2010

DETECTION OF ROOF BOUNDARIES USING LIDAR DATA AND AERIAL PHOTOGRAPHY

Andrew David Gombos
University of Kentucky, gombos@gmail.com

ABSTRACT OF THESIS

DETECTION OF ROOF BOUNDARIES USING LIDAR DATA AND AERIAL PHOTOGRAPHY

The recent growth in inexpensive laser scanning sensors has created entire fields of research aimed at processing this data. One application is determining the polygonal boundaries of roofs, as seen from an overhead view. The resulting building outlines have many commercial as well as military applications. My work in this area has created a segmentation algorithm where the descriptive features are computationally and theoretically simpler than previous methods. A support vector machine is used to segment data points using these features, and their use is not common for roof detection to date. Despite the simplicity of the feature calculations, the accuracy of our algorithm is similar to previous work. I also describe a basic polygonal extraction method, which is acceptable for basic roofs.

KEYWORDS: LiDAR data, Support Vector Machine, Segmentation,
Building extraction, terrain

Author's signature: Andrew David Gombos

Date: May 7, 2010

DETECTION OF ROOF BOUNDARIES USING LIDAR DATA AND AERIAL PHOTOGRAPHY

By
Andrew David Gombos

Director of Thesis: Ruigang Yang

Director of Graduate Studies: Raphael Finkel

Date: May 7, 2010

RULES FOR THE USE OF THESES

Unpublished theses submitted for the Master's degree and deposited in the University of Kentucky Library are as a rule open for inspection, but are to be used only with due regard to the rights of the authors. Bibliographical references may be noted, but quotations or summaries of parts may be published only with the permission of the author, and with the usual scholarly acknowledgments.

Extensive copying or publication of the thesis in whole or in part also requires the consent of the Dean of the Graduate School of the University of Kentucky.

A library that borrows this thesis for use by its patrons is expected to secure the signature of each user.

Name	Date

THESIS

Andrew David Gombos

The Graduate School
University of Kentucky
2010

DETECTION OF ROOF BOUNDARIES USING LIDAR DATA AND AERIAL PHOTOGRAPHY

THESIS

A thesis submitted in partial
fulfillment of the requirements for
the degree of Master of Science in
the College of Engineering at the
University of Kentucky

By
Andrew David Gombos
Lexington, Kentucky

Director: Dr. Ruigang Yang, Professor of Computer Science
Lexington, Kentucky 2010

To Stephanie

ACKNOWLEDGMENTS

I would like to thank Dr. Ruigang Yang, for his guidance and support during this project. I would also like to thank committee members Dr. Judy Goldsmith and Dr. Jinze Liu for their advice and support during this project. Finally I would like to thank Dr. Raphael Finkel for assisting as the Director of Graduate Studies, and the rest of the Computer Science department faculty and staff.

TABLE OF CONTENTS

Acknowledgments iii

Table of Contents iv

List of Figures v

Chapter 1 Introduction 1
1.1 Types of remote sensors 2
1.2 Related work 2
Point cloud only methods 2
Photograph Augmented methods 4
Related tasks 4
1.3 Goal and contribution 5
1.4 Outline 5

Chapter 2 Preliminaries 6
2.1 Introduction to Support Vector Machines 6
2.2 An overview of LiDAR systems 8

Chapter 3 Method 10
3.1 Training 10
3.2 Recognition 11
3.3 Basic data structures 13

Chapter 4 Feature vector creation 14
4.1 Feature selection 14
4.2 Calculation 15

Chapter 5 Implementation and Experimental Results 19
5.1 Results 19
5.2 Accuracy measurements 25

Chapter 6 Conclusions and future work 26
6.1 Acknowledgments 26

Bibliography 27

Vita 30

LIST OF FIGURES

2.1 Illustration of an support vector machine model. Colored circles indicate the feature vectors of two separate classes. Darker colored circles indicate the support vectors. 7
2.2 Linear, Polynomial, Gaussian Radial Basis, and Sigmoid kernels. u and v are feature vectors, γ, b, and d are constants. ^{T} denotes the transpose vector. 7
2.3 An example of a LiDAR data set. The green flat regions are roofs, and the red clouds are trees 8
2.4 Irregular spacing of LiDAR data points 9
2.5 Different LiDAR height returns for the same point 9

3.1 Accuracy percentages for varying γ and c values 11
3.2 A roof probability map. Roofs have a very strong signature, while the road and creek are barely visible 12
(a) Probability bitmap. White pixels indicate a higher probability of belonging to a roof 12
(b) Aerial photograph of the same region 12
(c) The probability bitmap overlaid on the aerial photo. The white dots representing the roofs are clearly visible 12

5.1 Aerial photographs of the training tiles 20
(a) Tile 1 20
(b) Tile 2 20
5.2 Three testing tiles, with different characteristics 21
(a) A dense neighborhood setting, with some more sparsely populated areas 21
(b) A suburban setting with a few large buildings 21
(c) The large highways and factories form one of the most prominent features of this tile 21
5.3 Resulting classification of the tile in Figure 5.2a. 22
(a) Classification of the tile in Figure 5.2a using a model trained on the tile in Figure 5.1b. Red areas are false positive areas, blue areas are false negatives, and purple areas are true positives. . . 22
(b) A region of the probability bitmap from the results shown in part (a). The detected roofs and sheds are clearly visible as whiter patches. No intensity correction was applied to this image. . . . 22
5.4 A comparison of the results between training tiles. The neighborhood based training tile provides much more accurate results. 23
(a) Using the tile of Figure 5.1a 23
(b) Using the tile of Figure 5.1b 23

5.5 Large roofs are missing due to the aggressive error reduction algorithm, even though they were properly detected in the LiDAR data. 23
(a) The large roofs, such as the blue region in the upper right corner, are not properly detected 23
(b) The blue region is easily detected in the probability bitmap . . 23
5.6 The large, incorrect region of highway was successfully removed from the output. Only smaller buildings, such as houses, remain. 24
(a) This image has some small noise artifacts, but large noise regions are successfully removed. The highway runs across the top of this image 24
(b) This section of the highway was strongly detected in the probability bitmap (Image enhancement applied for clarity) 24
5.7 Examples of detected roof polygons 25
(a) High quality detection results 25
(b) Detected roofs which were not in the reference map 25

Chapter 1 Introduction

This thesis presents a new technique for modeling polygonal roof outlines from laser scanned point cloud data and aerial photographs. An emphasis is placed on the simplicity of the feature calculations used to describe each point mathematically. This results in a method which is easy to understand, as well as easy to compute. A support vector machine is used to classify the data into roof and non-roof groups based on these features. I implemented and tested the algorithm on real-world data, and achieved high quality results on a wide variety of roof types.

The recent explosion of interest in urban data collection has provided a huge amount of information for accurate three dimensional models. Photographic imagery is available from satellites, airplanes, and vehicles with roof-mounted cameras. Similarly, high resolution point cloud data is available from airplanes and vehicle-mounted laser scanners. Accurate data describing roads, landmarks, protected areas such as parks, bodies of water, and other points of interest are generated by mapping companies. The most ambitious projects seek to combine all of this data together to create realistic models of urban environments. Much research has been dedicated to various components of this task, such as building reconstruction and tree identification.

This work takes a much more narrow scope than generating a complete model of a city. We are only interested in determining the outlines of a building roof from above. Despite the reduced complexity of this task, building outline information is still very useful. The outlines are painstakingly created by hand by surveying and mapping companies. The areas of geographical information systems (GIS) and photogrammetry involve the creation, manipulation, and interpretation of this data.

Wide scale roof outline information (collected over many square miles) may be analyzed by a property valuation authority (PVA), zoning commissions, real estate developers, land use managers, farmers, and census bureaus. For example, the city of Lexington, Kentucky recently passed an ordinance which charges property owners a fee based on the area of impervious surfaces on the property [10]. Automated methods to detect buildings, parking lots, sidewalks, and other impervious surfaces would make it much easier to calculate the required fees. A zoning commission may also want statistics about the average footprint of buildings in a neighborhood, to determine if a new building would meet the defined zone. Other statistics, such as the average distance from a building to the nearest road or the land use percentage are easily calculated. It is also straightforward to determine how many buildings, and their sizes, lie in a flood plain other other areas susceptible to natural disasters.

