LIDAR DATA ANALYSIS
FOR AUTOMATIC REGION SEGMENTATION
AND OBJECT CLASSIFICATION

Thesis
Submitted to
The School of Engineering of the
UNIVERSITY OF DAYTON

In Partial Fulfillment of the Requirements for
The Degree of
Master of Science in Electrical Engineering

By
Nina Marie Varney

UNIVERSITY OF DAYTON
Dayton, Ohio
December, 2015
LIDAR DATA ANALYSIS FOR AUTOMATIC REGION SEGMENTATION AND OBJECT CLASSIFICATION

Name: Varney, Nina Marie

APPROVED BY:

Vijayan K. Asari, Ph.D.
Advisor Committee Chairman
Professor, Department of Electrical and Computer Engineering

Eric J. Balster, Ph.D.
Committee Member
Associate Professor, Department of Electrical and Computer Engineering

Raul Ordonez, Ph.D.
Committee Member
Professor, Department of Electrical and Computer Engineering

John G. Weber, Ph.D.
Associate Dean
School of Engineering

Eddy M. Rojas, Ph.D., M.A., P.E.
Dean, School of Engineering
ABSTRACT

LIDAR DATA ANALYSIS FOR AUTOMATIC REGION SEGMENTATION AND OBJECT CLASSIFICATION

Name: Varney, Nina Marie
University of Dayton
Advisor: Dr. Vijayan K. Asari

Light Detection and Ranging (LiDAR) presents a series of unique challenges, the foremost of these being object identification. Because of the ease of aerial collection and high range resolution, analysts are often faced with the challenge of sorting through large datasets and making informed decisions across multiple square miles of data. This problem has made automatic target detection in LiDAR a priority.

A novel algorithm is proposed with the overall goal of automatic identification of five object classes within aerially collected LiDAR data: ground, buildings, vehicles, vegetation and power lines. The objective is divided into two parts: region segmentation and object classification. The segmentation portion of the algorithm uses a progressive morphological filter to separate the ground points from the object points. Next, the object points are examined and a Normal Octree Region Merging (NORM) segmentation takes place. This segmentation technique, based on surface normal similarities, subdivides the object points into clusters. Next, for each cluster of object points, a Shape-based Eigen Local Feature (SELF) is computed. Finally, the features are used as the input
to a cascade of classifiers, where four individual support vector machines (SVM) are trained to
distinguish the object points into the remaining four classes.

The ability of the algorithm to segment points into complete objects and also classify each point
into its correct class is evaluated. Both the segmentation and classification results are compared to
datasets which have been manually ground-truthed. The evaluation demonstrates the success of the
proposed algorithm in segmenting and distinguishing between five classes of objects in a LiDAR
point cloud. Future work in this direction includes developing a method to identify the volume
changes in a scene over time in an effort to provide further contextual information about a given
area.
ACKNOWLEDGMENTS

This work was supported by contract FA865012C6307: Cognitive Approaches to LiDAR integration into Intelligence Production. I would like to thank Mike O’Connor, Scott Cone, Dave Kancler, and Drew Bowers for their participation in research discussions and creative suggestions.

I would also like to thank my editors Michela Greco and Matthew Velazquez for their valuable critiques and suggestions. Finally, I would like to thank the members of the University of Dayton Vision Lab, especially my advisor Dr. Vijayan K. Asari, for their knowledge, patience, and encouragement throughout this process.
TABLE OF CONTENTS

ABSTRACT .................................................................................................................. iii

ACKNOWLEDGMENTS .............................................................................................. v

LIST OF FIGURES ..................................................................................................... viii

LIST OF TABLES ....................................................................................................... x

NOMENCLATURE ....................................................................................................... xi

I. INTRODUCTION .................................................................................................... 1

II. LITERATURE SURVEY .......................................................................................... 6

  2.1 Digital Terrain Model Generation .................................................................. 6
  2.2 Segmentation ..................................................................................................... 8
  2.3 Feature Extraction ............................................................................................ 9
     2.3.1 Local Features ...........................................................................................) 9
     2.3.2 Global Features ......................................................................................... 10

III. AUTOMATIC SEGMENTATION OF OBJECTS IN LIDAR POINT CLOUDS ........ 12

  3.1 Progressive Morphological Filtering .............................................................. 12
  3.2 Normal Octree Region Merging .................................................................... 17
     3.2.1 Surface Normal Calculation ................................................................... 17
     3.2.2 Octree Segmentation .............................................................................. 19
     3.2.3 Over Segmented Region Merging ............................................................ 20
  3.3 Segmentation Evaluation .................................................................................. 21

IV. FEATURE EXTRACTION AND CLASSIFICATION OF LIDAR POINT CLUSTERS 26

  4.1 Surface-based Eigen Local Features ............................................................... 26
     4.1.1 Histogram of Surface Normals ............................................................... 26
# LIST OF FIGURES

1.1 Structure of the proposed LiDAR segmentation algorithm ........................................ 3

1.2 Structure of the proposed LiDAR feature extraction algorithm ................................. 4

3.1 The image on the left shows the original point cloud while the image on the right shows the point cloud filtered with a small windowed morphological filter .................. 13

3.2 Framework of the progressive morphological filtering algorithm .............................. 15

3.3 DTM generation results using the outlined progressive morphological filtering al-
gorithm ......................................................................................................................... 16

3.4 NORM results before region merging on different scenes ........................................ 20

3.5 NORM results after region merging on different scenes ........................................... 22

3.6 NORM segmentation on a dense urban scene, where the figure on the left shows the input point cloud and the figure on the right shows segmented point cloud ............. 23

3.7 Comparison of automatic versus manual segmentation .............................................. 23

3.8 Segmentation comparisons between methods ........................................................... 25

4.1 Cascade of classifiers .................................................................................................. 30

4.2 Classification color codes .......................................................................................... 34

4.3 Object classification results on CVM dataset: Area 1 .................................................. 35

4.4 Object classification results on CVM dataset: Area 2 .................................................. 35

4.5 Object classification results on CVM dataset: Area 3 .................................................. 36
4.6 Object classification in Scene 1 of the Surrey dataset. Notice the misclassification of low vegetation and landscaping ........................................ 37

4.7 Object classification in Scene 2 of the Surrey dataset. Notice the misclassification of low vegetation and landscaping ........................................ 38

