# LIDAR DATA CLASSIFICATION AND COMPRESSION

A Dissertation

presented to

the Faculty of the Graduate School

at the University of Missouri-Columbia

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy

by

XIAOLING LI

Dr. Wenjun Zeng, Dissertation Supervisor

DECEMBER 2013

The undersigned, appointed by the dean of the Graduate School, have examined the dissertation entitled

LIDAR DATA CLASSIFICATION AND COMPRESSION

presented by Xiaoling Li,

a candidate for the degree of doctor of philosophy of Computer Science,

and hereby certify that, in their opinion, it is worthy of acceptance.

Professor WenJun Zeng

Professor Yunxin Zhao

Professor Jianlin Cheng

Professor Tony Xu Han

# ACKNOWLEDGEMENTS

Foremost, I would like to express my sincere gratitude to my advisor Dr. Wenjun Zeng for his continuous support of my PhD study and research, for his guidance, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis.

I would like to thank Dr. Jianlin Cheng, Dr. Tony Han, and Dr. Henry He for their wonderful lectures and insights that enable me to utilize concepts learned from class into challenge research problems. I would like to thank Dr. Yunxin Zhao for her support, encouragement and insightful comments for my thesis work.

Last but not least, I would like to thank my parents. I would not be able to complete this work without their unconditional love and encouragement.

TABLE OF CONTENTS

LIST OF TABLES

Table Page

LIST OF ILLUSTRATIONS

Figure Page

LIDAR DATA CLASSIFICATION AND COMPRESSION

Xiaoling Li

Dr. Wenjun Zeng, Dissertation Supervisor

**Abstract**

Airborne Laser Detection and Ranging (LIDAR) data has a wide range of applications in agriculture, archaeology, biology, geology, meteorology, military and transportation, etc. LIDAR data consumes hundreds of gigabytes in a typical day of acquisition, and the amount of data collected will continue to grow as sensors improve in resolution and functionality. LIDAR data classification and compression are therefore very important for managing, visualizing, analyzing and using this huge amount of data. Among the existing LIDAR data classification schemes, supervised learning has been used and can obtain up to 96% of accuracy. However some of the features used are not readily available, and the training data is also not always available in practice. In existing LIDAR data compression schemes, the compressed size can be 5%-23% of the original size, but still could be in the order of gigabyte, which is impractical for many applications.

The objectives of this dissertation are (1) to develop LIDAR classification schemes that can classify airborne LIDAR data more accurately without some features or training data that existing work requires; (2) to explore lossy compression schemes that can compress LIDAR data at a much higher compression rate than is currently available.

We first investigate two independent ways to classify LIDAR data depending on the availability of training data: when training data is available, we use supervised machine learning techniques such as support vector machine (SVM); when training data is not

readily available, we develop an unsupervised classification method that can classify LIDAR data as good as supervised classification methods. Experimental results show that the accuracy of our classification results are over 99%.

We then present two new lossy LIDAR data compression methods and compare their performance. The first one is a wavelet based compression scheme while the second one is geometry based. Our new geometry based compression is a geometry and statistics driven LIDAR point-cloud compression method which combines both application knowledge and scene content to enable fast transmission from the sensor platform while preserving the geometric properties of objects within a scene. The new algorithm is based on the idea of compression by classification. It utilizes the unique height function simplicity as well as the local spatial coherence and linearity of the aerial LIDAR data and can automatically compress the data to the desired level-of-details defined by the user. Either of the two developed classification methods can be used to automatically detect regions that are not locally linear such as vegetations or trees. In those regions, the local statistics descriptions, such as mean, variance, expectation, etc., are stored to efficiently represent the region and restore the geometry in the decompression phase. The new geometry-based compression schemes for building and ground data can compress efficiently and significantly reduce the file size, while retaining a good fit for the scalable "zoom in" requirements. Experimental results show that compared with existing LIDAR lossy compression work, our proposed approach achieves two orders of magnitude lower bit rate with the same quality, making it feasible for applications that were not practical before. The ability to store information into a database and query them efficiently becomes possible with the proposed highly efficient compression scheme.

## Chapter 1

# Introduction

In this chapter, we first briefly introduce LIDAR data, its collection system and applications. We then describe research motivation on LIDRA data classification and compression, followed by an outline of the dissertation.

## 1.1 LIDAR Data

LIDAR (light detection and ranging, also written Lidar or LiDAR) [5] is a remote sensing technology that measures distance by illuminating a target with a laser and analyzing the reflected light. It uses ultraviolet, visible, or near infrared light to densely sample the surface of the earth. LIDAR is an active optical sensor that transmits narrow laser beams toward a target while moving through specific survey routes. The reflection of the laser from the target is detected and analyzed by receivers in the LIDAR sensor. These receivers record the precise time from when the laser pulse left the system to when it is returned to calculate the range distance between the sensor and the target. The main components of LIDAR system include a collection vehicle such as aircraft, helicopter, vehicle, and tripod, a laser scanner system, GPS (Global Positioning System), and INS (inertial navigation system). An INS system measures roll, pitch, and heading of the LIDAR system [4]. Figure 1 is an illustration of a LIDAR data collection system.

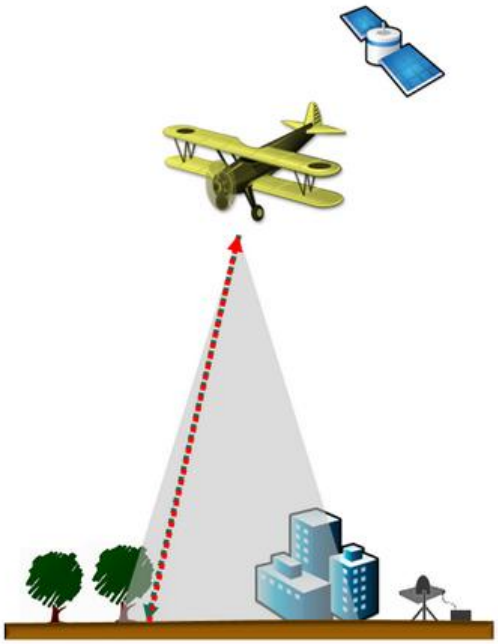


Figure 1. Illustration of a LIDAR data collection system

Low flying aircraft equipped with modern laser-range scanning technology collect precise elevation information for entire cities, counties, and even states. The scanner records the waveform of the returning reflection for each laser pulse that it sends out. The intensity peaks of this waveform correspond to points that were hit by the laser and that reflected significant portions back to the sensors on the plane. There can be multiple peaks because the laser may hit several surfaces such as wires or antennas, branches, leaves, or even birds in flight before reaching the ground. Each peak above a certain threshold is called a return. The coordinates of these returns together with intensity is the usual data of interest. A narrow laser beam can be used to map physical features with very high resolution. The point data is post-processed after the LIDAR data collection survey into highly accurate geo-referenced x,y,z coordinates by analyzing the laser time range, laser scan angle, GPS position, and INS information [4]. Post-processed spatially

organized LIDAR data is known as point cloud data. The point clouds are large collections of 3D elevation points, which include x, y, and z, along with additional attributes such as GPS time stamps. The specific surface features that the laser encounters are classified after the initial LIDAR point cloud is post-processed. Elevations for the ground, buildings, forest canopy, highway overpasses, and anything else that the laser beam encounters during the survey constitutes point cloud data.

The post-processes after the LIDAR data collection generate x,y,z measurements which form mass point cloud datasets and can be managed, visualized, analyzed and used with a wide range of applications, such as agriculture, archaeology, autonomous vehicles, biology and conservation, geology and soil science, meteorology and atmospheric environment, military and transportation, mining, physics and astronomy, etc [5]. Derivatives of LIDAR data such as digital elevation maps are subsequently used in applications such as remote sensing, assessing flood hazards, planning solar and wind installations, carrying out forest inventories, urban safety, surveillance and emergency management etc [45-50]. LIDAR can be used to help farmers determine which areas of their fields to apply costly fertilizer. It can create a topographical map of the fields and reveals the slopes and sun exposure of the farm land and categorized the farm land into high-, medium-, or low-yield zones; In the field of archaeology, it can aid in the planning of field campaigns, mapping features beneath forest canopy, and providing an overview of broad, continuous features that may be indistinguishable on the ground. LIDAR can also provide archaeologists with the ability to create high-resolution digital elevation models (DEMs) of archaeological sites that can reveal micro-topography that are otherwise hidden by vegetation; The ability of LIDAR technology to provide three-

dimensional elevation maps of the terrain, high precision distance to the ground, and approach velocity can enable safe landing of robotic and manned vehicles with a high degree of precision, enable robotics to obtain for the perception of the environment as well as object classification; Autonomous vehicles use LIDAR for obstacle detection and avoidance to navigate safely through environments; LIDAR has also found many applications in forestry. Canopy heights, biomass measurements, and leaf area can all be studied using airborne LIDAR systems; LIDRA is also used by many industries, including Energy and Railroad, the Department of Transportation, as a faster way of surveying. Topographic maps can also be generated readily from LIDAR; LIDRA has ability to detect subtle topographic features such as river terraces and river channel banks, to measure the land-surface elevation beneath the vegetation canopy, to better resolve spatial derivatives of elevation, and to detect elevation changes between repeat surveys. LIDAR have enabled many novel studies of the physical and chemical processes that shape landscapes, etc. Among these wide applications of LIDAR data, LIDAR classification plays important role to understand , analyze, visualize, manage and use of LIDAR data.

## 1.2 Research Motivation

LIDAR is now a mainstream in GIS technology [17-21]. Fresh off the scanner, LIDAR data is typically stored in a binary, vendor-specific format. But in order to exchange the data between users and across different software packages, it traditionally was converted into a simple ASCII representation where each line listed the floating points represented as attributes of a single return. A typical day of acquisition can result in a data set that is

hundreds of gigabytes in size; and the amount of data collected will continue to grow as sensors improve in resolution and functionality. Such large files mean storage and archive hassles as well as hampered workflows [51]. The whole file compression scheme, such as gzip, requires the entire file be decompressed before accessing any of the data. When dealing with gigabyte-sized files, such an approach might result in excessive I/O even when only a small spatial region needs to be decompressed. A preferred alternative would be to compress the file in such a way that, for a given spatial region, only the necessary subset of the file containing data for that region would need to be accessed from the disk and decompressed.

Although today e-Commerce is largely restricted to the use of text and 2D images, it is increasingly recognized that the true potential of e-commerce, for example, in its business to customer mode, can only be realized when the customer is able to access the product/service of interest by picking up, walking through and interacting with it as if in the traditional market place. This implies use and access of 3D data and model via the Internet [60, 61, 62, 81]. Medicine, genomics, bio-informatics, scientific collaborations are other domains where 3D data is in extensive use and has the need to be exchanged over the Internet [63]. Foremost of these issues are the large storage and long transmission time for the complex 3D data. An effective solution is to use 3D compression. Compressing 3D data is a challenge that has been studied for the last decade. For example, to compress 3D medical data, people have been enlightened by the success of 2D image data compression such as JPEG2000 and applied 3D wavelet compression along x axis, y axis and z axis [52, 53]. LIDAR compression has been studied in recent years. Despite the significant progress made in LIDAR data

compression, the compressed size still could be in the order of gigabyte, which is impractical for many applications such as storing information into database and querying, or in an e-competence site that customer can pick up a product/service and walk through, etc. We need to research on compressing the LIDAR data significantly and efficiently while maintaining good quality. This thesis proposes a novel LIDAR compression method that is based on geometry characteristics of different LIDAR data; it can compress LIDAR data significantly by classifying LIDAR data into more homogenous group first. One of the major goals of this thesis is to research on LIDAR classification.

As we discussed earlier, LIDAR technology has a wide range of applications in different fields such as remote sensing, urban safety, surveillance and emergency management, agriculture, archaeology, autonomous vehicles, robots, military and transportation, etc, LIDAR classification plays an important role in these applications, for example, to achieve better scene understanding, to enable robots to achieve the perception of the environment and object classifications. 3D point clouds of natural environments relevant to problems in geomorphology such as rivers, coastal environments and cliffs often require classification of the data into elementary relevant classes. A typical example is the separation of riparian vegetation from ground in fluvial environments, the distinction between fresh surfaces and rockfall in cliff environments, or more generally the classification of surfaces. Besides these applications, LIDAR classification is very important for LIDAR data compression as well, as shown in a later chapter in this thesis. It is one of the key steps to compress LIDAR data efficiently. This thesis researches on high accuracy LIDAR classification first - both on supervised learning and unsupervised classification, and then researches on effective compression methods.

## 1.3 Outline of the Dissertation

This thesis is organized into six chapters.

Chapter one is the introduction, which describes LIDAR data, the motivation and scope of the proposed research.

Chapter two provides an overview of data classification and compression.

Chapter three describes our work on LIDAR data classification by supervised learning [89]. Existing methods have problem in separating building boundary from trees. We use a new 3D feature to effectively distinguish building boundary from trees, and use a hybrid approach to preserve tree boundary and maintain data spatial coherence. Experimental results show that we can obtain over 99% of accuracy.

Chapter four describes our work on LIDAR data classification by unsupervised learning. Supervised learning schemes can classify LIDAR data well when the real data and training data are very similar. Its performance however will be degraded when real data is different from training data, for example, if one uses tree data that are full of leaves for training and classifies the winter trees that do not have leaves. This dissertation proposes a general approach that does not require training data. It uses unsupervised classification, based on super pixel likelihood computation to better utilize spatial coherence of LIDAR data and applies graph cut multi-label optimization to refine the final results. This work, to our best knowledge, is the first unsupervised learning method that can classify LIDAR data as accurately as or even better than supervised learning methods.

Chapter five addresses LIDAR data compression, and proposes two methods: image based compression and geometry based compression; It analyzes existing lossy compression methods, discusses that in low bit rate, the performance of our proposed image based compression is equivalent or better than existing lossy compression methods. Experimental results show that our geometry based compression [90] performs much better than our image based compression. Thus in low bit rates, our geometry based compression performs much better than existing methods.

Chapter six concludes the thesis and discusses future work.

## Chapter 2

# Background and Overview

In this chapter we describe background knowledge, including LIDAR data supervised and unsupervised classification, data compression in general, overview of 2D and 3D data compression, 3D point clouds compression and LIDAR data compression.

## 2.1 LIDAR Data Classification

LIDAR classification is important to better use, analyze, visualize and manage LIDAR data. LIDAR classification in general is a standalone research area for many years, it is a challenge task that existing classification mainly use supervised method to obtain higher accuracy to classify LIDAR data.

Among research work for Airborne LIDAR data classification, accurately and efficiently classifying airborne LIDAR tree points from building and ground points are important research tasks in urban planning, remote sensing, compression, etc. It has drawn interest from many researchers. One of the applications for LIDAR classification is for LIDAR compression. Among the existing supervised airborne LIDAR classification, researcher can achieve up to 96% accuracy. Despite their excellent work, they still has problem in distinguish building boundaries and tree points. Also supervised classification has limitation of not very generalized. For example, if we obtain training data from tree with leaves, the accuracy will be reduced if the actual tree data are from winter trees without leaves. This work also proposed an unsupervised method that to the author's best

knowledge, first time be able to classify LIDAR data as accurately as supervised method or even better. Chapter three and four present the two new methods in detail.

## 2.2 Data Compression

The main idea of data compression is to reduce the data correlation and replace with a simpler data form. The data compression system is called a source encoder where the input of the system is source, and output is a compressed bit stream. A source is a sequence of symbols to be compressed. The symbol can be considered as a random variable X, which takes its value from an alphabet X. The output of the source encoder is source code C; it is a mapping from X to a binary string, which is called a codeword. The data compression system can be described by Figure 2.

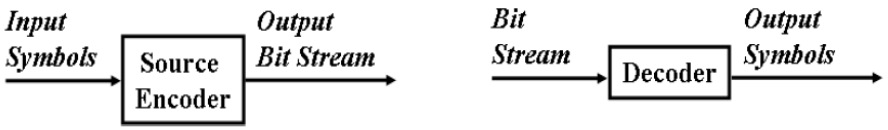



Figure 2 Data compression system

The source decoder takes the received bit stream as input and reconstructs the symbol sequence.

Let p(x) be the distribution of the discrete random variable X. The average length of the source code C, denoted by L(C), is given by L(C), which is the average number of bits per symbol.

$$L(C) = \sum_{x \in X} p(x) l(x)$$

Bit rate is defined as average number of bits per symbol. Bit rate and distortion are two important measurements in a data compression system. With the same distortion, a smaller bit rate indicates better compression efficiency. Distortion can be measured by mean square error (MSE) and peak Signal-to-Noise Ratio (PSNR). MSE is defined as

$$MSE = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - K(i,j)]^2.$$

Here I(i,j) is source data point and K (i,j) is the reconstructed data point.

PNSR is defined as

$$PSNR = 10 * log_{10}(\frac{MAX_I^2}{MSE}).$$

Here $MAX_I$ is the maximum possible value of the signal.

There are lossless compression and lossy compression. Lossless compression does not incur distortion after signal is reconstructed while lossy compression has distortion after signal is reconstructed. Rate-Distortion [38] (R-D curve) will be used to evaluate the distortion. Usually the average bit rate is x axis and distortion is y axis. A bigger bit rate will incur smaller distortion.

In data compression, there is often a significant amount of correlation or redundancy existing in the source data. A source symbol and its temporal or spatial neighborhood are often highly correlated. For example, in a 2D image, neighbor pixels for the same object will have similar values. One of the central tasks in efficient data compression design is to explore the source correlation and remove the redundant data so as to improve the rate-distortion performance of the compression system. The most commonly used approach to exploit source correlation is transform coding [39]. In particular, a Fourier-related

transform such as the Discrete Cosine Transform (DCT)[10] is widely used. The more recently developed wavelet transform [31-33] is also used extensively.

Entropy of a discrete random variable X, is denoted by H(X) [35],

$$H(x) = -\sum_{x \in X} p(x) log_2 p(x).$$

Entropy is a lower bound on the average number of bits needed to represent the symbols (the data compression limit). Entropy coding methods aspire to achieve the entropy for a given alphabet wherein a code achieving the entropy limit is optimal. Huffman coding [36] and arithmetic coding [37] are two examples of entropy coding.

Experience in the 2D raster world has been successful in effectively compressing large files. Compression algorithms have been widely used for well over decade in the 2D space. The most well know method is JPEG [11] and JPEG 2000 [12]. The key steps of JPEG and JPEG 2000 are illustrated in Figure 3.

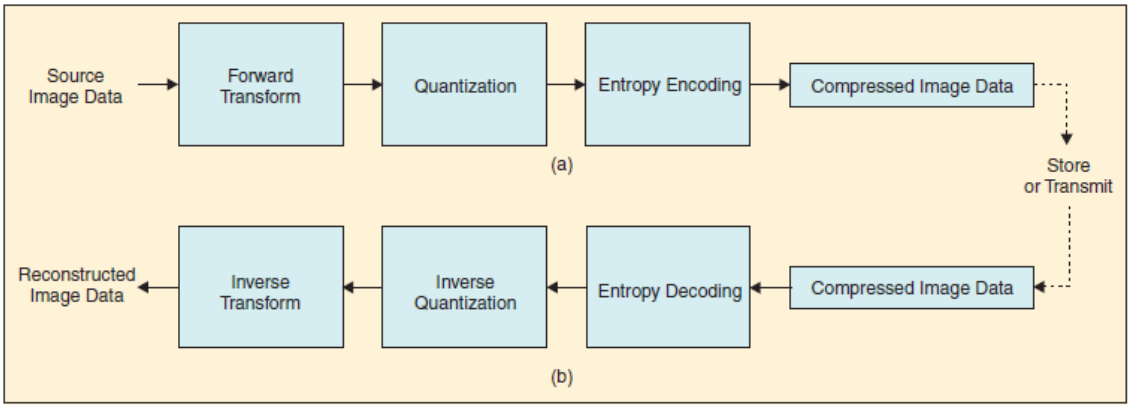



Figure 3. General block diagram for JPEG/JPEG2000 encoder and decoder

JPEG use Huffman encoder/decoder as entropy encoder and decoder, JPEG 2000 use arithmetic encoder and decoder as its entropy encoder and decoder.

JPEG uses a discretized version of the fourier cosine transform (DCT) as transformation coding. JPEG2000 use discrete wavelet transforms (DWT) [28-33]. Debauchies (5, 3) [34] filter bank include the following:

DWT analysis high pass filter : $H_0(z) = 1/8(-1+2z^{-1}+6z^{-2}+2z^{-3}-z^{-4})$

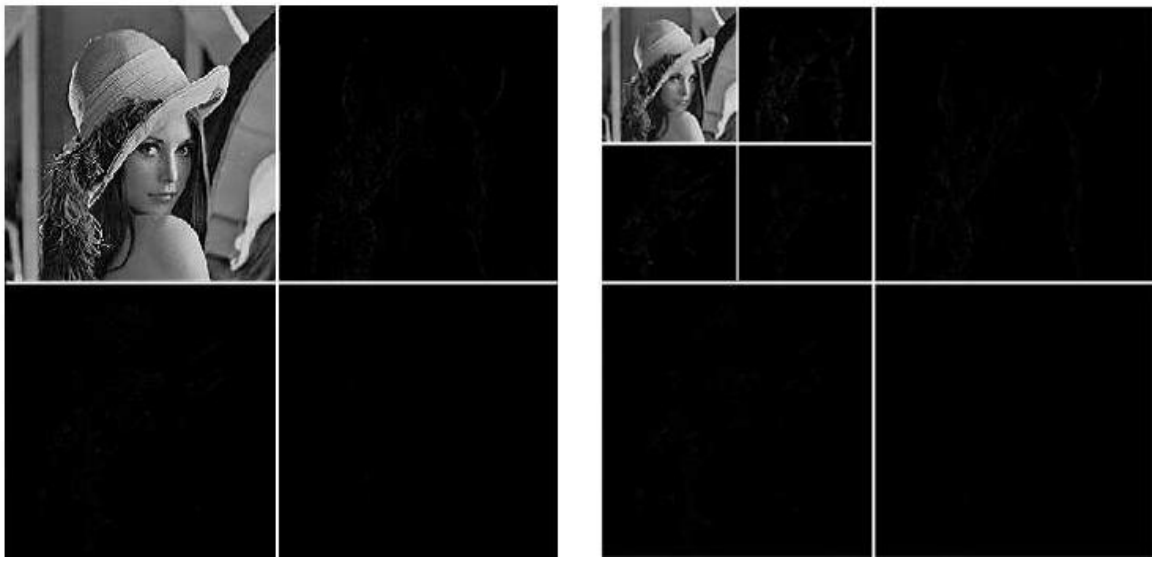
DWT analysis low pass filter $H_1(z) = 1/2(-1+2z^{-1}+z^{-2})$

DWT synthesis low pass filter: $G_0(z) = 1/2(1+2z^{-1}+z^{-2})$

DWT synthesis high pass filter: $G_1(z) = = 1/8(-1-2z^{-1}+6z^{-2}-2z^{-3}-z^{-4})$

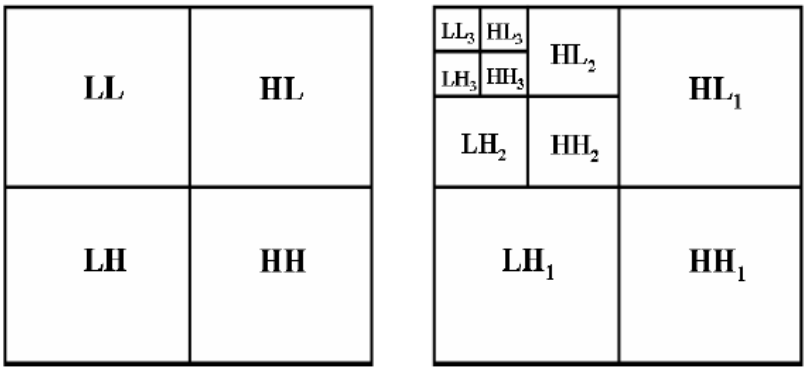
The 1-D DWT can be extended to a 2-D DWT by horizontal DWT of all rows followed by vertical DWT of all columns. Suppose we have an input image of size N·M, we must first apply horizontal DWT to each row of pixels of the original image of size. Due to down-sampling operation, the low-frequency and high-frequency subbands of the output have N/2coefficients. By putting the low-frequency subband coefficients in the left half and the high-frequency subband coefficient in the right, we will generate an image as shown in Figure 4(b). The next step is to apply vertical DWT by applying the 1-D DWT to each column of the image as shown in Figure 4(b). We put the low-frequency subband coefficients of this vertical transform in the top half and the high-frequency subband coefficient in the bottom half. This will produce an image as shown in Figure 4(c). These two steps forms one-layer of subband decomposition, i.e., 2-D DWT. One layer DWT will generate four subbands designated as LL, LH, HL, and HH as illustrated in Figure 5(a).

(a) (b)



(c) (d)

Figure 4. Subband decomposition of Lena image




(a) (b)

Figure 5. One layer (a) and three layer (b) subband decomposition

The LL subband image on the top-left corner is similar to the original image. This is because the low-frequency subband is the output of a low-pass analysis filter. We can apply the one-layer subband decomposition to this LL subband, and this procedure can be repeated until the LL is sufficiently small. Figure 5 (b) shows the output of three-layer subband decomposition. One nice feature of DWT is there exists parent-children dependency across the subband as illustrated in Figure 6.

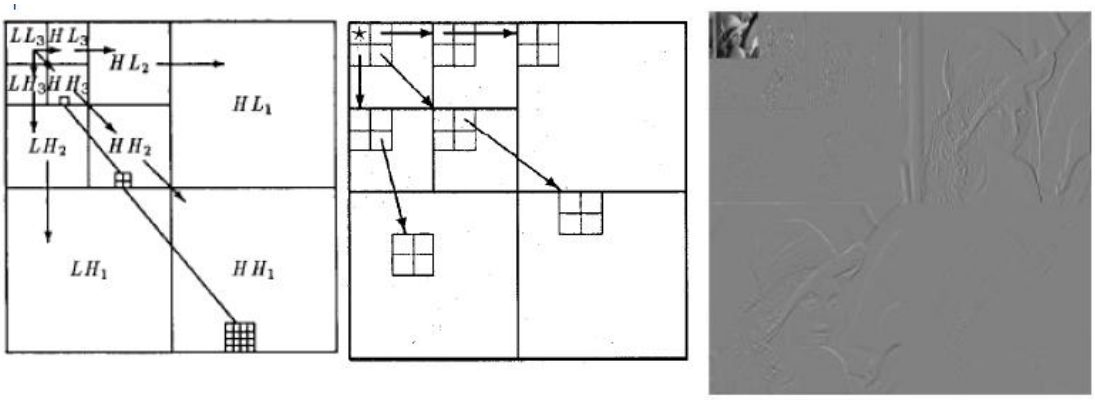



Figure 6. Parent-children dependency across subband

The embedded zero tree wavelet (EZW) [16] is based on the concept that discrete wavelet transform has an hierarchical subband decomposition, and parent-children has

dependency across sub-band, so it allows the successful prediction of insignificant coefficients across scales to be efficiently represented as part of exponentially growing trees by exploiting the self-similarity inherent in images. It uses zero-tree coding which provides a compact multi-resolution representation of the significance map, which consists of binary maps indicating the positions of the significant coefficients. EZW has the property enabling the bits in the bit stream to be generated in order of importance yielding a fully embedded code. Using an embedded algorithm, an encoder can terminate the encoding at any point by allowing a target rate or target distortion metric to be met accurately and precisely. Given a bit stream, the decoder can cease decoding at any point in the bit stream and still produce exactly the same image that would have been encoded at the bit rate corresponding to the truncated bit stream.

JPEG2000 is the new international standard for image compression, and it is much more efficient than the old JPEG international standard. For the same compression ratio / bit rate / file size, the JPEG2000 picture has much better quality. It represents advances in image compression technology where the image coding system is optimized not only for efficiency but also for scalability and interoperability in network and mobile environments. JPEG2000 standard offer a performance superior to current standards at low bit rates (e.g., below 0.25 b/p for highly detailed gray-scale images). This significantly improves low bit-rate performance which can now be achieved without sacrificing performance on the rest of the rate-distortion spectrum. Network image transmission and remote sensing are some of the applications that need this feature. One of our proposed compression methods implemented JPEG 2000's main steps.

To summarize, encoding included the following steps:

1. Transformation to remove redundancy
2. Quantization[40] (a mapping from a real number into a set of discrete points; this process is irreversible, and can thus incur distortion)
3. Entropy encoding

Decoding included the following steps:

1. Entropy decoder
2. De-quantization
3. Inverse transformation

## 2.3 3D Data Compression

3D compression is a new branch of data compression aimed at the 3D points, models and other geometric datasets used in computer graphics, virtual reality, geographic information system, video games, CAD/CAM, and many scientific, engineering, and medical applications. We use data compression in our everyday life to get data faster: ZIP for text, MP3 for music, JPEG or GIG for images and MPEG4 [67], QuickTime and DVD for movies, etc. The field of 3D compression is attempting to achieve similar gains for 3D models, animation, and other much more complex types of data [64]. Researchers around the world are addressing these problems [69-79, 82, 83]. Existing 3D compression algorithms can be categorized into two major parts:

(1) Apply techniques from 1D and 2D compression. For example, Yodchanan [80] use reversible KLT to remove redundancy to lossless compress 3D MRI data.