It is possible to automatically generate three dimensional roof models using only the outline. Many common types of roofs can be accurately described by the straight skeleton of the boundary, as described in Laycock and Day [15] and Ross et al. [20]. If the height of the outline is also stored, then a simple model of the building can be easily generated.

1.1 Types of remote sensors

To eliminate much of the tedious manual surveying work, our environmental data is collected by remote sensing methods. These methods are differentiated by their resolution, accuracy, and cost. The Shuttle Radar Topography Mission [5] used a radar system attached to the Space Shuttle to achieve approximately 30 meter resolution. While this resolution works well for determining the rough elevation changes for a digital elevation model (DEM), it is not acceptable for modeling fine details of a region. Similarly, the National Elevation Dataset [8] provides resolutions of three, ten, and 30 meters. While this data was specifically processed to represent the bare land elevation, much of it was collected using stereo imaging techniques. In this method, two photographs are taken of the same region at different positions. The parallax shift between these two images can be used to determine the elevation of specific points.

Laser based scanners are more appropriate for our needs. They provide a high resolution view of anything on the surface — trees, buildings, automobiles and power lines may all be detected by a laser based scanner. Note that most of these features are of items we are not interested in. Aerial scanning using a LiDAR (Light Detection and Ranging) system can provide a high resolution, top-down view of the ground. Photographs can be taken simultaneously, resulting in an orthographic image which is georeferenced to correspond exactly with the LiDAR points. These photographs usually have a much higher resolution than the returned pulses from the scanner, so they are important for determining the exact boundaries of building footprints on the ground.

Vehicle and tricycle mounted scanners and cameras can produce data similar to the aerial coverage. However, the different vantage point provides important side views of buildings. These can be used to create and texture map more complete models of buildings and other features. Data collected using this method is not very useful for determining roof outlines, since the roof itself is not generally sampled. Therefore, aerial collection of data is the most common for roof detection and segmentation.

1.2 Related work

Roof and building recognition algorithms can be classified into two main categories. The first category only uses point cloud information and various recognition techniques to classify buildings. The second category augments the point clouds with georeferenced photographic data. The reconstructed models may also be texture mapped if enough photographs are available of the sides of the building.

Point cloud only methods

Segmenting the various features which appear in a point cloud has many applications, such as automatically mapping the locations of different objects in an urban environment. Golovinskiy et al. [9] attached a laser scanner to a vehicle, and accurately map small elements of the environment such as mailboxes, light posts, and

stop signs. By using a vehicle mounted scanner, significant detail at the street level can be produced. However, aerial data is still necessary to get a complete picture of larger structures such as buildings. They also used sophisticated methods to segment different point clouds from one another, to properly detect different objects in the environment. This is not necessary for our project, since our data does not have a high enough density, or the correct perspective to distinguish smaller objects.

Several groups have also proposed methods to generate complete building models by fitting planes to the roof surfaces. Zhou and Neumann [27] first preprocess the input data to remove vegetation which would disrupt the rest of the process. Then a region-growing algorithm is used to discover all planar patches in the data. After the largest patches are combined to form the ground level patch, the remaining patches are combined to form the planar roof polygons. While this is an effective technique, it can only handle flat, planar roofs without user intervention. Finding a curved roof such as those on some barns and silos is not directly possible using this method.

Tarsha-Kurdi et al. [25] use simple mathematical principles to find probable roof planes. Their work compares two methods: 3D Hough transform and a RANdom SAmple Consensus (RANSAC) algorithm. The 3D Hough transform suffers from several problems similar to the two-dimensional version — there is a direct trade off between the step size though the parameter space and the required memory resources and computational time. If a large parameter step size is used, then planes which do not lie along the plane defined by the parameter values will probably not be detected. Their RANSAC formulation is much quicker, and several modifications to the basic algorithm are described to give higher-quality results. However, like the Hough transform method, the error each estimated plane must be calculated across the entire point set. Since identified plane points are removed after each iteration of the algorithm, the performance will improve over time. While our algorithm uses a RANSAC technique to determine the ground level, the number of points considered is limited to improve the performance of the feature generation algorithm.

Rottensteiner and Briese [21] introduce a curvature-based segmentation approach to locate probable roof planes. Planes are iteratively grown from seed points, and new points are added that meet certain tolerance requirements for the existing plane. After the planar regions are developed, several post-processing steps are performed to refine the collection of planes which actually belong to roofs. As with all methods which only utilize LiDAR data, the quality of the results is dependent on the original resolution of the data points. To address this problem, they propose an extension of the method which uses photographs to improve the edge resolution.

Verma et al. [26] attempt to match entire polyhedral roof structures, rather than simple planes, onto the data. If the polyhedra are able to model all of the possible roof sub-structures in the landscape, then very high quality models can be produced. However, roof sub-structures which do not have a corresponding polyhedral model cannot be recognized at all. The current work also does not address non-planar roof structures.

Pu and Vosselman [19] take a different approach, by recognizing individual features of a building, such as windows, doors, and walls. This method produces a much more detailed model which can then be texture mapped. While some roof

polygons can be determined from the ground level perspective data that this algorithm requires, the entire roof outline cannot be determined. However, it is useful as a complementary technique for generating a complete, detailed urban map.

These methods all rely on the point cloud data being as free of irrelevant data as possible. Since only geometric information such as curvature, point density, and the nearest neighbors to a point is available, any noise such as trees and power lines can prevent good results. Adding other layers of information, such as photographs of the region, can help to remove these irrelevant points.

Photograph Augmented methods

Recognizing and properly segmenting vegetation is one of the most important tasks for building reconstruction, since vegetation is the most likely source of noise in the point cloud. Secord and Zakhor [23] combine point cloud data with aerial imagery to remove vegetation such as trees which may interfere with the roof detection process. Their process is very similar to ours, with the goal of detecting trees rather than roofs. First, a feature vector is calculated for each point, using characteristics such as the image color (in the hue, saturation, value color space) and the approximate surface normal. The resulting feature vectors are used as the input into a weighted support vector machine model, which is described in more detail in Section 2.1.

Satellite imagery utilizing non-visible wavelengths of light can also be used to classify different regions as building and vegetation. Since different materials absorb infrared and ultraviolet light differently, they can be more accurately distinguished than by relying on visible attributes such as color. Sohn and Dowman [24] exploit this property by combining IKONOS imagery with LiDAR data. However, IKONOS imagery does not have the spatial resolution necessary for using the image to improve the accuracy of the building outline. Additionally, satellite imagery is an extra expense on top of the LiDAR collection cost, while an aerial photograph may be taken from the same airplane and at the same time as the LiDAR data.

Ameri and Fritsch [2] also use aerial photographs to refine the models generated from a point cloud and produce more accurate results. In addition to refining the detected edges, their system is also able to detect small features which are not well represented in the point cloud, such as dormer windows. A new regression algorithm is used to fit approximate roof planes to the points that are clustered by a region growing algorithm. The results of this algorithm are much more detailed than are necessary for our application, but would be a good choice for urban modeling.

Related tasks

The accurate detection of roof outlines can be made easier by preprocessing the data first. Koch et al. [14] describe a method to segment individual trees in a forest. While we are not interested in the exact tree count of an area, several of the techniques they describe may be used to segment trees in urban environments. They suggest an inverse-watershed segmentation method, where regions are created which have a monotonically decreasing height from the region's maximum.

Alharthy and Bethel [1] give several techniques that may be used to filter out artifacts in the data and produce more accurate results. One significant and simple method involves exploiting a property of LiDAR measurements. Each data point can have multiple associated height values, corresponding to different heights from which the laser beam was reflected back.[1]. By filtering out points which have a large difference in the return heights, points belonging to objects such as trees and cars are removed. However, this filtering technique may erroneously remove points belonging to glass roofs, such as green houses, skylights, and large atriums.

