in the province of British Columbia, Canada. The prominent coniferous species present in our data set were douglas fir, western redcedar, western hemlock, balsam fir and grand fir while the prominent deciduous were red alder, bigleaf maple and cottonwood. We visually inspected these files and hand-picked the 22 toughest files for testing the accuracy of our algorithm. A file is considered to be tough if the trees contained in the file showed a considerable diversity in shape and size and were densely packed with little gap between them. Henceforth these 22 files will be referred to as Data Set I. Another 47 files, where the trees showed relatively more uniformity in shape and size and were sparsely scattered, will collectively be referred to as Data Set II and can be considered easier of the two data sets. As discussed in Chapter 6, one of these data sets would be used to identify certain parameters needed by the algorithm, while the other data set would be used to test the effectiveness of the algorithm. Intuitively, this means that the data set used for training the algorithm is different from the one used for testing the algorithm.

Data Set I contained a total of 452 manually delineated trees while Data Set II contained 255 manually delineated trees. These LAS files contained an average of 19 tree points/meter$^2$

3.2 Preprocessing

Throughout the execution, our algorithm would need to calculate the height of any given tree point from the ground. Ideally, for each tree point, we would like to have a corresponding point on the ground, which has the same $X$ and $Y$ coordinates as that of the tree point. We call such a point a virtual ground point. The absolute difference between the $Z$ coordinates of the tree point and the corresponding virtual ground point would give us the height of the tree point. It should be noted that as suggested by our industry partner, tree points which are within a distance of 6 meters from ground, are simply ignored by our algorithm, since such points quite often belong to the undergrowth.

To achieve our goal, we proceed as follows (refer Figure 3.3 for visual details). We first construct the Digital Elevation Model (DEM), which is a regular 2D grid (of 0.5 meter grid spacing) that represents the absolute elevation of the ground at regularly spaced discrete locations. This is achieved by interpolating a surface through all ground points in the LAS file. The precise method used for interpolation is an active research area of its own. Often kriging is considered to be the state of art
Figure 3.3: Steps involved in the creation of virtual ground points for each tree point. It should be noted that a virtual ground point (shown in red) is calculated for each tree point (shown in green). Notice that in (a), there are very few ground points (shown in blue) directly underneath the tree. However, in (f) each green tree point has a corresponding virtual ground (red) point.
method for terrain modeling. However, it is usually slow and has a high memory requirement, which makes it less suitable for practical scenarios. At the same time, the precise choice of interpolation method would depend upon the nature the underlying land-form [2]. Based on the findings of the above mentioned work, we decided to use the nearest neighbor interpolation. However, it should be noted that the core idea of our proposed method assumes that an appropriate interpolation method suitable for the land-form was used.

Next, for each tree point in the LIDAR data, we find the DEM grid within which the tree point would fall if projected vertically downwards. A bilinear interpolation performed on the altitude values at the four corners of this DEM grid gives the virtual ground point for the corresponding tree point. However, it should be noted that as suggested in Kidner et al. [12], higher order interpolation, such as bicubic, biquadratic and biquintic, would almost always produce better results than bilinear interpolation and as such are generally preferred over bilinear interpolation. But since the core idea of our proposed method is not dependent on the choice of interpolation method, we opted to choose the bilinear interpolation for its speed and ease of implementation.

Our algorithm would also extensively perform nearest neighbor search in three dimensional euclidean space. As such, it is desirable to place the LIDAR points into an appropriate data structure which would speed up the process. We use a simple and straightforward implementation. We divide the entire 3D coordinate system into voxels of dimension 4 meter × 4 meter × 4 meter. To find the nearest neighbor of any point, we hence need to only query the points present in a total of 9 voxels, including the one in which the point lies. Since throughout our algorithm two points are considered close only when the distance between them is less than 2 meters, such a voxelization serves our purpose. However, more sophisticated data structures such as K-D tree could be opted for.
Chapter 4

“What makes a false-peak?”

In this chapter, we attempt to answer the question “What does a false-peak look like?” and provide the main intuition behind our algorithm. By definition, a true-peak must be the highest point of a tree. Hence only a locally highest point can be considered a potential true-peak of a tree. To begin with, our algorithm detects all locally highest points in the point cloud thus giving rise to a list of candidates for true-peaks. At this stage, our algorithm attempts to include all the true-peaks in the point cloud in this candidate list. Later, our algorithm would identify candidates which seem unlikely to be the true-peaks of any of the trees, and remove them from this list. In order to find the characteristics of these false-peaks, we studied the point cloud data, and concluded that false-peaks could broadly be categorized into two classes as described in the next two sections.

4.1 Branch tips

The first category is that of those peak points which are tips of elongated branches growing in upward direction. These are represented as green points in Figure 4.1. We call such peak points the branch tips. These are considered false-peaks since the true-peak of the tree is usually located close to the center of the tree as shown in red in the same figure. If an isolated tree is viewed from the top, branch tips tend to lie near the periphery of the individual tree. So intuitively, to distinguish a branch tip from the true-peak, we should be studying the distribution of the points in the vicinity of the peak. This can be achieved by grouping together nearby points to form a cluster. A peak would be called a branch tip if and only if it lies near the periphery of this cluster (Figure 4.1(b) lower image).

A natural question to ask would be “how many points should be included in the vicinity?” As explained in detail in the next chapter, our algorithm does not require to predefine such parameters, rather, it exploits the non-local structural information present in the point cloud data to its advantage;
Figure 4.1: Visual illustration of branch tips: (a) The top image is the side profile of a tree exhibiting branch tips, while the bottom is the top view of the same tree, along with the boundary of the tree marked by magenta polygon. Branch tips, which are shown in green, are locally highest points at the end of elongated branches growing in the upward direction. As is evident from the lower image, branch tips tend to lie near the boundary of the tree. (b) Close up view of the same tree shown in (a). To identify branch tips, we need to study the distribution of the points in the vicinity of the local-peaks. This is achieved by forming a cluster of nearby points. The lower image shows the cluster corresponding to each of the local-peak as seen from top. A peak would be called a branch tip only if it lies near the periphery of its cluster.)
Figure 4.2: Visual illustration of false central peaks: The image at the top is the side profile of a tree exhibiting a false central peaks, while the image at the bottom is the top view of the same tree, along with the boundary of the tree marked by magenta polygon. False central peaks, just like branch tips, are also locally highest points. However, the distribution of points in the vicinity of false central peaks is considerably different from that of branch tips. In the above image, the point in green is a false central peak. Unlike branch tips, false central peaks tend to lie near the center of the tree, and point distribution around false centers is similar to one around true-peak of the tree.
points in the vicinity of peak candidates are sequentially added to existing clusters, as a result of which these clusters grow in size. Eventually the clusters hit into each other, and at that point of time, we study the structural information related to these clusters to decide whether a local-peak is a branch tip or not.

4.2 False central peaks

The second category of false-peaks originate when a tree has more than one peak located close to the center of the tree. Such false-peaks are termed as false central peaks. Figure 4.2 shows a tree which has two peaks, and none of the them is close to the periphery of the tree. Thus none of them could be called a branch tip.