Zhang [82] use computer graphics pipeline to sample 3D range data into regular 2D images, encode three-dimensional range data into regular two-dimensional images, the 2D images can further be compressed with existing 2D image compression techniques.

(2) Use completely different approaches that take advantage of the properties of 3D surfaces. For example, Jarek Rossignac's Edgebreaker [69] is a finite state machine that walks from one triangle onto adjacent one, it encodes the complete connectivity of triangle mesh in a sequences of symbols. The methods for new 3D compression techniques can be further grouped into the following categories [64]:

- Mesh-based methods that traverse a polygon mesh representing an object's surface, represented by Edgebreaker [69], Java3D compression [65], Topological Surgery [70], etc.
- Progressive and hierarchical methods that transmit a base mesh and a series of refinements, for example Compressed Progressive Meshes [71], subdivision-based approaches [72], Compressed Normal Meshes [73].
- Image-based approaches that encode not an object but a set of pictures, for example QuicktimeVR [91] and IPIX [87].

In our research on airborne LIDAR compression via image based compression, we apply similar method as category (1), which convert 3D information into 2D image and use existing well established 2D compression algorithms. For methods in category (2), there is a rich body of literature in geometry compression with most techniques developed to date focusing on the compression of polygonal meshes where the connectivity information is required.

Geometry compression is one of the core problems in geometry processing [54]: its goal is to represent 3D models compactly for efficient storage, fast transmission, and reliable reconstruction. With increasing sizes and complexities of 3D models, there is an emerging demand for efficient compression techniques especially under networked environments. Since most techniques developed to date focusing on the compression of polygonal meshes where the connectivity information is required [55, 56, 74-79], such techniques are not applicable to the compression of point sets. Among newly-started research on point-set compression, local coordinate frames and least squares approximation are generally used to define approximating. However most of these existing methods are designed for clean point data and cannot deal with outliers inherently contained in the Light Detection and Ranging (LIDAR) data.

## 2.4 LIDAR Data Compression

LIDAR data compression is an active research area in 3D data compression. In a lossless compression, the decompressed point cloud must contain exactly the same set of points that were fed into the compressor. Under this constraint, some compression will be possible, but not much.

Geospatial imagery often needs to be able to store data at multiple resolution levels or scales so that it can more quickly access data at coarser granularities. This feature is often needed when users are quickly "scrolling through" a large data set at an "overview" level and then, when a feature of interest is found, they rely on geospatial imagery's level resolution capability to "zoom in" to that region and see more details. Lossy compression is usually allowed in these situations.

In a lossy compression that does not require full precision and accuracy, higher compression gains will be possible. For example, we might choose to reduce the precision of points or remove some points while retaining our ability to preserve the global surface or elevation characteristics and at the same time minimize less significant local changes. If points are to be moved, we need to provide the information about the statistical displacement of the points.

Despite the significant progress made in LIDAR data compression, the compressed size still could be in the order of gigabyte, which is impractical for many applications such as storing information into database and querying, or for e-competence site a customer can pick up a product/service and walk through from internet, etc. Also, there are increasing demands of applications that require semantic information as well as approximation of the objects for quick information retrieval; existing methods do not fit the new requirements well.

This dissertation develops LIDAR classification method that can accurately classify airborne LIDAR data into three classes: buildings, ground and trees. The resulting subsets of the classified LIDAR data can be used for applications that only interest certain class, for example, man-made structure investigation that is only interested in LIDAR data for buildings, it is lossless presentation of data. In our lossy compression scheme, we apply quantization of the data first. Since LIDAR data is not set on any inherent grid, we quantize them into a fixed grid on x and y direction. We develop two compression methods and compare their performance. Our geometry-based compression on building data provides an efficient lossy compression scheme that fits well for scalable

“zoom in” requirements. This method can achieve a much higher compression rate in a lossy compression scheme where error rate is not critical; however, file size is critical.

In the next few chapters, we describe details of our proposed LIDAR data classification and compression schemes. We start with LIDAR classification via supervised learning in the next chapter.

## Chapter 3

# LIDAR Classification by Supervised Learning

This chapter describes LIDAR data classification by supervised learning method; it first describes related work on LIDAR classification, then describes our proposed work for LIDAR classification via supervised method, and followed by experimental results.

### 3.1 Related Work

LIDAR data classification is an active and challenge research area that is very important for various applications including 3D urban scene modeling and understanding. Most of existing LIDAR data classification use supervised learning method, can obtain accuracy from 80% to 96%. Their methods could not be applied directly for our work due to lack of some of the main features in existing LIDAR classifications in our dataset. For example, some existing work[85,86] uses grey-scale aerial image as one of their features to classify LIDAR data, usually grey-scale aerial image can be obtained from separate data collection, it need to be registered with LIDAR data in order to be fused with related LIDAR data . We do not have grey-scale aerial image associated with the data, we tried the rest of the features proposed in their work, the experimental results is poor without the additional image data source, which indicates their feature set may not be optimal. Thus we need an effective classification method that use or can extract features from the limited, simple form of data we currently have, yet can classify with higher accuracy. Among existing works that employ supervised machine learning techniques, Lodha et al. [84, 85, 86] applied machine learning techniques such as AdaBoost, expectation

maximization and SVM to conduct airborne LIDAR classification that separates the data into four main categories. Carlberg et al. [6] performed segment wise classification of airborne LiDAR data based on PCA based saliency features. Chen et al. [7] performed 2D tree classification in airborne LIDAR data by combining over-segmentation and under-segmentation followed by classification. Despite their significant success, existing methods still have difficulty in separating the building boundary points from tree, for example, [7] defined future work as "involves finding better features to discriminate between building edges and trees". Supervised learning algorithms however would generally require a large set of training data which is not always readily available. The accuracy of the classification result on unknown dataset is also dependent on how close the unknown dataset to the original training data. Our preliminary work utilizes a new 3D shape feature [25] that works well to distinguish building boundary from trees.

### 3.2. Overview of the Proposed Scheme

In our system, the classification is performed on 3D points but the training data is based on the 2D depth image; this implementation simplifies the training process greatly but also results in difficulty in providing training data that includes points on the boundary of trees. This is due to the fact that several 3D points could be projected onto one 2D pixel, thus it is possible to have both building and nearby ground points projected onto the same 2D pixel, thus ground points can be used as training data for non-tree points. Similarly, tree boundary data can also include nearby ground point. Hence if we use building boundary (which may include ground points) as training data to represent non-tree points, we cannot use tree boundary (which may also include ground points) as training data to

represent the opposite class - tree. Figure 7 shows that building boundary and tree boundary are not mutually exclusive

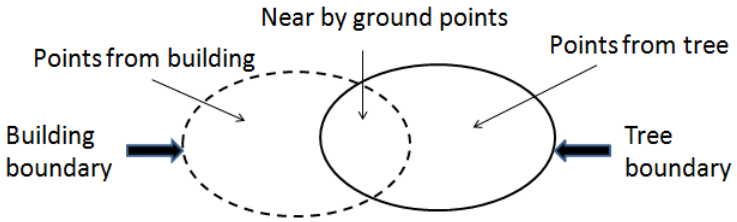



Figure 7. Building/tree boundaries are not mutually exclusive

In order to overcome the limitation on training data dilemma, we label building boundary to train building boundary and use unsupervised cluster method to re-label tree boundary based on their cluster information, the hybrid approach guarantees high accuracy on LIDAR data classification on both typical and untypical boundary data. Furthermore, this dissertation presents a novel 3D robust statistics-based shape feature that successfully overcame the limitations of existing methods in separating building boundary points from tree points.

### 3.3. Details of the Algorithm

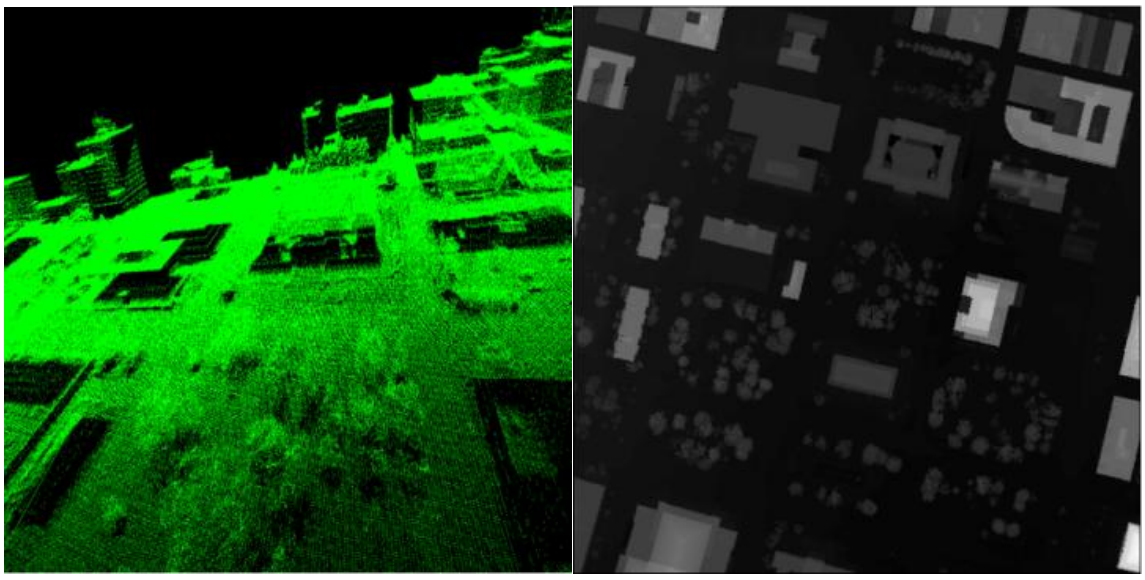
We research effective features to separate building, ground and trees. Normalized height is one of the features, since terrain elevation need to be adjusted, we use height based cluster to find the ground and normalize height. Due to the lack of building boundary and tree boundary co-exist in training samples, we create a hybrid approach to overcome this limits and obtain high accuracy to classify building boundary and trees.

There are three main steps in the algorithm:

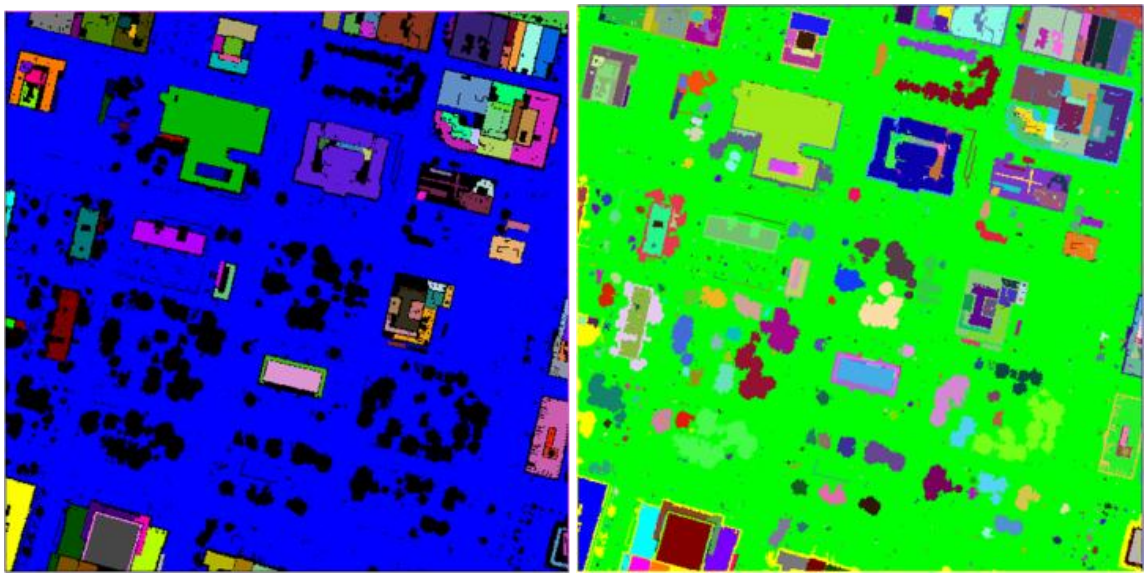
1. Clustering by spatial and range affinity
2. Classifying by supervised machine learning
3. Refining classification based on unsupervised clustering

### 3.3.1 Clustering

We first convert the 3D LIDAR points into a depth image by projecting the 3D points into a uniform grid on the (x, y) plane. The intensity of the pixel in the grid is decided by the average of the z values (height) of all the 3D points projected into the current grid. Figure 8(a)-(b) shows an illustration.



(a) (b)



(c) (d)

Figure 8: Depth image clustering based on spatial and range affinity. (a) Original LIDAR points; (b) Depth image converted from the LIDAR points; (c) Depth image clustering by region-growing based on the depth affinity. Different clusters are shown in different colors; (d) Connect component analysis for the leftover pixels from the previous clustering step. Different components are shown in different colors.

Next we conduct region-growing based clustering [3] on the depth image. Starting with a seed pixel of a cluster, we iteratively include its neighboring pixels into the current cluster if: (1) the intensity difference between the neighbor pixel and the current pixel was within a threshold, and (2) if the difference between the neighbor pixels' intensity and the pixel' mean intensity in the current cluster is within a threshold. Figure 8(c) shows the result after the clustering. Here the same color indicates the same cluster. The largest cluster is extracted as the ground cluster and is used in the later classification step. As can be seen from the figure, there are still some remaining pixels that are not assigned to any clusters. For example, pixels belonging to trees generally have large depth

variations preventing them from joining any clusters. After clustering we conduct connected component analysis on the remaining pixels to group them together based on spatial proximity. Figure 8(d) shows the result after connected component analysis.

### 3.3.2. Supervised Classification

We conduct classification using Support Vector Machine (SVM) to train a classifier that can separate tree points from building points. The features we use in training included the following four features: Normalized height, Normal vector, Normal vector variation, and LIDAR return intensity.

(1). Normalized height

Normalized height is calculated by subtracting the height value at the current point by the corresponding height value of the ground. The ground height is obtained from the ground cluster extracted from the previous clustering step followed by a hole-filling step to fill in some small gaps in the ground cluster. Figure 9 shows the ground extracted from the clustering step. Figure 10 shows the ground after the hole-filling step.

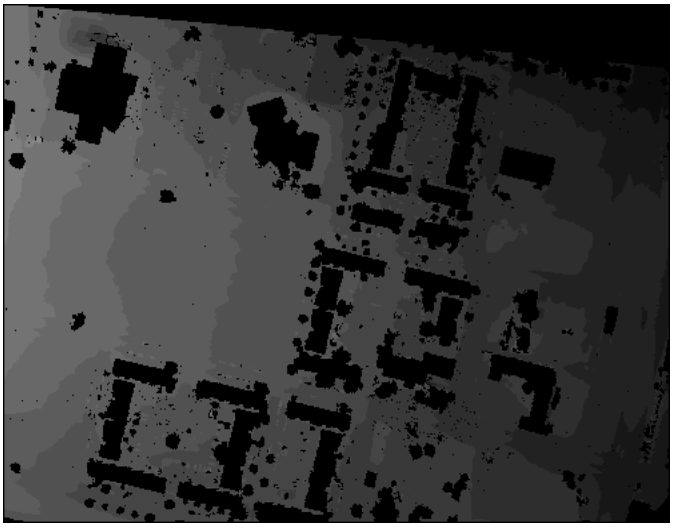


Figure 9. This figure represents extracted ground taken during clustering step.

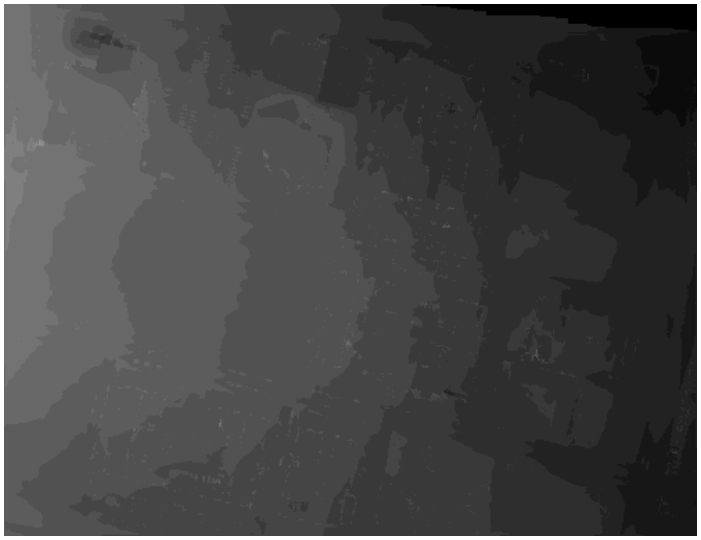


Figure 10. This shows the ground after it was filled to adjust terrain elevation.

(2). Normal vector

Normal vector is estimated using covariance analysis of the local neighborhood and is defined as the eigenvector of the smallest eigenvalue [1]. Normal vector can be calculated by PCA. First construct covariance matrix C by neighbor points as follows:

$$C=\begin{bmatrix} p_{i_1} & - & \bar{p} \\ & \cdots & \\ p_{i_k} & - & \bar{p} \end{bmatrix}^T \cdot \begin{bmatrix} p_{i_1} & - & \bar{p} \\ & \cdots & \\ p_{i_k} & - & \bar{p} \end{bmatrix}, \; i_j \in N_p$$

$$C \cdot V_l = \lambda_l \cdot V_l, l \in \{0,1,2\},$$

C is 3 x 3 covariance matrix of local neighbor points find eigenvalues of C. The eigenvector of the smallest eigenvalue will be normal vector. This can be shown in Figure 11.

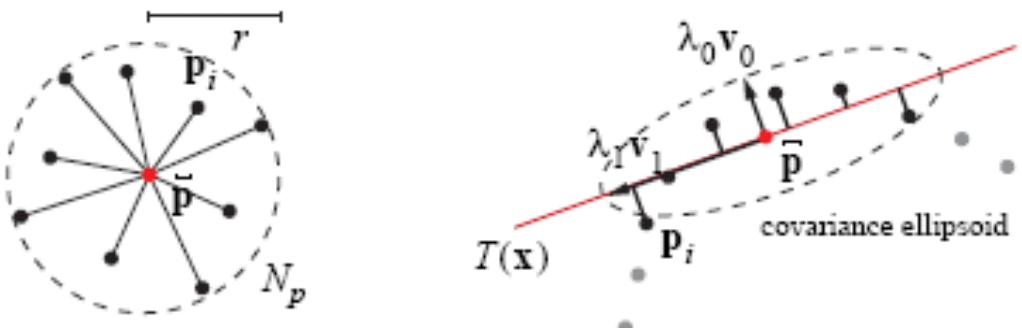



Figure 11 normal vectors from neighbor points

Here r is the radius of globe. To decide how many neighbors to include, calculate the normal vector. The normal vector needs to be normalized so that the length of the vector is 1.

Figure 12 is a visual display of the normal vector. Here (x, y, and z) is represented by red, green, and blue colors.

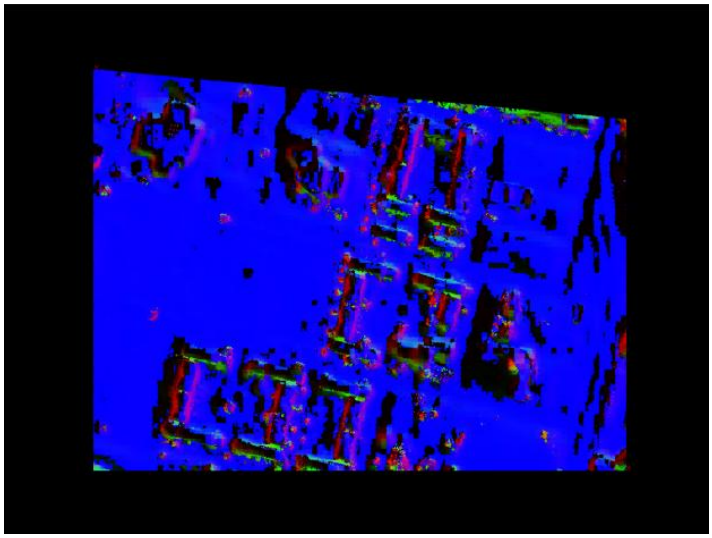


Figure 12. Normal vector with x, y, and z represented as red, green and blue colors

(3).Normal vector variation

Normal vector variation measures the variations within the local neighborhood of the current point and is defined as the mean cosine measure of the normal vectors in the local neighborhood:

$$S(\vec{X}, \vec{Y}) = \frac{\vec{X}^t \cdot \vec{Y}}{|\vec{X}||\vec{Y}|} \qquad (1)$$

Figure 13 illustrates a visual display of normal variation. Smaller variation will have bigger intensity.

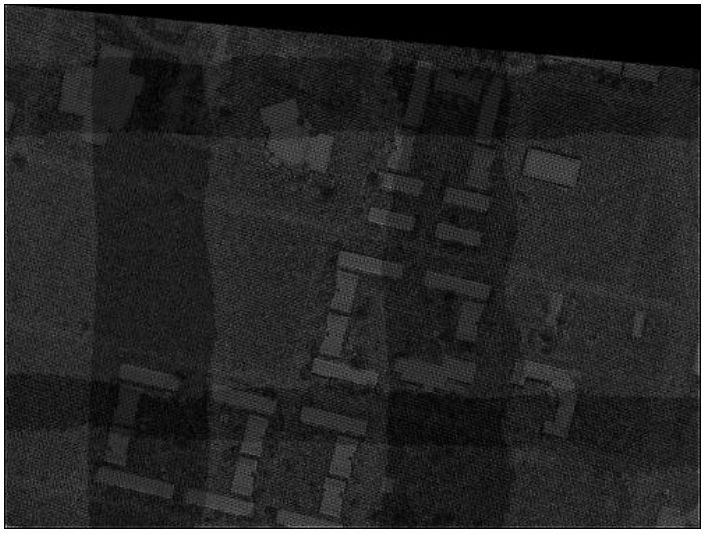


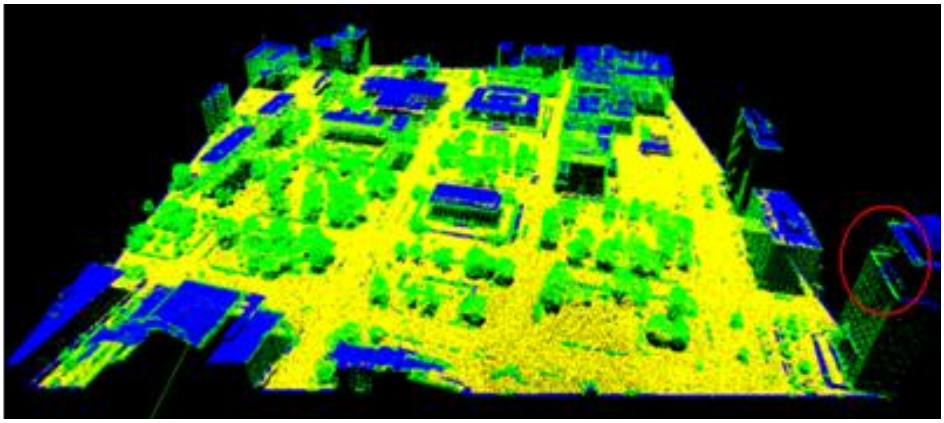
Figure 13. This is a visual display of normal variation.

(4). Laser returned intensity

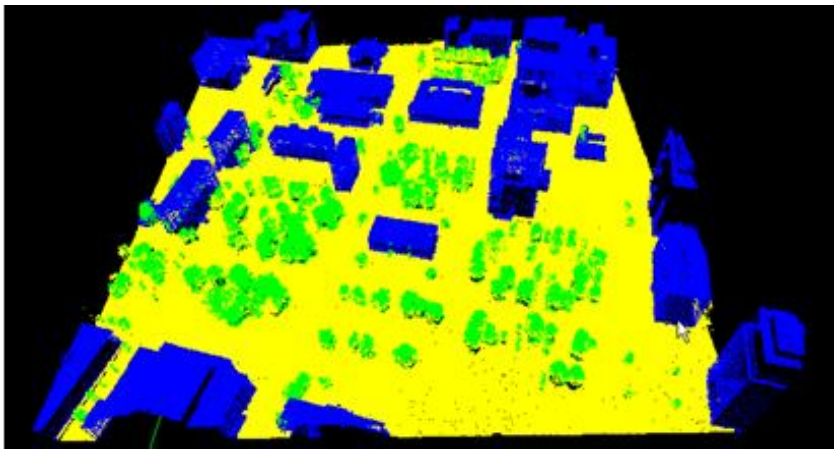
Laser returned intensity is the scalar value reflected from the object back to the LIDAR scanner that relates to the material reflectivity of the object. Figure 14 is a visual display of laser returned intensity.

Figure 14. This is a visual display of laser returned intensity.

The above four features are very effective in separating typical tree points and typical building points (i.e., when the data does not contain points in the building boundaries or tree boundaries). As shown in Table 1 (a)-(c), the training accuracy can reach over 99% when we train the classifier on tree/non-tree points, building/non-building points and ground/non-ground points, respectively. However since points near the building boundaries often have large normal variations, the above features cannot separate building boundary points from tree points very well. Figure 15(a) shows the classified result on downtown St Louis using the classifier trained by the four features. Here ground points are colored in yellow, and building points are shown in blue, while tree points are shown in green. A red circle highlights one of the areas where building boundary points are misclassified as tree points. If we include building boundary points in the training data to retrain the classifier using the four features, the test accuracy reduces to 92% as shown in the confusion matrix listed in Table 2. To overcome the difficulty in separating between tree points and building boundary points, we propose a new 3D shape feature based on robust statistics and Figure 15(b) shows the results after using the new feature.



(a)



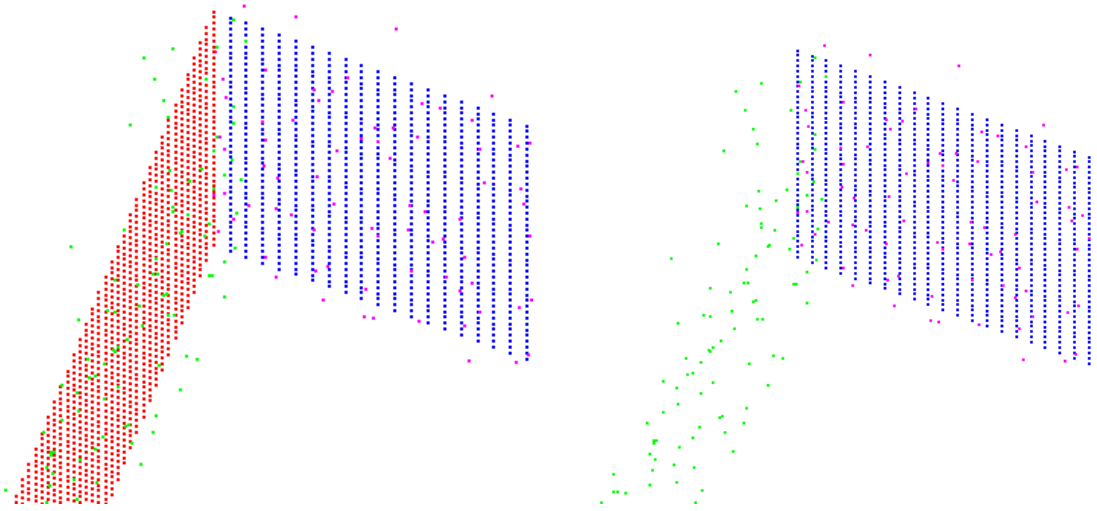
(b)

Figure 15: An example of the classification results on an area of around 420000 $m^2$ of downtown St Louis (a) using models trained without the new feature. Red circle highlights one of the areas where building boundary points are misclassified as tree points due to high normal variations. (b) Result after using the new feature.

### 3.3.3. 3D Shape Feature

The new 3D shape feature is based on the observation that there are generally multiple dominant planar structures near the building boundaries where there are no dominant structures in tree points. We employ iterative plane fitting with RANSAC [8] to explicitly detect the multiple dominant planes in the neighborhood of building boundaries. Figure

16 shows an illustration. In Figure 16(a) one dominant plane was fitted (based on the red points), while Figure 16 (b) shows where another dominant plane was fitted (based on the blue points). After the fitting, we compute the percentage of points in the neighborhoods that are inliers of one of the fitted planes and use it as a 3D shape feature in the tree classification. Adding this new feature causes classification accuracy to increase from 92% to 98.77%. Figure15 (b) shows the classification results on the same area of downtown St Louis with the new shape feature added in training.



(a) (b)

Figure 16: 3D shape feature. Synthetic data points simulate building boundary points (red points and blue points) with some noisy green and purple points added. We employed RANSAC based iterative plane fitting to detect the main dominant planes. (a): One of the planes is extracted based on the red points; (b): After removing the inliers (the red points), the other plane extracted based on the blue points. The percentage of the points in this neighborhood containing inliers of one of the two fitted planes was used as a 3D shape feature in the tree classification.