1.3 Goal and contribution

Many of the existing algorithms outlined above perform complex mathematical operations on the data, such as principle component analysis and regression algorithms to calculate the final roof polyhedra. Our method seeks to simplify the required manipulation of the data. This has two significant advantages over previous methods. The first advantage is that the calculation of feature vectors is relatively fast. Since the characteristics of each point are calculated using only local features, the problem is trivially parallelizable. While my implementation does not parallelize feature vector calculation, a significant speedup can be realized. Additionally, if the point cloud is broken down into smaller tiles which are processed in parallel, the height of the quad trees and nearest neighbor trees will be reduced, further improving the execution time.

The second advantage is that our method is not limited to specific shapes or geometries of roofs. By carefully choosing the training data to include uncommon roof structures such as domes and semi-cylindrical barn roofs, these regions can also be recognized without any modification to the underlying algorithm. This also means that the algorithm can successfully recognize a large variation in the sizes of roofs. Small roofs with only a few data points, such as a backyard shed, are recognized by the model. Large roofs, such as those for factories and warehouses, are also detected.

1.4 Outline

Section 2 begins with an overview of several critical components of our system. In Section 3 we give an overview of the algorithm used to identify building roofs. Section 4 will describe the algorithms as well as the necessary data structures in more detail. Section 5 will give details about our specific implementation of the algorithm. The section will also give some experimental results which show how well the method works on large areas of land. Finally, Section 6 will give some conclusions about the success and applicability of our method.

[1] Section 2.2 explains this idea in much more detail.

Chapter 2 Preliminaries

2.1 Introduction to Support Vector Machines

The support vector machine (SVM) was developed by Cortes and Vapnik in 1995 [4]. A SVM directly estimates the boundary surface model between two or more classes of data points. The model consists of parameters for a hyperplane that separates the two classes as completely as possible. The hyperplane model is trained using an iterative approach which is guaranteed to converge. Support vector machines perform well even when there are outliers in the data, so they work well for our application which has many potential outliers.

Each data point is represented as a *feature vector*, or a list of attributes which describe the characteristics each data point has. These attributes are typically floating point numbers, so the feature vector is an array of floating point numbers. Each element in the array describes a particular characteristic of the data, such as the height of a point or the red component of the RGB pixel. The feature vectors may be combined into a large, rectangular matrix with the characteristic labels on each column, and each row describes a feature vector.

These features must be selected so that they classify points into a specific class. The hyperplane chosen to define the boundary between two classes must maximize the *margin*, or the perpendicular distance from the boundary to the closest feature vector. Features which lie on the margin boundary are called *support vectors*. These features are used to define the boundary, and classify the unknown data. Since only the data points that are around the boundary from the two classes are examined, any outliers in the data are automatically eliminated from consideration without any preprocessing of the data. Figure 2.1 gives an illustration of the different components of a support vector machine model.

Linear boundary functions are easy to calculate, but they are not useful to describe many types of data classes. The shape of the hypersurface that is used is defined by a *kernel function*. This allows the SVM to easily support nonlinear boundaries by simply changing the kernel function that is used. Some common kernels are shown in Figure 2.2. The kernel function must be evaluated for training as well as unknown feature classification stages, so if efficiency is important it is necessary to choose the kernel with the least complexity that will achieve the desired classification accuracy.

Feature selection and parameter estimation are important stages to get the best results out of an SVM. One of the most important techniques I have found for getting good results out of an SVM is to scale the data to the range $[0, 1]$ or $[-1, +1]$. Each element of the feature vector is scaled independently. Thus, for a six element feature vector, there are six independent scaling factors which must be applied. By scaling the data before the model is trained, numerical instability is reduced when multiplying large matrices. Additionally, extremely high peaks in the data will not overshadow other elements in the feature set. Therefore, scaling the data is a useful way to make sure that all of the features that are used are seen as equally important (or weighted,

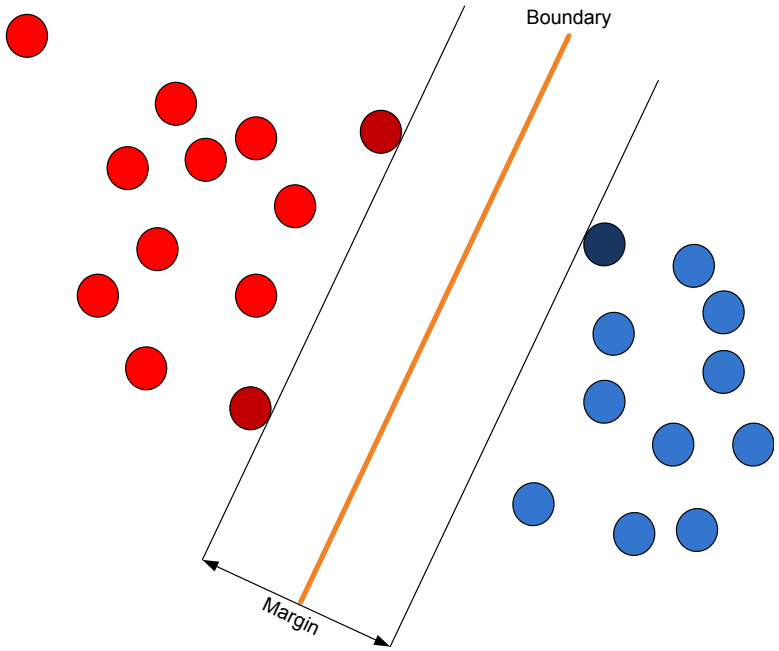


Figure 2.1: Illustration of an support vector machine model. Colored circles indicate the feature vectors of two separate classes. Darker colored circles indicate the support vectors.

Figure 2.2: Linear, Polynomial, Gaussian Radial Basis, and Sigmoid kernels. u and v are feature vectors, γ, b, and d are constants. ^{T} denotes the transpose vector.

if some features are more important than others).

Determining exactly which features produce the best result is also important. Unfortunately this must be done by testing random combinations of features and examining the validation accuracy, rather than some general heuristic. Grid search procedures should be used to optimize the parameters of the model (such as the kernel function parameters). Heuristic methods have been developed to reduce the size of the search space, however these may converge to a local maximum, which is not a global maximum. For more information about support vector machine training algorithms, libraries, and other information see [11], [3], and [12]. Section 3.1 gives details about our particular algorithm.

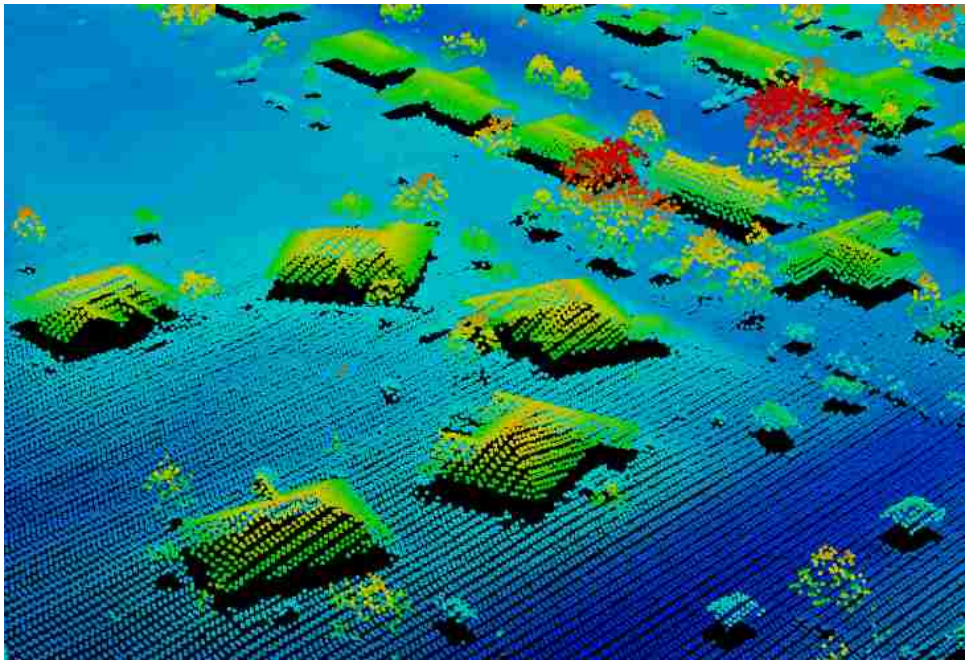


Figure 2.3: An example of a LiDAR data set. The green flat regions are roofs, and the red clouds are trees

2.2 An overview of LiDAR systems

Light Detection and Ranging (LiDAR) systems use the reflection from a laser pulse to determine the distance to objects. The laser beam is systematically scanned across an area to generate the point cloud. A example of such a point cloud is given in Figure 2.3. Since LiDAR data is usually collected from moving vehicles such as airplanes or automobiles, data collected during different passes may not be exactly aligned. This phenomenon can be seen in Figure 2.4. The east-to-west stripes are very consistent across the entire data set. However, the north-to-south grid is not uniform. This causes issues for data storage and representation and algorithms such as finding the nearest neighbor to a point. The large gaps between pairs of scan lines also reduces the effective resolution in the north-to-south direction.