4.8 Classification on CVM dataset. Some under segmentation can be seen in the upper left hand corner of the scene ........................................ 39

4.9 Classification results in Surrey dataset after adding contextual visual information . 40

4.10 Classification results in CVM dataset after adding contextual visual information . 42
## LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Region Segmentation Accuracy</td>
<td>24</td>
</tr>
<tr>
<td>4.1</td>
<td>Object Classification Results: Accuracy</td>
<td>33</td>
</tr>
<tr>
<td>4.2</td>
<td>Object Classification Results: Precision</td>
<td>33</td>
</tr>
<tr>
<td>4.3</td>
<td>Object Classification Results: Specificity</td>
<td>33</td>
</tr>
<tr>
<td>4.4</td>
<td>Object Classification Results: Sensitivity</td>
<td>33</td>
</tr>
</tbody>
</table>
NOMENCLATURE

\( \alpha \)  The ratio of principal components, calculated from global eigenvalues

\( \lambda \)  The eigenvalues associated with the covariance matrix calculated from points in the local neighborhood

\( \lambda' \)  The eigenvalues associated with the covariance matrix calculated from the global points, \( P_i \)

\( \phi \)  The elevation angle of the surface normal

\( \sigma \)  The curvature calculated from local eigenvalues

\( \theta \)  The polar angle of the surface normal

\( \vec{n}_i \)  The normal vector

\( C \)  The covariance matrix formed from a set of points

\( c \)  The cell size of the progressive morphological filter

\( d_p \)  The dilation of a group of points

\( e_p \)  The erosion of a group of points

\( f_i \)  The distance of point \( p_i \) to a plane \( q \)

\( H_i \)  The histogram of the elevation angles in bin \( i \)
\( h_{p,k} \)  The elevation difference between a LiDAR point \( p \) and the filtered surface at elevation \( k \)

\( h_{T,k} \)  The elevation difference threshold at iteration \( k \)

\( p_i \)  The points associated with cluster \( i \)

\( q \)  A plane fitted to a set of points

\( s \)  The slope of the terrain

\( t_e \)  Region merging histogram euclidean distance threshold

\( V_i \)  The variance associated with bin \( i \)

\( w_k \)  The window size of the progressive morphological filter at iteration \( k \)
Light Detection and Ranging (LiDAR) is a passive remote sensing technology that has gained popularity in recent years because of its ability to rapidly produce dense, high-precision aerial point clouds. Additionally, the presence of multiple returns allows pulses to penetrate vegetation canopies, providing an accurate description of the ground plane in areas that would not be visible in simultaneously collected aerial images. This gives LiDAR a distinct advantage over other methods like three-dimensional reconstruction, which is limited by the amount of information that can be obtained from an image in the visible spectrum. LiDAR has an additional advantage of being able to rapidly produce point clouds, whereas other methods like photogrammetry are extremely costly in both the time it takes to collect the images and the additional computation time required to orient the information in a meaningful way. Because of these advantages plus a significant reduction in cost in recent years, LiDAR has become a popular method of modeling, with applications such as surveying, urban planning, monitoring infrastructure changes, and automatic target detection. Military and pipeline industries are using LiDAR to precisely assess large areas and identify critical threats, with the overall goal being automatic object detection.

Automatic object detection in LiDAR has many challenges that must be overcome to provide an accurate and meaningful contextual description of a scene. One of the problems is the density of objects in the scene. In urban and rural areas alike, there are many instances of overlapping objects.
or simply different objects in close proximity to one another. For example, a tree overhanging onto a building is a common occurrence. The human visual system makes it easy to distinguish between these two objects because of their distinct color and texture signatures. However, the problem of distinguishing between these objects becomes non-trivial in LiDAR data because there are no longer any texture or color indicators to rely on, only shape. The problem is how to differentiate between two objects which share the same physical space. In typical two-dimensional image processing, segmentation techniques, such as watershed or automatic region growing methods can be used to make these distinctions. In the three dimensional case, new techniques must be developed to make these shape-based segmentations.

Even when a scene is properly segmented, it is often difficult to group the segments into distinct classes because of the wide variety of objects contained within the scene. It is often useful to identify objects within an overall class such as buildings or vegetation; however, each of these classes has many different subclasses which are distinctly dissimilar from one another. For example, a typical residential home and a skyscraper have very different physical qualities, yet they would both be considered objects which belong to the building class. Similarly, a pine tree and a maple tree can be grouped as vegetation despite their different shapes and leaf types. Even seasonal changes can make the same object look vastly different—a tree in the dead of winter will have a very different signature than the same tree in the spring. The problem then becomes whether a distinct set of features can be developed to fully describe the overreaching similarities between each class. In this particular application, the goal is to identify five classes of objects: ground, buildings, vehicles, power lines, and vegetation.

The solution to the problem of segmentation into object clusters can be solved in two steps. The first step is to separate the ground points from the object points, or the development of a digital terrain model (DTM). Previous studies have shown the removal of the ground plane leads to increased
success in the segmentation of the remaining objects [1]. Additionally the ground plane has the most distinct signature of all the classes, being the lowest and largest connected object within the scene. Because of this, it is often easiest to remove the ground points before the primary segmentation. One of the most commonly used DTM generation techniques is the progressive morphological filter. This procedure is used because of its success in removing object points while preserving the subtle local changes within the terrain. After the removal of the ground plane, the focus will then shift to separating the remaining object points into meaningful groups of individual objects using three-dimensional segmentation techniques [2]. The focus will be on using surface normals to group together objects with the conditions of local proximity and similarity of surface normal histogram signatures, specifically using the Normal Octree Region Merging (NORM) technique. The entire segmentation procedure is outlined in Fig. 1.1.
After the points have been separated into segments, the next challenge is to develop a feature vector to accurately describe the segments [3, 4, 5, 6]. These Shape-based Eigen Local Features (SELF) are built with the goal of identifying the signature shape and curvature both of each individual point and its surrounding neighboring points and the entire object cluster as a whole. The focus is on calculating normalized histograms to describe the local shape and curvature of the points as well as using Principal Component Analysis (PCA) on the set of points as a whole to determine meaningful relationships between the points, primarily using eigenvalues. By isolating individual shapes that describe each class, in some cases more than one shape per class, it is possible to distinguish between the remaining four classes. The overall construction of the feature vector can be

Figure 1.2: Structure of the proposed LiDAR feature extraction algorithm
seen in Fig. 1.2. Once the feature vectors are calculated, they can then be used as the input to a classifier. For this particular application, a cascade of support vector machines (SVM) is used. The final output is a LiDAR point cloud where each point has both an assigned classification value and red, green and blue (RGB) color code that matches the predicted class.