However, since both the peaks belong to the same tree, we expect these peaks to have a sense of strong connectivity between them. We now introduce the term gap which intuitively signifies the lack of tree points in the region between the two local-peaks and gives us a sense of how strongly the two peaks are connected. On careful visual inspection of our data, we concluded that if two peaks belong to different trees, they tend to have a large gap between them. On the other hand, the gap tends to be small if the peaks belong to the same tree. Thus the amount of gap present between the two peaks can give us an indication whether the two peaks belong to different trees or to the same tree.

We now provide the intuition that we use to measure this gap. Imagine a rodent perched at the top of a potential peak of a tree. Let us call this peak A. It wishes to go to a neighboring peak B. The rodent can either walk along the branches of the tree or make small jumps between two neighboring leaves, which are separated by a distance of, say, not more than a few centimeters. The rodent has no prior knowledge of whether the two peaks A and B belong to the same tree or to different trees. However, the rodent is competent enough to find out the shortest path between the two peaks.

Now assume that the two peaks in question belong to two different trees, placed far apart from each other. In such a case, the only way the rodent can go from peak A to peak B is by first traveling from peak A to the ground, moving a considerable amount of distance on the ground to reach the foot of the other tree, and then climb up that tree to reach peak B. Thus we could say that during the overall journey, the rodent would have to make a considerable amount of downwards movement followed by a considerable amount of upward movement; the path taken was certainly not flat. Let us compare this situation with another where the two peaks A and B belong to the same tree. No matter which path the rodent decides to choose, it will never have to touch the ground. In general, if peaks A and B belong to two different trees, the path would have a considerable amount of downward and upward portion, which results in a dip and indicates a large gap between the two local-peaks. On the other hand, if the peaks belong to the same tree, the path is relatively flat and
If the rodent wishes to travel between two peaks belonging to the same tree, e.g., the green peak and the black peak, the shortest path it would select will be relatively flat. One such path is shown in green. On the other hand, if the rodent wishes to travel between peaks belonging to different trees, e.g., the green and the blue peak, the path taken would have considerable amount of dip and may resemble the one shown in blue.

This indicates a lack of gap between the two local-peaks. This idea is schematically shown in Figure 4.3. Thus we could say, given a pair of local-peaks in our input point cloud, we could tell whether they belong to the same tree or not by studying the nature of path connecting the two peaks. The precise way we do this is explained in detail in the next chapter.
Chapter 5

Algorithm

This chapter explains in detail the working of our proposed algorithm for detecting true-peaks of the trees. As stated in Section 1.3, our algorithm can be viewed as a three stage pipeline which constitutes a series of filters that are sequentially applied to the point cloud data. Each subsequent filter applied to the point cloud identifies (and removes) points which are unlikely to be true-peaks of any of the tree. Thus each filter attempts to reduce the possible candidates for true-peak in a sequential manner. The three filtering stages are namely, identification of all candidate peaks, detection of branch tips and removal of false central peaks.

5.1 All candidate peaks

The input to our algorithm is a raw point cloud data. To begin with, each point in the point cloud could possibly be the true-peak of some tree. In the first stage, the algorithm identifies locally highest points in a reasonably small neighborhood, which is defined by the probe radius (see Figure 5.1 for details). The process of testing whether any point is locally highest point is as follows. Firstly, a vertical line running parallel to the Z axis is dropped from the point. Using this line as the axis, a cylinder of infinite height and radius equal to probe radius is constructed. If the initially selected point is higher than all other points inside this cylinder, it would be called a locally highest point and is a candidate for true-peak. An ideal choice for probe radius would be one that is smaller than the crown radius of most of the trees, so that at least one local-peak is identified per tree.

5.2 Detection of branch tips

During the second stage, a filter is used which identifies (and eliminates) branch tips among the candidate peaks. As stated in Section 4.1, the core idea is to group points that lie in the vicinity of a
local-peak to form a cluster and study the structural information related to these clusters to decide whether a local-peak is a branch tip or not.

To achieve our goal, we developed a plane sweep algorithm. Imagine a horizontal plane that sweeps through the point cloud in the downwards vertical direction (parallel to \(Z\) axis). To begin with, all points in the point cloud lie below this plane and are in the so called \textit{inactive state}. Whenever the plane hits a point, that point becomes \textit{active}. As soon as a point becomes active, it has two options. If it was identified as a local-peak (peak candidate) in the first stage, it becomes the \textit{seed point} of its own cluster. Else, it searches for the closest point amongst all the points that were previously active, and joins the cluster of that closest point. Intuitively, this means that this newly activated point joins the cluster that is closest to it. Thus existing clusters grow in size (refer Figure 5.2 for details). Now we define a few properties of a cluster.

- To find the \textit{boundary} of a cluster, we simply project all the points of the cluster onto a horizontal plane (\textit{defined by} \(Z = 0\)) and compute the convex hull of the projected points. The convex hull represents the boundary of this cluster.

- The \textit{distance} between two clusters is the smallest distance between all possible pair of points where the pair is made up of one point from each cluster.
Figure 5.2: Detailed explanation of the second filtering stage, i.e., branch tip detection. In each of the image, the top portion shows the side view of the point cloud. The black points are inactive points which so far, have not been touched by the sweep plane. The lower portion in each image shows the same point cloud as seen from the top. For sake of clarity the inactive black points are not shown in top view. Input to the second filtering stage is point cloud with all possible peak candidates marked in red (a). As the plane sweeps downwards, points become active, get associated with clusters and clusters grow (e.g., green cluster in (b)). Branches are clusters which have peak point near their boundary, and have a natural tendency to grow towards parent tree as shown in (c). We keep a track of the cluster boundary, represented by convex hull in top view. Branches eventually merge into parent tree (d). As the sweep plane moves further down, new branches are detected (e) and they too merge into the parent tree (f).
• We also introduce a notion of relative heights of the clusters. Given two clusters, the cluster which has the seed point with a larger $Z$ coordinate is called the higher cluster. The other cluster would naturally be called the lower cluster.

• A cluster would be a possible branch if its seed point, which also would be the highest point in the cluster, is close to the boundary of the cluster. In our current implementation, the horizontal distance of the seed point of the cluster is calculated from all the points that form the boundary (convex hull) of the cluster. If this distance is less than certain threshold, the seed point is considered to be at the boundary of the cluster and hence such a cluster qualifies to be called a branch. Although we would like this threshold to be as small as possible, we noticed that choosing a threshold of less than 0.25 meter considerably reduces the success rate of branch tip detection and hence in our current implementation, this threshold was selected to be 0.25 meter.

As clusters grow, they will eventually hit each other. If the distance between two clusters becomes less than a certain threshold, it implies that the clusters are trying to merge into each other. We found that a threshold of 0.5 meter was reasonable enough for our dataset. If the distance between two clusters is smaller than this threshold and the lower cluster is identified as a possible branch, then it is merged into the higher one, forming a single cluster and the seed point of the lower cluster would no more be considered a seed point. It should be noted that this seed point was identified as a potential peak in the previous step (Section 5.1) of the algorithm. However the merging process identified is as a branch tip and removed it from the list of potential peaks. As the plane sweeps downwards, all clusters representing the branch would eventually merge into some other (possibly non branch) cluster. Thus at the end of plane sweep, only those clusters which don’t resemble a branch would survive. It should also be noted that a cluster which got identified as a non branch at some point of time, would never become a branch again. Figure 5.2 shows this idea in detail. As visible in the figure, any branch in the point cloud has a natural tendency to get connected to the tree to which it belongs. Thus we see, our algorithm never explicitly defines a vicinity (as suggested in Section 4.1). Rather, it uses the natural tendency of a branch to merge into the parent tree to its advantage.