The laser beam also introduces several interesting features into the data. Since the laser beam must travel tens or hundreds of meters to reach the surface, beam divergence becomes a significant factor. The divergence of the laser beam is dependent on several factors, such as the wavelength of the laser light itself. At ground level, the beam width is approximately 20 cm. If some portion of the beam reaches the level of a tree, it may be reflected back to the receiver in the plane. However, another portion of the beam may continue down to the actual ground level before being reflected. Which return pulse should be used as the height for this coordinate? Current LiDAR systems store several possible reflected pulses, or *returns*, for each point. Figure 2.5 shows an example of this behavior. Two heights, corresponding to the height at the tip of each arrow, are returned. Multiple return values can be used in areas other

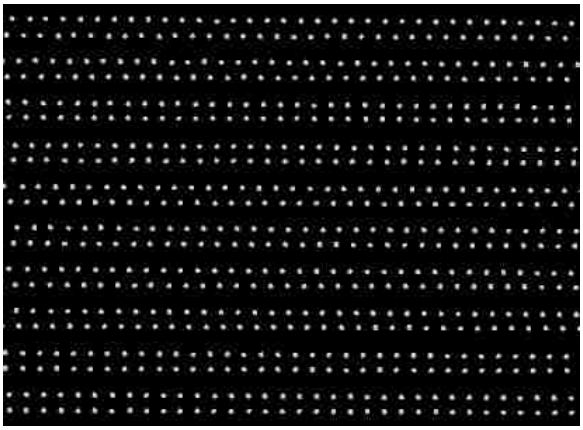


Figure 2.4: Irregular spacing of LiDAR data points

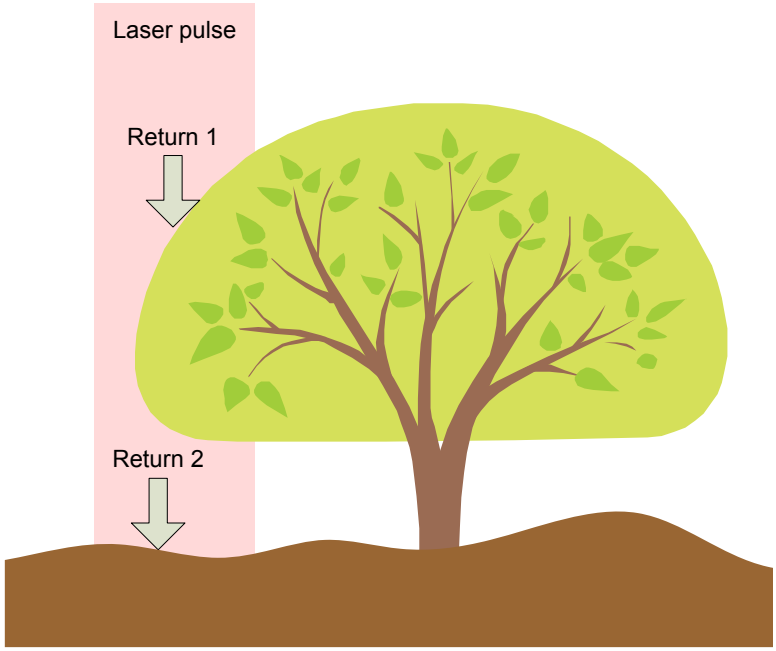


Figure 2.5: Different LiDAR height returns for the same point

than 3D modeling. By analyzing the difference between return values, scientists have been able to determine large scale characteristics of forests [16].

Chapter 3 Method

Training and recognition share many of the same steps. First, the data must be loaded into the necessary data structures. Then each data point must have several additional characteristics, such as the local curvature, calculated to determine the complete feature set. The training stage then generates the SVM model which will be used by the recognition stage.

3.1 Training

The training process begins by loading known roof outlines that were human generated. This supervised learning technique was chosen over data mining because thousands of these roof outlines had been accurately mapped. To generate the two data sets, any data point which falls inside of these roof outlines is marked as a roof point, otherwise it is a non-roof point. If the roof has any obstructions, such as an overhanging tree branch, then those points will also be included in the roof set.

After the data points have been partitioned, the optimal SVM model must be determined. There are several parameters that must be chosen for a given model. The most important decision is the selection of the kernel function used. Some common kernel functions used in SVM models were given in Figure2.2. Clearly a linear boundary function will not be able to cleanly separate data which has a quadratic correlation component. Using the radial basis function is suggested in SVM guide written by Hsu et al. [11]. Some experiments showed that the radial basis function is also the best kernel function to use in our problem.

The radial basis function has one parameter, γ. In addition, the SVM model has a cost parameter c. When calculating the boundary layer, the SVM training routine determines if any features have been misclassified. If c is very large, then this decision is a *hard decision*. That is, the determination of whether to use this boundary model or not is a binary choice based on whether any misclassified points exist. While this may make for a very accurate model (assuming one can be found), the model is likely to be *over fitted* to the data. The particular variances and coefficients in the model will be representative of the noise in the training data, rather than more generic patterns which are present in unknown data as well. To help eliminate the over fitting problem, the value of c can be made relatively small. Then the decision becomes a *soft decision*. Here, the training algorithm may accept several misclassified points if the overall result is accurate. The parameters γ and c are correlated, so the parameter space consists of a two dimensional grid of values.

To speed up the parameter estimation search, a subset of the total training data set is used. Several distinct regions are chosen to be representative of the complete data set, so that the final accuracy of the model is not significantly compromised. The following ranges were used for the parameter space:

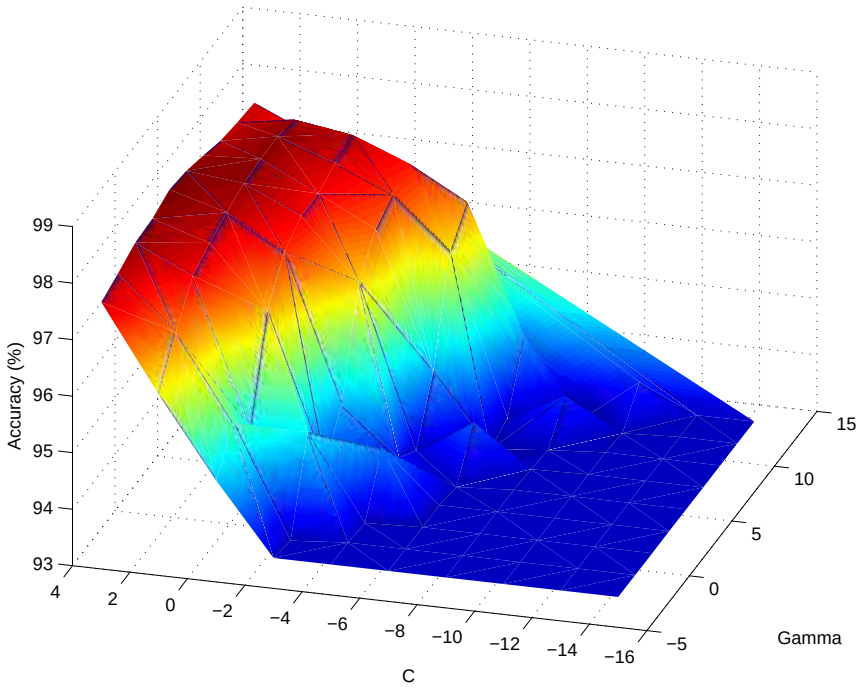


Figure 3.1: Accuracy percentages for varying γ and c values

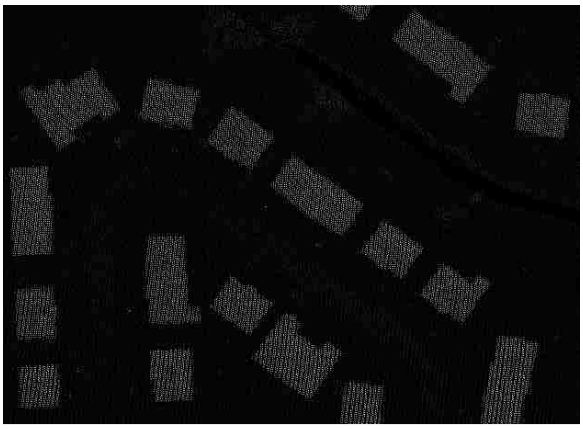
$$c : [2^{-5} \dots 2^{13}]$$
$$\gamma : [2^{-15} \dots 2^{3}]$$

At each iteration, the exponent values were incremented or decremented by 2. An example of the resulting accuracy over the parameter space is shown in Figure 3.1.