The thesis is structured as follows: Chapter 2 reviews the previous literature and methods that have been used for automatic object detection in LiDAR data, focusing on segmentation and feature extraction techniques. Chapter 3 discusses the DTM construction and NORM segmentation, and evaluates these against other popular approaches. Chapter 4 overviews the SELF construction and the cascade of SVM classification and examines the results achieved using this method. Finally, Chapter 5 presents the conclusions and plans for future work.
CHAPTER II

LITERATURE SURVEY

A survey of the current methods was performed in three major areas: digital terrain model generation, segmentation, and feature extraction techniques. The DTM survey focuses on RANdom SAmple Consensus (RANSAC) based plane fitting, skewness balancing, and progressive morphological filtering. The segmentation method focuses on automatic seeded region growing and strip histogram based segmentation. Finally, the feature extraction methods evaluate both local point-based features, including fast point feature histograms (FPFH) and the radius-based surface (RBS) descriptor as well as global features like volume component analysis (VCA).

2.1 Digital Terrain Model Generation

In their evaluation of LiDAR segmentation techniques, Douillard et al. have shown that in addition to being a valuable tool for analysts, extracting a DTM and separating the ground points from object points leads to a better overall segmentation and subsequent classification [1]. The general development of a DTM usually falls under three methods; Bartels and Wei present the first and most basic of these methods, skewness balancing [7, 8]. The method makes two assumptions. The first assumption is that the ground plane has a natural, Gaussian distribution. The second assumption is that all object points are higher than all ground points. The algorithm calculates the skewness of the height components of all of the points within the scene. If the skewness of the scene is greater than
zero, the highest point in the scene is removed. This process repeats iteratively until the skewness of the scene is zero and the only remaining points are ground points. This method is successful on non-complex scenes but obviously struggles on scenes which contain complex terrain like cliffs, hills, or other general dynamic changes. More recently, Bartels and Wei have expanded this to include a prediction model in which a LiDAR tile is predicted to contain complex terrain and the changes to the natural distribution expectations are adjusted accordingly [9].

The second method is a RANSAC-based plane fitting to estimate the ground plane [10]. The assumption behind this method is that in any particular scene, the majority of points will be ground points [11]. From this assumption, RANSAC can be used to determine the inliers, or ground points, and the outliers, or object points. The RANSAC method begins by randomly picking \( n \) points from the point cloud. From these \( n \) points, a plane model is calculated, then all of the points in the point cloud are examined and the distances from the plane model are computed and summed. The plane model with the smallest mean squared error is saved as the best model. This process is repeated, randomly selecting \( n \) points until the probability that all combinations of points have been selected is sufficiently high. Once the appropriate model is chosen, all the points that lie within a certain distance threshold are considered ground points while all of the other points are considered object points. This method is more successful than skewness balancing because of its ability to estimate sloped planes. For example, ground points of a tree canopy on a hill can be easily detected, where some ground points lie above some object points. However RANSAC plane fitting does not account for complex dynamic local changes in more than one direction.

The third algorithm is the progressive morphological filter. Killian et al. discussed the ability to separate ground and object points with the use of a morphological filter [12]. This method is effective in its ability to distinguish between points but requires manually choosing the window size. A window size that is too small leaves large objects misclassified as ground points, a commission
error, while a window size that is too large removes ground points as object points, an omission error. Zhang proposed an extension to this morphological filtering. By using a progressive morphological filter, iterating over the LiDAR cloud using an increasing window size, this problem of choosing an appropriate window size can be eliminated [13]. The progressive morphological filtering technique is as follows: first the LiDAR measurements are spaced into a grid; for each cell in the grid, the minimum elevation is chosen to represent the value of the cell. An opening operation is applied on the grid for the smallest window size and the minimum elevation of the surface is selected. This process is repeated, for a larger and larger window size, as the distance threshold is recalculated. The final grid will contain the value of all of the points contained within the ground plane.

2.2 Segmentation

The first segmentation method explored is automatic seeded region growing [2]. This method is borrowed from two-dimensional image segmentation [14]. The LiDAR scene is again spaced into a grid, where the value of each cell is chosen to be the maximum height of all the points contained within the cell. Then each cell is given a smoothness value; each cell is compared to its eight neighbors and the smoothness is calculated. The cells with the highest smoothness values are chosen as seed points. For each seed point, the surrounding cells are examined and if their heights are within a certain threshold of the average height for the region, they are added to the region. This process is repeated until no neighbors are remaining, and then is repeated again until all seed points are significantly grown. This process is generally successful in segmenting large regions but has two major flaws. The first flaw pertains to the threshold that dictates whether or not a cell belongs to a region. This threshold is scene-specific and will change depending on the object and resolution in a scene. The second flaw of this algorithm is that the operation functions on a two-dimensional grid. This prevents occluded objects from being segmented, losing one of the major advantages of LiDAR data.
The second segmentation method is the strip histogram method [15]. This method operates in a similar region growing approach as previously described. However, instead of splitting the scene into a two-dimensional grid, the scene is split into a three-dimensional voxel grid. After the two-dimensional region growing is performed, the scene is re-examined, this time looking at the three-dimensional voxels. If any voxel has a high density of points at a point which is not contained in the maximum height of the strip, that voxel is grown in three dimensions, allowing for a segmentation of objects that might be occluded from a nadir view. This segmentation method is successful and also has the advantage of being able to segment occluded objects in a truly three-dimensional sense; however, it still requires the user to manually select the grid size, which will vary based on the resolution and point density of the LiDAR point cloud.

2.3 Feature Extraction

There are two approaches to extracting features from a LiDAR data set. The first approach is a set of local features where a feature vector is developed for each point contained within the cloud, usually based on the properties of neighboring points [16]. The second method of feature extraction is to develop a global feature describing the entire cloud.