5.3 Removal of false central peaks

As stated in Section 4.2, if the gap between two potential peaks is large, then there is a high likelihood that the two peaks belong to different trees. Conversely, a lack of gap would indicate that the two peaks most probably belong to the same tree. Hence we need to find an adequate quantifiable measure of the amount of gap present between the two potential peaks. This section aims at developing such quantifiable measures.
5.3.1 Shortest path

In Section 4.2 we provided the intuition of the shortest path that the rodent would take if it had to go from one local-peak to another. We argued that its shape, which is a non-local geometrical piece of information, is related to the amount of gap between the two potential peaks and hence could be used to identify false central peaks. Figure 5.3 shows how this concept could be applied to our point cloud data. This section presents the precise method for developing such a path. Later in Section 5.3.2, we explore some of the characteristics of the path that could be used for quantifying the gap between two peak candidates.

To create a path between any two points, we require an underlying notion of connectivity, which accounts for how are the points connected to each other. We explore two different approaches for defining this connectivity. This resulted in two slightly different path definitions, namely:
• **2.5 Dimensional (2.5 D) surface assisted path**: An underlying 2.5 D surface helps in defining the path.

• **α (Alpha) shape assisted path**: An underlying α-shape helps in defining the path.

As is evident in Section 6.3, calculating the alpha shape assisted path is more time consuming than calculating the 2.5D surface assisted path. However, we decided to test the α-shape based approach primarily to support our argument that the nature of the path connecting two peaks could indeed be helpful in detecting the false central peaks. We now describe in detail the above mentioned approaches.

### 5.3.1.1 2.5 D surface assisted path.

In this approach, we make an attempt to construct a surface that would represent the outer most layer of leaves of individual trees as viewed from top. This layer of leaves, which immediately adjoins the atmosphere, would form a 2.5 D surface.

Unlike the urban modeling scenario, where the input LIDAR data can be treated as 2.5 dimensional for all practical purposes, the LIDAR data that we work with is inherently 3 dimensional. This is due to the fact that the laser beam can penetrate the foliage of trees. Hence we cannot uniquely define a 2.5 D surface for our data set. Moreover, majority of the existing methods for surface reconstruction work on point cloud derived from solid objects, where defining the reconstructed surface is relatively less ambiguous. Hence such methods do not serve our purpose. Intuitively, it may seem that the first-return LIDAR points would form the outermost layer of leaves (see Figure 1.3), but this is not always the case. Our observation was that quiet often the first-return pulse was buried deep inside the point cloud.

Hence, to create an approximation of the outermost surface of forest, we proceed as follows: We select all tree points which are locally highest in a radius of 0.5 meter. The procedure of selecting locally highest points is similar to the one described in Section 5.1 (Figure 5.1). Intuitively, this gives us the set of points which are very close to the outermost layer of leaves. We call such points *near surface points*. Then we calculate a 2.5 dimensional Delaunay Triangulation of these points, which results in a surface representing the outermost layer of leaves (refer to Figure 5.4 for details). To achieve a 2.5 dimensional Delaunay Triangulation, firstly all near surface points are projected onto the horizontal plane $Z = 0$. Then a 2 dimensional Delaunay Triangulation of these points is calculated. This 2 dimensional triangulation defines a connectivity between various points. Maintaining this connectivity, the points are moved back to their original position in the 3D point cloud, resulting in the creation of a mesh which represents the desired 2.5 D surface. It has been shown that amongst all possible triangulations, the Delaunay Triangulation is the one that minimizes the roughness property of the constructed 2.5 D surfaces and hence is considered desirable ([23]).
Figure 5.4: Figure showing the pipeline for creating the outermost surface of the forest. The input to the pipeline is the raw point cloud data (a), in which we select the near surface points as shown in (b). The near surface points are projected onto the horizontal plane $Z = 0$, and a Delaunay Triangulation of these points is calculated (c). Later the points are pushed back to their original position in the 3D space, maintaining the edge connectivity that was generated as a result of triangulation, giving rise to the 2.5 D surface in (d).
It should be noted that all the peak candidates found in the first step of our algorithm (Section 5.1) are also near surface points, and hence would lie on the 2.5 D surface that is constructed.

The 2.5 D surface assisted path between any two peak candidates is now defined as the shortest path between the two candidates as calculated over this 2.5 D surface. This shortest path would be the geodesic connecting the two potential peaks. If we treat the 2.5 D surface as a graph, where the near surface points are the nodes of the graph and weight of each edge is the 3D euclidean distance between the two near surface points that the edge connects, then a good approximation of the desired geodesic would be the shortest path in the graph between the peak candidates [27].

5.3.1.2 α-shape assisted path

Another approach that we explored was one that uses α-shapes to help us define the connectivity between various points. In the field of computational geometry, the α-shape provides one of the possible interpretations for the shape of a given set of points. The idea of α-shape for a set of points in a plane was first proposed by Edelsbrunner et al. [6], which was later extended to 3D space in [7].

Based on the intuitive description provided in [7], the α-shape for a set $S$ of points in $\mathbb{R}^3$ can be visualized as follows. Imagine that an infinitely large block of ice-cream fills up the entire space in $\mathbb{R}^3$ and contains the points in $S$ as small chocolate chips. Now assume that a spherical ice-cream spoon of radius $\alpha$ is being used to scoop out ice-cream without hitting any of the chocolate chips. The ice-cream spoon is capable of scooping out ice-cream even from internal of the ice-cream block, although it may not be practically feasible to do so in real life. The resultant shape that we get after having scooped out as much ice-cream as possible is the α-shape of the given set of point for the chosen $\alpha$.

In more formal terms, the definition of an α-shape is based on the underlying Delaunay Tetrahedralization of the given set of points. The Delaunay Tetrahedralization for any given point set $S$ is a simplicial complex, which can be viewed as structure formed by “gluing together points, line segments, triangles, and their n-dimensional counterparts” (Figure 5.5). For any given value of $\alpha$, the α-shape for the point set $S$ is a subcomplex of this simplicial complex, and includes all the geometrical simplexes in the Delaunay Tetrahedralization which have an empty circumscribing sphere with radius equal or smaller than $\alpha$. It should be noted that in general, the α-shape would be a disconnected structure, and for $\alpha = +\infty$, the resultant α-shape is the convex hull of the point set (Figure 5.6). It is hard to argue on what value of $\alpha$ parameter would correctly capture the essence of the shape of point cloud, and hence the interconnectivity of points. We decided to set the parameter $\alpha$ to 0.5 meter. We arrived at this value by visual inspection of the α-shape generated for different values of the α parameter. However, it should be noted that the optimum value of $\alpha$ would depend on the spatial density of the input point cloud; in general, higher the spatial density, smaller the value of $\alpha$ that could be opted for.
Figure 5.5: (a) A regular 3-simplex or tetrahedron. (b) A simplicial 3-complex. (c) An arrangement of simplexes that is not a valid simplicial complex (Image courtesy http://en.wikipedia.org).