### 3.3.4 Clustering Based Classification Refinement

In this work the classification is conducted on 3D points directly. However, the training data is based on the 2D depth image instead of 3D points. This implementation simplified the training process greatly, but it also created difficulty in providing training data that includes points in the boundary of trees. This is due to the fact that several 3D points could project into one 2D pixel; thus, it is possible to have both tree points and non-tree points projecting into the same 2D pixel. The lack of training data in the boundary of trees can result in some of misclassification near the boundary of trees as can be seen in Figures 18 (a) and (b). To overcome this limitation, this research methodology employs results from the previous clustering step to conduct classification refinement. More specifically if a point is classified as tree then it is labeled as tree; if a point is classified as non-tree points, then we check its corresponding cluster obtained from the previous clustering result. If the current point is located in a cluster whose majority members are trees then the current points' label is assigned as trees. Figure 17 shows the flow chart of the refined classification process. Figure 18 shows the classification with the refinement step.

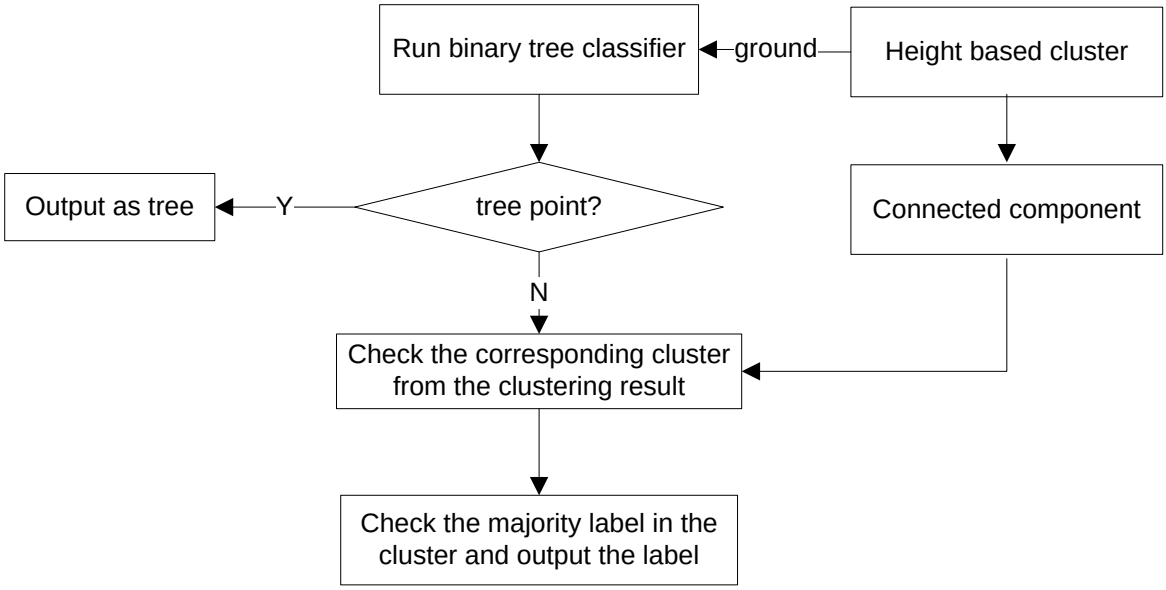
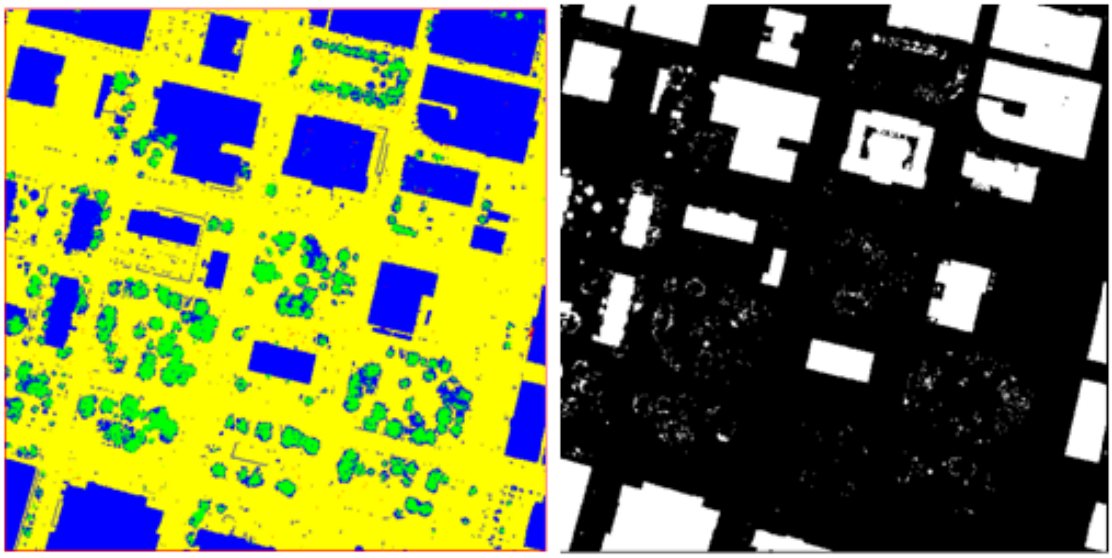
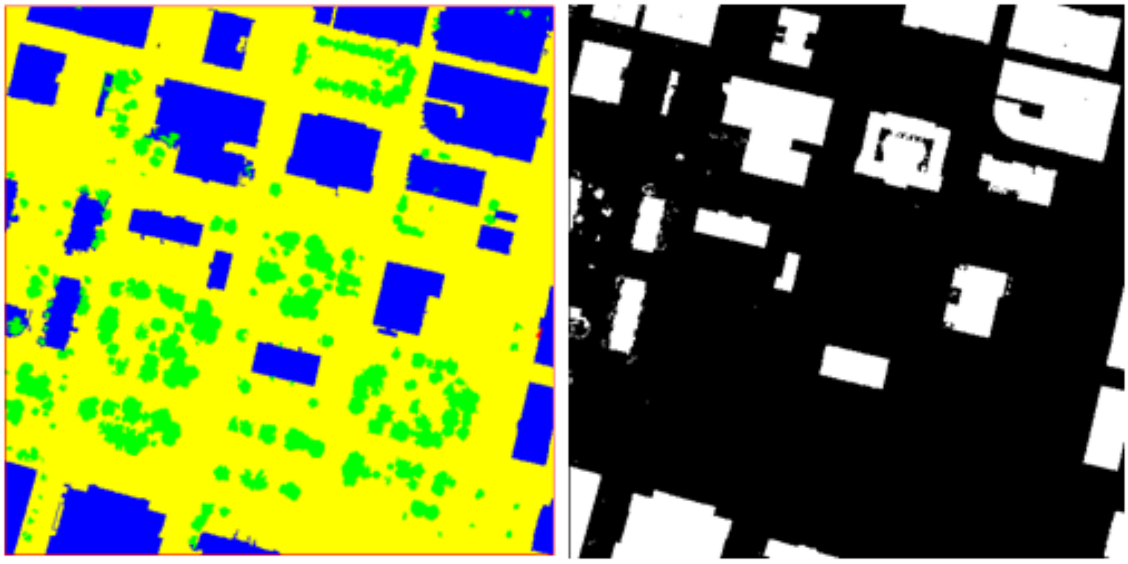


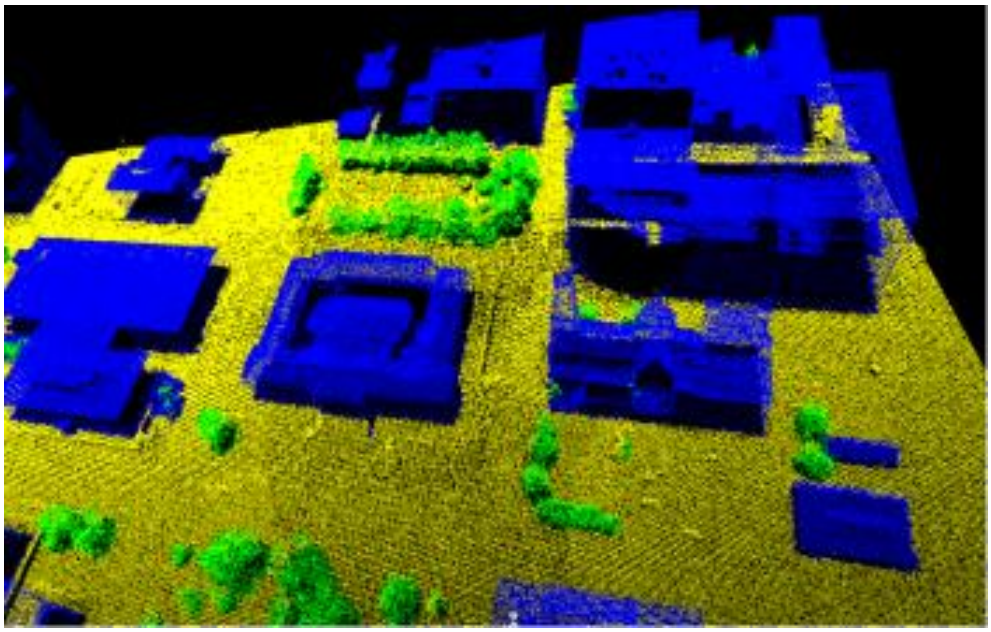

Figure 17 Flow chart of the classification process.



(a) (b)



(c) (d)



(e)

Figure 18: Classification results before and after refinement: (a) Results are based on the supervised classifier only. Some of the tree points are misclassified; (b) Binary depth image of points are shown after removing tree points classified in (a); (c) Results are shown after clustering based classification refinement; (d) Binary depth image of points are shown after removing tree points classified in (c) illustrating that tree boundaries can be classified accurately by comparing with (b). (e) Image shows a3D view of refined classification result.

## 3.4 Experimental Results

To verify our method, we use 5-fold cross validation for the supervised classification based on Support Vector Machine (SVM-light) with RBF kernel. Table 1 shows the confusion matrix when the first four features are used and the training data includes typical trees and buildings. Here the accuracy is above 99%. If the training data include the building boundaries, then the accuracy reduce to about 92% (Table 2). When the new 3D feature is used in addition to the four features, the accuracy rate goes back up to above 98% (Table 3). The final accuracy after the hybrid approach is applied is 99.21%. We apply the model to multiple unknown datasets and find the results to be very accurate. Figure 19 shows another example.

| | | | | | | | | Accuracy |
|---|---|---|---|---|---|---|---|---|
| TP | 3150 | FP | 38 | FN | 201 | TN | 48396 | 99.54% |
| TP | 3165 | FP | 44 | FN | 186 | TN | 48390 | 99.56% |
| TP | 3169 | FP | 32 | FN | 181 | TN | 48402 | 99.59% |
| TP | 3152 | FP | 43 | FN | 198 | TN | 48391 | 99.53% |
| TP | 3128 | FP | 41 | FN | 222 | TN | 48393 | 99.49% |

(a)

| | | | | | | | | **Accuracy** |
|---|---|---|---|---|---|---|---|---|
| TP | 42485 | FP | 183 | FN | 27 | TN | 9090 | **99.59%** |
| TP | 42492 | FP | 176 | FN | 20 | TN | 9097 | **99.62%** |
| TP | 42490 | FP | 186 | FN | 21 | TN | 9087 | **99.60%** |
| TP | 42496 | FP | 188 | FN | 15 | TN | 9085 | **99.61%** |
| TP | 42499 | FP | 192 | FN | 12 | TN | 9081 | **99.61%** |

(b)

| | | | | | | | | Accuracy |
|---|---|---|---|---|---|---|---|---|
| TP | 5894 | FP | 16 | FN | 29 | TN | 45846 | 99.91% |
| TP | 5897 | FP | 15 | FN | 26 | TN | 45847 | 99.92% |
| TP | 5899 | FP | 10 | FN | 24 | TN | 45852 | 99.93% |
| TP | 5894 | FP | 18 | FN | 28 | TN | 45844 | 99.91% |
| TP | 5891 | FP | 15 | FN | 31 | TN | 45846 | 99.91% |

(c)

Table 1. Confusion matrix for training data including typical tree points, typical building points, and ground points using the first four features on: (a) buildings vs. non-buildings; (b) ground vs. non-ground; and (c) trees vs. non-trees, respectively.

| | | | | | | | | Accuracy |
|---|---|---|---|---|---|---|---|---|
| TP | 44973 | FP | 2359 | FN | 1293 | TN | 2102 | 92.80% |
| TP | 44995 | FP | 2333 | FN | 1271 | TN | 2128 | 92.90% |
| TP | 44919 | FP | 2332 | FN | 1347 | TN | 2129 | 92.75% |
| TP | 44917 | FP | 2319 | FN | 1349 | TN | 2142 | 92.77% |
| TP | 44908 | FP | 2336 | FN | 1357 | TN | 2124 | 92.72% |

Table 2 Confusion matrix for trees vs. non-trees using the first four features and the training data include building boundary points.

| | | | | | | | | Accuracy |
|---|---|---|---|---|---|---|---|---|
| TP | 46148 | FP | 505 | FN | 118 | TN | 3956 | **98.77%** |
| TP | 46150 | FP | 515 | FN | 116 | TN | 3946 | **98.76%** |
| TP | 46128 | FP | 543 | FN | 138 | TN | 3918 | **98.66%** |
| TP | 46148 | FP | 549 | FN | 118 | TN | 3912 | **98.69%** |
| TP | 46136 | FP | 520 | FN | 129 | TN | 3940 | **98.72%** |

Table 3 Confusion matrix for trees vs. non-trees when the new 3D shape feature is used in addition to the four features with the training data including building boundary points.

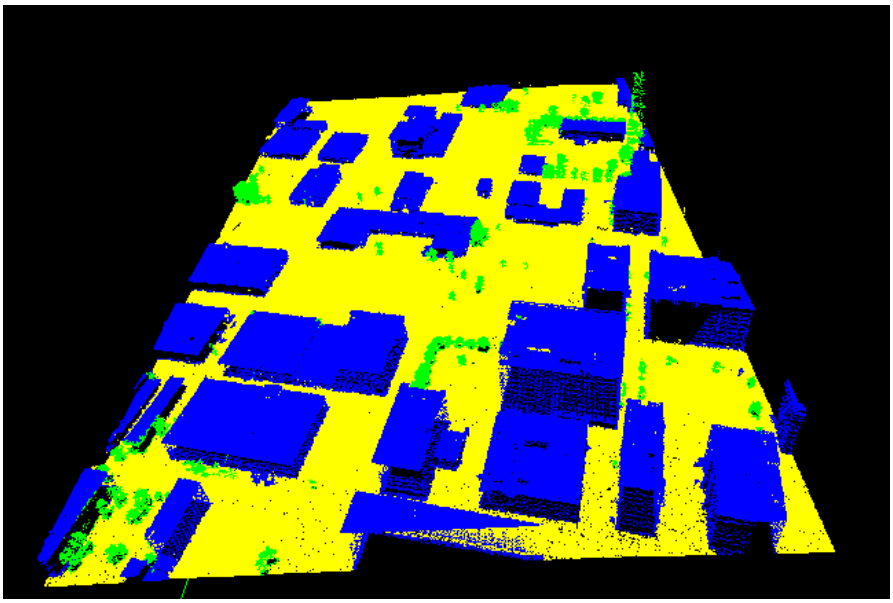


Figure 19: Classification result on another dataset.

## 3.5 Conclusion

In summary, we develop an algorithm which succeeds in overcoming the limitations of existing methods in distinguishing between building boundary points and tree boundary points by using a novel 3D shape feature. Our method of labeling training data in 2D simplifies the labeling process significantly. However since data is classified in 3D, this makes building boundary points and tree boundary points not mutually exclusive. We integrate point wise supervised classification with region-based unsupervised clustering to overcome this limitation and the results show that both building boundary and tree boundary points are classified accurately, even though they are not mutually exclusive in the training data. The experimental results show that we overcome the limitation of existing methods, and achieve a higher accuracy than what can be found in similar work by other LIDAR classification experts.

Supervised classification performs great when the training data and real data are similar. When the real data is not similar to the training data, the performance of supervised learning is expected to degrade, for example, when we use training data from trees full of leaves to classify winter trees which do not have leaves. Or even worse, in some cases, the training data might not available at all. Hence we need to research on a more general approach that do not rely on training data while still can classify the data accurately. In the next chapter, we describe how we achieve this goal by proposing an unsupervised classification scheme.

# Chapter 4

# Unsupervised LIDAR Classification

In this chapter, we discuss unsupervised classification on LIDAR data. We describe related work first, followed by an overview of the proposed scheme, and then we discuss algorithms and give the experimental results.

## 4.1 Related Work

Unsupervised classification on LIDAR data is a more general classification method than supervised classification. Supervised method use training data to train the classifier, allow it to learn from the characteristics of each class from sampled training data, it normally can obtain higher accuracy than unsupervised method if the classified data are very close to training data; unsupervised clustering method, on the other hand, does not use any training data for learning, thus is a more general approach. For example, the training data on trees from our current supervised classification have leaves, for the winter tree that without leaves, its performance on accuracy is expected to reduce, because we train the classifier by trees with leaves; There are not too much peers work in unsupervised method, the most of the state of art LIDAR classification are based on supervised methods. This work proposed an unsupervised method that to our best knowledge, first time be able to classify LIDAR data as accurately as supervised method or even better. It is based on super pixel likelihood computation to better utilize spatial coherence of LIDAR data and applied graph cut multi-label optimization to refine the final results.

### 4.2 Overview of the Proposed Method

To better utilize the spatial coherence between neighboring data points, we perform classification on super pixel instead of data point. Super pixel is obtained as a result of over segmentation on the LIDAR depth image based on height similarity using algorithm such as Mean Shift [9]. Figure 20 shows the flowchart of the unsupervised classification process.

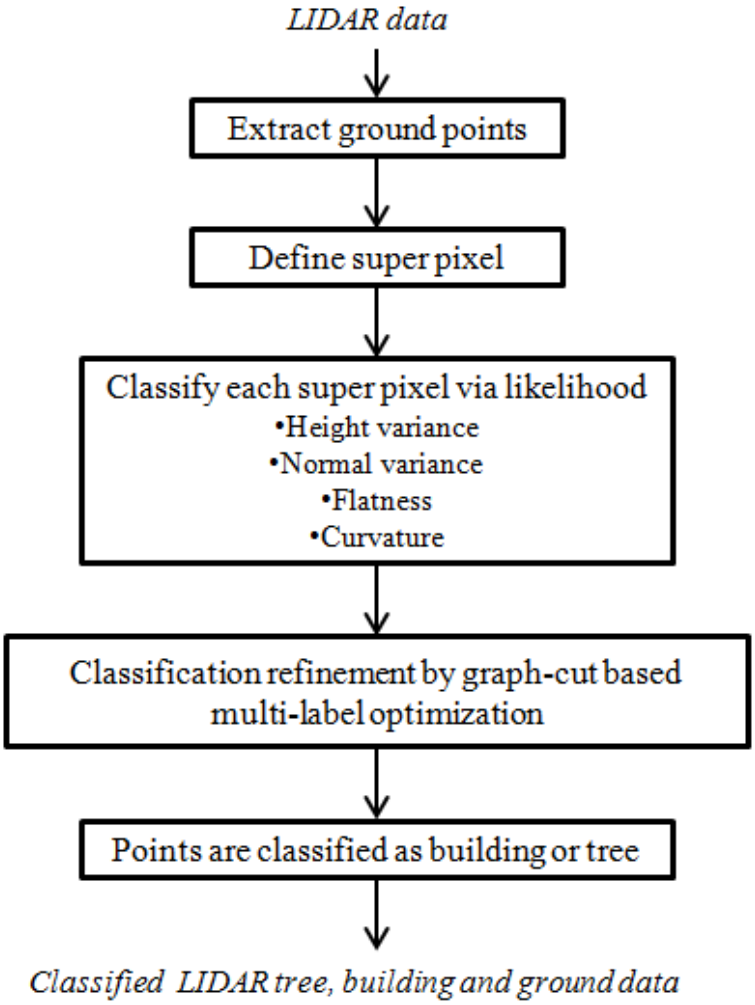


Figure 20. Flow chart of the classification process

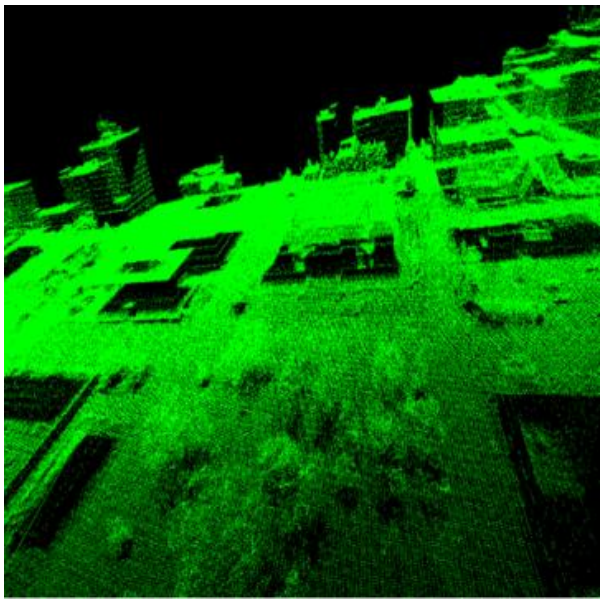
We remove the ground as a preprocessing step to simplify the classification into a simpler binary classification problem. First ground points are extracted via a region-growing based clustering algorithm, followed by super pixel extraction. After that we conduct unsupervised machine learning to classify each super pixel as either building or tree based on the following four features: height variance, normal variance, flatness, and curvature. After classification, we perform a post processing step to further refine the classification. We will describe each of these steps in more details in the following.

### 4.3 Algorithm

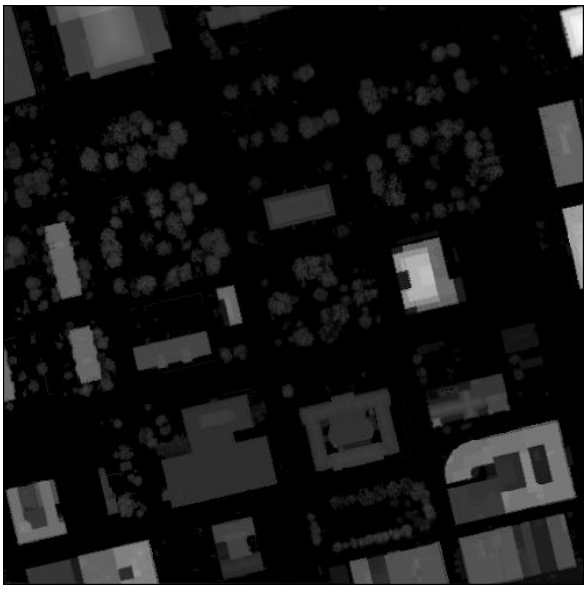
The algorithm is composed of the following steps.

#### 4.3.1 Clustering by Spatial and Range Affinity for Ground Extraction

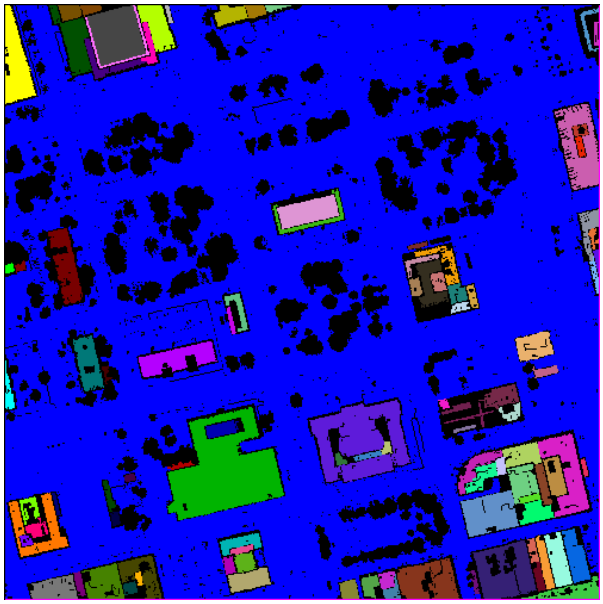
We first convert the 3D LIDAR points into a depth image by projecting the 3D points into a uniform grid on the (x, y) plane. The intensity of the pixel in the grid is decided by the average of the z values (height) of all the 3D points projected into the current grid. Figure 21(a)-(b) shows an illustration.



(a)



(b)



(c)

Figure 21. Depth image clustering based on spatial and range affinity. (a) Original LIDAR points; (b) Depth image generated from the LIDAR points; (c) Depth image clustering by region-growing based on the spatial and range/depth affinity. Different clusters are shown in different colors.

Next we conduct region-growing based clustering [9] on the depth image. Starting with a seed pixel of a cluster, we iteratively include its neighboring pixels into the current

cluster based on whether: (1) the intensity difference between the neighbor pixel and the current pixel is within a threshold, and (2) if the difference between the neighbor pixels' intensity and the pixel' mean intensity in the current cluster is within a threshold. Figure 21(c) shows the result after clustering. Here the same color indicates the same cluster. The largest cluster (shown in blue in Figure 21(c)) is extracted as the ground cluster and is used in the latter classification step. As can be seen from the figure, after clustering, some of the pixels will not be assigned to any clusters. For example, pixels belonging to trees generally have large depth variations thus preventing them from joining any clusters. After ground is extracted, we only need to classify two classes: building and tree.

Finally, after the ground cluster is extracted, we will conduct connected component analysis on all the remaining pixels to group them together into connected components based on the spatial connectivity. This information will be needed in the likelihood-based classification stage to be described later.

### 4.3.2. Compute Super Pixel

Mean Shift [9] is a segmentation method that can group points with similar features together. In this research we use Mean Shift algorithm to segment LIDAR depth image into small homogenous patches based on height similarity. Here we define the segmented small patch as a super pixel, which will be the unit to be used in the later classification stages. Comparing with classification on each data point, using super pixel in classification enables us to take consideration of local neighborhood information and better utilize spatial consistency. Figure 22 shows super pixels generated by Mean Shift

segmentation from the depth image of Figure 21(b). Different super pixels are shown by different colors; the ground is shown in red. Figure 23 (a) shows a close up view of the lower left of Figure 22. Figure 23(b) is the corresponding aerial view from Google Map®. We can see building points with flat roof are clustered into the same super pixel. If height changes too dramatically, Mean Shift will segment them into multiple super pixels.

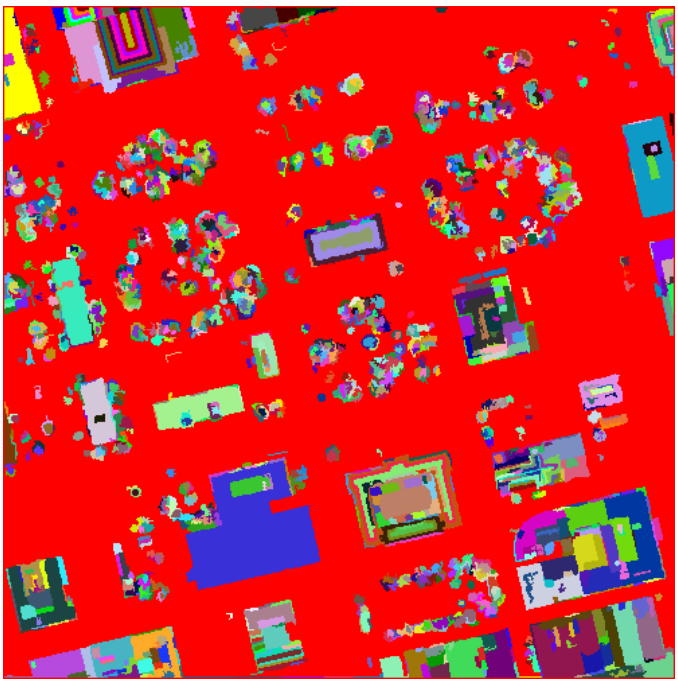
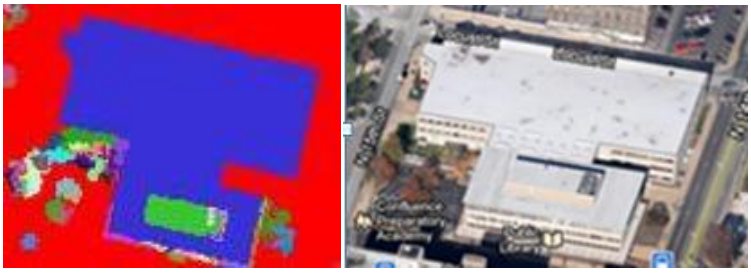


Figure 22. Super pixels generated from the depth image of Figure 21(b).



(a) (b)

Figure 23. Close up view of Figure 22 (lower left). (a) Different super pixels are shown in different colors (red color is ground). (b) Corresponding aerial view from Google Map®.

### 4.3.3 Compute Feature Vector

In this dissertation, we use the following four features to classify each super pixel as either tree or building: height variance, normal variance, flatness, and curvature.

(1). Height variance

We calculate the average height variance within the local neighborhood of each super pixel. For building super pixels, height variance should be small. For tree super pixels, height variance should be large. Figure 24 shows the height variance map of Figure 21(b).