2.3.1 Local Features

The first local feature is the Fast Point Feature Histogram (FPFH) descriptor [17]. For each point, all the points within a specified radius are examined. For each pair of points, the distance between the two points and the difference between the angles of the surface normals are calculated. This process is repeated for each pair of points within the neighborhood and all of the values are concatenated to form a feature vector describing the original point. This feature extraction method is commonly used as a robust method for extracting local features, although it is computationally expensive.
The second feature extraction methodology is the radius-based surface (RBS) descriptor [18, 19]. This descriptor is calculated as follows: each point in the point cloud is iterated over and all of the neighboring points within a radius are collected. For each neighboring point, the distance between the two points and the angle between their surface normals are calculated. These two values—the distance and the angle—are used to populate two histograms which, when concatenated together form the feature vector for that particular point. This process is repeated for all points within the point cloud.

2.3.2 Global Features

The alternative to the local point descriptors is the global cloud descriptor, where one feature vector is used to describe an entire cloud of points. Two global features are defined in this thesis research. The first method is a Volumetric Grid Shape Descriptor (VGSD) [6]. The object regions are normalized to a $N \times N \times N$ voxel space, where each voxel contains information about the location and density of points within that voxel. A set of volumetric features is extracted to represent the object region; these features include: three-dimensional form factor, rotation invariant local binary pattern (RILBP), fill, stretch, corrugation, contour, plainness, and relative variance. The form factor, fill, and stretch provide a series of meaningful relationships between the volume, surface area, and shape of the object. RILBP provides a textural description from the height variation of the LiDAR data. The corrugation, contour, and plainness are extracted by three-dimensional eigen analysis of the object volume to describe the details of the object’s surface. Relative variance provides an illustration of the distribution of points throughout the object. The new feature set is robust, and scale and rotation invariant for object region classification.

The second method is volume component analysis (VCA) [3] which is based on PCA. PCA provides a reduced feature representation that computes a covariance matrix from the original input vector. The eigenvectors corresponding to the largest eigenvalues of the covariance matrix are used
to describe an image. VCA proposes that a LiDAR point cloud can be represented as a series of voxels whose values correspond to the point density within that relative location. From this voxelized space, the point cloud can be considered in layers with block-based PCA used to analyze sections of the space. The sections, when combined, represent features of the entire three-dimensional object.
CHAPTER III

AUTOMATIC SEGMENTATION OF OBJECTS IN LIDAR POINT CLOUDS

The process of region segmentation within point clouds is discussed in this chapter. First, a progressive morphological filtering technique is presented for the purpose of identifying and removing ground points within the LiDAR point cloud. Next, a Normal Octree Region Merging (NORM) segmentation is presented to further segment the remaining object points. Finally, the combination of these two methods will be compared to the automatic seeded region growing and strip histogram segmentation methods.

3.1 Progressive Morphological Filtering

Progressive morphological filtering is a way to classify and remove object points, leaving only the ground points. This method draws on the idea of dilation and erosion, which are common filtering methods in two-dimensional image processing. These methods can be extended to LiDAR in the form of a max and min operation, if the scene is analyzed as a continuous surface. The dilation, \( d_p \), of a measurement is defined by Eq. 3.1 where \((x_p, y_p, z_p)\) are the X, Y, and Z locations of \( p \) neighbors within a window \( w \). Erosion is defined similarly in Eq. 3.2

\[
d_p = \max_{(x_p, y_p) \in w} (z_p)
\]  

(3.1)
\[ e_p = \min_{(x_p,y_p) \in w} (z_p) \] (3.2)

Additionally, these dilation and erosion operations can be combined to perform opening and closing, where opening is the dilation of an erosion of a given set and closing is the erosion of the dilation of a given set. Earlier methods suggest using a simple morphological filter, where an opening operation is performed to detect the lowest point within the data set and all points below a height threshold of that point are considered ground points. This method requires caution when picking both the window size and the height threshold. A window size that was too small would leave out significant portions of larger items, such as buildings. This can be seen in Fig 3.1 where vegetation and power lines are removed, but because the widow size was not large enough, many of the buildings remain. Similarly, a window size that is too large would over remove ground points, taking out sections of higher ground points.

Figure 3.1: The image on the left shows the original point cloud while the image on the right shows the point cloud filtered with a small windowed morphological filter
In order to remove large objects while also preserving small subtle changes in the ground terrain, a progressive morphological filter is proposed. In this method, morphological filtering is used iteratively, gradually increasing the window size on each iteration. The process is as follows, the entire scene is loaded and the grid is created. For each cell within the grid, the minimum value of all the points within the cell is chosen to represent the value of the cell. If a cell does not contain any points, the value of the nearest cell containing points is chosen to represent that cell. For each window, an erosion and then dilation is performed according to Eq 3.2 and 3.1 respectively. Next, $Z$ is taken to be the original height of the cell and $Z_f$ is the height obtained from the opening operation. $Z - Z_f$ is examined for each point. If $Z - Z_f > h_{T,k}$, where $h_{T,k}$ is the difference height threshold at iteration $k$, the point is considered an object point. The remaining points (which are still considered ground points) are filtered again, with an increasing window size.

For this algorithm to be effective, it is important to choose the appropriate window size, $w$, and elevation difference threshold, $h_{T,k}$. For this particular application the window size is increased linearly, using Eq 3.3, where $k = 1, 2, ..., M$. $M$ is the maximum window size and $b$ is the initial window size.

$$w_k = 2kb + 1$$ (3.3)

Similarly, it is important to choose a distance threshold which is appropriate for each window. The elevation difference threshold can be determined according to Eq. 3.4, where $h_{T,0}$ is the initial distance threshold, $h_{T,\text{max}}$ is the maximum distance threshold, $s$ is the slope of the terrain, and $c$ is the size of the individual cell within the grid.

$$h_{T,k} = \begin{cases} 
  h_{T,0} & w_k \leq 3 \\
  s(w_k - w_{k-1})c + h_{T,0} & w_k > 3 \\
  h_{\text{max}} & dh_{T,k} > h_{\text{max}}
\end{cases}$$ (3.4)
After the data is gridded, an opening operation is performed on the grid. For each point within the grid, a $h_{p,1}$, can be calculated which is the difference between the height of the original point and the height of the filtered surface. This can then be compared to $h_{T,1}$ which is the elevation difference threshold. If the $h_{p,1} \leq h_{T,1}$ then the point is a ground point. Similarly, if the $h_{p,1} > h_{T,1}$ then the point is an object point. This process is repeated. For each iteration, the window size and elevation threshold are incremented according to Eqs. 3.3 and 3.4, respectively, and the surface from the previous iteration is used as the input. The framework of this progressive morphological filter is shown in Fig. 3.2.