Figure 5.6: The $\alpha$-shapes for set of points randomly generated on the surface of two linked tori. Six different $\alpha$-shapes for values of $\alpha$ decreasing from top to bottom and left to right are shown. The first shape with $\alpha = +\infty$ is the convex hull while the last shape is the point set itself for $\alpha = 0$. The $\alpha$ value used in the fourth frame neatly separates the two tori. Further decreasing disassembles the shape. Image source [7].
Similar to the case of 2.5 D surface, the \( \alpha \)-shape can also be viewed as a graph; all the points in the input point cloud are treated as the nodes of a graph, and an edge is present between two points if and only if they are the end points of the edge of any of the simplexes forming the \( \alpha \)-shape. The edge weight is equal to the euclidean distance between the end points. It should be noted that all points in the input point cloud are also present in the \( \alpha \)-shape (This was not the case with the 2.5 D surface). This implies that all potential peaks are necessarily present in the \( \alpha \)-shape. The \( \alpha \)-shape assisted path between any two peak candidates is now defined as the shortest path between those two candidates as calculated over this graph. Since the \( \alpha \)-shape may be disconnected, it implies that the corresponding graph could have more than one connected components and hence a path may not exist between certain pair of peak candidates.
5.3.2 Path properties for detecting false central peaks

While the previous section focused on defining the optimized path between two peak candidates, this section aims at exploring various properties of that path that could be used to quantify the gap between two peaks. We explore three different properties, namely Z-Variance of the path, Range of the path and Normalized Dip of the path. These are described below in detail.

- The Z-Variance of the path is calculated as the variance of the Z coordinates of all the LIDAR points that form the path. Since variance measures the spread in a given data set, a small value of Z-Variance would indicate that points forming the path are at more or less the same height. On the other hand, a large value of Z-Variance would mean a considerable spread in the Z coordinates of the points forming the path. Since peak candidates are locally highest point, a large value of Z-Variance, would suggest that points lying between two neighboring peak candidates have considerably smaller value of Z coordinates, which can be interpreted as a dip in the path connecting the two peaks.

- The Range of the path is defined as the difference in the Z coordinates of the highest and the lowest point in the path. Since the peak candidates are locally highest points, the highest point in a given path is always one of the peak candidates. If the given path is relatively flat, the range would be small. On the other hand, if the given path has a large dip, the range would be a numerically large number.

- If we define the mean-peak as the mid-point between the two potential peaks, then the difference between the Z coordinate of the mean-peak and the lowest point in the path connecting the two potential peaks can be treated as the mean-dip in the path. This mean-dip when normalized with height of the mean-peak gives the Normalized Dip of the path.

The intuition behind using these properties is that all of them tend to capture whether the given path has a considerable dip or is relatively flat (as shown in Figure 5.3). To investigate how these parameters could be used to distinguish false centers from true ones, we performed the following study. We randomly selected 128 pairs of peaks from Data Set II (refer Section 3.1). We ensured that the horizontal distance between peaks forming each pair was less than 5 meters. Thus we could say that each pair was formed by peaks which were in close vicinity of each other. With the help of our ground truth data, which was provided in the form of shape files (see Section 3.1), we found that in 73 of the pairs, both the peaks belonged to the same tree, while in the remaining 55, the peaks belonged to different trees.

We calculated various properties of the paths connecting peaks in each of the pair. The value of the properties were plotted against the height of the higher peak as well as against the height of the lower peak. These plots are shown in the next few pages. The main intuition of these plots was to seek if a meaningful relation existed between various properties described above and the height
of the potential peaks. In each of the plot, a red data point corresponds to a pair where both the peaks belonged to the same tree, while a blue data point corresponds to a pair where the peaks forming the pair belong to different trees.

Figure 5.8: Variance of path over 2.5 D surface plotted against the height of the higher peak.
Figure 5.9: Variance of path over 2.5 D surface plotted against the height of the lower peak.

Figure 5.10: Range of path over 2.5 D surface plotted against the height of the higher peak.
Figure 5.11: Range of path over 2.5 D surface plotted against the height of the lower peak.

Figure 5.12: Normalized dip of path over 2.5 D surface plotted against the height of the higher peak.
Figure 5.13: Normalized dip of path over 2.5 D surface plotted against the height of the lower peak.

Figure 5.14: Variance of path over $\alpha$-shape plotted against the height of the higher peak.
Figure 5.15: Variance of path over $\alpha$-shape plotted against the height of the lower peak.

Figure 5.16: Range of path over $\alpha$-shape plotted against the height of the higher peak.
Figure 5.17: Range of path over $\alpha$-shape plotted against the height of the lower peak.

Figure 5.18: Normalized dip of path over $\alpha$-shape plotted against the height of the higher peak.
A close inspection of the above plots reveals a few salient properties. In each of the plot, red data points are clustered towards the bottom of the plot, more specifically towards bottom-right region. On the other hand, the blue data points tend to lie towards the upper portion of the plots. Thus we could say that the red and the blue data points show a tendency to separate out from each other. This tendency is most noticeable in the plots for Range parameters, followed by the plots for Normalized Dip, and least noticeable in the plots for Z-Variance. In the next subsection, we explain how we use this observation to predict if a peak is likely to be a false central peak.

### 5.3.3 Prediction of false central peaks

We first provide the intuition behind the approach that is used for predicting whether a peak is a false central peak or not. Given a pair of peaks from our Data Set I, we would like to answer the question whether the two peaks belong to the same tree or to different trees. Amongst all the properties mentioned above, we prefer to choose a property where the blue data points and the red data points show a considerable amount of separation. One such property could be the Range in path as plotted against the height of the lower of the two peaks present in the pair. We will focus on this graph for the rest of our discussion and hence re-present it below.

If we allow both the axes of the plot presented in Figure 5.20 to assume negative values, then the graph can be treated as a 2 dimensional vector space. Consequently, each data point shown on the graph can be represented by a 2 dimensional feature vector. Now, assume that we randomly
select a pair of peaks from our Data Set I. We calculate the Range of the path connecting these
peaks and the height of the lower of the two peaks to form the 2 dimensional feature vector. If
this feature vector lies in the lower right region of the graph, then it would be likely that the pair of
peaks belong to the same tree. On the other hand, if it gets placed in the upper left portion of the
graph, then it is likely that the two peaks belong to different trees. Hence we would like to have a
demarcator that separates the lower right region containing predominantly red data points from rest
of the graph. One such boundary is shown as the green dotted line.

Since our filters are conservative (as explained in Section 1.3), we would like this demarcator
to be such that there is very small probability that a blue data point would ever lie below this line.
It should be noted that the green line drawn in Figure 5.20 is for illustration purpose only. A closer
inspection of the plot suggests that a higher order curve, similar to one shown in black dotted line,
would probably serve as a better demarcator as compared with the straight line. The exact method
for calculating this demarcator is explained below.