3.2 Recognition

The recognition process begins by loading in an already trained SVM model. Recognition is split into two phases: Roof outline generation, and outline culling. To generate the roof outline data, a feature vector is created for each point. The SVM model is used to classify it, and the resulting probability that the point belongs to a roof is stored in a bitmap. This bitmap is then processed to retrieve the actual roof contours. The outline culling phase removes non-roof contours to produce the final roof polygon set.

Instead of a binary label classification of roof or non-roof, the SVM algorithm is also able to compute the probability that a data point is a roof. By using the probability value, the determination of the contours of the roof becomes a soft decision problem. Various image processing and morphology techniques may also be used



(a) Probability bitmap. White pixels indicate a higher probability of belonging to a roof

(b) Aerial photograph of the same region

(c) The probability bitmap overlaid on the aerial photo. The white dots representing the roofs are clearly visible

Figure 3.2: A roof probability map. Roofs have a very strong signature, while the road and creek are barely visible

to enhance the probability image for easier roof recognition. An example of the probability bitmap is shown in Figure 3.2.

After creation of the probability bitmap, a combination of erosion and dilation operations were used to remove the outlier noise pixels from the bitmap. Additional dilation steps also help to fill in holes in the roof regions. Unfortunately these dilations also eliminate some of the potential fine features of the roof outlines. However, most roofs do not have many fine features to preserve, so it is likely that noise is being smoothed instead.

After most of the fine noise has been removed (false positive patches are still present), the OpenCV [18] library is used to extract the edge contours from the image. These contours will become the final roof outlines after more processing. The contours are converted into polygons with an allowable error of one meter. This

corresponds to one pixel in the probability bitmap, so the generated polygons follow the detected edges very closely. However, roof outlines which do not have noise on the boundary will still produce polygons with a low vertex count, and long, straight edges.

Statistical processing is used to reject the outlier polygons. The number of vertices and the average length of edges are used to classify the polygons into three groups: too small to be a roof, probably a roof, and too large to be a roof. Some care is necessary as small roofs such as garden sheds are detected by the algorithm. However, large false positive segments can be much more easily classified as incorrect. After filtering out invalid polygons, the final set of polygons is returned by the algorithm.

3.3 Basic data structures

Two similar data structures handle fast, efficient access to the LiDAR data. A quad tree [6, 22] is used to rapidly search for a given point, as well as retrieve all of the points that lie in a given rectangle. An approximate nearest neighbor library [17] was used to efficiently calculate the k points which are closest to an initial point, within some threshold.

Chapter 4 Feature vector creation

4.1 Feature selection

In any classification problem, different feature sets may greatly affect the potential accuracy of the method. After some experimentation to determine the optimal combination, I chose six features their accuracy, ease of calculation, simplicity, and robustness over different types of land:

- Horizontal and vertical curvature components
- Relative height
- The interpolated color of each point
- The distance to the nearest road

This feature set has significant variation over the residential areas and farmland of the data set. Each feature is discussed in more detail below.

Curvature

The curvature of a point is a useful metric to describe the rate of change in the shape of a surface at a point. For a point which is in the middle of a roof plane, the curvature value will be close to zero, since most roofs are approximately planar. However, trees will have wildly varying curvatures, as the LIDAR points describe individual leaves and branches, rather than the potentially much smoother general shape of the tree.

Relative Height

The relative height of each point is determined by approximating the shape of the ground in a neighborhood surrounding the point. By using the relative height of a point as a feature instead of the absolute height, a building on the top of a hill can be recognized alongside a building that is at the base of the hill. However, this does not account for buildings of different heights. For example, a one story ranch house, a two story commercial building and a tall grain silo must all appear in the training data for them to be accurately recognized.

Interpolated color

The color of each point provides an important clue to the type of each point. Since the spatial resolution of the aerial photograph is much higher than the LIDAR data, each point is colored using the bi-linear interpolation of the nearest pixel values. Care must be taken to weight the color information less than other descriptors. While a residential roof is generally a neutral gray, black, or brown, these are also the colors of roadways, dead grass, and gravel parking lots.

Distance to the nearest road

Many roofs are near roads, whether they are residential or commercial buildings. The use of road proximity does not work as well in a rural area, where long private access roads are more common.

4.2 Calculation

Overview

Several different procedures are used to generate the necessary features.

1. Initialize the quad tree and approximate nearest neighbor data structures
2. Compute the ground plane mesh
3. For each data point to be processed
    a) Calculate the principle curvatures
    b) Calculate the bi-linearly interpolated color
    c) Estimate the relative height to the ground
    d) Determine the distance to the nearest road
4. Determine the probability data point belongs to a roof
5. Create an image where each pixel represents the calculated probability
6. Process the image to improve the roof edge definition, and threshold to a binary image
7. Find the closed contours in the image
8. For each contour
    a) Approximate the contour by a polygon
    b) Classify each polygon as a roof contour or not

Using a probability measure allows the contour decision function to become a soft decision, and allow roof areas that were not as strongly detected to be added to the final roof polygon.

Ground plane mesh

Estimating the shape of the ground is important for determining the relative height of a data point. An iterative approximation method is used to estimate the ground plane for a grid of rectangles, which may be different sizes. The steps of our implementation are given in Algorithm 1. It is important to choose the two thresholds so that a building with a large, flat roof such as a warehouse is not marked as the ground plane. Additionally, slight bumps and dips in the ground level must be tolerated. In large areas of farmland, the roads are often raised above the ground level by several meters. However, the roads should still be considered part of the ground plane.

To satisfy these two conditions, T_{ground} is chosen to be two meters. T_{window_size} is chosen to be 300 meters. This seems to be a good balance between ignoring any large roof profiles, and allowing any hills or other significant altitude deviations to be modeled. A more accurate ground model could be determined if the fitting planes were instead allowed to be paraboloids. However, there is a much greater risk for modeling the noise (such as trees and buildings) with higher order surfaces. Additionally, the computational cost for determining the best fit paraboloid for a window is also higher than with a simple plane. Since our sample data consists of a relatively flat landscape, the plane is used as the fitting element.

Algorithm 1 Estimation of the ground plane mesh

1. Let $Windows = \{training\,region\}$
2. For each $window \in Windows$
   a) Choose three random points in $window$, and determine the fitting plane
   b) Count the number of points which lie around this plane (within a tolerance T_{ground})
      i. **If** the number of points is lower than a threshold **and** $size(window)$ is greater than a threshold T_{window_size}
      then $Windows = Windows \cup \{$window subdivided into four rectangles$\}$
      else $Windows = Windows \setminus window$
3. Repeat (2) until $Windows = \varnothing$

Principle Curvatures

The principle curvatures of a point in a 3D cloud are useful for determining the likelihood that a point lies on a plane, like that of a roof. Garimella and Swartz [7] present several algorithms to determine the curvature of triangular mesh surfaces. With some modifications their method can be implemented on unstructured point clouds as well.

We estimate the curvature of a point $\mathbf{p}$ by fitting a quadric surface to the surrounding points, aligning the Z axis of the surface along the approximate surface normal of the point in the cloud. The normal at $\mathbf{p}$ is estimated by fitting a least-squares plane to the approximate nearest neighbors of $\mathbf{p}$. An outline of the algorithm used is given in Algorithm 2.

