

Tree Detection in Urban Regions Using Aerial LiDAR and Image Data

John Secord and Avideh Zakhor
EECS Department, U.C. Berkeley
{secord,avz}@eecs.berkeley.edu

Abstract

In this paper, we present an approach to detecting trees in registered aerial image and range data obtained via LiDAR. The motivation for this problem comes from automated 3D city modeling, in which such data is used to generate the models. Representing the trees in these models is problematic because the data are usually too sparsely sampled in tree regions to create an accurate 3-D model of the trees. Furthermore, including the tree data points interferes with the polygonization step of the building roof top models. Therefore, it is advantageous to detect and remove points that represent trees in both LiDAR and aerial imagery. In this paper we propose a two-step method for tree detection consisting of segmentation followed by classification. The segmentation is done using a simple region-growing algorithm using weighted features from aerial image and LiDAR, such as height, texture map, height variation, and normal vector estimates. The weights for the features are determined using a learning method on random walks. The classification is done using weighted support vector machines (SVM), allowing us to control the misclassification rate. The overall problem is formulated as a binary detection problem, and the results presented as receiver operating characteristic curves are shown to validate our approach.

1 Introduction

There has been a great deal of interest in the construction of 3-D models of urban and suburban environments. Traditionally, stereo imaging methods have been used since aerial imagery is readily available and relatively inexpensive to obtain[1]. However, interest in aerial LiDAR data is emerging due to the higher achievable accuracy than in the past, and the increased number of algorithms to process the data. One such approach has been developed in the Video and Image Processing Lab at the University of California, Berkeley over the past five years[9]. This approach involves segmenting aerial LiDAR data, and applying Random Sample Consensus (RANSAC) polygonization algorithm to delineate roofs of individual buildings. While this

approach works well on urban regions with few trees, there is substantial performance degradation in suburban regions with a large number of trees. Therefore, it is conceivable to improve the accuracy and appearance of the overall models by removing all data points corresponding to trees from the aerial imagery and LiDAR data prior to applying the RANSAC-based polygonization algorithm.

There is a large body of work on detection and classification of LiDAR data and aerial imagery. In [2], classification of land cover into four different classes, namely building, tree, grassland, and bare soil is achieved by combining LiDAR data and multispectral images and fusing data based on the theory of Dempster - Shafer. In [3], the authors use a vegetation index called the Normalized Difference, derived from the first and last pulse data to classify both trees and buildings. In [4], a segment based classification method has been developed, which applies fuzzy logic to features of each segment in order to obtain a reliable classification of building, vegetation, and terrain.

In this paper, we propose a new approach to detecting trees in aerial imagery registered with airborne LiDAR data. Our proposed algorithm consists of segmentation followed by classification. The segmentation is a region growing algorithm that grows if adjacent data points have a point-wise similarity above a threshold, and stops if the similarity is below the threshold. The similarity is calculated from weighted features, such as height, color, and normal vector, with the weights being determined by a learning algorithm on a random walk [10].

The segments resulting from the region growing algorithm are then classified using Support Vector Machines (SVM)[6]. Features for the SVM algorithm are calculated for each segment. The resulting segments from the segmentation step vary in size from two points to over a thousand points. Since the statistical properties of a given segment feature such as mean hue, vary widely as a function of segment size, we separate our segments into four different bins depending on their size. The training and classification is then carried out on each bin separately to improve results.

The outline of this paper is as follows: Section 2 explains how the LiDAR data is acquired and stored. In Sections 3

and 4 we describe the segmentation and classification algorithms, respectively. Lastly, Section 5 presents the experimental results.

2 Data Structure

The LiDAR data used in this paper was acquired in a flight in Fall of 2002 by Airborne 1 Incorporated, and represents a $3.5\text{km} \times 3.5\text{km}$ area of Berkeley, California, including residential, commercial, and University of California, Berkeley campus areas. In this paper, we use two subsets of the above data for testing our algorithms: campus data, and residential data. Our previous papers address the problem of 3D urban modeling for commercial regions such as downtown Berkeley[9]. The residential data set is to the southwest of the U.C. Berkeley campus, and consists mostly of residential homes with trees in their yards¹. The tree variety in both residential and campus data sets are varied, and include Eucalyptus, Redwood, Oak, and Pine. The density of the scan points used in this paper is roughly four points per square meter. The scan density is high enough to discern and model large objects, such as buildings, but too sparse to arrive at accurate 3D models for individual trees and irregular geometries to enable photorealistic virtual walk-throughs.

The format of the data acquired by Airborne 1, is an unstructured point cloud, and each point is represented by a simple coordinate system, (x, y, z) . The resulting point cloud is then processed as follows. First, a grid of $0.5\text{m} \times 0.5\text{m}$ is superimposed on the region covered by the airborne LiDAR, and the z value associated with each scan point is re-gridded accordingly. If multiple points fall into the same grid square, the highest and lowest z values are stored. If a grid square has no scan points, it is assigned with the z value from its nearest neighbor.