We first calculate the convex hull for the blue data points present in the plot. Intuitively, a blue
data point is more likely to fall inside this convex hull rather than outside. We identify the lowest and
the right most point of this convex hull, and all the points between them. These points are shown as
magenta disks in Figure 5.21 and the portion of the convex hull they span is shown in black. This
portion of the convex hull clearly acts as a valid demarcator since all blue data point lie over it and
the region underneath this demarcator contains only red data points.
However there are two concerns with such a demarcator. Firstly, it is a polygonal chain (a connected series of line segments), whereas we would like to have a curve which is much smoother. Secondly, since this demarcator touches the blue data points, it certainly cannot be called conservative. Hence we proceed as follows to generate a smoother and more conservative demarcator.

The Range corresponding to each magenta disk is halved (divided by 2) to produce a new set of points, shown as green disks in Figure 5.21. These points, along with the origin, form the support for a much conservative demarcator. A 2nd order curve is fitted to these points which results in a highly conservative demarcator, shown as the black dotted curve. A similar approach can be used for creating demarcator curves for other any other desired property.

Once a suitable demarcator curve has been generated, it is used to filter out false central peaks in the last stage of our algorithm as follows. The input to the third filtering stage is the point cloud data with potential peaks that were not filtered out during the first two filtering stages. We take a pair of potential peaks which are separated by a horizontal distance of less than 10 meters. We then calculate the path connecting the two peaks in this pair. If a path does not exist (as may be the case if the underlying structure used is the $\alpha$-shape), then the two peaks are assumed to belong to different trees. If the lower of the two peaks is also the lowest point in the path, then the two peaks are assumed to be belonging to the same tree and the lower peak in the pair is removed from the list of potential peaks. If none of the above two conditions are met, then we calculate the two dimensional feature vector corresponding to the path joining the two peaks in the pair and place it in the 2 dimensional vector space represented in Figure 5.21. If the feature vector gets placed below
the black demarcator curve, the lower peak in the pair is removed from the list of potential peaks. By repeating this process for all possible pairs in the point cloud, the third filtering stage makes an attempt to remove most of the false central peaks.
Chapter 6

Results and discussion

6.1 Evaluation methodology

Our algorithm, in the current form, attempts to identify the true-peaks of the trees, rather than delineating them. The true-peaks identified would act as cues for any subsequent tree delineation algorithm, where each peak would give rise to a separate tree. Thus our proposed method, in its current form, can still give a count of the number of trees present in the point cloud. Based on this argument, we can make a quantitative comparison of our approach with existing tree delineation algorithms by comparing the total count of trees correctly computed in a data set by various algorithms. Please note that we are not making any qualitative comparison regarding the quality of delineation.

We chose to compare our algorithm with approaches stated in Li et al. [17] and Yao et al. [28]. Within the Photogrammetry and Remote Sensing community, these methods are generally regarded as the state of art algorithms for delineating trees (Lu et al. [18]). While [17] can be considered to be a representative method for all algorithms that use the concept of locally highest point as the initial seed (including those which are based on CHM), the normalized cut based approach presented in [28] deserves special consideration since it is one of the few methods which do not rely on locally highest seed points and hence could outperform traditional methods which use locally highest points as the starting seed.

As stated in Chapter 3, we had been provided with ground truth in the form of shape files. These files contained the ground truth polygons for individual trees (Section 3.1). We implemented the algorithms proposed in Li et al. [17] and Yao et al. [28]. The input to both the algorithms was raw point cloud data and the output was delineated trees. The highest point in each delineated tree was selected as the peak of the tree. In fact, the highest point would indeed be the peak of the tree if the delineation produced by these methods was perfect. We then placed the shape file containing
Figure 6.1: Figure showing our methodology of testing various algorithms against the ground truth. (a) shows the ground truth polygons, which are present in a shape file, in black. (b) shows the peaks identified by any of the three algorithms as red dots. (c) shows the ground truth polygons placed over the point cloud, after the point cloud has been processed by one of the algorithms. Ideally, each ground truth polygon should have exactly one red peak inside it.
the ground truth polygon over the delineated point cloud, and counted the number of peaks within each ground truth polygon (Figure 6.1).

In order to develop an evaluation framework for various approaches, we first show that the true-peak detection problem can be viewed as a binary classification problem as follows. All the LIDAR points that fall inside the ground truth polygons either belong to the class of true-peaks or do not belong to the class of true-peaks. Under ideal circumstances, exactly one point inside each ground truth polygon should belong to the class of true-peaks. Based on this argument, we could define the following terms that help in evaluation of a binary classifier.

- If a ground truth polygon contained exactly one peak, then it would mean that this peak was correctly identified as belonging to the class of true-peaks and hence would be considered a true-positive.
- If a ground truth polygon contained more than one identified peaks, the highest amongst them would be the true-positive while the rest would be regarded as false-positive.
- Each ground truth polygon that contains none of the identified peaks contributes to a false-negative.

Consequently, any metric commonly used for evaluating the performance of a binary classifier, such as the F1 score, could be used to evaluate the performance of our algorithm vis-à-vis other approaches. However, any such evaluation methodology would not reveal whether the true-peak detected by our method would indeed coincide with the highest point of the tree. We could have measured precisely the accuracy of the location of a true-peak detected by our algorithm, but our industrial partner could not provide us with the relevant ground truth data. Hence, below we provide an argument in support of the fact that if there is exactly one true-peak detected by our algorithm in a given ground truth polygon of the tree, then this detected peak would indeed be the highest point within that polygon, and consequently would be the true-peak of that tree.

By the very definition, the true-peak of the tree would be the highest point of the tree, and hence should be a locally highest point in a reasonably small neighborhood. Hence, after the first filtering stage when we collect all the peak candidates that fall within the ground truth polygon of any given tree, one of them would indeed be the true-peak of the tree. Now we provide arguments to support that if after application of the subsequent filtering stages exactly one peak survives inside a ground truth polygon, then it has to be the highest point of the tree.

The second filtering stage eliminates branches by identifying the branch tips in the tree. A branch gets eliminated only when it merges into a cluster higher than itself (refer Section 5.2). If a branch tip was the highest point in the tree, its corresponding cluster would not find a suitable higher cluster to merge into and hence the branch tip would never be eliminated. Thus the highest local-peak of the tree survives after the second filtering stage, irrespective of the fact whether it was
Figure 6.2: A cluster, if identified as a branch, is allowed to merge only into a cluster higher than itself. (a) The image shows the top view of a tree with two local peaks. (b) The blue peak actually shows characteristics similar to a branch tip as is evident in this image. However since the branch tip is the highest point in the tree, it does not find a suitable cluster to merge into and consequently is not filtered out during second stage of filtering.
a branch tip or not (see Figure 6.2). The third filtering stage tests possible pair of peaks in the point cloud to detect if one of the peaks in the pair could be considered a false central peak because of the presence of other. The distance between the two peaks forming this pair could be as high as 10 meters. Since the widest trees in our data had a crown diameter of less than 10 meters, we can say that we do test all possible pairs of local-peaks belonging to the same tree. In case the property of the path connecting peaks within a pair suggests that one of the peaks is a false central peak, then it is always the lower of the two peaks which is removed from the list of possible true-peaks. Hence at the end of this filtering stage, if there is exactly one local-peak left inside a ground truth polygon, then it has to be the highest point within that tree. By extension of this logic, we could say that if there are two or more peaks identified within a ground truth polygon, one of them would be the highest point within that tree.