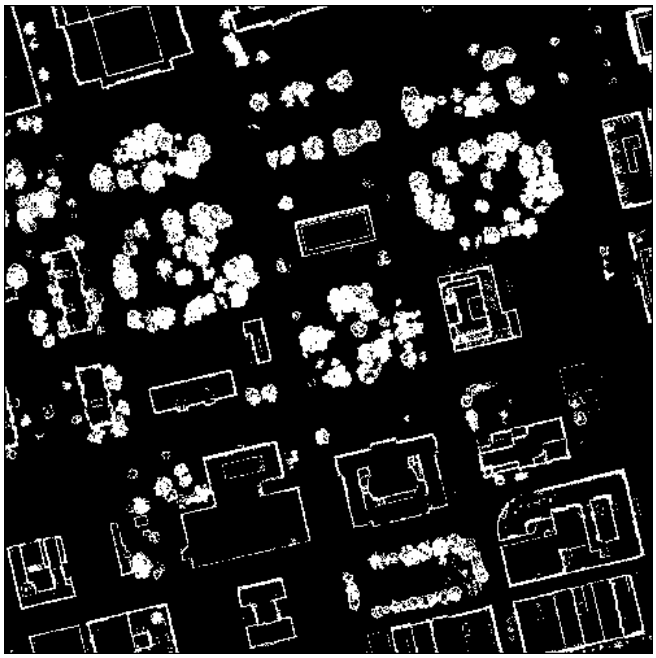
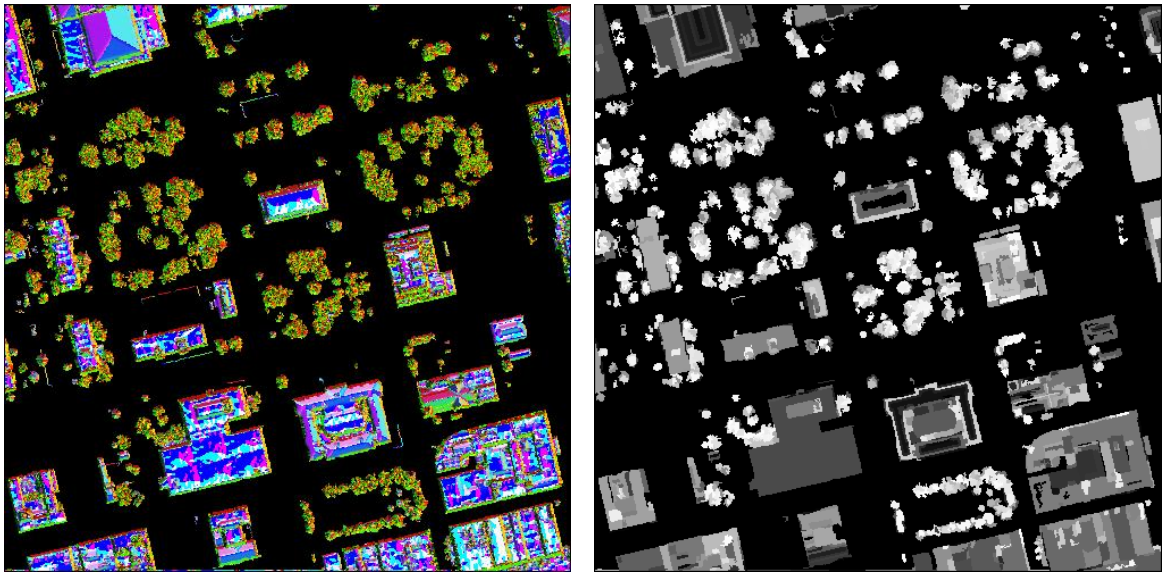


Figure 24. Height variance map of Figure 21(b).

(2). Normal Variance

We estimate the normal vector of each data point by Principle Component Analysis (PCA). We then calculate the average normal variance within the local neighborhood of

each super pixel. For building, normal variance should be small. For tree, normal variance should be large. Figure 25 (a) shows the normal map at each point, and figure 25(b) shows the normal variance for each super pixel.



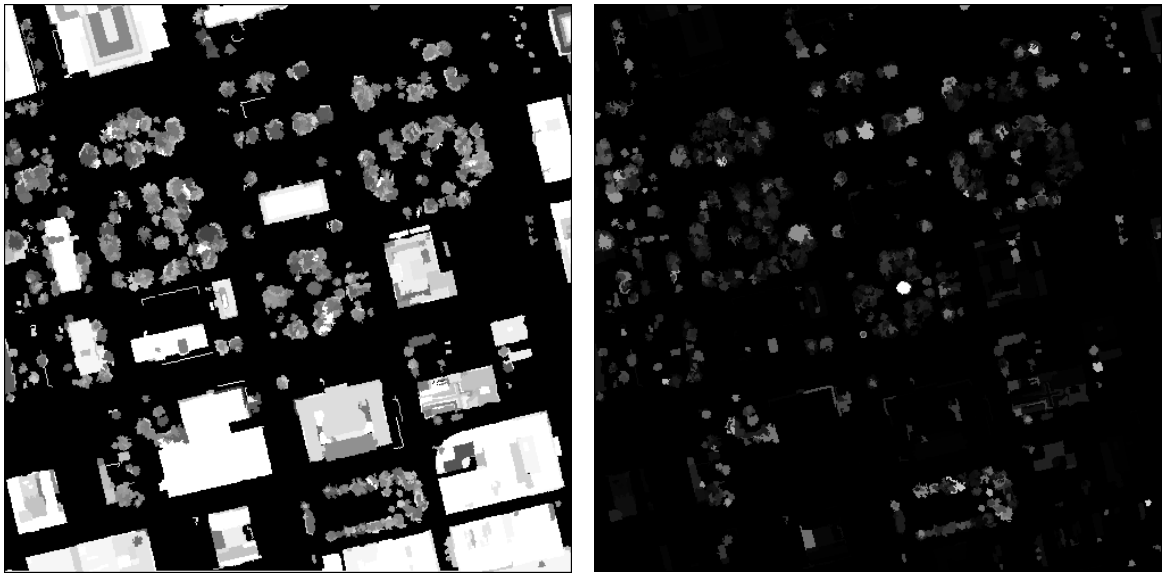
(a) (b)

Figure 25. Normal and normal variance map of Figure 21(b). (a) Normal map. (b) Normal variance map.

(3). Flatness and curvature

For 3D points belong to the same super pixel, we will conduct PCA based plane fitting with Random Sample Consensus (RANSAC). We define the flatness measurement as the percentage of points that are inliers to the fitted plane over the total number of points in the current super pixel. In addition to the flatness measurement, we also computed the curvature of each super pixel as: $\lambda_0 /(\lambda_0 + \lambda_1 + \lambda_2)$, with $\lambda_0 \leq \lambda_1 \leq \lambda_2$ are the three eigenvalues obtained from the PCA. Figure 26 shows the flatness map (a) and the curvature map (b). As can be seen (Figure 26(a)), buildings have high flatness (i.e. large percentage of inliers) while trees have low flatness (i.e. low percentage of inliers).

Similarly, building super pixels should have smaller curvature value, while tree super pixels should have larger curvature value.



(a) (b)

Figure 26. Flatness and Curvature map of Figure 21(b). (a) Flatness map (b) Curvature map.

#### 4.3.4. Likelihood Based Classification

Based on the four features- height variance, normal variance, flatness and curvature, we will compute the building likelihood and tree likelihood for each super pixel. The four features will first be linearly normalized to [0, 1] and then be multiplied together.

$$L_b = (1 - p_{height\_\mathrm{var}}) \cdot (1 - p_{normal\_\mathrm{var}}) \cdot p_{flatness} \cdot (1 - p_{curvature})$$

$$L_t = p_{height\_\mathrm{var}} \cdot p_{normal\_\mathrm{var}} \cdot (1 - p_{flatness}) \cdot p_{curvature}$$

$p_{height\_\mathrm{var}}$, $p_{normal\_\mathrm{var}}$, $p_{flatness}$, $p_{curvature}$ are the normalized values of height variance, normal variance, flatness and curvature, respectively.

For each super pixel we then compare its building likelihood $L_b$ and tree likelihood $L_t$ :

If $L_b > L_t + \Delta$ => it will be classified as building,

If $L_t > L_b + \Delta$ => it will be classified as tree,

Otherwise, it will be classified as unlabeled.

$\Delta$ is a user-defined margin between the classifier. For those super pixels that are classified as unlabeled, we will further refine their classification based on the following observation: In urban setting, the heights of the trees adjacent to buildings are usually quite different than the heights of the adjacent building. Thus for each unlabeled super pixel, we will find its current connected component (Section 4.3.1), compute the histograms of the heights of labeled tree super pixels and labeled building super pixels in the current connect component, and model the histograms by 1D Gaussian function:

$$G_{\text{tree}}(x) = \frac{1}{\sqrt{2\pi\sigma_{\text{tree}}^2}} e^{-\frac{1}{2\sigma_{\text{tree}}^2}*(x-\mu_{\text{tree}})^2}$$

and

$$G_{\text{building}}(x) = \frac{1}{\sqrt{2\pi\sigma_{\text{building}}^2}} e^{-\frac{1}{2\sigma_{\text{building}}^2}*(x-\mu_{\text{building}})^2}$$

Here $\sigma$ is standard deviation of the model and $\mu$ is mean of the model, they can be calculated from the histograms of already labeled trees and buildings in current connected

component. Then for each unlabeled super pixel u, we will compute its tree likelihood $L_t'$ and building likelihood $L_b'$ by plugged in its mean height x into 1D Gaussian functions for tree and building respectively. If $L_t' > L_b' + \Delta$, then it will be classified as tree; Otherwise, it will stay as unlabeled. In experimental results section we show that the classification is robust to user defined margin Δ variation.

Finally we resolve the remaining unlabeled super pixels based on the spatial coherence and smoothness constraints commonly used in the Markov Random Field algorithms such as [43]. Thus we need to assign the unknown label to labels that vary smoothly but change dramatically at object boundary, i.e. preserve boundary discontinuity. This problem can be formulated in energy minimization problem. In energy minimization framework, a label *f* should minimize the energy

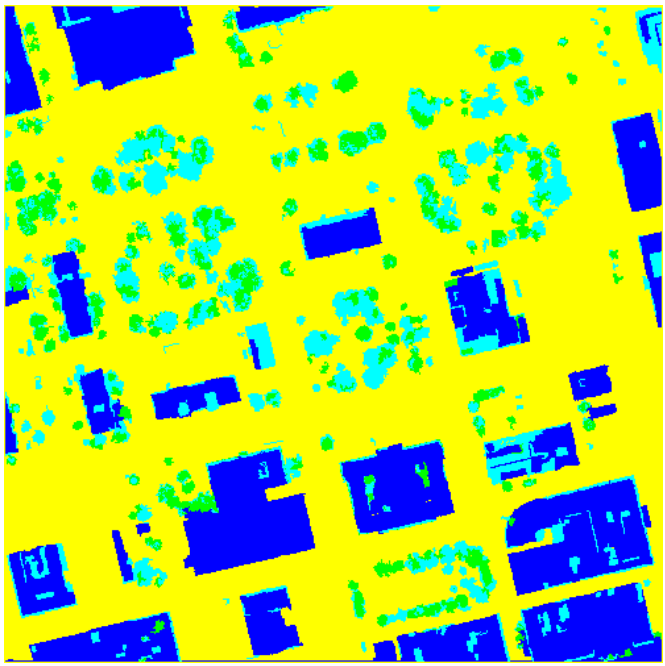
$$\mathrm{E}(f) = \mathrm{E}_{\mathrm{data}}(f) + \mathrm{E}_{\mathrm{smooth}}(f)$$

Here $E_{data}(f)$ measure the disagreement between label and observed data, and $E_{smooth}(f)$ measure the extent to which *f* is not piecewise smooth. More specifically, we define an energy function E as:

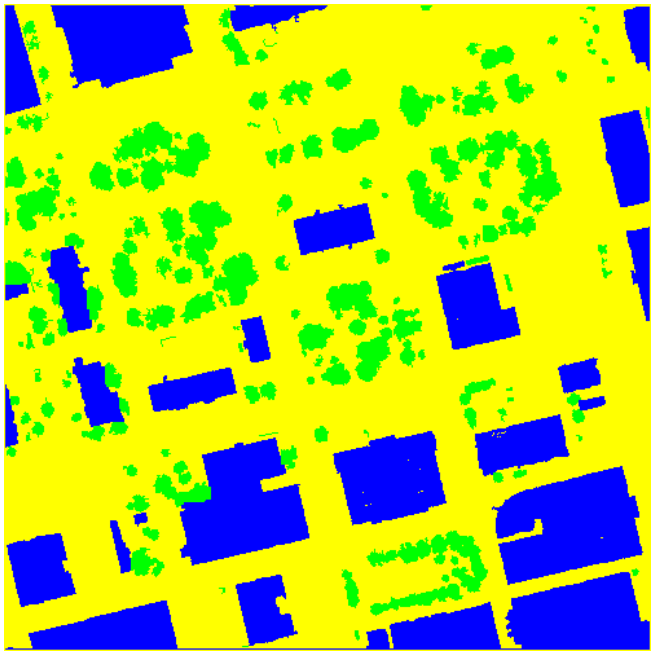
$$\mathrm{E} = \sum \mathrm{D}(\mathrm{p}) + \mu \sum \mathrm{V}(\mathrm{p}, \mathrm{q})$$

The first term $\sum \mathrm{D}(\mathrm{p})$ is $E_{data}(f)$ and second term $\mu \sum \mathrm{V}(\mathrm{p}, \mathrm{q})$ is $E_{smooth}(f)$. Here p and q are adjacent to each other, V(p, q) is the smoothness term and is equal to zero if the labels of p and q are the same, otherwise it is equal to one. D(p) is the data term and is defined as a small constant (e.g. 1) for all the labeled super pixels, and is assigned a large number

(e.g. 100) for all the unlabeled super pixels. μ is a coefficient that controls the effect of the smoothness term. We set μ as 2 in all experiments. We use the Graph-Cut algorithm [59] to find an approximate solution of the global optimum of this energy function. Figure 27 and Figure 28 shows two examples of the unsupervised classification before and after the graph cut optimization.

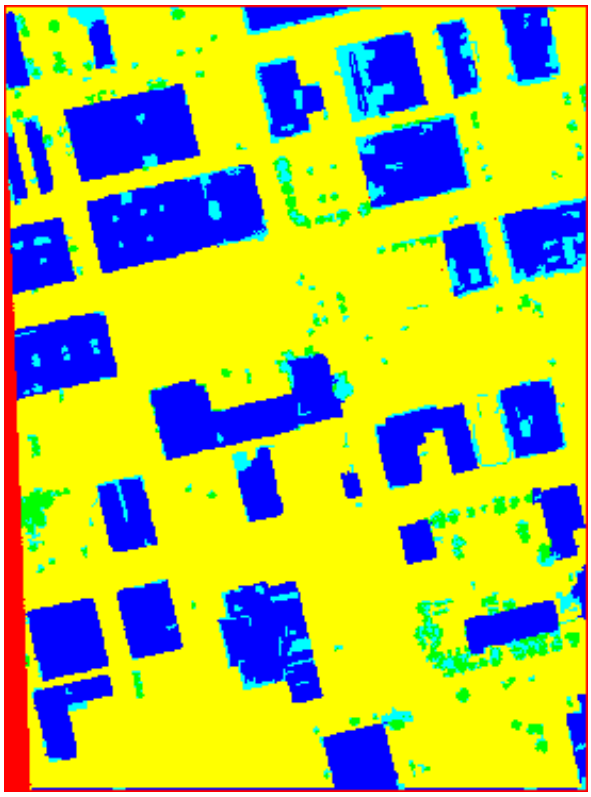


(a)

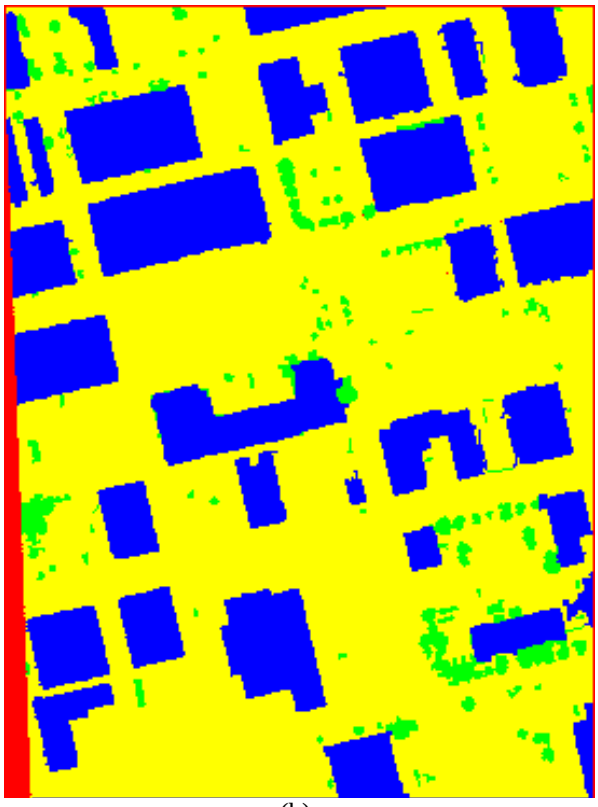


(b)

Figure 27. Unsupervised classified result. (a) Classification result before graph cut based optimization. Different labels are shown in different color: tree as green, building as blue, ground as yellow, and unlabeled as cyan. (b) Classification results after graph cut based optimization.



(a)



(b)

Figure 28. Unsupervised classified result for another data. (a) Classification result before graph cut based optimization. Different labels are shown in different color: tree as green, building as blue, ground as yellow, and unlabeled as cyan. (b) Classification results after graph cut based optimization. Areas shown in red are the regions where there are no data.

In our work on supervised classification as described in Chapter 3, we use Figure 30(a) as our labeled training data (ground truth) for tree classifier, here points in green are labeled as tree and points in blue are labeled as not tree. We use this data to test the accuracy of our unsupervised classification scheme.

## 4.4 Experimental Results

Table 1 shows the corresponding confusion matrix on data from Figure 30(a) using $\Delta$=0.06. The testing result on this piece of data can achieve an accuracy of 99.04%.

| | | | | | | | | Accuracy |
|---|---|---|---|---|---|---|---|---|
| TP | 1573 | FP | 0 | FN | 193 | TN | 18293 | **99.04%** |

Table 4. Confusion matrix of test data shown in Figure 28(a).

We calculate true positive rate and false positive rate [44] of the unsupervised classifier on the test data with different Δ. The sensitivity of the classifier related to Δ is shown by the Receiver Operating Characteristic (ROC) curve [88] in Figure 29. X axis is the false positive rate, Y axis is the true positive rate. The area of the ROC curve can indicate whether or not the classifier is robust to Δ. The area of the obtained ROC curve is very close to 1, which means it is robust to user-defined margin variation Δ.

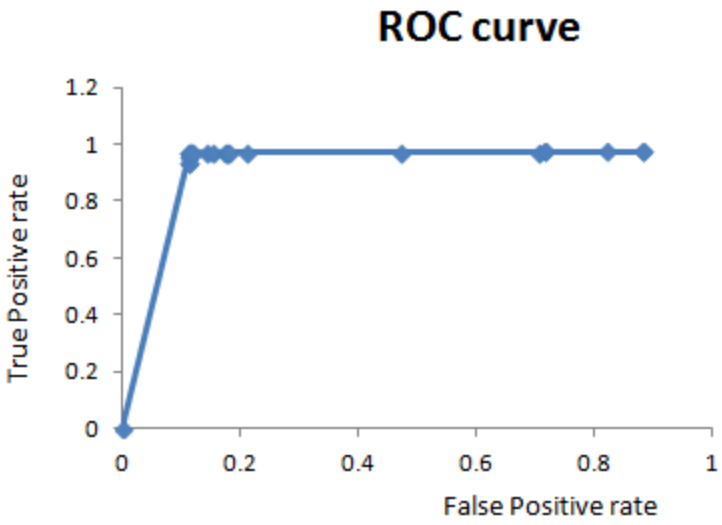



Figure 29 ROC curve of the unsupervised classifier for margin variation.

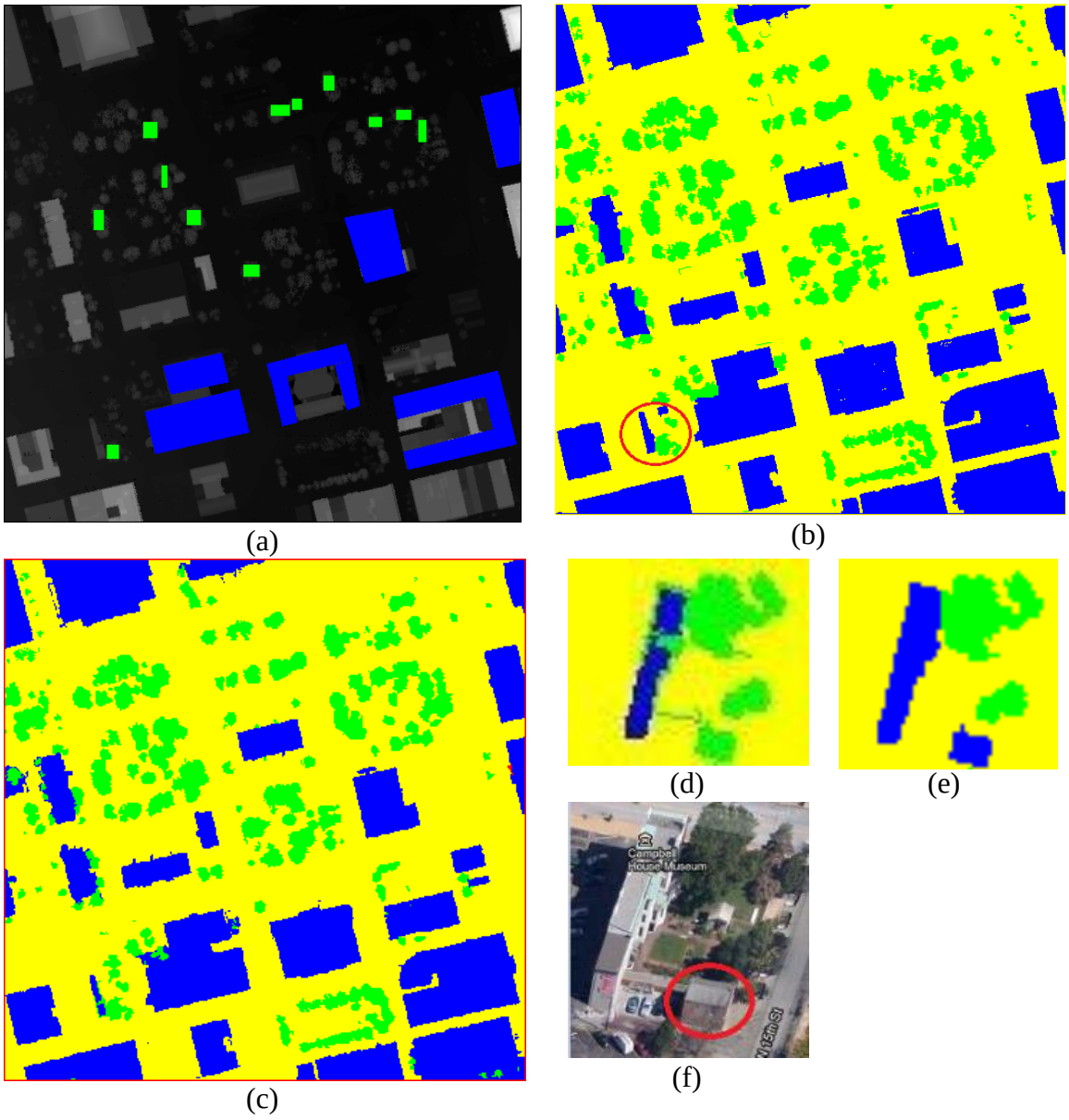



Figure 30. Comparing the unsupervised classification with the supervised classification [25]. (a) Ground truth data used in supervised classification. Green points are labeled as tree and blue points are labeled as non-tree. (b) Result of unsupervised classification. All the labeled points of (a) are correctly classified. (c) Classification result from previous supervised learning . (d) and (e) are the close up views of (b) and (c) in the highlighted area (shown in red circle in (b)), respectively. (f) The corresponding aerial image from Google map ® that can serve as the ground truth for (d) and (e).

Figure 30(b) is the 2D view of the unsupervised classification result, yellow indicate ground points, green is tree points, and blue is building points. Figure 30(c) is the result

of our previous supervised classification work via SVM. Compare Figure 30(b) with 30(c), we can observe that the result is very close to each other. There are some classification difference between Figure 30 (b) and Figure 30(c) as highlighted by the red circle in (b). Figure 30 (d) and 30 (e) are the close up views of 30 (b) and 30 (c) in the highlighted area, respectively. 30 (f) is the corresponding aerial image from Google map ® that can serve as the ground truth. Compared with the ground truth, a tiny small building shown in red circle in Figure 30(f) is misclassified as tree by supervised learning, and it is correctly classified as building in our new unsupervised method. Figure 31 shows a 3D view of the LIDAR points of Figure 30 after unsupervised classification.

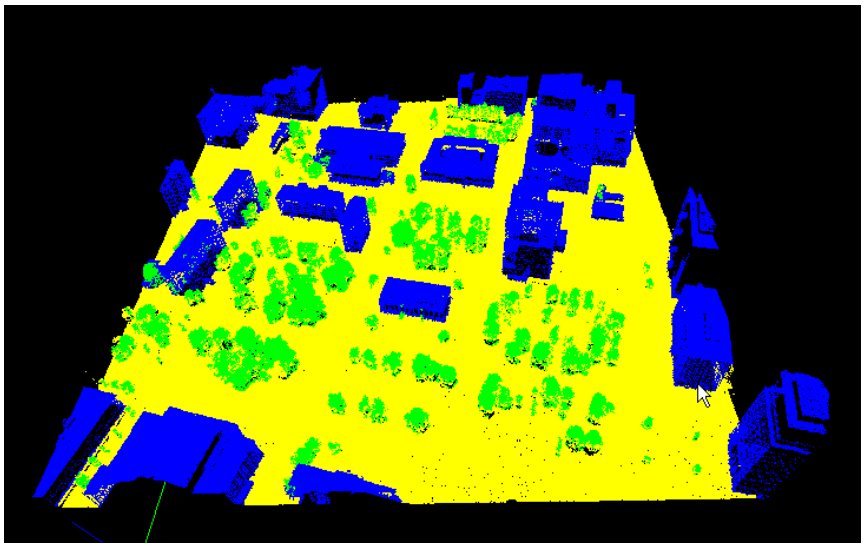
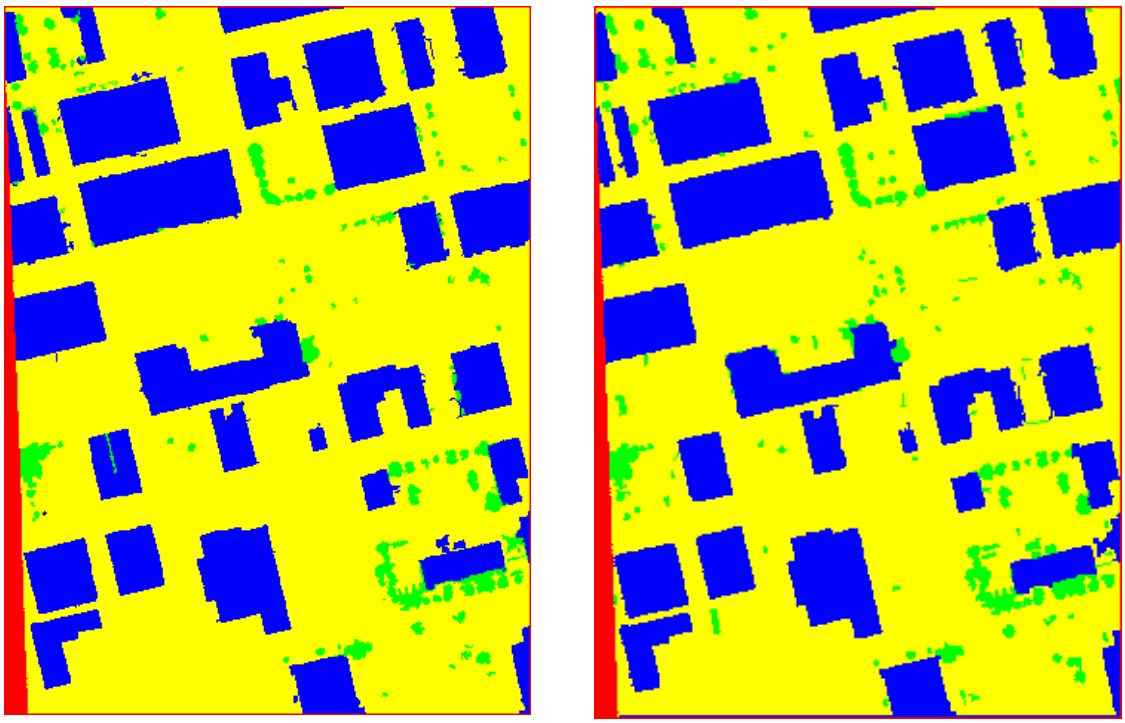


Figure 31. 3D view of the LIDAR points of Figure 30 after unsupervised classification. Ground points are shown in yellow, green points are tree points are shown in green, and building points are shown in blue.

Figure 32 shows the comparison between supervised classification and unsupervised classification on another set of LIDAR data. Figure 33 shows the corresponding 3D view of the unsupervised classification results.



(a) (b)

Figure 32. Comparison of the proposed unsupervised classification with supervised classification on another set of LIDAR data (Areas shown in red are the regions where there are no data). (a) Result from supervised classification. (b) Result from unsupervised classification.