It is possible to use the calculated quadric surface to estimate a new normal for the points, and iterate until the normal estimation converges. However, our application does not require curvature estimations which are accurate to several decimal places; after all, the model must be able to usefully use such precision. Therefore, the curvatures at each point are estimated using only the plane which is fit through the neighborhood.

Algorithm 2 Estimation of the principle curvatures

Input: A neighborhood $\mathbf{N} = \{$a point $\mathbf{p}$ and its four (approximate) nearest neighbors$\}$

1. Least-squares fit a plane to the neighborhood $\mathbf{N}$, and determine the outward-facing normal $\hat{n}$ to this plane

2. Transform the points in $\mathbf{N}$ to a local coordinate system where $\hat{n}$ is the positive Z axis.

3. Compute a least-squares quadric surface to the transformed points. The quadric surface is given by the equation

$$\begin{bmatrix} x_0^2 & x_0y_0 & y_0^2 & x_0 & y_0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ x_k^2 & x_ky_k & y_k^2 & x_k & y_k \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \end{bmatrix} = \begin{bmatrix} z_0 \\ z_1 \\ z_2 \\ z_3 \\ z_4 \end{bmatrix} \tag{4.1}$$

4. Compute the principle curvatures of the quadric surface

$$\kappa_1 = \frac{a + c + \sqrt{(a-c)^2 + b^2}}{(1 + d^2 + e^2)^2} \tag{4.2}$$

$$\kappa_2 = \frac{a + c - \sqrt{(a-c)^2 + b^2}}{(1 + d^2 + e^2)^2} \tag{4.3}$$

Color interpolation

Color cues sampled from the aerial photograph are important signifiers of potential roofs. For example, a point with a green color is probably vegetation, while white and black areas can be roofs, roads, and gravel areas. The color for each pixel is

bi-linearly interpolated, since the resolution of the photographs is much higher than the LiDAR data.

The colors are also analyzed using the Y'UV color space using the following conversion equations:

$$Y' = 0.299R + 0.587G + 0.114B \tag{4.4}$$

$$U = 0.436\frac{B - Y'}{0.886} \tag{4.5}$$

$$V = 0.615\frac{R - Y'}{0.701} \tag{4.6}$$

Relative height above ground estimation

During the initialization of the algorithm, a ground plane was estimated for varying sized tiles. Each point is checked against each bounding tile to determine which ground plane approximation to use. The absolute value of the distance to the ground plane is used so that the direction of the ground plane normal is not important. However, this has the potential to cause issues if there are deep depressions in the ground below the approximated ground plane. Fortunately, such depressions are not likely to be formed in the shape of an inverse roof, so this potential issue was ignored.

Distance to the nearest road

The distance from a given point to the nearest road is strongly correlated with the probability that the point describes a roof (assuming other the criteria are met, such as the relative height and curvature). The lines which describe a given road can be obtained from public sources, so including them in the feature set is not problematic. Since the number of road polylines is relatively low, a simple brute force search produces acceptable performance.

Chapter 5 Implementation and Experimental Results

I implemented the algorithm as a C++ program running on a computer with an Intel Core i7 920, 6 GB of RAM, and Windows Vista SP2. The SVM library used was LibSVM [3]. LiDAR data as well as a human-generated building and road maps were provided by GRW Inc. The DGNdirect library from the Open Design Alliance was used to read the roof and road data. Approximately 320 million square meters of data was available for testing. The average point spacing was 2.1 meters, which is dense enough to recognize roofs without requiring extremely large processing times.

The calculation of feature vectors took about 45 minutes for each 4000 meter square tile, which contained about 4,000,000 points on average. This time was dwarfed by the training time for the SVM. Depending on the contents of each particular tile, training times of two to four days were common. During the classification phase of each tile, about 180 points were processed per second. However, this still resulted in times of several hours to generate the probability bitmap. The computational requirements were increased dramatically by the use of probability information in the SVM model. Training times without probability information were on average less than one half of a day, and several thousand points could be classified per second. The probability bitmap processing component only took a few seconds, so it was dwarfed by the other processing components.

The slow training and evaluation times show that this SVM library (and perhaps the particular algorithms used) does not handle a very large number of data points. Input samples of a few hundred thousand points executed much faster. The implementation was trained on two tiles from the data, which are given in Figure 5.1. Three additional testing tiles were used to examine different aspects of the algorithm's performance. These tiles are shown in Figure 5.2.

5.1 Results

On a point by point basis, the SVM classifier achieved approximately 95% accuracy. However, determining the correct roof polygons is still not an easy task, due to the relatively low point density. The SVM model has the most trouble distinguishing the ridge lines of roofs where two roof planes meet. This is due to the significantly different curvature at this points compared to the curvature of the flat roof planes. This complication results in holes around the ridge lines, some of which extend to the edge of the roof profile. These few incorrectly classified points can result in the entire roof polygon being determined incorrectly. In the following result images, three different colors are shown. Red polygons are areas which were only detected by the algorithm. Blue areas are the human generated reference roof polygons. Purple areas indicate roofs which were successfully detected by the algorithm.

The classification accuracy was very good on some images. For example, the tile in Figure 5.2a was tested on a model trained on the tile in Figure 5.1b. The results are shown in Figure 5.3a. The results show very little noise and misdetected roofs. It

(a) Tile 1 (b) Tile 2

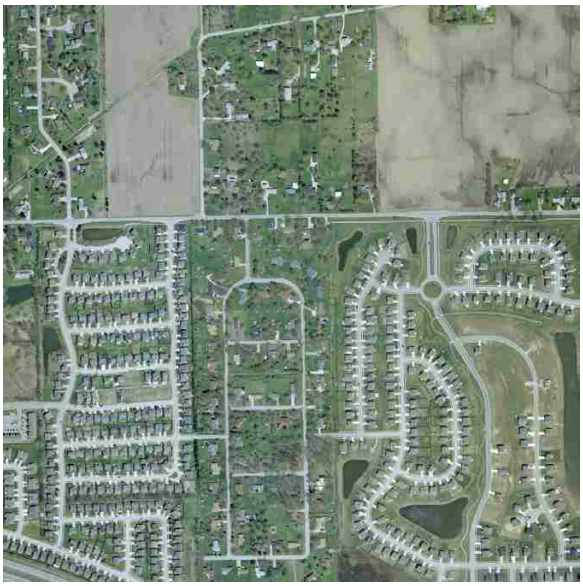
Figure 5.1: Aerial photographs of the training tiles

is possible to see several small out-buildings and sheds in the blue reference polygons. The noise reduction algorithm unfortunately removes any detection of these areas, although they are easily visible in the probability bitmap, shown in Figure 5.3b. Most of the regions completely in red are valid roofs which were not present in the reference data.

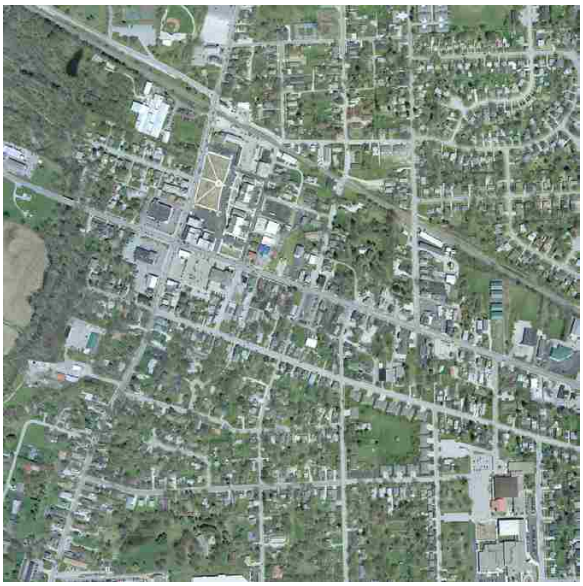
The particular training data used for the model also can dramatically determine the quality of the results. Figure 5.4 shows the dramatic difference choosing a tile with similar data can have.