![Figure 3.2: Framework of the progressive morphological filtering algorithm](image)

Figure 3.2: Framework of the progressive morphological filtering algorithm
(a) The image on the left shows the original input cloud and the image on the right shows the output ground points from the progressive morphological filter.

(b) A profile view of the clouds from Fig 3.3 (a).

(c) The image on the left shows the original input cloud of a more complex scene and the image on right shows the output ground points from the progressive morphological filter.

(d) A profile view of the clouds from Fig 3.3 (c).

Figure 3.3: DTM generation results using the outlined progressive morphological filtering algorithm.
Fig. 3.3 shows the results of two scenes filtered through a progressive morphological filter. The scenes on the left show the original point clouds with both ground and object points. The scenes on the right show the results of the progressive morphological filtering where the object points have been removed, leaving only the ground points. Each point cloud is colored by height with red representing the highest points within the point cloud and blue representing the lowest. Underneath each scene is a side view of the scene for perspective. It can be observed that the progressive morphological filter is successful in removing the unwanted ground points. Additionally, these scenes are of dense urban areas, some places with a significant amount of tree cover and generally complex local terrain changes. In the first scene, Fig. 3.3 (b) and (d) it can be observed by the color changes that the progressive morphological filter was very successful in keeping gradual changes in the local terrain, with the overall height difference of the scene being around 33 meters. This result would have been impossible to achieve with a method such as skewness balancing or RANSAC-based planes estimation.

Once the progressive morphological filtering is performed, the points identified as ground points are treated as classified and the next step is to consider the object points and segment those points into individual distinct objects.

### 3.2 Normal Octree Region Merging

The next sections focus on the segmentation of the remaining points using the proposed NORM method.

#### 3.2.1 Surface Normal Calculation

In order to separate the remaining points into meaningful groups, the first step is to calculate the surface normals for each point as described by Rusu [17]. The idea behind this method of surface normal calculation is that for a neighborhood around each point, \( p \), a plane which best fits that set
of points can be calculated and the normal of that plane can be considered to be the surface normal at that particular point. In this case, the plane is represented by $q$ and the distance to the plane, $f_i$, is given by $f_i = (p_i - q) \cdot \vec{n}$, where $\vec{n}$ is the normal vector. The covariance matrix, $C$, is evaluated according to Eq. 3.6 where $\overline{p}$ is the average of the points within the neighborhood, as seen in Eq. 3.5. From the covariance matrix, both the eigenvectors and eigenvalues can be determined. For the eigenvectors representing the eigenvalues, $\lambda_2 \leq \lambda_1 \leq \lambda_0$, the smallest eigenvectors will represent the normal vector, in the form $\vec{v}_2 = \vec{n} = \{n_x, n_y, n_z\}$. Because it is impossible to know the sign of the normal vector, when calculating from principal components, the sign is arbitrarily assigned to be from a nadir viewpoint, satisfying Eq. 3.7, where $v_p$ is the viewpoint. Finally, the normal vector is transformed into spherical coordinates, as shown in Eqs. 3.8 and 3.9, where $\phi$ is the elevation angle and $\theta$ is the polar angle.

\[
\overline{p} = \frac{1}{k} \cdot \sum_{i=1}^{k} p_i \quad (3.5)
\]

\[
C = \frac{1}{k} \cdot \sum_{i=1}^{k} (p_i - \overline{p}) \cdot (p_i - \overline{p})^T \quad (3.6)
\]

\[
\vec{n}_i \cdot (v_p - p_i) > 0 \quad (3.7)
\]

\[
\phi = \arctan \frac{n_z}{n_y} \quad (3.8)
\]

\[
\theta = \arctan \frac{\sqrt{(n_y^2 + n_z^2)}}{n_x} \quad (3.9)
\]
3.2.2 Octree Segmentation

After the surface normals are calculated, an octree-based segmentation approach is applied. The entire scene is considered and is split up into eight bins, $2 \times 2 \times 2$ in the X, Y, and Z direction. The variance of all of the elevation angles corresponding to the surface normals within that bin is calculated according to Eq. 3.10, where $\phi_i$ is the elevation angle of the surface normal and $\mu_{bin}$ is the average elevation angle of the bin. If the variance is below a threshold $t_v$, then the current bin is subsequently split into eight more bins. This process is repeated until the entire scene consists of different sized bins, all containing surface normals where $V_{bin} < t_v$. The results of this can be seen in Fig. 3.4; where each color represents a distinct segment whose points have surface normals where $V_{bin} < t_v$. A few properties can be observed from Fig. 3.4; the first is that there are no instances of under segmentation or segmentations where there is more than one object that is considered as one object. The second property that can be observed is that there is an obvious over segmentation of most, if not all, of the objects. In order to correct this over segmentation, a region merging approach is applied.

$$V_{bin} = \frac{1}{n} \cdot \sum_{i=1}^{n} (\phi_i - \mu_{bin})^2$$  (3.10)
Figure 3.4: NORM results before region merging on different scenes

3.2.3 Over Segmented Region Merging

This region merging approach works much like the automatic seeded region growing technique proposed in [2, 14] with the exception that instead of using height to measure similarity, a histogram
signature is used. For each bin, a ten-element histogram of the elevation angles of the containing points are calculated, $H_1, H_2, ... H_i$. The process of region merging can be summarized as follows:

1. Organize the histograms into a list, $S$, in descending order of the variance of their elevation angles.

2. For each histogram, create a list of neighbors, $L$, of bins which share one or more sides with the histogram of interest.

3. While $L > 0$, calculate the Euclidean distance between the seed bin and the nearest neighbor bin. If the distance < $t_e$, join the neighboring bin to the current bin and recalculate the histogram. Add the neighbors of the neighboring bin the list $L$ and remove the corresponding histogram from the list $S$.