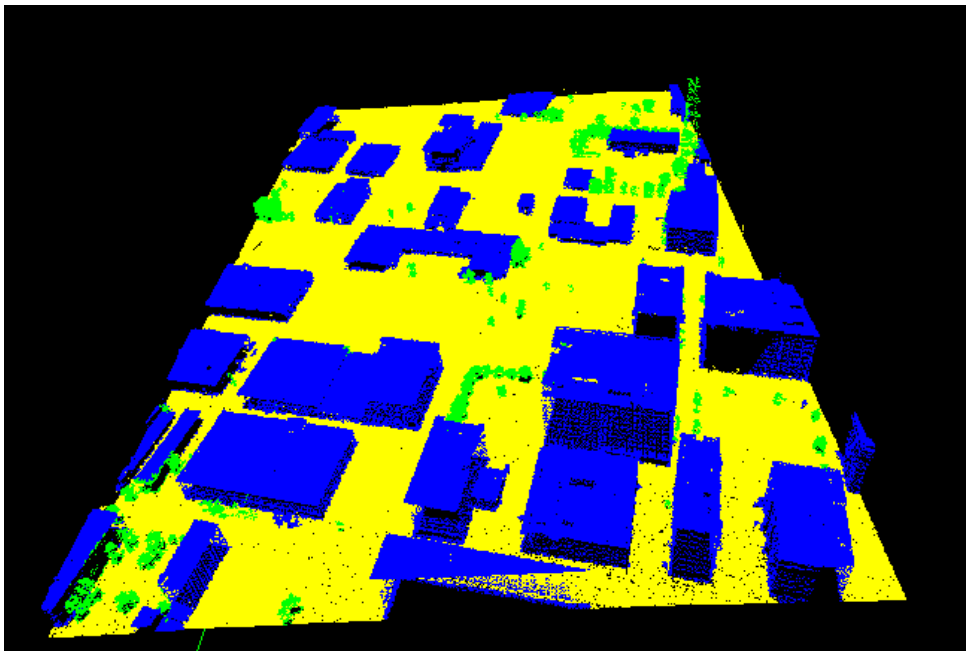


Figure 33. 3D view of the unsupervised classification results for LIDAR points of Figure 32. Ground points are shown in yellow, green points are tree points are shown in green, and building points are shown in blue.

## 4.5 Conclusion

In summary, in order to better utilize the spatial coherence between neighboring data points, we extract the ground first by a height based clustering method, then we perform classification on super pixels instead of data points. We computer super pixels based on over segmentation on the LIDAR depth image based on height similarity. Then for each super pixel, we compute the likelihood based on four features: height variance, normal variance, flatness and curvature. After performing likelihood based classification, we have data labeled as tree, building, ground and unlabelled. We resolve the remaining unlabeled super pixels based on the spatial coherence and smoothness constraints and apply graph cut multi-label optimization to refine the final results. Experimental results show that our research can obtain high accuracy for unsupervised classification. The results are as good as supervised classification, and in some cases, even better.

Hence, we have LIDAR classification schemes that can classify airborne LIDAR data via both supervised and unsupervised methods. The results are very accurate based on our experimental results. This builds a solid foundation for our next step toward compressing LIDAR data efficiently. The next chapter describes the details of our proposed methods for LIDAR compression.

## Chapter 5

# Geometry Based LIDAR Compression

In this chapter, we describe the proposed LIDAR compression methods. First we introduce and discuss related work, pointing out that our image based compression scheme has performance at least equivalent to the performance of one of the state-of-the-art LIDAR compression methods; We then describe the detail of our two new LIDAR compression methods - image based LIDAR compression and geometry based LIDAR compression, followed by experimental results and result comparison.

### 5.1 Related Work

Among existing LIDAR data compression, lossless or lossy compression to 5%-23% of the original file size has been reported [13, 14, 51]. For example Isenburg's work on lossless LIDAR compression predicts the attributes of a point from previous points with a set of prediction rules and compresses the corrective deltas with adaptive, context-based arithmetic coding. Pradhan et al [13] use second generation wavelet. They first split a triangle into several sub-triangles and the elevation step has been used to 'modify' the point coordinates for geometry after the splitting. Then compress resulting dataset using second generation wavelets. Their results shows geographical surface representation after compression however does not perform error analysis as they indicated "We are currently investigating the detailed error analysis for the different sets of data sets at different scales".

Gerlek [51] was enlightened by 2D space raster image compression and proposed to use wavelets to transform the points from (x,y,z) space into a space in which the data are represented at multiple levels of resolution. They then lessen the precision of the data points so that each point can be represented in less space. They adjust the amount of loss by throwing away a number of least significant bits and can achieve compression rate ranging from 5-17%. [51] Applied the same concept as JPEG2000, except as they described, "2D raster datasets are regularly gridded while the LIDAR datasets are not." For 2D raster image, we don't need to explicitly encode the x and y coordinates, just the pixel values at each grid point. For LIDAR data, which is 3D case, [51] thus encode the x and y coordinates as well as z coordinates. The rest of their methods are same with JPEG2000, applying arithmetic encoding and bit planning, the most significant bit of each piece of data forms the first "bit plane", the second most significant bit of each item forms the second bit plane and so on. They adjust the amount of loss by throwing away a number of least significant bits. Our analysis shows that in lower bit rate, after throwing away many less significant bits, their loss on accuracy of x and y coordinates are usually bigger than projecting the 3D points into a higher resolution uniform grid on the (x, y) plane. In this thesis, we propose an image based compression, which will project 3D LIDAR points into 2D depth image. In our depth image generation method, we can define very high resolution uniform grid on (x, y) plane, this equivalent to very small quantization step and thus the loss due to quantization of x and y could be very small. Although a very high resolution depth image could potentially has cells that has no LIDAR data, as shown in Figure 34, this can be overcome by simple image smoothing by neighbor pixels. Figure 35 is depth image with empty cell filled by neighbor value.

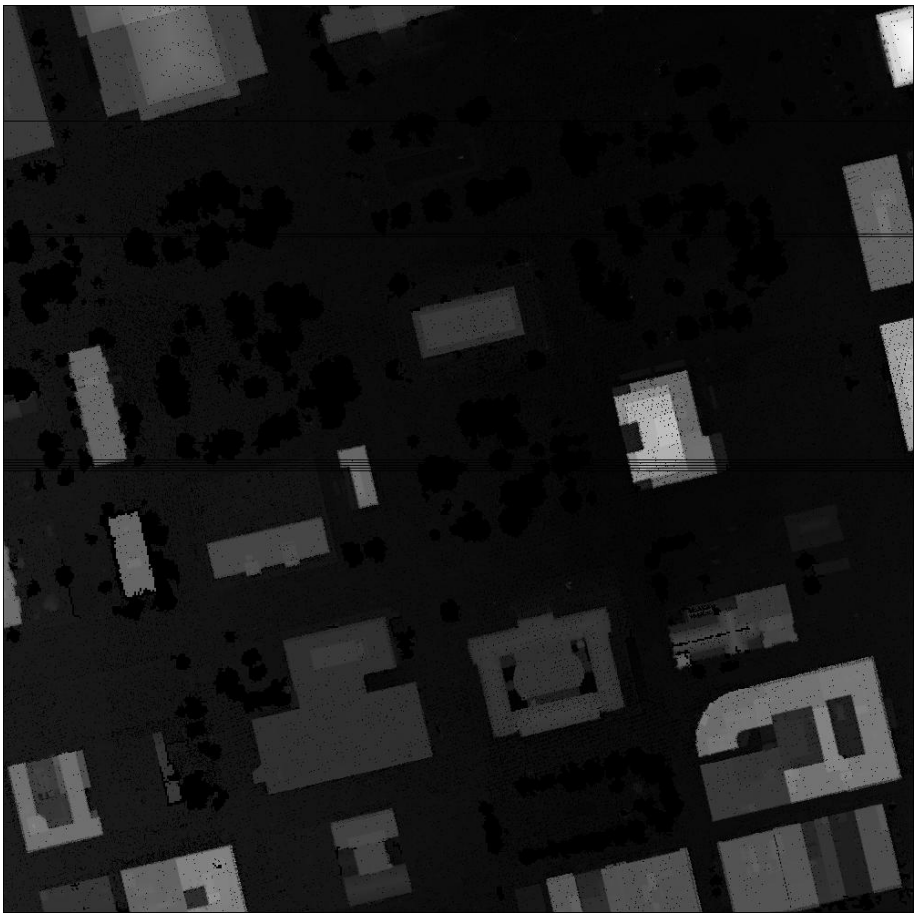


Figure 34  Depth image of higher resolution image with empty cell

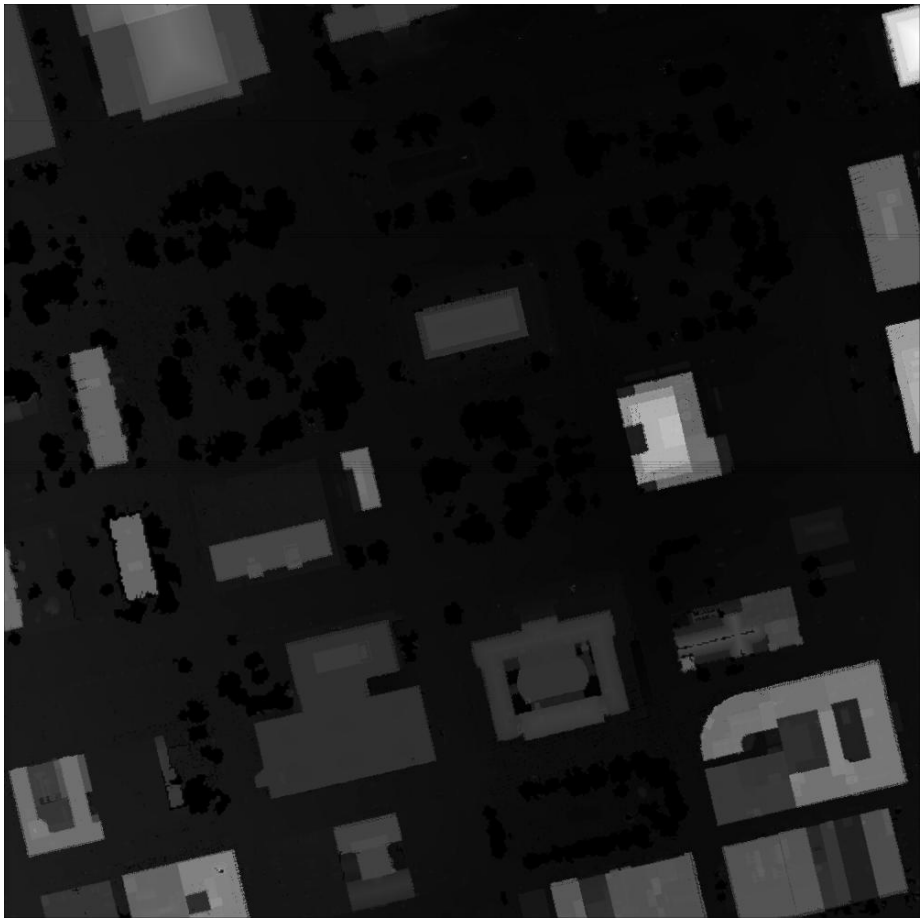


Figure 35 Depth image of higher resolution with empty cell filled by neighbor value

In the next section, we will describe our image based compression method. We note that the distortion on x and y are determined when they are quantized on a uniform grid of the (x, y) plane; it is fixed and does not change even on lower bit rates. For high resolution this loss can be very small, while [51] discards many significant bits for x and y in lower bit rates, thus the loss will be worse than our method for x and y encoding. If we apply the same compression method on Z as [51] does, in lower bit rates, our projection method for x and y is better than [51]. This can indicate that the performance of our image based compression method will be equivalent to or better than [51] in lower bit rates.

## 5.2 Image based Compression

In our lossy compression scheme, we apply quantization to the data first. Since LIDAR data is not set on any inherent grid, we quantize them into a fixed grid on the x and y directions. We first project the 3D data points into 2D depth image, this is equivalent to quantize (normalize) the data to a fixed number of bits in the integer domain. We then compress the depth image using existing image compression methods. Compared with JPEG, JPEG2000 has better performance, especially in lower bit rates. Figure 36 shows an example.

Figure 36 JPEG2000 perform better than JPEG in lower bit rates (left: original image; middle: JPEG right:JPEG2000)

Sub-band decomposition enables the lower frequency part of the whole image to be transmitted first, while DCT is performed on 8x8 blocks in JPEG. When the bit rate is low, the quantization step is high, which causes the blocky effect in low bit rate JPEG compression. As we mentioned earlier, geospatial imagery often needs to be able to store data at multiple resolution levels or scales so that it can more quickly access the data at coarser granularities, EZW enables the bits in the bit stream to be generated in order of importance yielding a fully embedded code. Thus JPEG2000 fits our goal better and we implemented major steps of image based compression based on JPEG2000.

The performance of this method can be measured by the Rate-Distortion (RD) curve, where the error is measured by the mean square error for each pixel re-projected to a data point in 3D. The procedure is illustrated in Figure 38.

Here are more details of each step.

Step 1: Project LIDAR data into 2D depth Image. The LIDAR data do not lie on an implicit regular grid, as an image does. Therefore, each LIDAR point must be stored explicitly. It is not possible to map each LIDAR point into each pixel. Usually we have several 3D LIDAR points mapped into one 2D pixel. The value of the pixel is the average depth of the 3D points.

Step 2: Apply 5 level sub-band decomposition using (5,3) wavelet. Down sample and up sample can be shown in the Figure 37. Here left is encoder side and right is decoder side

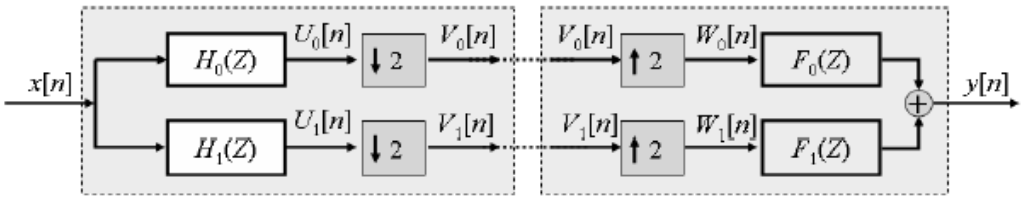



Figure 37. Daubechies (5, 3) filter bank illustration

DWT analysis high pass filter : $H_0(z) = 1/8(-1+2z^{-1}+6z^{-2}+2z^{-3}-z^{-4})$

DWT analysis low pass filter $H_1(z) = 1/2(-1+2z^{-1}+z^{-2})$

DWT synthesis low pass filter: $F_0(z) = 1/2(1+2z^{-1}+z^{-2})$

DWT synthesis high pass filter: $F_1(z) = = 1/8(-1-2z^{-1}+6z^{-2}-2z^{-3}-z^{-4})$

There is no compression error in this step. Because it is a lossless transformation, we can get perfect reconstruction if we ignore all other steps and use steps 2 and 9 exclusively.

Step 3: Quantization on DWT coefficients using round (coefficient/stepSize). There is compression error introduced in this step due to quantization. The bigger the quantization step is, the more compression error is introduced. For example, Figure 39(a) shows a quantization step size of 1 on building ground depth image vs. Figure 39 (b) which shows a quantization step size of 50 on the same depth image.

**Step 4**: For lowest sub-band, use prediction encoding where the first pixel of the first row is the reference value; then, the difference of the next pixel with its immediate previous same-row neighbor pixel is recorded. Do this until the end of the row. For the next row, the first value is predicted by the difference of the pixel where its immediate previous row's same-column pixel is recorded. Then the difference of the next pixel with its immediate previous same-row neighbor pixel is recorded. Do this until the end of current row. For other sub-bands, use the scan order specified similar as EZW [16] as shown in Figure 40 below. The scan order is pre-specified in the following order: $HL_5$, $LH_5$, $HH_5$, $HL_4$, $LH_4$, $HH_4$, $HL_3$, $LH_3$, $HH_3$, $HL_2$, $LH_2$, $HH_2$, $HL_1$, $LH_1$, $HH_1$ during scanning. The process is shown in Figure 41.

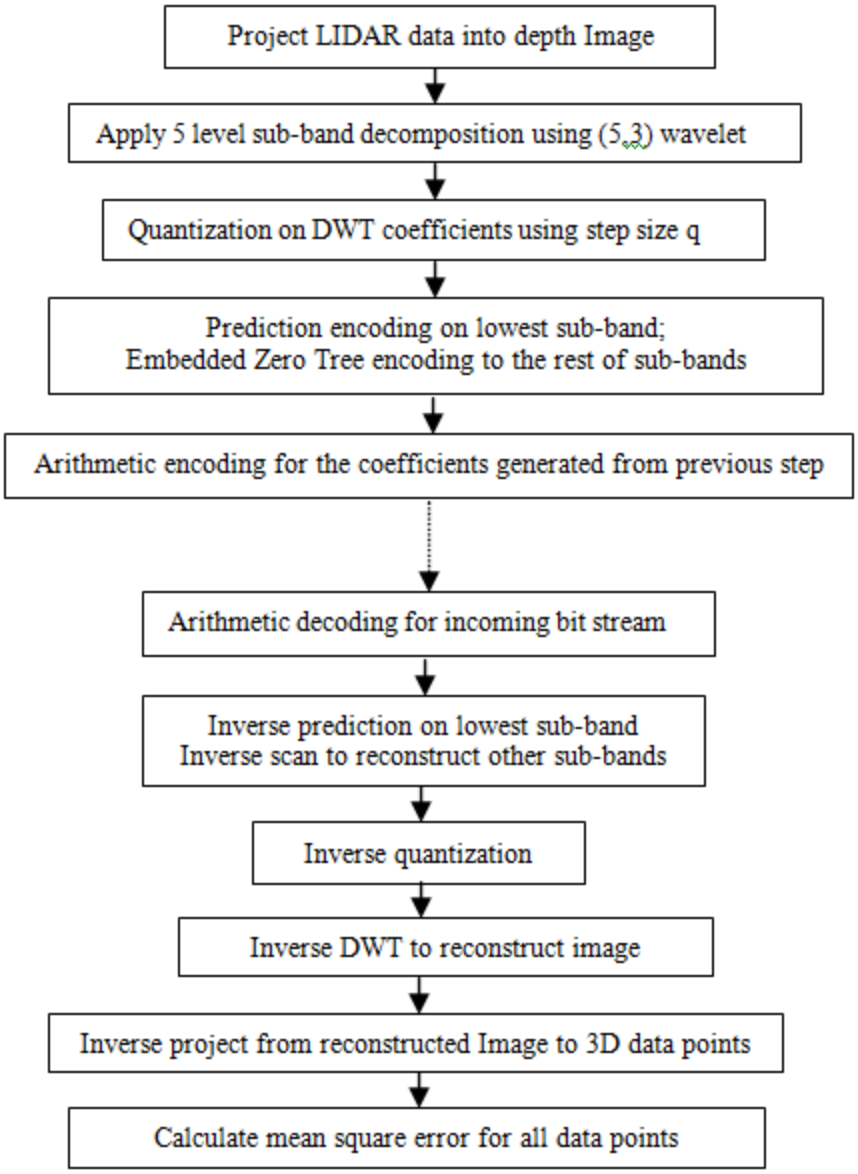
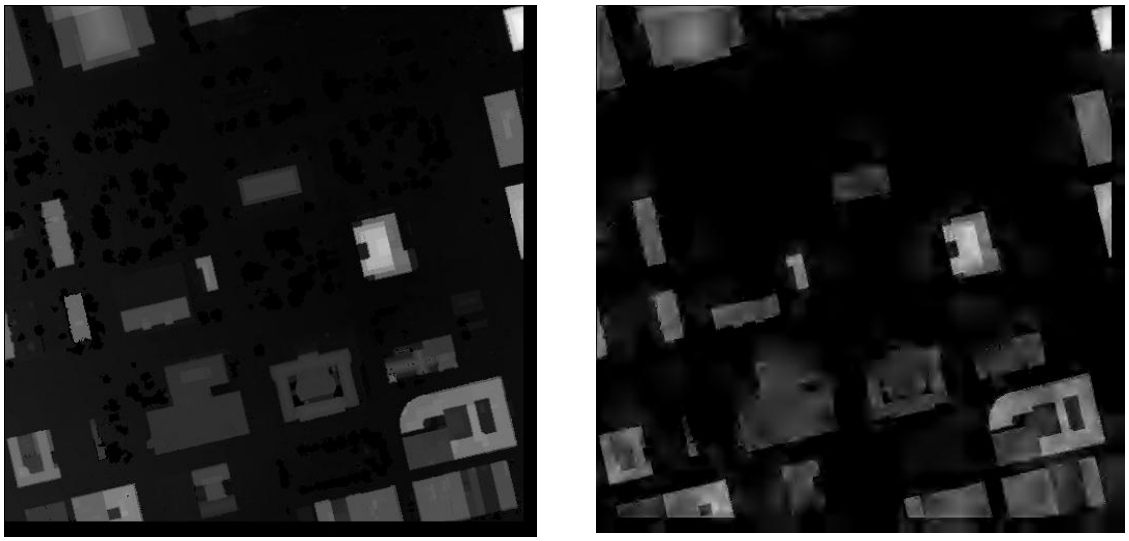



Figure 38 Image based compression block diagram



(a) (b)

Figure 439 Distortion from quantization step illustration

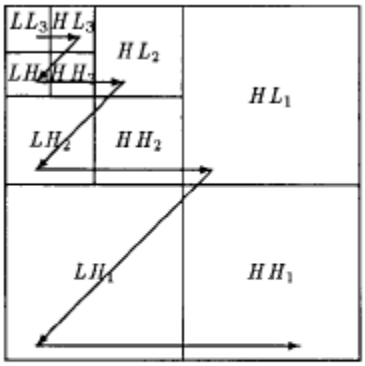



Figure 40 Scan order illustrated when level=3. Note that parents must be scanned before children. Also note that all positions in a given sub-band are scanned before the scan moves to the next sub-band.

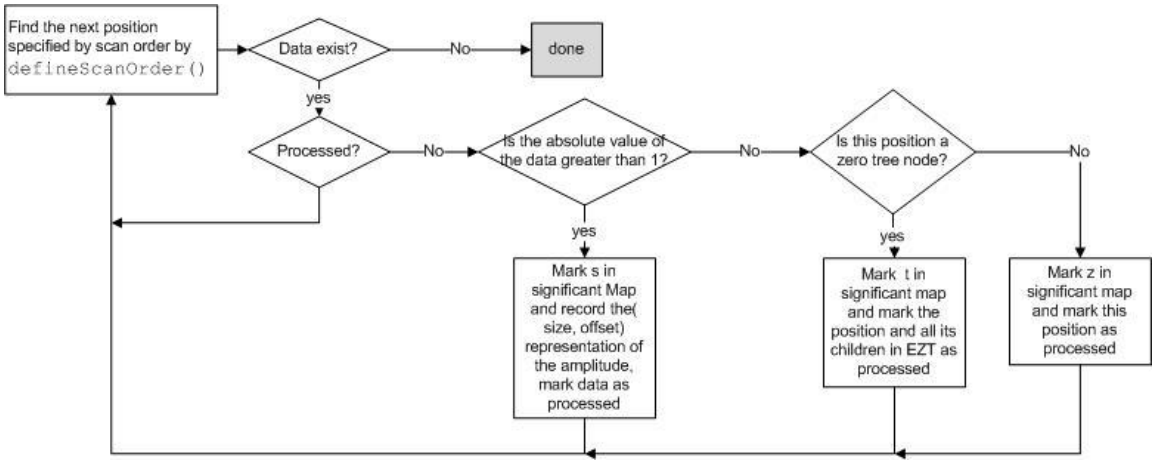



Figure 41. EZT scanning process for sub-band ($HL_5$, $LH_5$, $HH_5$, $HL_4$, $LH_4$, $HH_4$, $HL_3$, $LH_3$, $HH_3$, $HL_2$, $LH_2$, $HH_2$, $HL_1$, $LH_1$, $HH_1$)

**Step 5**: Utilize well known Arithmetic encoder, and the bit rate is calculated in the following way:

```
For each coefficients in LL1, find (Size offset) representation for each amplitude.
AmplituteBitsCt = AmplituteBitsCt + size fileSize = fileInfo.bytes*8 + AmplituteBitsCt
Bitrate = fileSize/ImagePixelSize
```

**Step 6**: The output of step 6 is the significance map and amplitude list.

**Step 7**: Once $LL_5$ sub-band amplitude is parsed, perform inverse prediction to recover original $LL_5$ sub-band coefficients. For other sub-band, according to scan order for each element, if it is marked as s, find its amplitude. If it is a EZT then assign it as well as its children as 0. If it is z, assign its value as 0.

**Step 8**: inverse quantization is calculated by `coefficients = coefficients*stepSize.`

**Step 9**: After reconstructing 2D image, PSNR is calculated by `10*log10(255*255/mean_square_error)`

Figure 42 shows the original depth image of Downtown St Louis and Figure 43 shows the resulting PSNR chart after Step 9.

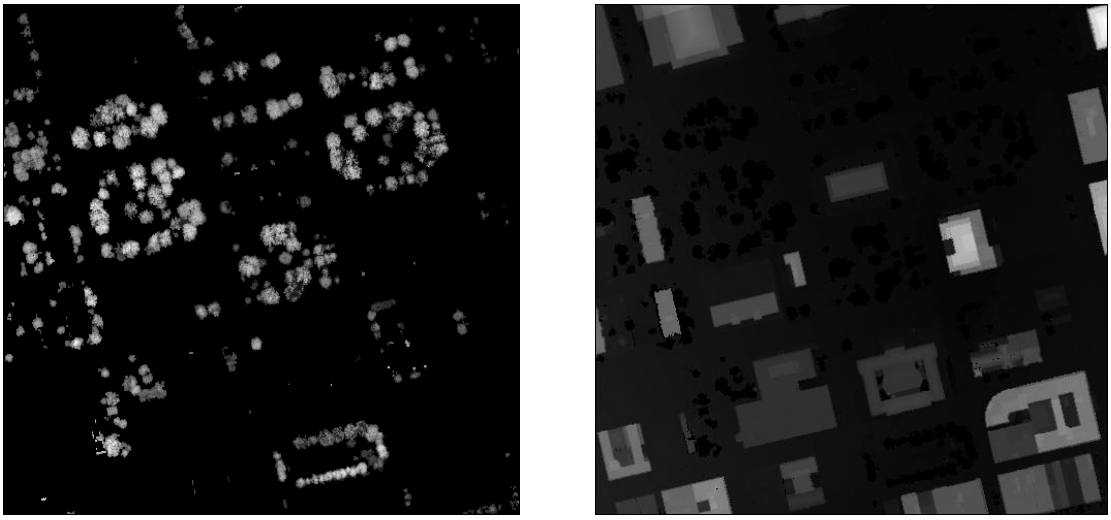


Figure 42. Depth Image for downtown St Louis for trees (left) and buildings/ground (right)

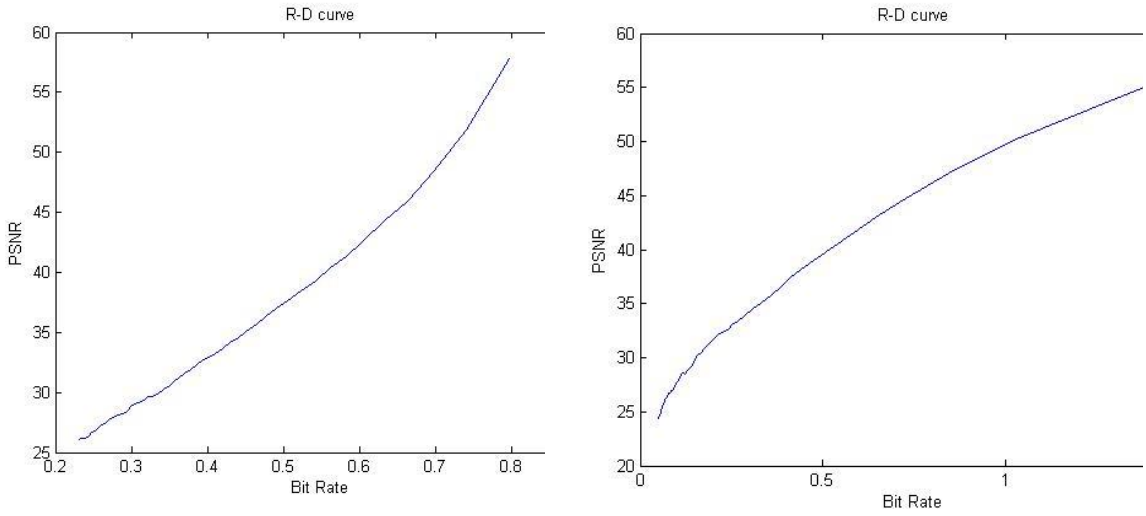



Figure 43. PSNR chart for downtown St Louis for tree (left) and building/ground (right)

**Step 10**: Inverse project from 2D points into 3D space. This is implemented by reading in each original 3D LIDAR points and by finding their mapped 2D x coordinates and y coordinates. The corresponding pixel value will be converted into the depth (z coordinates) of the resulting 3D points.