Large roofs, such as those on factories, posed a problem for the final noise-reduction step. Figure 5.5 shows this issue. The large blue regions at the top right corner are large missing roofs, which should have been easily detected. In fact, the probability image shows that the roof was cleanly detected. However, the resulting polygon contained too much fine detail (too many vertices and edges), so it was eliminated from the final results. This gives a clear indication of where future work should be directed. The other large blue regions show similar behavior.

Some legitimate noise patches are also eliminated using the same algorithm. Figure 5.6 shows a highway segment which was strongly detected by the SVM model. However, it is not present in the final output polygons, which means it was successfully detected as noise and removed. This model was trained using the tile of Figure 5.1b. Using the tile of Figure 5.1a resulted in a much weaker signature of the road, but this also resulted in a much weaker signature on most buildings in the image.



(a) A dense neighborhood setting, with some more sparsely populated areas

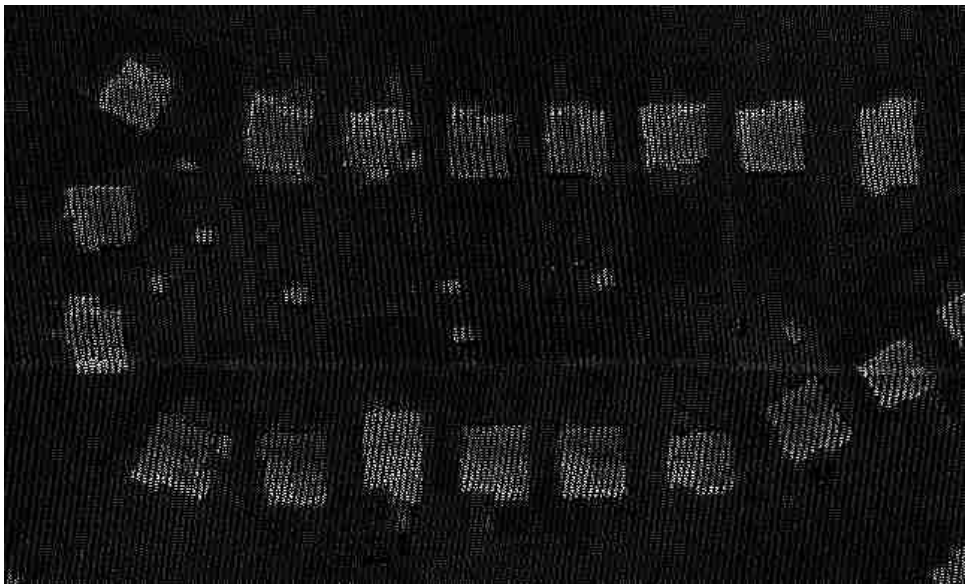


(b) A suburban setting with a few large buildings

(c) The large highways and factories form one of the most prominent features of this tile

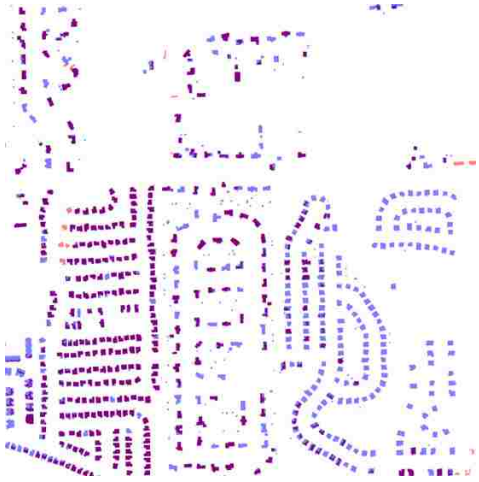
Figure 5.2: Three testing tiles, with different characteristics

(a) Classification of the tile in Figure 5.2a using a model trained on the tile in Figure 5.1b. Red areas are false positive areas, blue areas are false negatives, and purple areas are true positives.



(b) A region of the probability bitmap from the results shown in part (a). The detected roofs and sheds are clearly visible as whiter patches. No intensity correction was applied to this image.

Figure 5.3: Resulting classification of the tile in Figure 5.2a.



(a) Using the tile of Figure 5.1a

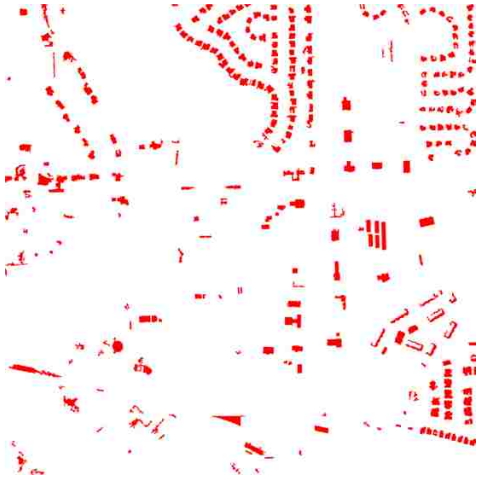
(b) Using the tile of Figure 5.1b

Figure 5.4: A comparison of the results between training tiles. The neighborhood based training tile provides much more accurate results.

(a) The large roofs, such as the blue region in the upper right corner, are not properly detected

(b) The blue region is easily detected in the probability bitmap

Figure 5.5: Large roofs are missing due to the aggressive error reduction algorithm, even though they were properly detected in the LiDAR data.



(a) This image has some small noise artifacts, but large noise regions are successfully removed. The highway runs across the top of this image

(b) This section of the highway was strongly detected in the probability bitmap (Image enhancement applied for clarity)

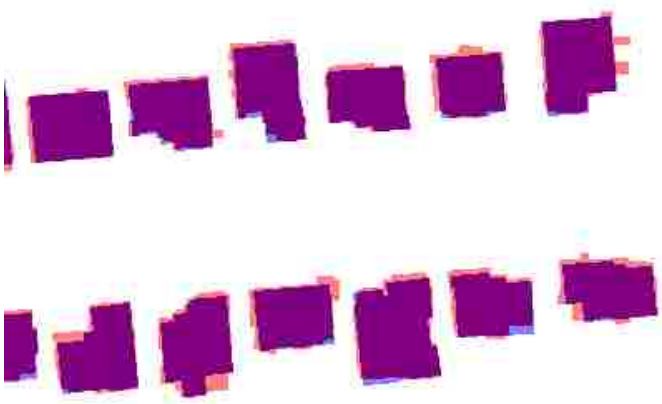
Figure 5.6: The large, incorrect region of highway was successfully removed from the output. Only smaller buildings, such as houses, remain.

The model trained on the neighborhood setting tile shown in Figure 5.1b performed consistently better than the tile of mostly farm areas in Figure 5.1a. Unfortunately there were not any other farm tiles present in the data set to test the performance of that model on similar data. The results show that the neighborhood tile performs well as a general roof classifier for a wide variety of buildings, without optimizing the particular content the SVM was trained on. Selecting additional areas to add to the training data set, such as a large highway, would probably improve the performance of the model. However, the execution time of the SVM would also be increased significantly.

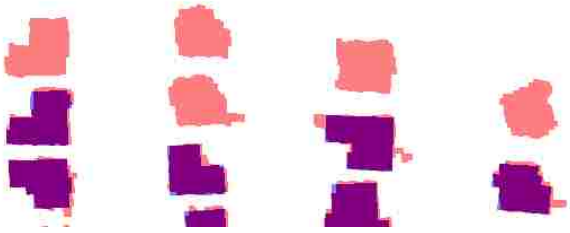
5.2 Accuracy measurements

Unfortunately, much of the previous work for detecting roof outlines does not present an analysis of how well the algorithm worked. Instead, renderings are used to show the quality of the computed results. This makes comparing different approaches challenging, especially if one algorithm generates fitting planes for each roof while another only calculates the outline. Sohn and Dowman [24] present a simple technique of counting the number of pixels which are appropriately classified in the output image. However, this results in misleading accuracy numbers for our example data set. Each tile of data contains several roofs which are not marked on the ground truth map of the building outlines, resulting in false positives. Additionally, every building is marked in the reference map, even though we are not interested currently in detecting small out buildings or sheds. This results in an artificially high false negative rate.