4. Repeat step 3 until all neighbors have been examined

5. Repeat for all histograms remaining in list $S$

The results of the region merging portion of the algorithm can be seen in Fig. 3.5. The region merging provided meaningful connections between previously segmented bins. Fig. 3.6 shows the NORM applied on a dense urban scene. Each color represents a different object cluster.

3.3 Segmentation Evaluation

The following evaluation is proposed by Douillard et al [1] to compare segmentation methods against a manually segmented scene. Each group of points in the manually segmented data will be examined. The largest section of points, segmented with the algorithm, that falls within the portion of the manually segmented object will be considered correct. All other segmentations within the object will be considered as errors. Once an object is considered correct for a single object it must be an error for all other objects. For example, take the tree in Fig. 3.7; 3.7 (a) is the manually
Figure 3.5: NORM results after region merging on different scenes
Figure 3.6: NORM segmentation on a dense urban scene, where the figure on the left shows the input point cloud and the figure on the right shows segmented point cloud.

Figure 3.7: Comparison of automatic versus manual segmentation.

(a) Manually segmented object  
(b) Automatically segmented object
segmented object while 3.7 (b) is the object obtained by our algorithm. In this instance all points in the dark green would be considered a correct segmentation while all points in the light green would be an error. This particular segmentation has a point score range of 67%. The results were compared on a set of two manually segmented data sets of urban areas. The NORM algorithm was compared against the automatic region growing [2] and strip histogram grid methods [15]. Fig. 3.8 shows example results.

For both scenes in the progressive morphological filtering the maximum window size was chosen to be 30, the cell size was \(1m\) and the initial distance and maximum distance were \(0.5\) and \(3.0m\) respectively. The threshold for the variance was \(0.5m\) and the Euclidean distance threshold was 1.4. Table 3.1 shows that the NORM algorithm out-performed the automatic region growing [2] and strip histogram [15] methods in the Surrey dataset and in the CVM dataset. The metrics of the segmentation evaluation are challenging because of the overwhelming instances of false positives in the situation of an under segmentation. However, even though the segmentation results have relatively low accuracy, it is still possible to correctly classify regions in the false positive segmentation range as long as enough relative shape information is present in the under segmentation.
Figure 3.8: Segmentation comparisons between methods

(a) Automatic Seeded Region Growing Segmentation

(b) Strip Histogram Segmentation

(c) NORM Segmentation
The process of extracting shape-based features from the segmented objects and classifying them is discussed in this chapter. First, the SELF extraction technique is discussed. Next, the classification approach, specifically the cascade of classifiers, is described. Finally, the entire classification algorithm will be evaluated.

4.1 Surface-based Eigen Local Features

This section presents the procedure for creating a 51-element feature vector that is produced by the SELF algorithm as outlined in Fig 1.2.

4.1.1 Histogram of Surface Normals

The first set of features is a twenty-element normalized histogram of the elevation angles of the surface normals contained within the object cluster. These surface normals are the same ones calculated in Chapter 3 and used for the octree segmentation and are based on the local point neighborhoods. By evaluating these elevation angles, it will be possible to obtain a shape signature for each cluster. In the case of a vehicle like a sedan, the object can be divided up into three sections the engine, body, and trunk. Each of these three sections will have distinct surface normals associated
with it and when combined, these combinations of surface normals will be similar to other surface normal combinations associated with sedans in general. Similarly, a commercial building will have a flat root, an almost completely vertical surface normal signature, while the surface normal signature of vegetation is randomly distributed.

### 4.1.2 Point, Curve and Surface

Next, three elements are calculated to describe the global distribution of the points within the segment. First the covariance matrix of the points are calculated as in Eq. 3.6. From this covariance matrix, it is possible to solve for the eigenvalues and eigenvectors. The eigenvectors are grouped in terms of decreasing value, $\lambda'_2 \leq \lambda'_1 \leq \lambda'_0$.

From these eigenvalues, several meaningful relationships can be developed to represent the shape of the cloud [20]. For example, a cloud cluster whose eigenvalues are similarly valued, $\lambda'_0 \approx \lambda'_1 \approx \lambda'_2$ would represent a cloud cluster of randomly scattered points. The point value as expressed in Eq. 4.1 would have a value that is close to one, while the curve and surface, as expressed in Eq. 4.2 and 4.3 respectively, would have low values close to zero. If points lie on a line, similar to an object cluster belonging to a power line, those points would have the eigenvalue signature of $\lambda'_0 \gg \lambda'_1 \approx \lambda'_2$. These object clusters would have a point and surface that are small or near zero while the value for curve would be close to one. Finally, in object clusters that are spread across a plane, similar to a rooftop, the eigenvalues would have the following relationship $\lambda'_0 \approx \lambda'_1 \gg \lambda'_2$. The point and curve values would be low and the surface value would be close to one. These point, curve, and surface values will make up the next three elements of the feature vector.

$$\Lambda_1 = \frac{\lambda'_2}{\lambda'_0}$$  \hspace{1cm} (4.1)

$$\Lambda_2 = \frac{\lambda'_0 - \lambda'_1}{\lambda'_0}$$  \hspace{1cm} (4.2)
\[ \Lambda_3 = \frac{\lambda'_1 - \lambda'_2}{\lambda'_0} \] (4.3)

4.1.3 Length, Width and Height

Next, the physical dimensions of the cloud cluster are considered. In order to get the dimensions of the cloud cluster, a minimum oriented bounding box is constructed around the point cluster. For this minimum oriented bounding box, the length, width and height dimensions are observed and these values are the next three elements of the feature vector.

4.1.4 Histogram of Curvatures

The next feature is a histogram of the curvatures of each point within its local area. For each point within the entire point cloud, a local neighborhood is considered and the eigenvalues for the covariance matrix are calculated, the same way that the surface normals were calculated in Chapter 3. The covariance matrix can be seen in Eq. 4.4. Once the eigenvalues are found where \( \lambda_2 < \lambda_1 < \lambda_0 \) the curvature \( \sigma \) can be found using Eq. 4.5. Once the curvature is found for each point, all points within the object cluster can be considered. An equally spaced, twenty-bin normalized histogram can be formed to represent the object cluster. This will be the next twenty elements of the feature vector.

\[ C = \frac{1}{k} \cdot \sum_{i=1}^{k} (p_i - \bar{p}) \cdot (p_i - \bar{p})^T \] (4.4)

\[ \sigma = \frac{\lambda_2}{\lambda_0 + \lambda_1 + \lambda_2} \] (4.5)
4.1.5 Variance of Histograms

The next two elements of the feature vector are the variance of the surface normal histogram and the curvature histogram.