### 6.2 Comparative analysis

All our experiments were performed on an Intel(R) Core(TM) i7-3770CPU @3.40GHz with 16.0GB RAM (64 bit OS). Table 6.1 shows how well our algorithm performs vis-à-vis the other two algorithms on the Data Set I (refer Section 3.1). The initial probe radius for our algorithm (see Section 5.1) used for generating these results was 2.0 meters and the demarcator curve was created using the property Range as plotted against the height of the lower peaks (Figure 5.10 and Figure 5.17) for the Data Set II as explained in Section 5.3.3.

We could test the algorithm proposed in Yao et al. [28] only on the smaller LAS files. This was because of high memory cost of the algorithm: \(O(N^2)\), where \(N\) is the number of points in the LAS file. Although the method proposed by Yao et al. [28] voxelises the input point cloud data, which reduces the memory requirement, the memory required to process a small LAS file containing just 100,000 tree-points was still in excess of 20 GB. Also, as mentioned in Section 2.2, the algorithm has a large number of parameters that need to be fine tuned. All these factors make this algorithm less suitable for tree delineation on large scale point cloud data. Figure 6.3 shows some improper segmentations produced by this algorithm.

As is evident from Precision, Recall and F1 score in Table 6.1, the overall results of our algorithm after the first filtering stage (Post 1st filter) are comparable with those obtained by the method proposed by Li et al. However, the subsequent filtering stages of our algorithm (branch tip and false central peak detection) are able to identify peak candidates which are unlikely to be the true-peaks of any of the trees. Removal of these peak candidates results in a considerable reduction in the overall number of false-positives. Since the second and the third filtering stages of our method are conservative, we witness only a small increase in the number of false-negatives. The two different approaches that we used for the third filtering stage seem to be equally effective. In the end, the F1 score of our method shows a considerable improvement over the F1 score calculated for other two
Figure 6.3: Some of the noticeable failure cases with the normalized cut approach used by Yao et al. [28]. In (a) the algorithm is not able to distinguish between the two smaller trees in green, and also tips of the branches of the red tree end up as part of the green trees. In (b) the red tree branches end up being part of the smaller tree in cyan. Such cases will almost never arise in an algorithm that uses locally highest points as the seed point.
methods.

In order to test the stability of our approach we ran another set of experiments in which the demarcator curves for the third filtering stage were generated using Data Set I comprising 452 trees and the effectiveness of the algorithm was tested on the Data Set II comprising 255 trees. The probe radius used in the first filtering stage was 2.0 meters. The results for this experimental setup are shown in Table 6.2.

As is evident from Table 6.2, our method shows marginal improvement over the method proposed by Li et al. Thus we can conclude that the advantage of using our proposed method is more noticeable on complex data set where trees are closely packed and show a considerable amount of diversity in shape and size. On data sets where trees were sparsely scattered and showed more uniformity in shape and size, our method results in only a marginal improvement.

As stated in Section 2.1, traditional approaches for tree delineation such as the one proposed by Popescu et al. [20], are very much sensitive to the initial size of neighborhood selected to detect locally highest points. On the other hand, our method shows more robustness to changes in the value of the initial probe radius. Table 6.3 shows the results of our algorithm vis-à-vis other approaches when the initial probe radius is reduced to 1.0 meter.

<table>
<thead>
<tr>
<th></th>
<th>Li et al.</th>
<th>Yao et al.*</th>
<th>Ours</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Post 1st filter</td>
<td>Post 2nd filter</td>
<td>Post 3rd filter</td>
<td>Using 2.5D surface assisted path</td>
<td>Using α-shape assisted path</td>
</tr>
<tr>
<td>True-positives</td>
<td>421</td>
<td>95</td>
<td>418</td>
<td>413</td>
<td>410</td>
</tr>
<tr>
<td>False-positives</td>
<td>134</td>
<td>9</td>
<td>125</td>
<td>66</td>
<td>46</td>
</tr>
<tr>
<td>False-negatives</td>
<td>31</td>
<td>118</td>
<td>34</td>
<td>39</td>
<td>42</td>
</tr>
<tr>
<td>Precision</td>
<td>0.7586</td>
<td>0.9135</td>
<td>0.7698</td>
<td>0.8622</td>
<td>0.8991</td>
</tr>
<tr>
<td>Recall</td>
<td>0.9314</td>
<td>0.4460</td>
<td>0.9248</td>
<td>0.9137</td>
<td>0.9071</td>
</tr>
<tr>
<td>F1 score</td>
<td>0.8361</td>
<td>0.5994</td>
<td>0.8402</td>
<td>0.8872</td>
<td>0.9031</td>
</tr>
</tbody>
</table>

Table 6.1: Table comparing the effectiveness of Li et al., Yao et al. and our proposed algorithm on Data Set I. Due to resource constraint we could test Yao et al. on approximately half of the data set, consisting a total of 213 manually delineated trees. For these set of experiments, the initial probe radius used in the first filtering stage was 2.0 meters and demarcator curves for the third filtering stage were generated using the Data Set II.
<table>
<thead>
<tr>
<th></th>
<th>Li et al.</th>
<th>Yao et al.*</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Post 1\textsuperscript{st} filter</td>
<td>Post 2\textsuperscript{nd} filter</td>
<td>Post 3\textsuperscript{rd} filter</td>
</tr>
<tr>
<td></td>
<td>Using 2.5D surface assisted path</td>
<td>Using (\alpha)-shape assisted path</td>
<td></td>
</tr>
<tr>
<td>True-positives</td>
<td>246</td>
<td>246</td>
<td>245</td>
</tr>
<tr>
<td>False-positives</td>
<td>67</td>
<td>68</td>
<td>41</td>
</tr>
<tr>
<td>False-negatives</td>
<td>9</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>Precision</td>
<td>0.7859</td>
<td>0.7834</td>
<td>0.8566</td>
</tr>
<tr>
<td>Recall</td>
<td>0.9647</td>
<td>0.9647</td>
<td>0.9608</td>
</tr>
<tr>
<td>F1 score</td>
<td>0.8662</td>
<td>0.8647</td>
<td>0.9057</td>
</tr>
</tbody>
</table>

Table 6.2: Table showing results of our second set of experiments, where the demarcator curve was generated using the Data Set I and the algorithm effectiveness was tested on the Data Set II. As is evident from the F1 score, our proposed method shows a marginal improvement over the method proposed by Li et al.