**Step 11**: The mean square error of each point is calculated by the square sum of difference between the resulting Z coordinates for each point obtained from step 10 and the real Z value of the original point, divided by total points.

The error introduced in this method is mainly from two factors:

(1) Projection error from 3D LIDAR points to 2D Depth Image (Step 1) and after re-constructing 2D image, it re-projected into 3D point's space (Step 10).

(2) The error introduced from traditional image compression method by quantization step size was greater than 1 (Step 3).

Figure 44 shows the RD curve - mean square error calculated after Step 11 vs. bit rate for LIDAR data from downtown St Louis for trees and building/ground. Both have the trend that when bit rates increase, the errors decrease.

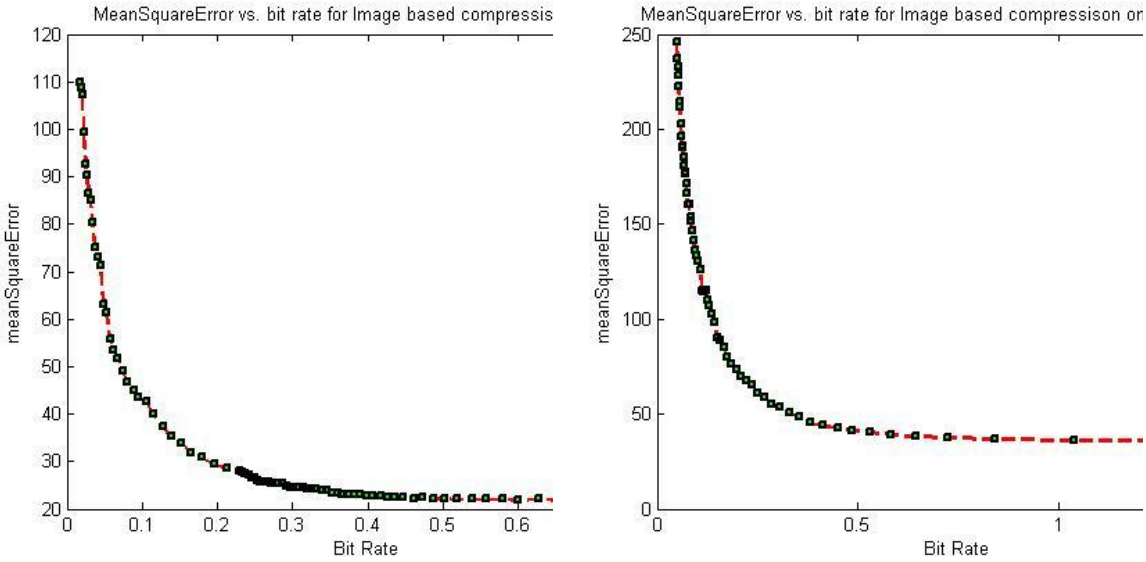



Figure 44 Mean Square Errors vs. bit rates for trees (left) and building/ground (right)

From the above chart, we observed that for building/ground type of 3D LIDAR points, using 500 x 998 depth image seemed to introduce significant error when using image based compression. We regenerated the depth image of 1000 x 998, as shown in Figure 45.

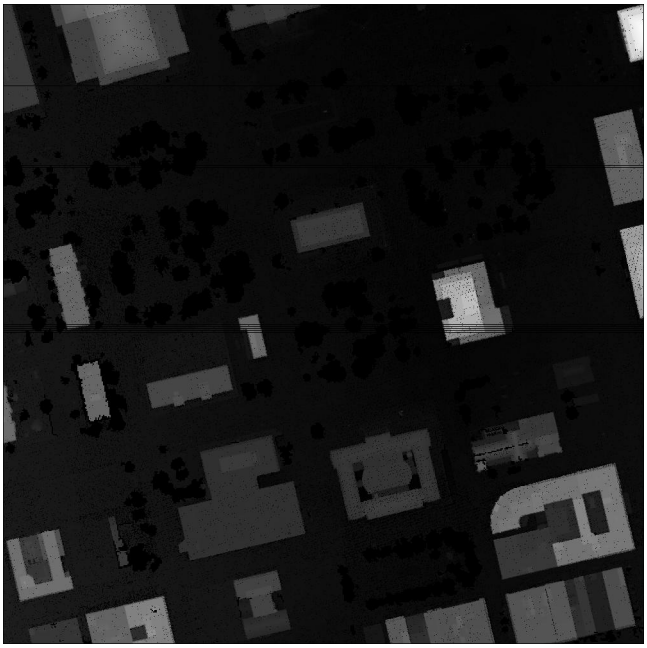


Figure 45. 1000 x 998 depth image for building/ground

The RD curve for 1000 x 998 vs 500 x 499 on building/ground points is shown in Figure 46.

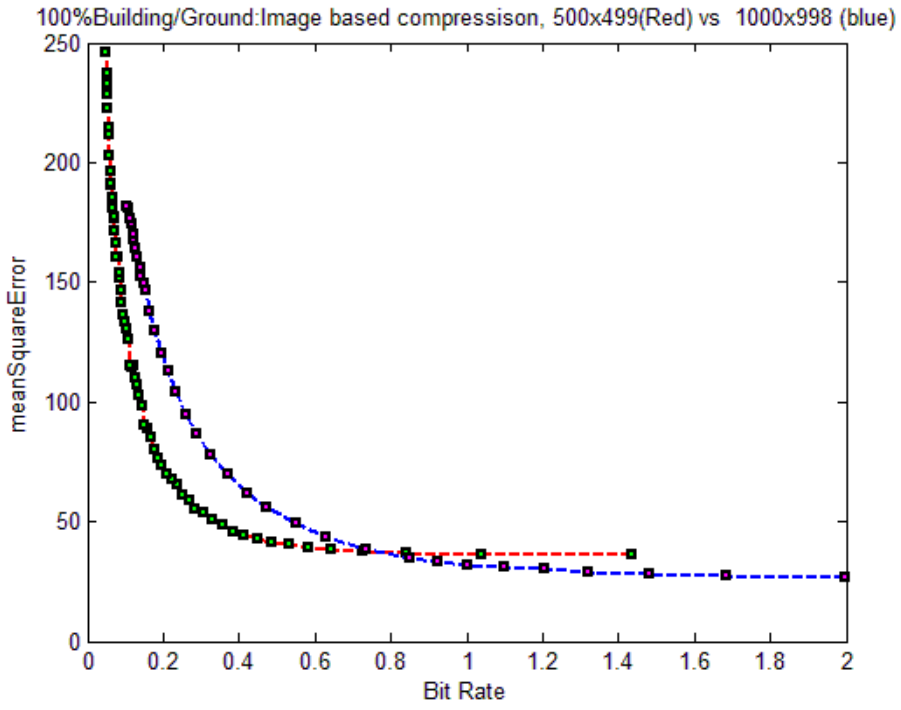


Figure 46. Image based compression for 500 x 499 vs 1000 x 998

Figure 46 shows 1000 x 998 resolution has lower error for high bit rate, but for a bit rate < 0.7, high resolution image compression requires more bits for same error than lower resolution compression.

Figure 46 has lots of empty cells, filled these cells by the neighbor value and then generated another depth image as shown in Figure 47.

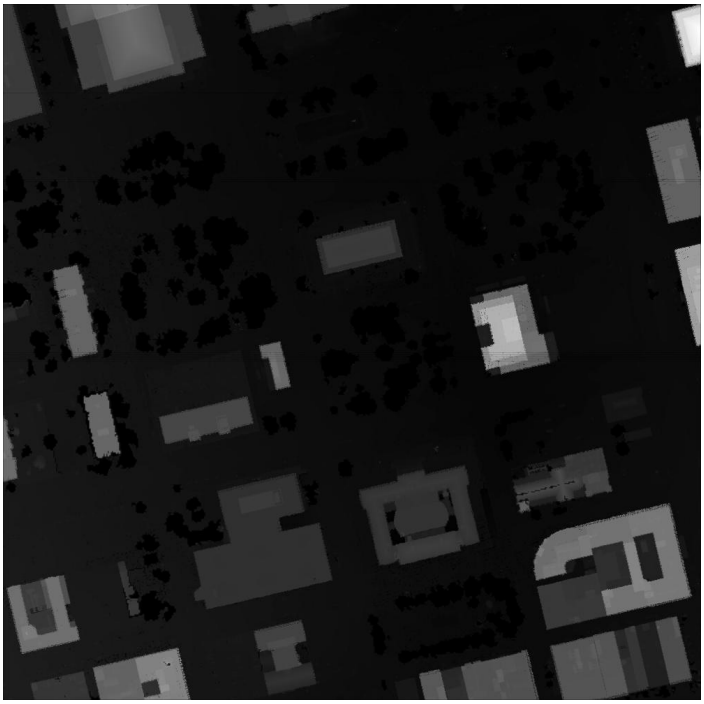


Figure 47. 1000 x 998 depth image for building/ground after “smoothing”

Figure 48 is the RD curve for 1000 x 998 image based compression for empty cells filled by neighbor value (red line) vs. empty cells unfilled.

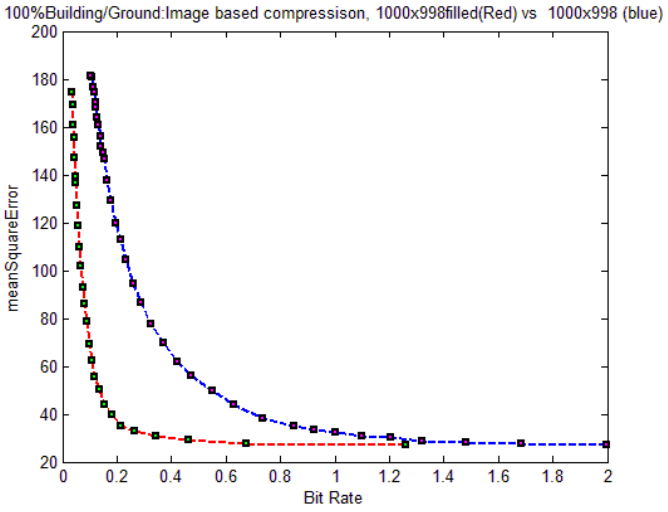



Figure 48 1000x998 image based compression RD curve (empty cells filled vs. empty cells unfilled)

The filled image can be compressed much better than unfilled image because unfilled images have higher frequency components, and filled images have a smoother effect. In addition, JPEG 2000 performs better at lower bit rates. To better utilize this property, we use a filled image as the baseline for 1000 x 998 image based compression.

In our image based compression method, the distortion on x and y are determined when they are quantized on a uniform grid of the (x, y) plane. It is fixed and does not change even on lower bit rates. For high resolution this loss can be very small. Existing work such as [51] discards many significant bits for x and y in lower bit rates. The loss will be worse than our method for x and y encoding. Thus if we apply the same compression method on Z as [51] does, in lower bit rates, our projection method for x and y is better than [51]. This can illustrate that the performance of our image based compression method will be equivalent to or better than [51] in lower bit rates.

Besides compressing LIDAR via projection into 2D and use traditional 2D compression method, we also investigate novel ways to compress the data effectively based on what they are. For example, to better represent the data points by their statistical displacement, if the data are from tree, they can be modeled by a Gaussian function; if the data are from man-made structure such as buildings, it is possible to compress the data by remembering boundary displacement, etc. We call this approach geometry based compression.

### 5.3 Geometry based LIDAR Compression

In this section, we first explain why geometry based LIDAR compression based on LIDAR classification could potentially compress LIDAR data more efficiently than compressing the whole LIDAR data directly, we then give an overview of the proposed system, followed with the details of the proposed scheme and experimental results analysis.

#### 5.3.1 Overview of the Proposed System

Our goal is to achieve high compression ratio while maintaining good reconstruction quality. The fundamental rationale of compression is to explore and remove the redundancy exists in the data. Current active research areas in compressive sensing/sparse coding/low rank matrix theory [57, 58] are based on the following observation: in a high dimensional space, the data are actually concentrated in low dimensional spaces/manifolds. The key challenge is to efficiently and accurately find these low dimensional spaces/manifolds that the data are located in. In our study, the

airborne LIDAR data follows the same trend. In our research, we put data with similar properties into more homogenous groups to allow more efficient exploitation of data redundancy. We divide the airborne LIDAR data into three homogenous groups: tree, building and ground. Then apply different compression schemes for these different types of data which can achieve a much higher compression ratio, instead of directly compress the whole LIDAR data which can be very in-homogenous and can have dramatically different geometric and statistical properties. For example trees and building roofs have totally different geometry and statistics properties.

Our geometry-based compression on building data provides an efficient lossy compression scheme that fits well for scalable "zoom in" requirements. This method can achieve a much higher compression rate in a lossy compression scheme where error rate is not critical. Figure 49 is the high level overview of our proposed geometry based compression system.

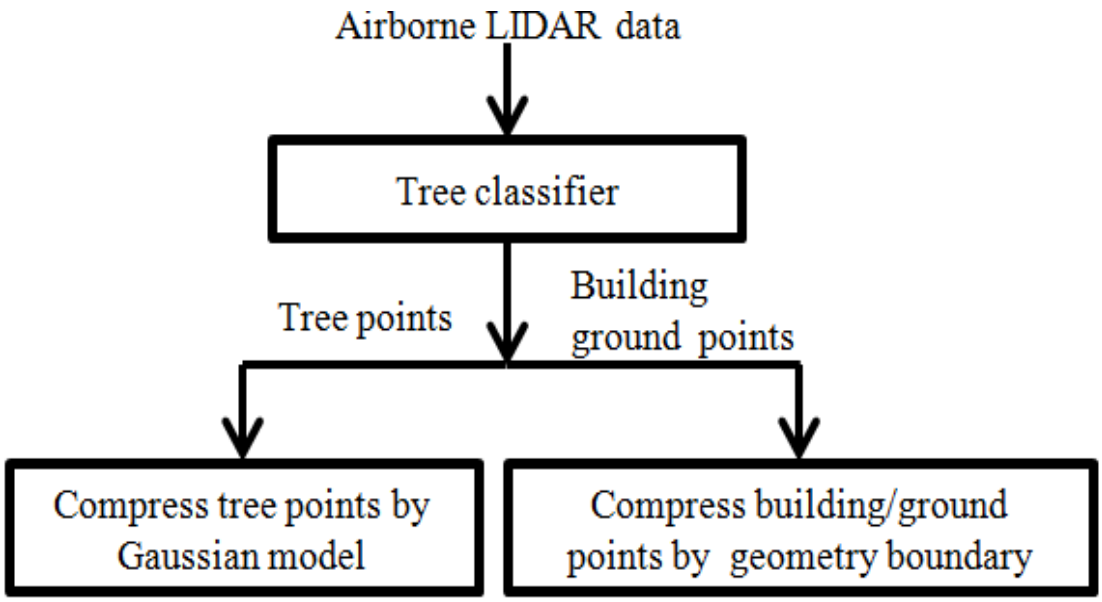


Figure 49 High level process flow of the proposed geometry based system

In this system, the first step is to perform LIDAR classification. To be able to accurately classify LIDAR data is very important for extracting/estimating coherent geometric and statistical information for compression use. As described in previous chapters, we have developed LIDAR classification methods in both supervised and unsupervised way that can accurately classify airborne LIDAR data into three classes: buildings, ground and trees. After the data is classified, the distribution of the data is more predictable. The bell shape of trees and clear boundary geometry characteristics of buildings enable us to model them effectively with simple parameters. This can reduce the redundancy of the original data source significantly, thus making it possible to obtain lower bit rate. To be more specific, for the tree data, it first uses mean shift [3] to segment it into individual trees, then uses Gaussian model to fit each tree; for the buildings and ground, it uses a height based cluster to extract the ground and building part, then finds the building boundary, and then simplifies the boundary by line fitting. The following sections describe the algorithms in detail.

### 5.3.2 Algorithm for Tree Points Compression

We model the trees by their approximation to bell shaped Gaussian distribution. If we could find points that belong to the same tree, then these points can be approximated by Gaussians model. This way, instead of store each 3D point, we can just store an index map of all trees and each tree's Gaussian parameter. Here we apply Mean shift method to cluster each tree and compressed tree by its statistics measured by the Gaussian model. The encoder tasks included the following steps:

1. Project 3D tree points into a 2D depth Image.

2. Apply mean Shift algorithm to segment 2D depth Image
3. Record each region location
4. Use the least square error to fit each region into 2D Gaussian distribution
5. Entropy encode the label map and Gaussian parameter

The decoder included the following steps:

1. Decode the label map and Gaussian parameter
2. Use the label map and Gaussian parameter to reconstruct 2D depth image
3. Inverse project from reconstructed 2D image to 3D points

Calculate mean square error for all data points

Details of some steps are listed here:

**Step 2**: Apply mean shift algorithm to segment 2D depth Image.

Mean Shift is a powerful general purpose technique for clustering scattered data. Instead of assuming a fixed number of clusters, as is common with other clustering methods, e.g., *K*-means, Mean Shift extracts the modes of the density function [9], which allowed us to work with an arbitrary set of $n$ points $\{x_1, \ldots, x_n\}$ in the $d$-dimensional Euclidean space $R^d$. The multivariate kernel density estimate obtained with kernel $K(x)$ and window radius $h$, computed in the point $x$ is defined as:

$$\hat{f}(x) = \frac{1}{nh^d} \sum_{i=1}^{n} K(\frac{x - h_i}{h})$$

where $K(x)$ is the spherically symmetric kernel function satisfying

$$K(x) \geq 0, \qquad \int_{R^d} K(x)dx = 1$$

and $h$ is a smoothing parameter called the bandwidth. Since we are interested in subdividing scattered data into a set of clusters, we consider the points where $\hat{f}(x)$ has local maxima as centers of the clusters. The simplest method to find the local maxima of $\hat{f}(x)$ is to compute the gradient of $\hat{f}(x)$ and use a hill-climbing process to map each input point to its local maxima (i.e., mode) defined by $\hat{f}(x)$. This process can be shown in Figure 50.

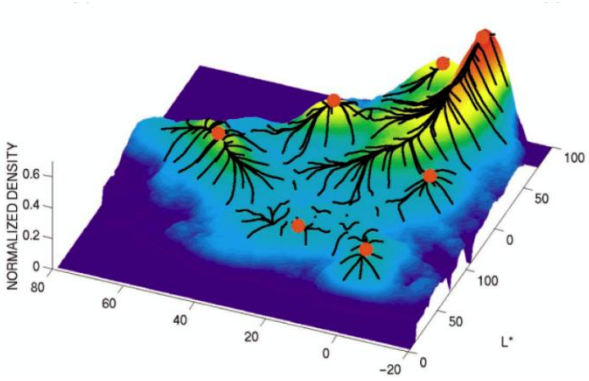



Figure 50 Mean shift process and mode illustration

These resulting modes can then be used to select cluster shapes using basins of attraction, and can have very nontrivial shapes--unlike *K*-means clustering where points are simply assigned to the nearest cluster center. The single bandwidth parameter *h*, allows the number of clusters to be chosen in terms of a length scale in the input point space.

The general Mean Shift clustering procedure consists of the following two steps:

1. Initialize: $y_0 = x$;

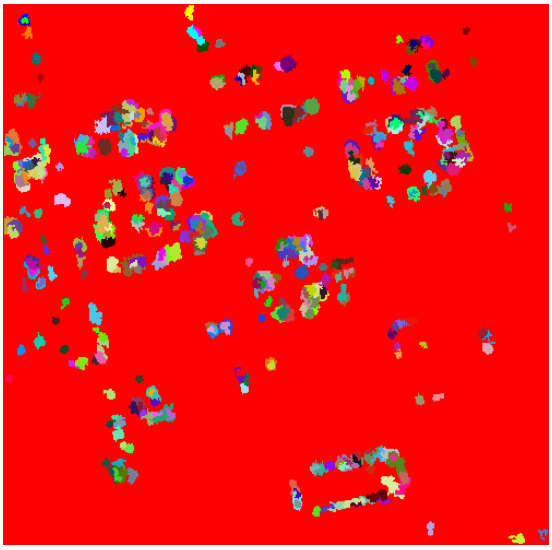
2. Update by hill climbing: $y_{i+1} = y_i + m(y_i)$ until convergence.

$m(x)$ is the Mean Shift vector and is given by

$$m(x) = \frac{\sum_{i=1}^{n} x_i g(\| \frac{x - x_i}{h} \|^2)}{\sum_{i=1}^{n} g(\| \frac{x - x_i}{h} \|^2)} - x,$$

It is the difference between the weighted mean, with $g(x)$ as the gradient kernel function derived from kernel function $K(x)$ as weights, and $x$, the center of the kernel (window).

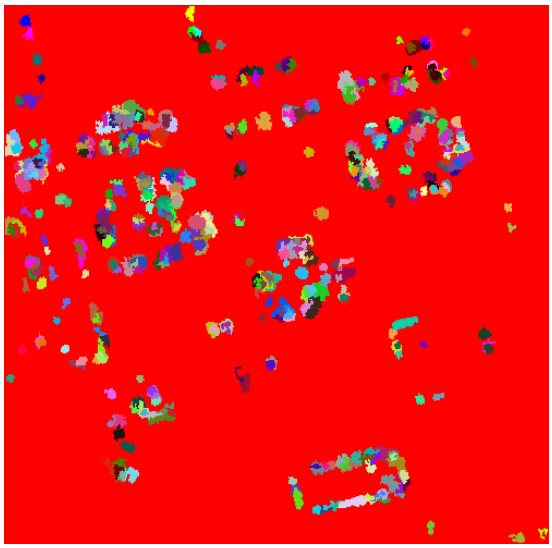
The magnitude of the mean shift vector converges to zero; the mean shift iterations satisfy the conditions that the trajectories of such gradient method are attracted by local maxima if they are unique (within a small neighborhood) stationary points. Once $y_j$ gets sufficiently close to a mode, it converges to it. Once mean shift is in the gradient direction of the density estimate, successive iterations will converge to a local maxima of the density, i.e., a stationary point: m(x) = x. It is a steepest-ascent like procedure with variable size steps that lead to fast convergence to mode. Here, one must represent each pixel x as spatial location $x^s$ and range $x^r$ (color, intensity), then the algorithm look for modes in the joint spatial-range space. Once converged to a mode, all the points that are converted to the same mode will be labeled as the same segmentation. Here different spatial/range thresholds will generate different results. Figure 48 illustrates some mean shift results for different spatial/range thresholds.



SpatialBandwidth = 9;

RangeBandwidth = 12.5;

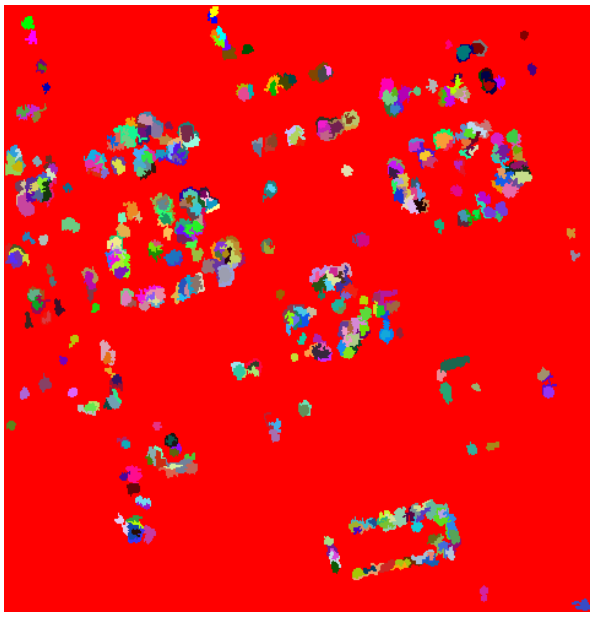
MinimumRegionArea = 30



SpatialBandwidth = 9;

RangeBandwidth = 28.5;

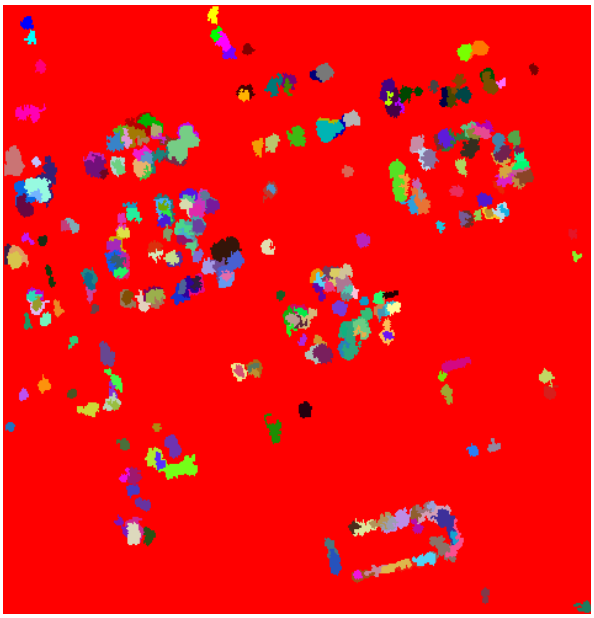
MinimumRegionArea = 30



SpatialBandwidth = 38;

RangeBandwidth = 28.5;

MinimumRegionArea = 30



SpatialBandwidth = 16;

RangeBandwidth = 46.5;

MinimumRegionArea = 40

Figure 51: Different segmentations result based on different thresholds

**Step 3**: Record each region location. From Step 2, we obtain information about modes and all pixel locations that converged to the same mode. The information will be used in Step 4.

**Step 4:** Use non liner least square fitting to fit each region by 2D Gaussian model. The parameter of the 2D Gaussian model and each label will be compressed in Step 5.

**Step 5:** The text file can be compressed by either Zip or arithmetic encoding. We chose the size with the smaller value for the two methods.

**Step 6:** Use the label map and Gaussian parameter to reconstruct 2D depth image. First find the label by the pixel's location, then its Gaussian parameter can be obtained. Thus, the depth of this pixel can be computed by Gaussian model. Figure 52 shows the original tree depth image (left) and reconstructed tree depth image (right) based on Gaussian model.

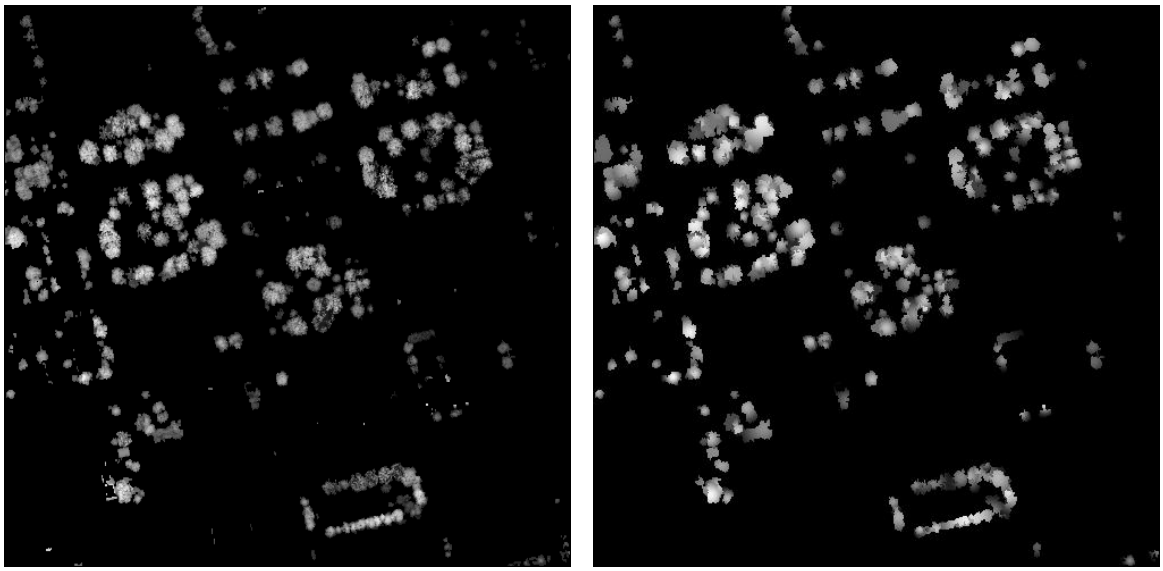


Figure 52: These images compare the original tree depth image (left) and reconstructed tree depth image based on Gaussian model (right).

The result of the geometry-based tree compression method (after Step 8) can be illustrated by Rate-Distortion curve as shown in Figure 53.