4.1.6 Ratio of Principal Components

Finally, the last set of features is the ratio of the direction of the eigenvalues, similar to the curvature but on a global level across the entire cluster, the eigen values from the cloud cluster will be used from Section 4.1.2. For each of the three eigenvalues, the ratios can be computed based on Eqs 4.6, 4.7, and 4.8.

\[
\alpha_0 = \frac{\lambda_0'}{\lambda_0' + \lambda_1' + \lambda_2'} \quad (4.6)
\]

\[
\alpha_1 = \frac{\lambda_1'}{\lambda_0' + \lambda_1' + \lambda_2'} \quad (4.7)
\]

\[
\alpha_2 = \frac{\lambda_2'}{\lambda_0' + \lambda_1' + \lambda_2'} \quad (4.8)
\]

4.2 Cascade of Support Vector Machines

Once a feature vector for each object has been calculated, the next step is to use a cascade of classifiers for our classification [21, 22, 23]. A support vector machine (SVM) was chosen because of its accuracy, robustness, and ability to handle high dimensions of data [24, 25, 26]. The decision to use a cascade of classifiers was chosen after the close evaluation of each individual SVM, the cascade approach utilizes the strong classifiers higher in the cascade to help eliminate
the false positives of the weaker classifiers lower in the cascade. A visual example of the cascade of classifiers can be seen in Fig. 4.1. A group of feature vectors corresponding to object segments are given as inputs to a classifier trained to identify vegetation. The linear one-class SVM will give a binary output determining whether the prediction for the given input is vegetation or not. If the prediction is true, that particular object is labeled as vegetation. If the prediction is false, that feature vector is used as the input to the classifier trained for buildings. This continues down the cascade of SVM’s until any remaining segments that are rejected from the power line SVM are labeled as unknown. Once every feature vector is given a label, that label is assigned to each point within the corresponding cluster and a corresponding RGB value is given to represent the class as well, according to Fig. 4.2.

![Figure 4.1: Cascade of classifiers](image)

4.3 Object Identification Evaluation

This section will discuss the two data sets used, as well as present the results qualitatively as well as quantitatively according to the measures of accuracy, precision, specificity, and sensitivity. Finally, the strengths and weaknesses of the algorithm will be evaluated.
4.3.1 Datasets

The classification algorithm is evaluated on two ground truthed datasets. The first of these datasets is from a public data collect in Surrey, British Columbia. The data contains a dense urban scene including commercial and residential buildings, vehicles, vegetation, and complex terrain, with the ground varying up to 45m across the scene. The data was taken in April 2013, when vegetation was in full bloom and significant amounts of overhang and occluded objects occur because of the vegetation. The point density is 15.72 pts/m². The second set of data is the Civil Visual Maps (CVM) data set taken in an urban area near San Francisco, CA. It contains several areas of dense forestry, buildings, power lines, and a complex local terrain, varying up to 33m across the scene. The point density of this scene is 31.04 pts/m². The radius of the local neighborhood for the SELF surface normal and curvature calculations was chosen to be 1 meter.

4.3.2 Evaluation Criteria

The effectiveness of the algorithm in classifying these datasets is evaluated in terms of four criteria: accuracy, precision, sensitivity, and specificity. Accuracy is defined in Eq. 4.9 and can be expressed as the number of true positives and true negatives over the total number of predictions. Precision is defined in Eq. 4.10 and can be expressed as the number of true positives over the number of true positives and false positives. Sensitivity is defined in Eq. 4.11 as the number of true positives over the number of true positives and false negatives. Finally, specificity is defined in Eq. 4.12 as the number of true negatives over the number of true negatives and false positives.

\[
\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \quad (4.9)
\]

\[
\text{Precision} = \frac{TP}{FP + TP} \quad (4.10)
\]

\[
\text{Sensitivity} = \frac{TP}{TP + FN} \quad (4.11)
\]

\[
\text{Specificity} = \frac{TN}{TN + FP} \quad (4.12)
\]
Sensitivity = \frac{TP}{TP + FN} \quad (4.11)

Specificity = \frac{TN}{FN + FP} \quad (4.12)
Table 4.1: Object Classification Results: Accuracy

<table>
<thead>
<tr>
<th>Class</th>
<th>CVM Data Set</th>
<th>Surrey Data Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buildings</td>
<td>93.91</td>
<td>94.62</td>
</tr>
<tr>
<td>Vehicles</td>
<td>–</td>
<td>89.27</td>
</tr>
<tr>
<td>Vegetation</td>
<td>95.97</td>
<td>95.88</td>
</tr>
<tr>
<td>Ground</td>
<td>99.98</td>
<td>99.73</td>
</tr>
<tr>
<td>Power lines</td>
<td>95.09</td>
<td>–</td>
</tr>
<tr>
<td>Totals</td>
<td>96.23</td>
<td>94.87</td>
</tr>
</tbody>
</table>

Table 4.2: Object Classification Results: Precision

<table>
<thead>
<tr>
<th>Class</th>
<th>CVM Data Set</th>
<th>Surrey Data Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buildings</td>
<td>92.98</td>
<td>93.95</td>
</tr>
<tr>
<td>Vehicles</td>
<td>–</td>
<td>88.98</td>
</tr>
<tr>
<td>Vegetation</td>
<td>96.90</td>
<td>92.14</td>
</tr>
<tr>
<td>Ground</td>
<td>94.24</td>
<td>91.74</td>
</tr>
<tr>
<td>Powerlines</td>
<td>89.80</td>
<td>–</td>
</tr>
<tr>
<td>Totals</td>
<td>93.48</td>
<td>91.70</td>
</tr>
</tbody>
</table>

Table 4.3: Object Classification Results: Specificity

<table>
<thead>
<tr>
<th>Class</th>
<th>CVM Data Set</th>
<th>Surrey Data Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buildings</td>
<td>97.01</td>
<td>94.56</td>
</tr>
<tr>
<td>Vehicles</td>
<td>–</td>
<td>96.61</td>
</tr>
<tr>
<td>Vegetation</td>
<td>96.08</td>
<td>98.10</td>
</tr>
<tr>
<td>Ground</td>
<td>98.08</td>
<td>99.57</td>
</tr>
<tr>
<td>Powerlines</td>
<td>99.90</td>
<td>–</td>
</tr>
<tr>
<td>Totals</td>
<td>97.76</td>
<td>97.21</td>
</tr>
</tbody>
</table>