<table>
<thead>
<tr>
<th></th>
<th>Li et al.</th>
<th>Yao et al.*</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Post 1\textsuperscript{st} filter</td>
<td>Post 2\textsuperscript{nd} filter</td>
<td>Post 3\textsuperscript{rd} filter</td>
</tr>
<tr>
<td></td>
<td>Using 2.5D surface assisted path</td>
<td>Using (\alpha)-shape assisted path</td>
<td></td>
</tr>
<tr>
<td>True-positives</td>
<td>421</td>
<td>95</td>
<td>447</td>
</tr>
<tr>
<td>False-positives</td>
<td>134</td>
<td>9</td>
<td>1232</td>
</tr>
<tr>
<td>False-negatives</td>
<td>31</td>
<td>118</td>
<td>5</td>
</tr>
<tr>
<td>Precision</td>
<td>0.7586</td>
<td>0.9135</td>
<td>0.2662</td>
</tr>
<tr>
<td>Recall</td>
<td>0.9314</td>
<td>0.4460</td>
<td>0.9889</td>
</tr>
<tr>
<td>F1 score</td>
<td>0.8361</td>
<td>0.5994</td>
<td>0.4195</td>
</tr>
</tbody>
</table>

Table 6.3: Table showing effect of reducing the initial probe radius for our algorithm to 1.0 meters. A smaller probe radius results in smaller number of false-negatives in every filtering stage of our algorithm. When compared with results in Table 6.1, we notice a considerable increase in overall number of false-positives, but when compared with the algorithm proposed by Li et al., our algorithm still performs better, indicating that unlike traditional methods of tree delineation, our proposed method is more robust to initial choice of parameters.
### 6.3 Running times

In this section, we present the typical running time our algorithm takes to process files of various size (Table 6.4), and discuss the asymptotic complexity. The overall running time of our algorithm for any given data set is not just dependent on the number of points in the data set, but also on various factors such as actual number of trees within the point cloud, shape of the trees, how closely the trees are spaced, number of branches detected etc.

<table>
<thead>
<tr>
<th>Number of inputs points</th>
<th>Li et al. Time required for application of 1st filter</th>
<th>Time required for application of 2nd filter</th>
<th>Time required for application of 3rd filter</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Using 2.5 D Surface For creating 2.5 D Surface</td>
<td>For detecting false centers</td>
<td>Using α-shape For creating α-shape</td>
</tr>
<tr>
<td>34,967</td>
<td>0.379</td>
<td>0.197</td>
<td>14.136</td>
</tr>
<tr>
<td>71,243</td>
<td>1.148</td>
<td>0.462</td>
<td>43.562</td>
</tr>
<tr>
<td>143,946</td>
<td>1.925</td>
<td>1.523</td>
<td>395.138</td>
</tr>
<tr>
<td>181,742</td>
<td>2.567</td>
<td>1.690</td>
<td>195.004</td>
</tr>
<tr>
<td>280,172</td>
<td>3.326</td>
<td>2.740</td>
<td>44.205</td>
</tr>
<tr>
<td>304,084</td>
<td>3.895</td>
<td>3.382</td>
<td>34.927</td>
</tr>
<tr>
<td>426,524</td>
<td>14.372</td>
<td>13.511</td>
<td>4217.796</td>
</tr>
<tr>
<td>513,475</td>
<td>5.878</td>
<td>5.027</td>
<td>678.282</td>
</tr>
<tr>
<td>526,982</td>
<td>7.358</td>
<td>5.853</td>
<td>988.540</td>
</tr>
</tbody>
</table>

Table 6.4: The table shows the time taken by our algorithm (in seconds) on files of various size and compares the running time with the method proposed by Li et al. The left most column shows the number of tree-points in our input file, the second column shows the time required by method proposed by Li et while the remaining columns show the time required for an individual stage of our proposed method.

As is evident from Table 6.4, our method is much slower than the one proposed by Li et al. It should be noted that the time reported in the second column of this table are for complete delineation of the point cloud using method proposed by Li et al. The method proposed by Yao et al. could be run only on smaller LAS files. To process the files containing 34,967 and 71,243 tree-points (first and second row in Table 6.4) there method took 40.57 and 89.99 seconds respectively and hence was slower than ours at least on the smaller LAS files.

The first filter identifies all locally highest points. To identify whether a point is locally highest or not, we need to compare its height with the height of points in a total of 9 voxels (see Section 3.2). Since the number of points within a voxel is dependent on the scanning resolution of the LIDAR...
system, the total number of points present in any group of 9 voxels is bounded by a constant. Hence the time required to test whether a point is locally highest or not would require a constant time. Thus the asymptotic complexity would be \( O(N) \), where \( N \) is the number of tree-points in the LIDAR file.

The second filter, which detects branch tips, is clearly the most time consuming stage of our algorithm and this is evident from the 4th column of Table 6.4. The algorithm has to continuously keep track of the boundary (convex hull) of each growing cluster. It uses Andrew’s monotone chain convex hull algorithm [1] for doing so, which has a running time complexity of \( O(N \log N) \). In fact, it can be shown that the asymptotic complexity of any convex hull algorithm cannot be better than that of the comparison sort, which is \( O(N \log N) \). Our method re calculates the convex hull for every cluster whenever the sweep-plane moves downwards. Since the number of clusters would be \( O(N) \) and the convex hull recalculations are called each time the sweep plane moves downwards, i.e. \( O(N) \) time, the overall complexity of this step would be \( O(N^3 \log N) \).

However, it is easy to notice that when a point joins any existing cluster, and does not form a part of the convex hull, then it implies that the point lies in the internal of the cluster and hence will never ever become a part of the convex hull of the cluster. Hence to compute the new convex hull of a cluster after a point joins it, it is sufficient to compute the convex hull of the points that previously formed the boundary of this cluster and the newly added point. Table 6.5 shows reduction in the running time of the second filtering stage after incorporating these changes. The fourth column shows the maximum size of the convex hull computed in the corresponding data set. Since the expected size of the convex hull seems to be bounded by a numerical value, the expected running time for calculating the convex hull of an individual cluster should be bounded by a constant rather than \( O(N \log N) \), and we would expect a considerable reduction in the running time after incorporating the above mentioned changes. However, as we can see, there seems to be only marginal reduction in the running time of the algorithm (Columns 2 vs. Column 3).

This can be explained as follows. The branch detection stage has primarily two components. The first component calculates the convex hull of the growing clusters, as explained above. The second component checks whether two clusters should be merged into one or not. For this, we need to calculate the smallest distance between all possible pair of points where the pair is made up of one point from each cluster. This computation has an \( O(N^2) \) complexity. Since there \( O(N^2) \) possible pair of clusters, the overall complexity of the second filtering stage is \( O(N^4) \) rather than \( O(N^3 \log N) \). Hence, although we use axis aligned bounding boxes as an initial test for collision detection, the overall time required to execute the second filtering stage is still on the higher side.

The third filtering stage, identifying false central peaks, consists of two sub steps: construction of the underlying structure that defines connectivity between points (either the 2.5 D surface or the \( \alpha \)-shape) and calculating the shortest path between pair of potential peaks. In order to construct the underlying structure, we need to construct either a Delaunay Triangulation in 2D or an \( \alpha \)-shape
Table 6.5: The table shows the time taken by the second filtering stage of our algorithm (in seconds) on files of various size. The left most column shows the number of tree-points in our input file. The second column shows the time taken by our initial implementation, while the third column shows the running time after incorporating possible optimization for convex hull computations. The fourth column shows the maximum size of the convex hull formed in the corresponding file.