A detailed view of some of the extracted roof polygons can be seen in Figure 5.7. Even though the point by point accuracy is not perfect, for most application purposes of counting roofs and their sizes, the results are very good. In these examples, every roof that should be detected was marked with a red blob of the correct size. A stronger polygon regularization scheme would enhance the overall shape of the detections further.



(a) High quality detection results



(b) Detected roofs which were not in the reference map

Figure 5.7: Examples of detected roof polygons

Chapter 6 Conclusions and future work

Determining the roof boundaries from point clouds and aerial photography has many uses. While other research has also combined LiDAR data as well as photographic images to generate a more accurate result, our method generates results which are roughly equivalent in quality, while the feature vector values require much less manipulation of the actual data. The feature calculation is very quick, compared to the time required to train the SVM model.

The main SVM point segmentation model works well, correctly distinguishing between most roofs and the surrounding environment. However, the model which classifies polygons as roofs or noise needs further refinement. Large, complex roofs may easily be rejected as noise if they were not perfectly separated from the background points. However, some regions of noise (especially smaller lumps) pass through the filter easily. More work is required to create an accurate model for classifying the actual roof polygons after segmentation.

Further research is also needed to improve the SVM performance. The performance of the SVM library significantly limits the overall performance of the algorithm. There have been recent advances in algorithms which can train certain types of support vector machines faster. Thorsten Joachims' work [13] in this area is promising, and a different formulation of the feature set may produce acceptable results using a linear kernel. Alternate implementations of SVM algorithms may also be optimized for very large data sets. Current implementations appear to be targeted towards data sets with a few hundred thousand elements at once, and our data is an order of magnitude larger.

Despite these issues, a simple feature vector set combined with a support vector machine can reliably segment roofs from the surrounding environment. By keeping each feature relatively simple to calculate, the overall system accuracy is high, even on local neighborhood configurations, such as highways, which the system may have not been trained on originally. This ability to scale feature traits to objects of an arbitrary size is also an important capability of the system. We have also shown that the support vector machine is a suitable learning model for this problem. While SVMs have been used extensively for other classification tasks such as document classification, their use in point cloud classification is not common. Further work to classify the resulting polygons as roof or noise will further improve the overall system results.

6.1 Acknowledgments

GRW Inc. provided the data set used in the experiments. The Open Design Alliance provided the DGNdirect library used to load the human generated training data. The open source projects LibSVM, OpenCV, and Eigen provided optimized implementations of many of the algorithms necessary for the project.

Bibliography

[1] A. Alharthy and J. Bethel. Heuristic filtering and 3D feature extraction from LIDAR data. *INTERNATIONAL ARCHIVES OF PHOTOGRAMMETRY REMOTE SENSING AND SPATIAL INFORMATION SCIENCES*, 34(3/A):29–34, 2002.

[2] Babak Ameri and Dieter Fritsch. Automatic 3d building reconstruction using plane-roof structures. In *Proceedings of ASPRS Annual Conference*, pages 22–26, 2000.

[3] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[4] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3): 273–297, 1995.

[5] Tom G. Farr, Paul A. Rosen, Edward Caro, Robert Crippen, Riley Duren, Scott Hensley, Michael Kobrick, Mimi Paller, Ernesto Rodriguez, Ladislav Roth, David Seal, Scott Shaffer, Joanne Shimada, Jeffrey Umland, Marian Werner, Michael Oskin, Douglas Burbank, and Douglas Alsdorf. The shuttle radar topography mission. *Rev. Geophys.*, 45(2), May 2007. ISSN 8755-1209. URL `http://dx.doi.org/10.1029/2005RG000183`.

[6] R. A. Finkel and J. L. Bentley. Quad trees a data structure for retrieval on composite keys. *Acta Informatica*, 4(1):1–9, March 1974. URL `http://dx.doi.org/10.1007/BF00288933`.

[7] R. V. Garimella and B. K. Swartz. Curvature estimation for unstructured triangulations of surfaces. Technical report, Los Alamos National Laboratory, 2003.

[8] D. Gesch, M. Oimoen, S. Greenlee, C. Nelson, M. Steuck, and D. Tyler. The national elevation dataset. *Photogrammetric Engineering and Remote Sensing*, 68(1):5–11, 2002.

[9] Aleksey Golovinskiy, Vladimir G. Kim, and Thomas Funkhouser. Shape-based recognition of 3D point clouds in urban environments. In *International Conference on Computer Vision (ICCV)*, 2009.

[10] Lexington-Fayette County Urban Government. Water quality management fees. Chapter 16, Article XIV. URL `http://library5.municode.com:80/default-now/template.htm?view=browse&doc_action=setdoc&doc_keytype=tocid&doc_key=fb9f0fac1bfdadf04db32b178188f8af&infobase=11163`.

[11] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin. A practical guide to support vector classification. Technical report, National Taiwan University, October 2008. URL `http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf`.

[12] T. Joachims. Making large-scale svm learning practical. In Bernhard Schölkopf, Christopher J. C. Burges, and Alexander J. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1999.

[13] T. Joachims. Training linear SVMs in linear time. In *ACM SIGKDD International Conference On Knowledge Discovery and Data Mining (KDD)*, pages 217–226, 2006.

[14] B. Koch, U. Heyder, and H. Weinacker. Detection of individual tree crowns in airborne lidar data. *Photogrammetric Engineering and Remote Sensing*, 72(4): 357, 2006.

[15] RG Laycock and AM Day. Automatically generating roof models from building footprints. In *Proceedings of WSCG*. Citeseer, 2003.

[16] K. Lima, P. Treitza, M. Wulderb, B. St-Ongec, and M. Floodd. LiDAR remote sensing of forest structure. *Progress in Physical Geography*, 27(1):88–106, 2003.

[17] David M. Mount and Sunil Arya. ANN: A Library for Approximate Nearest Neighbor Searching. Online. URL `http://www.cs.umd.edu/~mount/ANN/`.

[18] OpenCV. Online. URL `http://opencv.willowgarage.com/wiki/`.

[19] Shi Pu and George Vosselman. Automatic extraction of building features from terrestrial laser scanning. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 36(5), 2006.

[20] Lutz Ross, Birgit Kleinschmit, Jürgen Döllner, and Henrik Buchholz. Automated transformation of 2d vector-based plans to 3d geovirtual environments. In *International Conference on Information Technologies in Landscape Architecture*, 2006. URL `http://www.kolleg.loel.hs-anhalt.de/studiengaenge/mla/mla_fl/conf/pdf/conf2006/43ROSS_L.pdf`.

[21] F. Rottensteiner and C. Briese. Automatic generation of building models from lidar data and the integration of aerial images. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences of the ISPRS*, 34(3/W13):174–180, 2003.

[22] Hanan Samet. *The design and analysis of spatial data structures*. Addison-Wesley series in computer science. Addison-Wesley, 1990.

[23] John Secord and Avideh Zakhor. Tree detection in urban regions using aerial lidar and image data. *Geoscience and Remote Sensing Letters, IEEE*, 4(2):196 –200, April 2007. ISSN 1545-598X. doi: 10.1109/LGRS.2006.888107.

[24] Gunho Sohn and Ian Dowman. Data fusion of high-resolution satellite imagery and lidar data for automatic building extraction. *ISPRS Journal of Photogrammetry and Remote Sensing*, 62:43–63, 2007.

[25] F. Tarsha-Kurdi, T. Landes, and P. Grussenmeyer. Hough-transform and extended RANSAC algorithms for automatic detection of 3D building roof planes from Lidar data. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 36(3/W52):407–412, 2007.

[26] V. Verma, R. Kumar, and S. Hsu. 3d building detection and modeling from aerial lidar data. In *Computer Vision and Pattern Recognition*, volume 2, 2006.

[27] Qian-Yi Zhou and Ulrich Neumann. Fast and extensible building modeling from airborne lidar data. In *GIS '08: Proceedings of the 16th ACM SIGSPATIAL international conference on Advances in geographic information systems*, pages 1–8. ACM, 2008. doi: http://doi.acm.org/10.1145/1463434.1463444.

Vita

- Date and Place of Birth: May 24, 1986 Charleston, WV
- Education
  - University of Kentucky, Bachelor of Science in Mathematics, Magna Cum Laude
  - University of Kentucky, Bachelor of Science in Computer Science, Magna Cum Laude