Table 4.4: Object Classification Results: Sensitivity

<table>
<thead>
<tr>
<th>Class</th>
<th>CVM Data Set</th>
<th>Surrey Data Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buildings</td>
<td>83.97</td>
<td>97.63</td>
</tr>
<tr>
<td>Vehicles</td>
<td>–</td>
<td>93.50</td>
</tr>
<tr>
<td>Vegetation</td>
<td>95.57</td>
<td>86.32</td>
</tr>
<tr>
<td>Ground</td>
<td>98.61</td>
<td>96.32</td>
</tr>
<tr>
<td>Powerlines</td>
<td>95.09</td>
<td>–</td>
</tr>
<tr>
<td>Totals</td>
<td>93.31</td>
<td>93.44</td>
</tr>
</tbody>
</table>
4.3.3 Uncertainties, Errors and Accuracies

Overall the results in Tables 4.1 to 4.4 show that the classification algorithm was effective. Table 4.1 shows an accuracy of 94.87% in the CVM data set and 96.23% in the Surrey Data set. The ground classification by morphological filtering was extremely effective, identifying 99.98% and 99.73% of points in the CVM and Surrey datasets respectively. These results can be seen on the CVM dataset in Figs. 4.3, 4.4 and 4.5. The power line and vegetation classes also did well, with accuracies in the mid 90s for all cases. The building class achieved good accuracy results in the Surrey dataset but only average results in CVM dataset. One possibility for this is a commission error in the progressive morphological filter. Overall the CVM data set has much larger buildings than the Surrey data set, making it possible for errors to occur if the maximum window size of the progressive morphological filter is too small.
Figure 4.3: Object classification results on CVM dataset: Area 1

Figure 4.4: Object classification results on CVM dataset: Area 2
Figure 4.5: Object classification results on CVM dataset: Area 3
Additionally the accuracy of the vehicles in the Surrey dataset was high, but lower than other classes, this can be accounted for visually in Figs. 4.6, and 4.7 where it can be seen that there is a significant amount of lower vegetation and landscaping that is being classified in the realm of vehicles, this is understandable because it is of a similar size and has a curvature that resembles that of a vehicle. This is a definite area of improvement for future work on this algorithm.

Figure 4.6: Object classification in Scene 1 of the Surrey dataset. Notice the misclassification of low vegetation and landscaping

Table 4.2 shows the precision results for the proposed algorithm. Precision is an especially important metric because it takes into effect the negative impacts of false positives. Because of the nature of this research, military applications and pipeline threat detection, the presence of false positives are extremely detrimental, the higher the precision, the better the algorithm will be for this
Figure 4.7: Object classification in Scene 2 of the Surrey dataset. Notice the misclassification of low vegetation and landscaping.

particular application. In examining the precision results on the two data sets the trends between precision and accuracy are similar. The vegetation and ground classes have high precision in both data sets, while there is a slight deterioration in the building and power line classes.
Figure 4.8: Classification on CVM dataset. Some under segmentation can be seen in the upper left hand corner of the scene.
Figure 4.9: Classification results in Surrey dataset after adding contextual visual information
The final evaluation metrics are sensitivity and specificity, shown in Tables 4.3 and 4.4. The specificity results are all extremely high with the exception of some buildings and vegetation results in the CVM and Surrey datasets respectively. From a visual examination of the results, this is probably due to a segmentation error. Often in an urban or suburban scene the vegetation and buildings are very close in proximity. This can affect the surface normal calculation and lead to region merging too aggressively, which leads to false negatives and false positives in relation to one of the objects. An example of this under segmentation can be seen in Fig. 4.8. Overall, the results of the classification algorithm were successful, contributing a significant amount of contextual information to the scene as can be seen in Fig. 4.9. In this particular instance it becomes apparent how difficult it can be to visually parse this information without some additional contextual information. In Fig. 4.9 (a) and (c) it is extremely difficult to pick out the vehicle under the vegetation. (b) and (d) exhibit how the algorithm can make it easier to gain key information. Fig 4.10 shows another scene where significant contextual information is added.
Figure 4.10: Classification results in CVM dataset after adding contextual visual information
CHAPTER V

CONCLUSIONS AND FUTURE WORK

The goal of this thesis research was automatic segmentation and detection of objects within LiDAR point clouds. A novel segmentation method named Normal Octree Region Merging (NORM) and a robust feature extraction technique named Shape-based Eigen Local Features (SELF) were proposed in an effort to handle the unique challenges of LiDAR data, including instances of occlusion and the distinctive segmentation of objects in close local proximity.

A progressive morphological filter used in conjunction with the proposed NORM segmentation provided the framework for the segmentation technique. The ability to differentiate between objects based on local shape is a novel feature which works to distinguish objects based on a complex feature rather than one distinctive quality like height or local proximity. This segmentation technique was implemented on two data sets with drastically different resolution and proved to outperform other state of the art algorithms. Additionally, even in a situation where there is an error in the segmentation, such as an under segmentation, it is still possible to have an accurate classification and the segmentation error will not necessarily affect the results.

After a segmentation into distinct objects a SELF extraction is performed on each of the segmented clusters. This feature extraction method looks to combine previous research by concentrating on extracting the global features of each cluster, while simultaneously collecting information
about each point on a local level. After the SELF extraction, these feature vectors are classified using a cascade of SVM approach and the results are evaluated on a point-by-point basis.

The results prove that the overall segmentation and classification technique is effective in classifying five classes of objects for these dense urban scenes. While the overall classification is effective there is always room for improvement. Future work involves further development of the feature vector to help distinguish between the vehicle classes and any low vegetation or landscaping. Another area that will be investigated will be the field of adding contextual information to LiDAR point clouds and involves a volume change detection between two point clouds. By evaluating two classified point clouds and examining the changes between time 1 and time 2, not only could changes be detected but by knowing the classification of the points associated with the change, the level of threat associated with the change could be evaluated as well.
BIBLIOGRAPHY