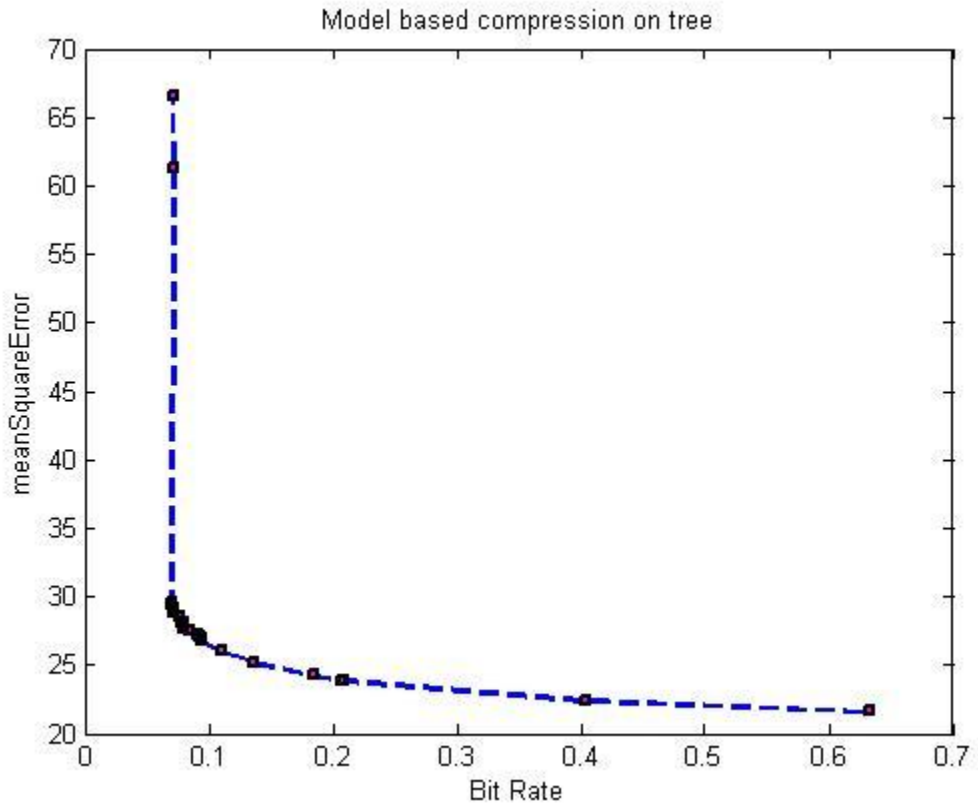



Figure 53 Rate-distortion curve for geometry-based tree compression

### 5.3.3 Algorithm for Building/Ground Points Compression

The simplest form of manmade structures such as building is its geometry boundary. To compress building and ground 3D data, we would like to obtain each building's boundary and store those boundaries instead of individual points, this will reduce the data complexity and redundancy significantly. We cluster building blocks by height based cluster, then extract the boundary of each building part which share the same height. By control cluster threshold, we can obtain different level of detail as needed; this will

satisfy the requirement such as quickly "scrolling through" a large data set at an "overview" level and then, when a feature of interest is found, "zoom in" to that region and see more details. Encoding on building and ground points includes the following steps:

1. Project 3D building/ground points into 2D depth Image.
2. Apply height-based clustering to cluster buildings.
3. Use least square fitting to fit each cluster of pixels into a plane.
4. Find boundary pixel based on each building cluster
5. Compress the boundary pixel location and plane parameter by WinZip or arithmetic encoder.

Decoding on building and ground points includes the following steps:

1. Unzip/arithmetic decode the compressed boundary pixel location and plane parameter
2. Use the boundary pixels and plane parameter to refit to 3D points
3. Calculate mean square error for all data points

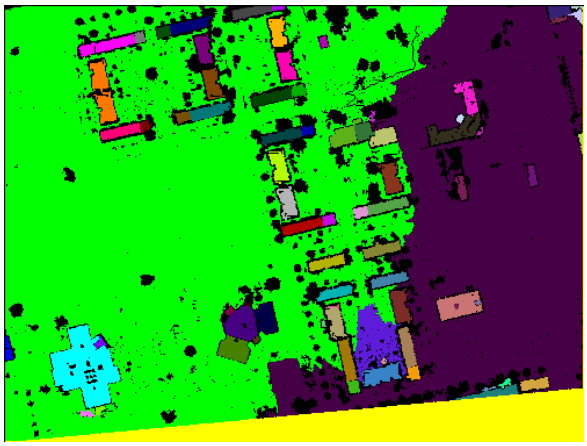
Details of each step are described in the following.

**Step 2**: Apply height-based clustering to cluster buildings

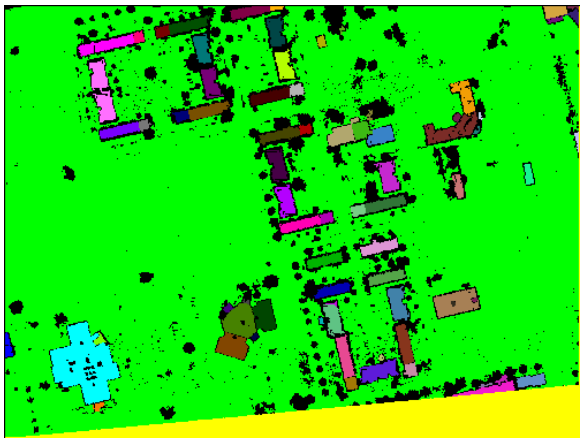
Ground and building points are processed by height-based cluster. Use algorithm similar to density-based clustering and perform cluster algorithm based on 2D depth image. Here region growing criteria is mainly the height criteria: (1) if the difference between

neighbor's height and the pixel's height is within a threshold, and (2) the difference between neighbor's height and the center's height ( i.e., the average height of this cluster) is within some threshold, then this neighbor will join the same cluster with this pixel.

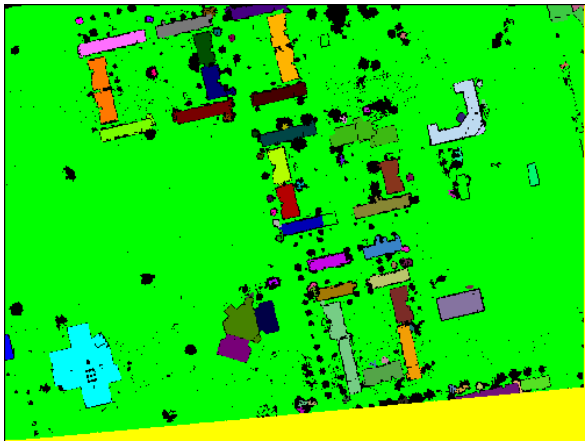
Figure 54 illustrate some results of the cluster method. The ground will be the cluster that will have the most pixel counts.



Threshold with neighbor: 0.5, threshold with center of this cluster: 3, minimum cluster region: 15



Threshold with neighbor: 0.5, threshold with center of this cluster: 7, minimum cluster region: 15

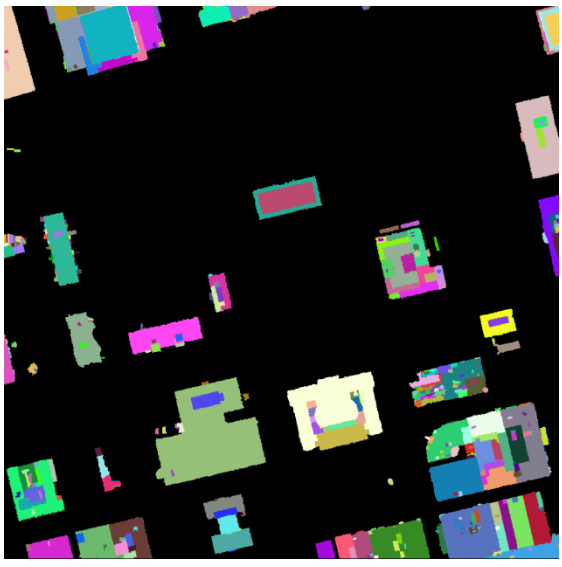


Threshold with neighbor: 0.7, threshold with center of this cluster: 4.5, minimum cluster region: 15

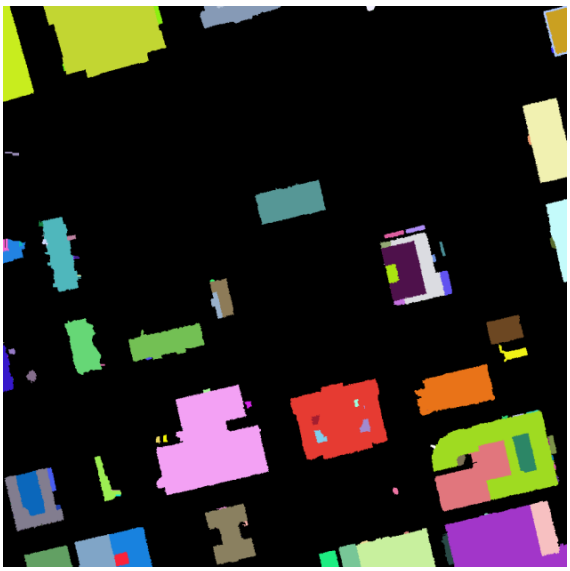
Threshold with neighbor: 1, threshold with center of this cluster: 7, minimum cluster region: 15

Figure 54 Height-based clustering results for different threshold

Height-based clusters can extract ground. A building block can be obtained by finding connected components of non-ground cluster, so one building block can be composed of smaller buildings as long as they are connected together. After extracting each building block, we perform height-based cluster again on this building block to find different building parts which have the same height and thus belong to the same cluster. Figures 55(a) and (b) show result after performing height-based cluster on each local building with different thresholds.



(a) Height based cluster with smaller threshold



(b) Height based cluster with bigger threshold

Figure 55 Height based cluster result on each local building with different threshold.

In Figure 55, data with same color share the same cluster. When comparing (a) and (b), we observe (a) has more refined clusters and (b) has coarse clusters. The information is stored in hierarchy structure: We store each big building block first; then, inside of each block, the encoder stores each individual building information. This method supports the multi-structure of the data inherently, and it fits the aforementioned geospatial imagery requirement that needs to store their data at multiple resolution levels or scales and be able to "zoom in" easily.

**Step 3**. Use least square fitting to fit each cluster pixels into a plane

A plane can be defined as $AX + BY + CZ + D = 0$. For the building part that has the same cluster, the roof plane parameter A, B, C, D is fitted by least square fitting.

**Step 4.** Find boundary pixel based on each building cluster.

The boundary includes points whose neighbor's cluster is different. Boundary points are extracted and recorded in this step. In Figure 55, boundary file size from (a) is larger than boundary file size from (b). Since (b) is coarse, we expect, (b) to have a smaller bit rate from output of encoder but bigger distortion from output of decoder.

Figure 56 is an illustration of the boundary points

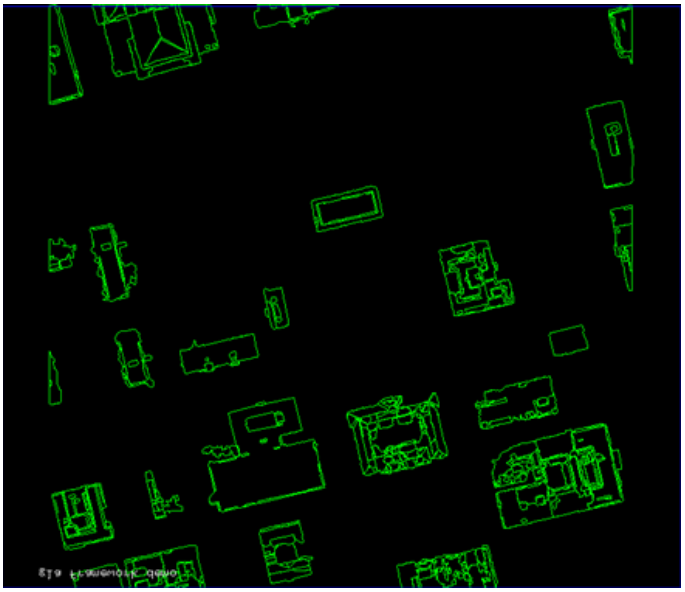


Figure 56  Building boundary points illustration

**Step7**. Use the boundary pixels and plane parameter to refit to 3D points.

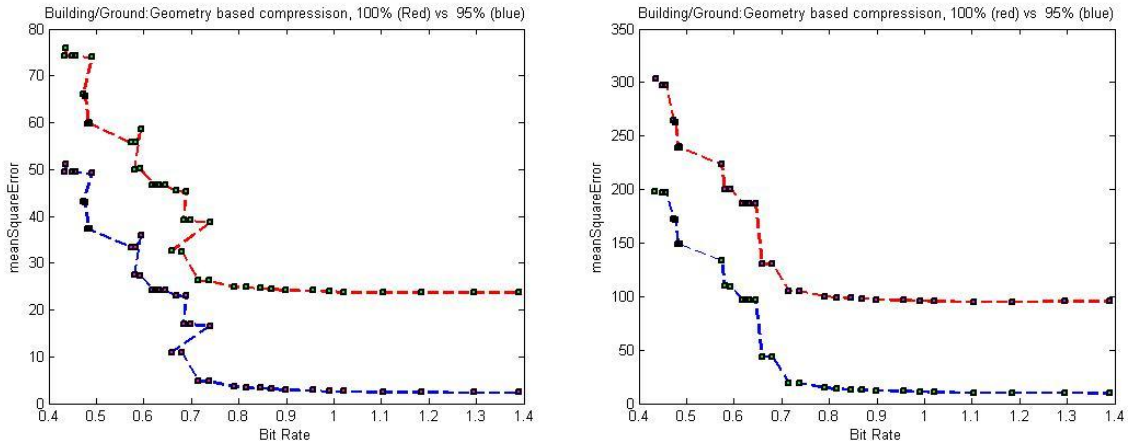
Building boundary pixels can be overlapped. For example, the building block in Figure 57 has two clusters and thus will have two sets of boundary points. The inner boundary points are surrounded by the outer boundary.

Figure 57 Building block with overlapped boundaries

In order to correctly recover cluster information, we first find all the points inside of each boundary, then sort by its point count in descent order within the current building block. This allows all the points included by the outside boundary to be assigned first. The points inside the inner boundary are updated by the new cluster information, which

guarantees a cluster for each building part in blocks with overlapped boundary are assigned correctly. Then each point's height can be calculated by plane parameter.

It is worth mentioning here is that when we refit to 3D points, the algorithm has the capability to find/remove noise if needed. As mentioned earlier, airborne LIDAR data are from intensity peaks of its waveform which correspond to points that were hit by the laser and then reflected significant portions back to the sensors on the plane. There can be multiple peaks because the laser may hit several surfaces such as wires or antennas, branches, leaves, or even birds in flight before reaching the ground. This data collection process causes airborne LIDAR data to contain noise. Due to the nature of manmade buildings, LIDAR data for building should be close to each other within a near neighborhood. When refit to 3D points, if some individual point from the original LIDAR data is suddenly very far away (in height) from its close neighbor. This could indicate signal collection noise, and the above mentioned method can efficiently detect these outliers. We perform experiment by filtering out 5% of noise data with the most fitting error, then apply geometry-based building/ground compression on the remaining 95% LIDAR points vs. 100% original LIDAR points. Figure 58 illustrates the rate-distortion curve for geometry-based building/ground methods on 100% vs. 95% of the LIDAR points. Here we use an internal parameter to control height based cluster to obtain different building boundary file sizes and fitting error, this parameter (threshold, as illustrated in Figure 55) is not strictly linear with respect to the file size ( the file size may oscillate within a small range of the parameter variation). It is possible to observe local oscillation in RD curve due to this nonlinearity as shown in Figure 58(a). Figure 58 (b) is obtained when we find the minimum mean square error for the same bit rate.




(a) (b)

Figure 58 Rate distortion curve for geometry based compression on 100% LIDAR points vs. 95% LIDAR points(500x499 resolution)

From Figure 58, we observe that compression on 95% can obtain much less error than 100% case. We also compare the traditional image based compression on 100% LIDAR points vs. 95% LIDAR points. Figure 59 illustrate the rate-distortion curve comparison for traditional image-based compression.

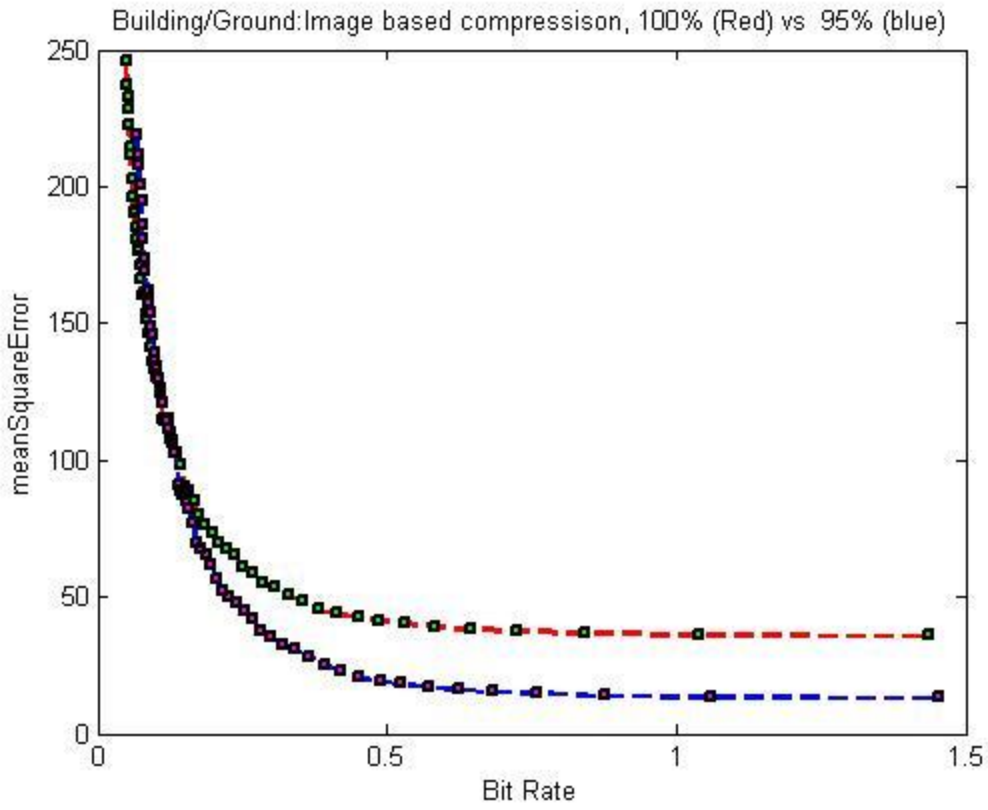



Figure 59 Rate distortion curve for image based compression on 100% LIDAR points vs. 95% LIDAR points

Figure 60 is the RD curve for filled 1000 x 998 image compression vs. geometry-based compression. When bit rates are large enough (infinity), theoretically, geometry-based compression and image-based compression should converge, but here we could still observe some small gap due to the implementation of image-based compression for JPEG2000. The z values were quantized into a fixed grid, and prediction encoding and embedded zero tree representation required the coefficients be rounded to nearest integer. However, this is not required in geometry-based compression, so the difference was mainly due to the fact that floating points lose precision on image-based compression.

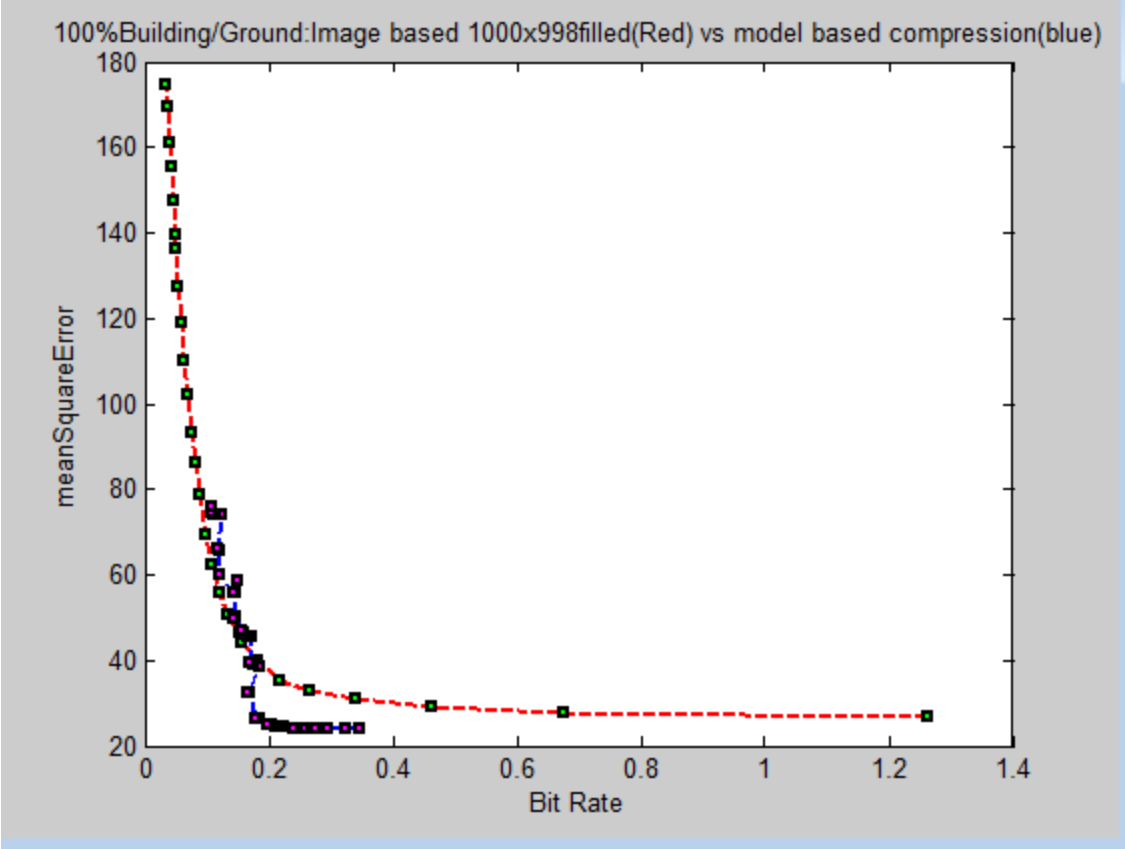



Figure 60. Image based compression vs. geometry based compression on building/ground (resolution 1000 x 998)

Geometry based compression file size is determined by boundary points location. If the building's boundaries are composed of straight lines (in most cases this is the real-world situation), instead of recording all the boundary points, say 100 zigzag points, one needs only to record the two end points to reduce the file size significantly without introducing much error. The following process can achieve this.

### 5.3.4. Boundary Line Extraction

If the building contour is not all straight lines, the contour boundary can be approximated by concatenate of segment of lines , in such a way that more segments are needed if

require less approximation error and less line segments is needed if more approximation error is allowed. To implement this, Boundary lines were extracted by the following method. Line points were accumulated along the line until at certain point the maximum deviation for each point to the line was greater than a predefined threshold. A line can be defined by its begin point (x1,y1) and end points (x2,y2) as

$$x*(y1-y2) + y*(x2-x1) + y2*x1 - y1*x2 = 0.$$

For point(x,y), the deviation from the line was calculated as

$$| x*(y1-y2) + y*(x2-x1) + y2*x1 - y1*x2 |.$$

A line segment stops and a new segment starts from a point when the maximum deviation for the points along the line is greater than allowable tolerance. This way, we can control the error by specifying different tolerance values. The smaller the value, the less error and the more line segments are generated. Figure 61 is the flow chart of the algorithm.

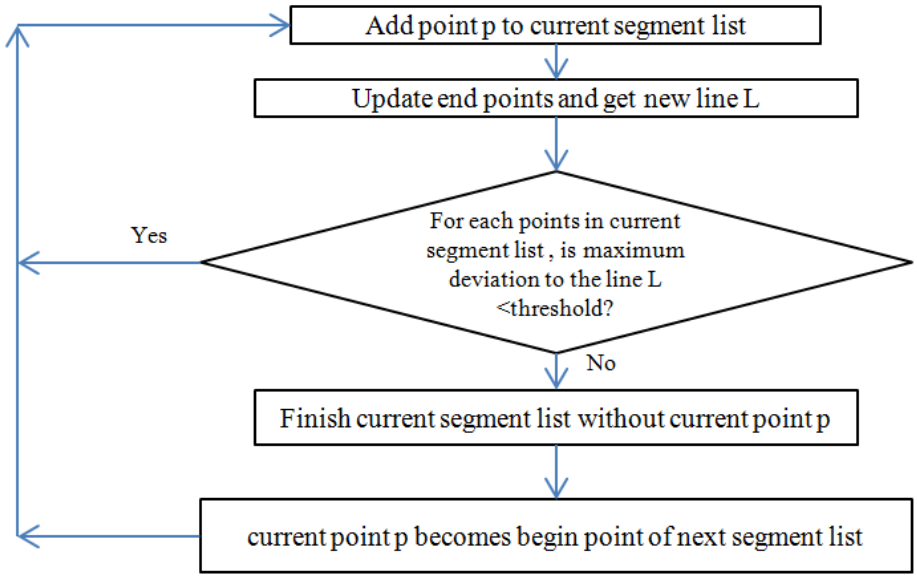



Figure 61. Extract line from contour points process.

#### 5.3.4.1 Line Extraction Experimental Results

To illustrate line extraction and the experimental results encountered, we use the building in Figure 62's red circle as an example.

Figure 62 Building line extract example

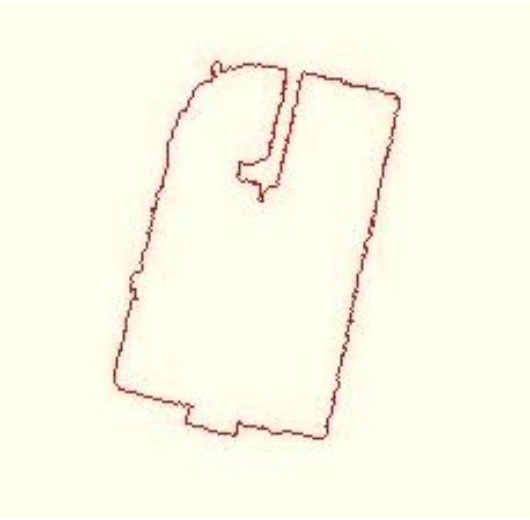
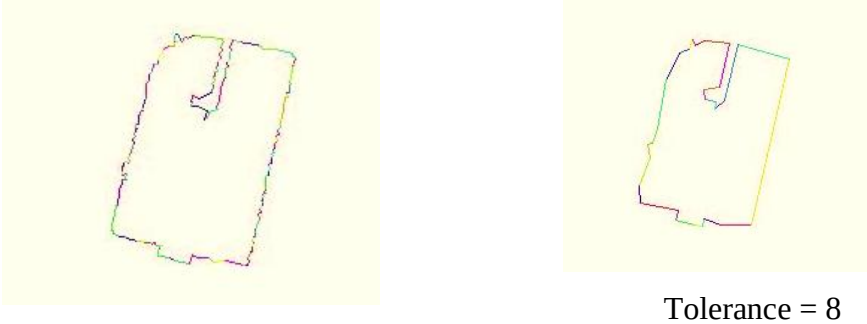


Figure 63 contour of the building

The original contour of the building is shown in Figure 63.

The experiment results for different thresholds are shown in Figure 64. Points that fall on the same line segment will have the same color. Here we can see that bigger tolerance has longer lines and a smaller number of line segments generated.



Tolerance = 2 Tolerance = 8

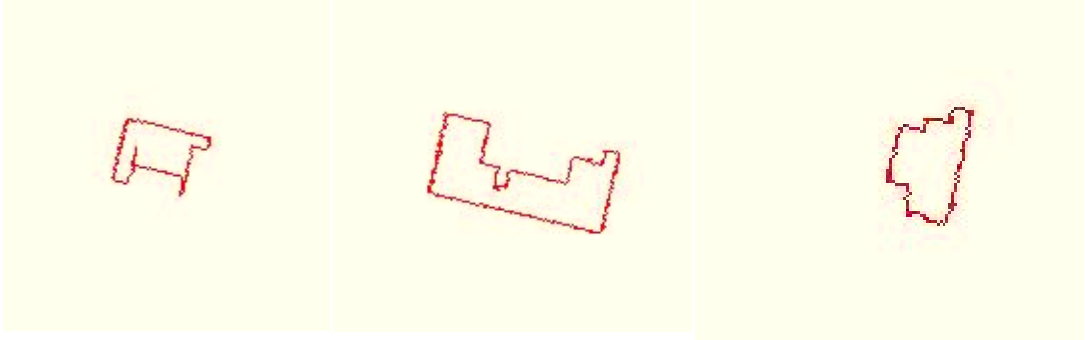
Figure 64 Line segment generated by different tolerance value

Figure 65 shows the mask of all the points in white that falls inside of the line segments for different tolerance.

Tolerance = 2 Tolerance = 8

Figure 65 points shown in white are inside of the generated line segment.

Figure 66. The illustrations below represent three different original boundary contour points for building.



a b c

Figure 66. These are portions of the building's roofing class boundary contour.

Figure 67 is line extraction results for Figure 66(a) with different thresholds.

Tolerance = 2 Tolerance = 8

Figure 67 Line segment generated by different tolerance value

Figure 68 shows all the points inside the line segments boundary for Figure 66(a).

Tolerance = 2 Tolerance = 8

Figure 68 Points shown in white are inside of the generated line segment

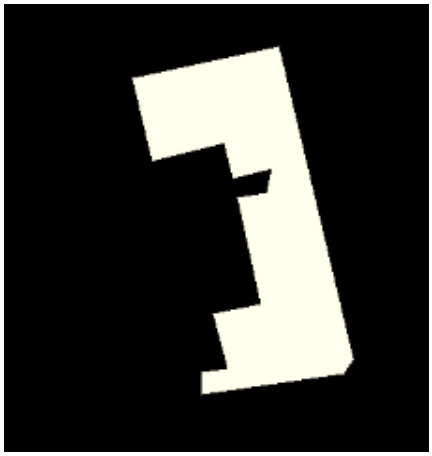
Figure 69 shows line extraction results for Figure 66(b) with different thresholds.

Tolerance = 2 Tolerance = 8

Figure 69 Line segment generated by different tolerance value

Figure 70 shows points inside the line segments boundary for Figure 66(b).