<table>
<thead>
<tr>
<th>Number of points</th>
<th>Time required for application of 2nd filter</th>
<th>Time required for application of 2nd filter with improvement</th>
<th>Maximum number of points in any convex hull</th>
</tr>
</thead>
<tbody>
<tr>
<td>34,967</td>
<td>14.136</td>
<td>13.078</td>
<td>21</td>
</tr>
<tr>
<td>71,243</td>
<td>43.562</td>
<td>39.864</td>
<td>22</td>
</tr>
<tr>
<td>143,946</td>
<td>395.138</td>
<td>380.314</td>
<td>27</td>
</tr>
<tr>
<td>181,742</td>
<td>195.004</td>
<td>182.479</td>
<td>25</td>
</tr>
<tr>
<td>280,172</td>
<td>44.205</td>
<td>40.300</td>
<td>24</td>
</tr>
<tr>
<td>304,084</td>
<td>34.927</td>
<td>31.505</td>
<td>23</td>
</tr>
<tr>
<td>426,524</td>
<td>4217.796</td>
<td>4129.213</td>
<td>31</td>
</tr>
<tr>
<td>513,475</td>
<td>678.282</td>
<td>649.557</td>
<td>26</td>
</tr>
<tr>
<td>526,982</td>
<td>988.540</td>
<td>736.001</td>
<td>25</td>
</tr>
</tbody>
</table>

in 3D. The complexity of 2D Delaunay Triangulation is bounded by $O(N \log N)$ while that of 3D $\alpha$-shape is bounded by $O(N^2)$. To construct the optimized path between the pair of potential peaks, we use Dijkstra’s Algorithm, the complexity of which is also bounded by $O(N^2)$. Since the pair of peaks that can be formed within a point cloud itself is bounded by $O(N^2)$, the theoretical asymptotic complexity of this step would be $O(N^4)$ (for $\alpha$-shape based approach). However, since in a given point cloud data, the number of locally highest points as identified in the first filtering stage is a very small (usually less than 1 % of the total number of tree-points), the overall running time of this stage remains under a couple of minutes.

It should be noted that since only the near surface points are used in construction of 2.5 D surface (Section 5.3.1.1), the $\alpha$-shape is actually a much denser structure than the 2.5 D surface. Consequently, the running time for Dijkstra’s Algorithm is expected to be more for the $\alpha$-shape assisted path, which is evident from Table 6.4 (Column 6 vs. Column 8).
Chapter 7

Conclusion and future work

In this thesis we have proposed a novel approach for tree peak identification based on non local geometrical analyses. To the best of our knowledge, this is the first work that attempts to extract and utilize information about more global structure of trees beyond height values of individual points. Our proposed method shows possible improvements over existing tree delineation algorithms. However, there are a few areas that still have room for improvement and these are discussed in this chapter.

7.1 Limitations

Trees, even those belonging to the same species, could show a considerable variation in their structure and shape. This, coupled with the fact that the LIDAR data used in forestry has a low spatial density (less than 20 points/m²), makes the problem of true-peak identification challenging. Consequently, each of the filtering stage in our algorithm has certain limitations as discussed below.

As stated in Krzystek et al. [22], any tree delineation method that relies on locally highest points for detecting peaks suffers from the drawback that it would not be able to detect small trees in the lower forest layer since the true-peak of the smaller trees would almost certainly be covered by some bigger tree. Thus the true-peak of the smaller tree would go undetected. However, choosing a smaller probe radius in the first stage of our algorithm could reduce the omission of such peaks.

The second filtering stage, branch tip detection, assumes that the locally highest point of the branch would lie close to the convex hull boundary. Although this seems to be a reasonable assumption, as evident from reduction in the number of false-positives after application of 2nd filter (see Table 6.1), there are cases which can not be handled by this simple approach. Some of the failure cases are shown in Figure 7.1 and 7.2. The last filtering stage of our pipeline assumes that peaks belonging to the same tree will not have a considerable gap between them, which is not always true. Our ground truth data had enough examples which violated this assumption. Figure
7.3 and Figure 7.4 show a few such cases.

Figure 7.1: The side view of a tree showing branch structures that can easily be identified by human beings. However as seen in the top view (image on right) the highest points (in cyan) of the red and blue clusters, do not lie close to the boundary of the cluster (shown in yellow). Consequently, the locally highest points of these clusters are not identified as branch tips. The low spatial density of the point cloud prevents the smallest cluster (in green) from connecting to the main body of the tree, and thus local-peak of the cluster is not filtered out.

7.2 Future work

The most important aspect that needs to be addressed is related to the running time of the algorithm. As stated in Section 6.3, the second filtering stage of the algorithm has a worst case time complexity of $O(N^4)$. So, although the algorithm is decently fast for the smaller LAS files, and was able to process them in less than a couple of minutes, it would easily take hours to process a LAS file that contains a million points. This clearly makes the algorithm less suitable for practical usage when compared with other conventional methods. Hence we would certainly like to improve upon our naive branch detection algorithm, both in terms of accuracy of detecting branches as well as
Figure 7.2: The images on the left show two different side views of the same tree. The right most image shows the top view of the same tree, with the boundary of the two clusters drawn in yellow. Although a human would identify the red and blue clusters as branches, the algorithm fails to do so since the local-peaks of these clusters are not close to the yellow boundary as shown in the right most image.
Figure 7.3: Image showing considerable amount of gap between two local-peaks of the same tree. (a) shows an example from our Data Set I where the local-peaks from the same tree (marked in red) have considerable gap between them. Even human beings would find it hard to judge whether the two peaks shown here belong to two different trees or to one. (b) Such scenarios may arise because of human intervention as depicted in this image. The central portion of this tree was chopped off so that it does not interfere with the overhead wires, resulting in formation of two local-peaks (shown using arrows) with a considerable amount of gap between them.
Figure 7.4: It could be extremely hard to extract any kind of structural information for some of the trees in our data set. One such case is shown here. All the red peaks marked belong to the same tree, but lack of any kind of regularity in the structure of the tree makes the detection of false-peaks extremely challenging.
The existence of an $O(N \log N)$ algorithm for finding the closest pair of points, and its use in a variety of problems in the field of computational geometry [24] suggests that we might be able to reduce the complexity of the second filtering stage to at least $O(N^3 \log N)$. It would also be interesting to see if we could use the concepts of Principal Component Analysis (PCA) or Hough transform to detect linear clusters that resemble branches in the point cloud (see Figure 7.5). The use of PCA and Hough transform to detect structures resembling branches was explored by Ko et al. [13]. However, the input LIDAR data they worked with was of better quality (approximately 40 points/m$^2$), and their PCA implementation needed manual supervision.

The manner in which path connecting two potential peaks is created could have a considerable effect on the effectiveness of the third filtering stage of our algorithm (detecting false central peaks). To create the underlying structures that defines the connectivity between points, an optimum set of parameters needs to be selected. However, we would like to reduce the dependency of our algorithm on various parameters, and hence it would be interesting to see if we could come up with a definition of an shortest path which is independent of any parameter.

Lastly, in our current implementation, the only criterion for deciding whether a peak is a false central peak or not is a two dimensional feature vector comprising the Range of the path connecting two potential peaks and the height of the lower of the two peaks. We could think of incorporating other properties of the pair, such as the distance between the two peaks, height of the higher of the two peaks etc. to construct a higher dimensional feature vector. The use of a higher dimensional feature vector may be able identify false central peaks much more accurately.
Figure 7.5: As suggested in [13], Principal Component Analysis (PCA) or Hough transform could be used for identifying linear patterns in the point cloud which resemble branches. (a) shows linear branch-type patterns detected by PCA. (b) shows a possible use of Hough transformation for detecting such linear patterns in the cloud.
Bibliography