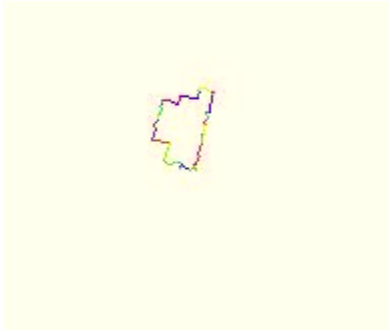
Tolerance = 2



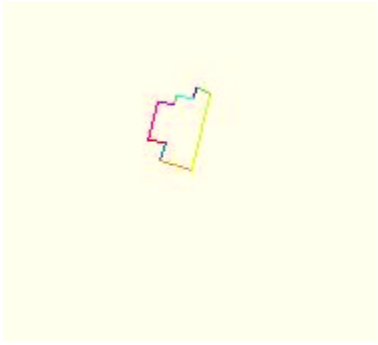
Tolerance = 8

Figure 70 points shown in white are inside of the generated line segment

Figure 71 is line extraction results for Figure 66(c) with different thresholds.



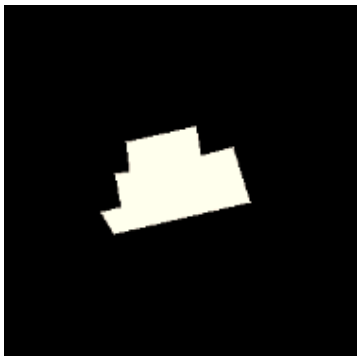
Tolerance = 2



Tolerance = 8

Figure 71  Line segment generated by different tolerance value

Figure 72 shows points inside the line segment boundaries for Figure 66(c).



Tolerance = 2 Tolerance = 8

Figure 72 points shown in white are inside of the generated line segment

Figure 73 is the whole building roof's different class boundary contours. In this case, boundary points for same class share the same color.

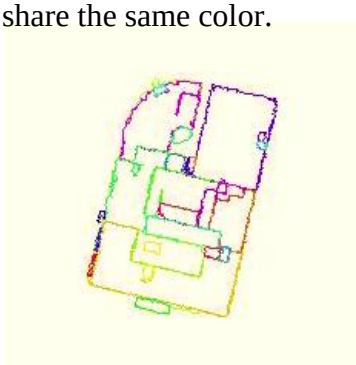


Figure 73 Whole building roof's different class boundary contours

Figure 74 shows line segments where tolerance = 2, and the same line segments share the same color. Figure 75 shows line segments where tolerance = 8. We observed that when tolerance = 8, it results in less lines.

Figure 74 Line segments (Tolerance = 2)

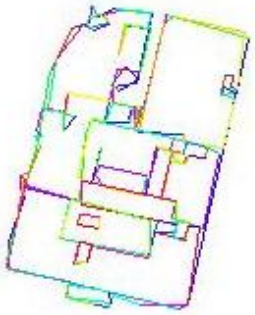


Figure 75 Line segments (Tolerance = 8)

For this building block, the original boundary file size is 175 k, after line extraction where tolerance = 2, file size is 11k where tolerance = 8, and file size is 1k.

### 5.3.5. Experimental Results

Figure 76 shows a rate-distortion curve comparison on tree points between image-based compressions vs. geometry-based compression.

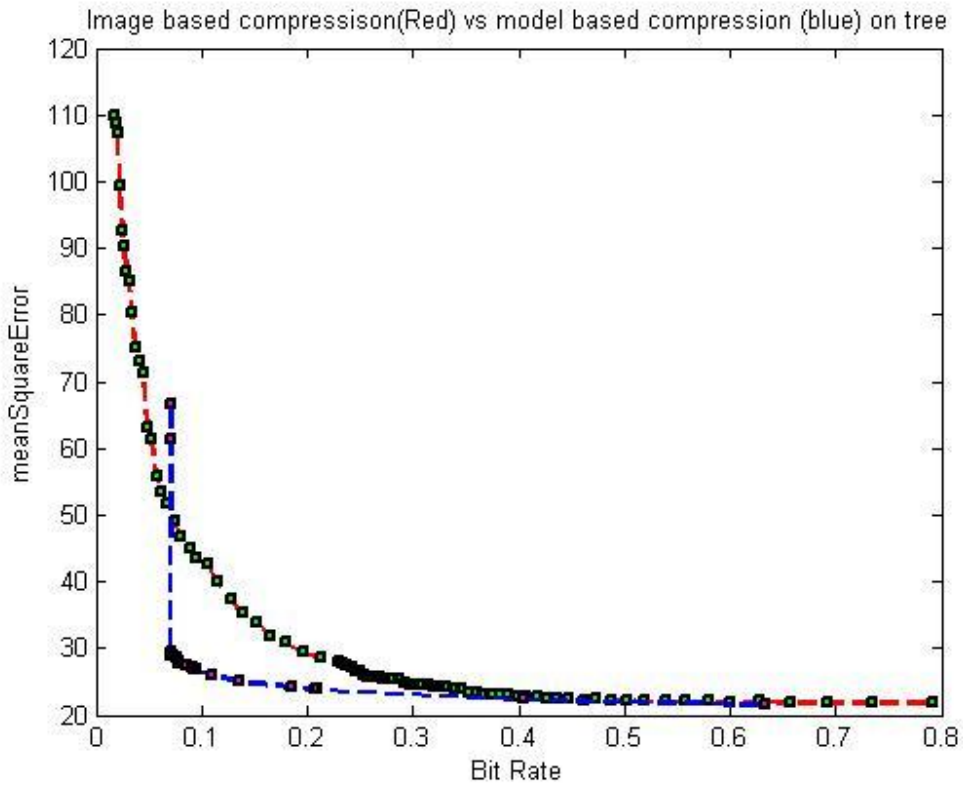



Figure 76: Tree compression by image based compression (red line) vs. geometry based tree compression (blue line)

Figure 76 shows that when bit rate is large, image-based tree compression is comparable to geometry-based tree compression, but when bit rate is less than 0.4 bits, geometry based-tree compression performs better than image-based tree compression.

Figure 60 shows a rate-distortion curve comparison on 100% building/ground points between image-based compressions vs. geometry-based compression. After using the line fitting method to further process boundary file, we use an arithmetic encoder to

encode the resulting fitted line boundary file and recalculate error. We obtain the following RD curve shown in Figure 77:

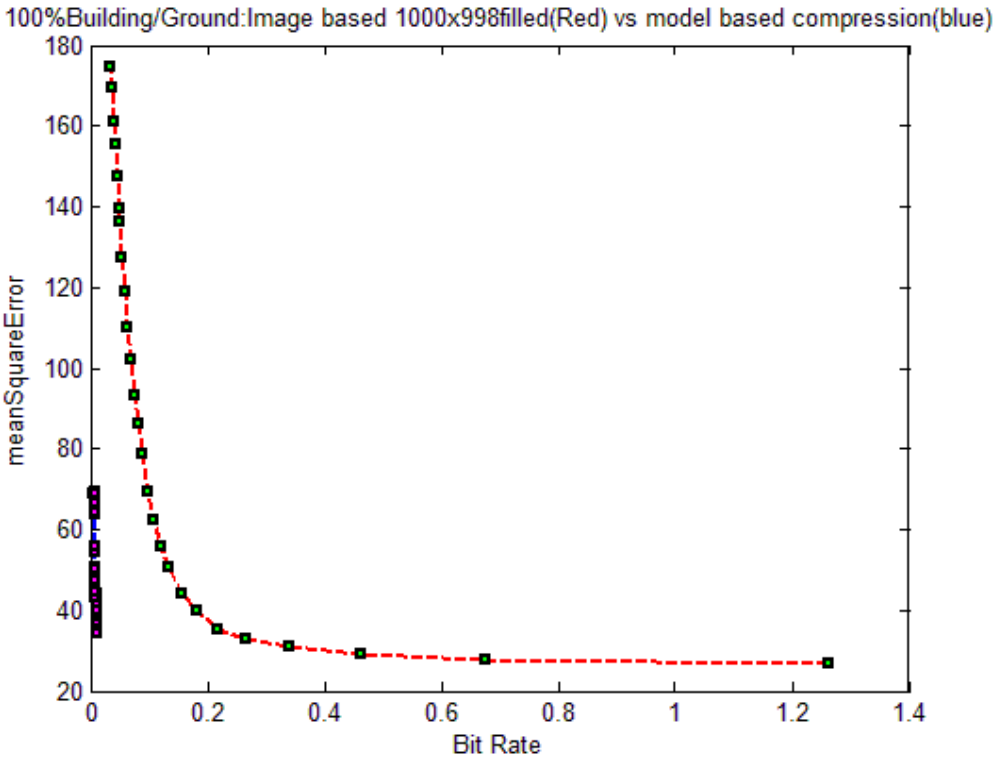



Figure 77. RD curve for image-based compression (in red line) vs. geometry-based compression (blue line) on building/ground points.

Figure 77 illustrates the line fitting method is very effective to reduce file size while not introducing significant error. The RD curve indicates that geometry based compression can achieve a much better compression rate in a lower bit rate range with the same error as image based compression. The largest file size in Figure 77 after geometry-based compression was 11210 bytes, while the original LIDAR text data is 79656000 bytes, and its zipped version is 12182000 bytes. Our proposed method obtains a compression rate of 1.4e-6.9.

## 5.4 Conclusion

We propose two new compression schemes- an image based compression scheme and a geometry based compression scheme. Compared to the existing LIDAR lossy compression methods, our image based compression scheme performs better in lower bit rates. Compared to [51], their encoding on z value is equivalent to our image based compression method, in which we obtain the depth image and then perform JPEG2000 on the depth image; their RD curve on z will be similar to our RD curve on z. The loss of our x and y will be less than some of the existing methods such as [51] while loss on z will be equivalent to [51]. So in summary, in lower bit rates, our image based compression method based on JPEG2000 is at least equivalent or better than the prior wavelet based methods.

Our new geometry-based compression classifies points based on what they are: tree points, building points and ground points, and then uses different methods to geometrically fit these points. When the bit rate is high enough, both the image based compression and the geometry based compression should have similar performance with geometry based compression slightly better. This is due to the fact that image based compression has quantization on the Z dimension but the geometry based compression retains the floating point on z. Since different objects have special characteristics, trees are generally bell shaped, and buildings have clear geometry boundaries, under low bit rates, we can use geometry based compression to reduce the size significantly, making the points on RD curve move to the left side significantly. Our experimental results show

that in lower bit rates, our new geometry based compression, which leverages the accuracy of our classification results, provides significant improvement for low bit rate compression compared to our image-based compression. Thus compared to the existing LIDAR lossy compression methods [13, 51], our new geometry based compression scheme achieves two orders of magnitude lower bit rate at the same quality. In addition, our geometry based method has a hierarchy structure giving it the ability to meet the geospatial imagery requirement of storing data at multiple resolutions and being able to "zoom in" easily.

## Chapter 6

# Conclusion and Future Work

In this dissertation, we develop two accurate LIDAR data classification methods that overcome the shortcomings of existing methods and obtain higher accuracy. We conduct LIDAR compression via traditional image-based compression and geometry-based compression, illustrate that our geometry-based compression has significant performance improvement in low bit rate compression compared with image-based compression. Compared with existing LIDAR compression methods, our compression rate is two orders of magnitude lower at the same quality. The original data can be reconstructed with geometric and statistical accuracy using the compressed information. Our geometry based method can identify/eliminate noise effectively based on manmade structure characteristics. Furthermore, our geometry- based compression has a hierarchy structure giving it the ability to facilitate the geospatial imagery requirement needed to store data at multiple resolution levels or scales and being able to "zoom in" easily. The semantic information embedded in the results is useful for urban scene modeling & understanding.

Currently we only use the LIDAR data to conduct the classification. If the corresponding airborne optical images are also available, we could conduct data fusion by integrating the optical images with the LIDAR data and conduct the classification using both the LIDAR data and the optical images. This could potentially further improve the classification performance.

For LIDAR data compression, besides airborne LIDAR data, the proposed geometry based compression scheme could be applied to other types of data as well. For example, not only airborne LIDAR data, but also ground based LIDAR data could be compressed following the same principle. More specifically, we will need to conduct spherical mapping to map the ground based LIDAR data into a spherical depth image, and then conduct segmentation and classification on the spherical depth image. Finally the classified LIDAR data could be compressed using the same idea as the proposed geometry based compression scheme. Furthermore we believe the proposed compression scheme can be applied beyond LIDAR data. Due to the advancement of sensor technology such as the Microsoft Kinect ®, RGB+D (i.e. Color + Depth) cameras that can capture real time RGBD image and/or image/video sequences become more and more popular. To efficiently compress these RGBD image and/or image sequences is very important for a lot of applications such as entertainment, virtual navigation, location based services, teleconferencing, etc. Traditional video based compression techniques might not be the best approach for the depth image/video. The proposed geometry based compression scheme might be more suited to compress this type of depth image/video. These would be the focus of our future research.

# References

[1] Surface reconstruction from unorganized points. Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, Werner Stuetzle. ACM SIGGRAPH 1992 Proceedings, 71-78.

[2] SVM-Light Support Vector Machine. svmlight.joachims.org

[3] R. C. Gonzalez and R. E. Woods, Digital Image Processing (3rd Edition), Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2006.

[4] http://resources.arcgis.com/en/help/main/10.1/index.html#//015w00000041000000

[5] http://en.wikipedia.org/wiki/Lidar

[6] M. Carlberg, P. Gao, G. Chen, and A. Zakhor, "Classifying urban landscape in aerial lidar using 3d shape analysis," in IEEE International Conference on Image Processing, 2009.

[7] G. Chen, and A. Zakhor, "2D tree detection in large urban landscapes uses aerial LiDAR data," in IEEE International Conference on Image Processing, 2009.

[8] Martin A. Fischler and Robert C. Bolles (June 1981). "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography". Comm. of the ACM 24 (6): 381–395.

[9] Comaniciu, D., and Meer, P. "Mean Shift: A Robust Approach Toward Feature Space Analysis". IEEE PAMI (24), No. 5, 603-619, May 2002.

[10] N. Ahmed, T. Natarajan and K.R.Rao, “Discrete Cosine Transform”, IEEE Trans. Computers, 90-93, Jan. 1974

[11] Gregory K. Wallace, “The JPEG Still Picture Compression Standard” , IEEE Transactions on Consumer Electronics, Apr. 1991

[12] Athanassios Skodras, Charilaos Christopoulos, and Touradj Ebrahimi, “The JPEG 2000 Still Image Compression Standard”, IEEE Signal processing magazine, Sep. 2001

[13] Biswajeet Pradhan, Sandeep Kumar, Shattri Mansor, Abdul Rahman Ramliand

Abdul Rashid Bin Mohamed Sharif, “LIGHT DETECTION AND RANGING (LIDAR) DATA COMPRESSIONLIDAR compression”, KMITL Sci. Tech. J. Vol. 5 No. 3 Jul. – Dec. 2005.

[14] Martin Isenburg, “LASzip: lossless compression of LiDAR data”, http://laszip.org

[15] S. Laky,P. Zaletnyik, C.. Toth, “LAND CLASSIFICATION OF WAVELET-COMPRESSED FULL-WAVEFORM LIDAR DATA”, IAPRS, Vol. XXXVIII, Part 3A Sept. 2010

[16] Jerome M. Shapiro, “Embedded Image Coding Using Zerotrees of Wavelet Coefficients”, IEEE Transaction on signal processing, Vol. 41, No. 12 Dec. 1993.

[17] Axelsson, P., 1999. Processing of laser scanner data—algorithms and applications. ISPRS Journal of Photogrammetry and Remote Sensing 54 (2– 3), 138– 147.

[18] Briese, C., Pfeifer, N., 2001. Airborne laser scanning and derivation of digital terrain models. In: Gruen, A., Kahmen, H. (Eds.), Optical 3D Measurement Techniques V. Technical University, Vienna, Austria, pp. 80– 87.

[19] Elmqvist, M., Jungert, E., Lantz, F., Persson, A., Soderman, U., 2001. Terrain modelling and analysis using laser scanner data. International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences XXXIV (Pt. 3/W4), 219– 227.

[20] Flood, M., 2001. LIDAR activities and research priorities in the commercial sector. International Archives of the Photogramm

[21] Reutebuch, S.E.; Andersen, H.-E.; McGaughey, R.J. Light detection and ranging (LIDAR): an emerging tool for multiple resource inventory. J. Forest. 2001, 103, 286-292.

[22] Baltsavias, E. A comparison between photogrammetry and laser scanning, ISPRS J. Photogramm. Remote Sens. 1999, 54, 83-94.

[23] Wehr, A.; Lohr, U. Airborne laser scanning—an introduction and overview. ISPRS J. Photogramm. Remote Sens. 1999, 54, 68-82.

[24] Romano, M.E. Innovation in LiDAR processing technology. Photogramm. Eng. Remote Sens. 2004, 70, 1202-1206.

[25] Flood, M. Laser altimetry—from science to commercial lidar mapping. Photogramm. Eng. Remote Sens. 2001, 67, 1209-1211.

[26] D. M. Cobby. The use of airborne scanning laser altimetry for improved river flood prediction. PhD thesis, University of Reading, 2002.

[27] I. J. Davenport, R. B. Bradbury, G. Q. A. Anderson, G. R. F. Hayman, J. R. Krebs, D. C. Mason, J. D. Wilson, and N. J. Veck. Improving bird population models using airborne remote sensing. International Journal of Remote Sensing, 21(13 & 14):2705–2717, 2000.

[28] R. C. Gonzalez and R. E. Woods. Digital Image Processing. 3rd ed. Upper Saddle River, N.J.: Prentice Hall, 2008.

[29] R. T. Ogden. Essential wavelets for statistical applications and data analysis. Birkhaeuser, 1997.

[30] M. Misiti. Wavelet toolbox : for use with MATLAB. 2nd ed. The MathWorks Inc., 2000.

[31] S. G. Mallat and W. L. Hwang. Singularity detection and processing with wavelets. IEEE Transactions on Information Theory, 38(2):617–643, 1992.

[32] S. G.Mallat. A theory for multiresolution signal decomposition: The wavelet representation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 11(7):674–693, 1989.

[33] J. C. Goswami and A. K. Chan. Fundamentals of wavelets : theory, algorithms, and applications. New York, Chichester: Wiley, 1999.

[34] I. Daubechies. Ten lectures on wavelets. CBMS-NSF regional conference series in applied mathematics, 1992

[35] Thomas M. Cover, Joy A. Thomas, “Elements of Information Theory” ,Wiley-Interscience, 2 edition, July 18, 2006

[36] D.A. Huffman, "A Method for the Construction of Minimum-Redundancy Codes", Proceedings of the I.R.E., September 1952, pp 1098–1102.

[37] Witten, Ian H., Neal, Radford M.; Cleary, John G. (June 1987). "Arithmetic Coding for Data Compression" Communications of the ACM 30 (6): 520–540, Sept. 2007

[38] Toby Berger. Rate Distortion Theory: A Mathematical Basis for Data Compression. Prentice Hall, 1971.

[39] J. Saghri, A. Tescher, and J. Reagan, "Practical Transform Coding of Multispectral Imagery"," IEEE signal processing magazine , January 1995.

[40] A. Gersho and R. M. Gray, Vector Quantization and Signal Compression, Kluwer Academic Publishers, 1991.

[41] M. A. Fischler, R. C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. Comm. of the ACM, Vol 24, pp 381-395, 1981.

[42] Jean-Francois Lalonde, Nicolas Vandapel , Daniel F. Huber, and Martial Hebert, Natural Terrain Classification using Three-Dimensional Ladar Data for Ground Robot Mobility.

[43]A. Delong, A. Osokin, H. Isack, Y. Boykov, “Fast Approximate Energy Minimization with Label Costs”, In International Journal of Computer Vision (IJCV), vol. 96, no. 1, pp. 1-27, Jan. 2012.

[44] Tom Fawcett, “ ROC Graphs: Notes and Practical Considerations for Researchers,” HP Laboratories, March 16, 2004

[45] E. Baltsavias, “A comparison between photogrammetry and laser scanning, ISPRS J. Photogramm. Remote Sens. vol. 54, pp. 83-94, 1999.

[46] A. Wehr and U. Lohr, “Airborne laser scanning—an introduction and overview,” ISPRS J. Photogramm. Remote Sens., vol. 54, pp. 68-82, 1999.

[47] M. E. Romano, “Innovation in LiDAR processing technology,” Photogramm. Eng. Remote Sens., vol. 70, pp. 1202-1206, 2004.

[48] M. Flood, “Laser altimetry—from science to commercial LIDAR mapping,” Photogramm. Eng. Remote Sens., vol. 67, pp. 1209-1211, 2001.

[49] D. M. Cobby, “The use of airborne scanning laser altimetry for improved river flood prediction,” PhD dissertation, University of Reading, 2002.

[50] I. J. Davenport, R. B. Bradbury, G. Q. A. Anderson, G. R. F. Hayman, J. R. Krebs, D. C. Mason, J. D. Wilson, and N. J. Veck, “Improving bird population models using airborne remote sensing,” International Journal of Remote Sensing, vol. 21, no. 13 & 14, pp. 2705–2717, 2000.

[51] Michael P. Gerlek, “Compressing Lidar Data”, Photogrammetric engineering & remote SenSing, November 2009, 1253-1255.

[52] T. Vijayaraghavan and K. Rajan, "Image coding of 3D volume using wavelet transformfor fast retrieval of 2D images", IEE Proc.-Vis. Image Signal Process., Vol. 153, No. 4, August 2006

[53] Wei, J., Saipetch, P., Panwar, R., Chen, D., and Ho, B.K.T., "'Volumetric image compression by 3D discrete wavelet transform", Proc. SPIE, 1995, 2431, pp. 184–194

[54] Gu, X., Gortler, S. J., and Hoppe, H. "Geometry images", ACM Transactions on Graphics , 21(3), 2002.

[55] Gu, X., Zhang, S., Zhang, L., Huang, P., Martin, R., and Yau, S. T, "Holoimages", ACM Solid and Physical Modeling, 2006

[56] Zhang, S. and Yau, S.-T., " Three-dimensional data merging using holoimage". Optical Eng., 47(3):033608. (Cover feature), 2008

[57] Donoho, D.L. "Compressed sensing". IEEE Transactions on Information Theory. Vol. 52, No. 4, page 1289, April 2006

[58] E. J. Candes, J. Romberg, and T. Tao, "Robust uncertainty principles: exact signal reconstruction from highly imcomplete frequency information," IEEE Transactions on Information Theory, vol. 52, no. 2, pp. 489–509, 2006

[59] Yuri Boykov, Olga Veksler and Ramin Zabih, "Fast Approximate Energy Minimization via Graph Cuts", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 23, no. 11, pp1222-1239, November 2001

[60] Jarek Rossignac, "The 3d revolution: CAD access for all!". In Invited key-note paper at the International Conference on Shape Modeling and Applications, Aizu-W akamatsu, Japan, IEEE Computer Society Press, pages 64–70,March 1997

[61]N.Haval, "Three-dimensional documentation of complex heritage structures", IEEE Multimedia, 7(2):52–56, 2000.

[62]V.J. DeLeon and H.R. Berry, "Vrnd: Notre-Dame cathedral– globally accessible multi-user real-time virtual reconstruction", In Proceedings of Virtual Systems & Multi Media (VSMM98), Ogaki, Gifu, Japan, November 1998.

[63]J.D. Foley, A.van Dam, S.K. Feiner, and J.F. Hughes, "Computer Graphics:Principles and Practice", (Second Edition in C),Addison-Wesley Publishing Company, 1996.

[64] Davis King, http://www.3dcompression.com/index.php

[65]Sun Microsystems, "Java3d API specification", http://java.sun.com/products/java-media/3D, June 1999.

[66] IBM Research (Visualization Group). VRML compressed binary format. http://www.research.ibm.com/vrml/binary/.

[67] R.Koenen.MPEG-4: Multimedia for our time. IEEE Spectrum, 36(2):26–33, February 1999

[68] L. Chiariglione, MPEG home page, http://www.cselt.it/mpeg

[69] Jarek Rossignac, "Edgebreaker: Connectivity compression for triangle meshes", IEEE Transactions on Visualization and Computer Graphics, Vol. 5, No. 1, pp. 47-61, January - March 1999

[70] Gabriel Taubin and Jarek Rossignac, "Geometric Compression Through Topological Surgery" ,ACM Transactions on Graphics, Vol. 17, No. 2, April 1998, Pages 84 –115

[71] Renato Pajarola and Jarek Rossignac, "Compressed Progressive Meshes", IEEE Transactions on Visualization and Computer Graphics, Volume 6 Issue 1, January 2000, Page 79-93

[72] Guillaume Lavoué, Florent Dupont and Atilla Baskurt, "Subdivision surface fitting for efficient compression and coding of 3D models", Proceeding of SPIE, vol 5960, PP1159-1170

[73] Sridhar Lavu, Hyeokho Choi and Richard Baraniuk, "Geometry Compression of Normal Meshes Using Rate-Distortion Algorithms", Eurographics Symposium on Geometry Processing (2003)

[74] S. Gumhold and W. Strasser, "Real-time compression of triangle mesh connectivity", In SIGGRAPH 1998, pages 133–140

[75]N. Haval, " Three-dimensional documentation of complex heritage structures", IEEE Multimedia, 7(2):52–56, 2000

[76] M. Isenbueg and J.Snoeyink, "Face Fixer: Compressing polygon meshes with properties", In SIGGRAPH 2000, pages 263–270,2000

[77] C.Touma and C.Gotsman, "Triangle mesh compression", In Proceeding of Graphics Interface 98, June 1998.

[78] D. King and J.Rossignac, "Connectivity compression irregular quadrilateral meshes", Technical Report GIT-GVU-99-36, GVU Center, Georgia Tech., Atlanta, USA, 1999.

[79] A. Guiziec, F.Bossen, G.Taubin, and C.Silva, “Efficient compression of non-manifold polygonal meshes”, In IEEE Visualization 1999, pages 73–80,1999

[80] Yodchanan, W., "Lossless compression for 3-D MRI data using reversible KLT", International Conference on Audio, Language and Image Processing, 2008. pp1560 – 1564

[81] Dinesh Shikhare, S. Venkata Babji, S.P.Mudur, "Compression Techniques for Distributed Use of 3D Data: An Emerging Media Type on the Internet", Proceeding ICCC '02 Proceedings of the 15th international conference on Computer communication Pages 676-696

[82] Zhang S., "Three-dimensional range data compression using computer graphics rendering pipeline", Applied Optics, Vol. 51, Issue 18 2012, pp. 4058-4064

[83] Nikolaus Karpinsky, Song Zhang, "3D range geometry video compression with the H.264 codec", Optics and Lasers in Engineering 51, 2013,pp620-625

[84] Amin P. Charaniya, Roberto Manduchi, and Suresh K. Lodha, “Supervised parametric classification of aerial lidar data,” in IEEE Conference on Computer Vision and Pattern Recognition Workshop, 2004, pp. 25–32.

[85] S. K. Lodha, D. M. Fitzpatrick, and D. P. Helmbold, “Aerial lidar data classification using adaboost,” in 3DIM ’07: Proceedings of the Sixth International Conference on 3-D Digital Imaging and Modeling,Washington, DC, USA, 2007, pp. 435– 442, IEEE Computer Society.

[86] Suresh K. Lodha, Edward J. Kreps, David P. Helmbold, Darren Fitzpatrick , "Aerial lidar data classification using Support Vector Machines (SVM)", Third International Symposium on 3D Data Processing, Visualization, and Transmission, 14-16 June 2006.

[87] http://en.wikipedia.org/wiki/IPIX

[88] Charles E. Matz, "Basic Principles of ROC Analysis", http://www.umich.edu/~ners580/ners-bioe_481/lectures/pdfs/1978-10-semNucMed_Metz-basicROC.pdf

[89] Xiaoling Li, Wenjun Zeng, Ye Duan, "A Hybrid approach for tree classification in airborne LIDAR data", 2013 IEEE Inter. Conf. on Acoustic, Speech, and Signal Processing, Vancouver, Canada.

[90] Xiaoling Li, Wenjun Zeng, Ye Duan, "Geometry based airborne LIDAR data compression", 2013 IEEE Inter. Conf. on Multimedia and Expo., San Jose.

[91]. http://en.wikipedia.org/wiki/QuickTime_VR

# VITA

Xiaoling Li received her bachelor and master degrees from Tsinghua University, Beijing, China, in 1992 and 1996, respectively, both in Electronic Engineering. She received her master degree in Computer Science from State University of New York at Stony Brook in 1998 and her MBA degree from Long Island University in 2004. From 2009 to 2013, she studied in the Computer Science Department, University of Missouri-Columbia, under the advice of Professor Wenjun Zeng, and was granted her Ph.D. degree in 2013.

From 1992 to 1994, she was an assistant professor in Electronic Engineering department in Tsinghua University. From 1998-2004, she was a software engineer in Motorola Solutions (formally Symbol Technologies Inc). From 2004-2006, she worked as a consultant for SAIC. She worked as a database administrator in University of Missouri from 2004 to 2012 and in IBM from 2012 to 2013. She currently works on research projects in School of Medicine, Washington University in St Louis. Her research interests include data mining, data compression, machine learning, medical imaging, visualization and signal/information processing.