The next step in processing the LiDAR data is texture mapping the Digital Surface Model (DSM) using aerial photography. The aerial photographs are shot from a helicopter using Nikon digital camera. The resolution of images is between one and two mega pixels, with density of about 12 pixels per square meter. The LiDAR data and aerial photography acquisitions were carried out at different times i.e. within few months of each other, resulting in discrepancies between the LiDAR data and the aerial image. While these discrepancies can affect the tree detection, we anticipate the effect to be minimal for our data set.

3 Segmentation

3.1 The Segmentation Algorithm

Our proposed segmentation algorithm selects a point, p , in the data that has not been previously assigned to a seg-

¹To view the relative locations of residential, campus, and downtown Berkeley data, interested reader can view the 3D models of these areas inserted into Google Earth by downloading the following: <http://www-video.eecs.berkeley.edu/~avz/DowntownResidentialCampus.kmz>

ment, and assigns a new segment identifier to it. The algorithm then analyzes the eight neighboring points of p . If a neighbor point to p already belongs to a segment, the algorithm moves on to the next neighboring point. When it identifies a neighboring point not yet assigned to a segment, it computes the similarity between it and p . The similarity between points i and j is defined by [10]:

$$S_{ij} = e^{-(\mathbf{fv}_i - \mathbf{fv}_j)^T * \text{diag}(\lambda) * (\mathbf{fv}_i - \mathbf{fv}_j)}, \quad (1)$$

where \mathbf{fv}_i is a vector of features for point i , and $\text{diag}(\lambda)$ is a matrix whose diagonal entries are the weights in the vector λ for each feature, to be discussed shortly.

If the similarity is greater than the threshold the neighbor is added to the segment p belongs to. The algorithm continues until all eight neighboring points of p are processed, after which it iterates the above steps for each of the points newly added to the segment until no new points are added to this particular segment. Then, the algorithm once again, identifies a data point that has not been assigned to a segment, and continues this process until all points in the data set have been assigned to a segment.

The parameters in the above algorithm include features, the associated weights and the similarity threshold. Ideally, the features should correspond to physical quantities that help identify differences between tree and non-tree data points. The weights should be derived to provide more emphasis to the traits that are the most distinguishing, and less emphasis to those that are less important. Adjusting the threshold controls the average segment size. On average, the lower the threshold, the larger the segments.

3.2 Feature Selection

Feature selection is important as meaningful features facilitate accurate segmentation of the data. We have chosen three features, hue (h), saturation (s), and value (v) from the aerial imagery, and four features from the LiDAR data, namely height value (z), local height variation (hv), and the x , and y component of a normal vector denoted by n_x and n_y .

Height variation is calculated as the difference between the maximum and minimum height value over a $1.5\text{m} \times 1.5\text{m}$ area[8]. This is a meaningful feature as it is common for the laser from the LiDAR to pierce the top canopy of a tree, and reflect off a lower part of the tree or even the ground. This results in a larger height variation for trees than for a solid object, such as a building. n_x and n_y are estimated using finite differences. Thus, the resulting feature vector for each point is given by $\mathbf{fv} = [z \ h \ s \ v \ hv \ n_x \ n_y]^T$. We now need to find the optimal weights to combine these features.

3.3 Feature Weights

Since, no known methods for optimizing the feature weights in the region-growing algorithm exist, we proceed

indirectly. Parameter optimization methods, also known as learning methods, do exist for other segmentation algorithms; such as spectral clustering. There are a number of approaches to parameter learning for this class of segmentation [10, 5]. The learning method detailed in [10], has been applied to a specific spectral clustering algorithm known as normalized cuts [12]. We have chosen the learning method in [10] to arrive at a set of feature weights to be used in our region growing segmentation algorithm.

The framework for the learning method for normalized cuts is as follows [10]: A similarity measure is defined as a binary operator on pairs of points. A similarity matrix S may then be constructed for the data set of interest, I , where entry s_{ij} corresponds to the similarity between points i and j . In our case, the data set is the LiDAR data, and the similarity measure is identical to the one used in the region-growing segmentation as defined in (1). This assignment results in similarity matrix, $S = [S_{ij}]$. Each row in the similarity matrix is then normalized such that each row sums to one. The "normalized" similarity matrix therefore satisfies the requirements of a stochastic matrix, and consequently may be treated as the transition probability matrix, P , of a discrete-time Markov chain.

An "ideal" or target transition probability matrix, \hat{P} , is defined as:

$$\hat{P}_{ij} = \begin{cases} 0, & j \notin A \\ \frac{1}{|A|}, & j \in A \end{cases} \quad (2)$$

where point i is assumed to belong to segment A with $|A|$ elements. The segment A corresponds to either the tree segment or the non-tree segment in the training data which is assumed to be manually generated. Given the observed transition probability matrix P , and the target transition probability matrix \hat{P} described above, we minimize of the Kullback-Leibler (KL) divergence between the two to obtain feature vector weights. Since \hat{P} is fixed, minimizing the KL divergence simplifies to maximizing the cross entropy between P and \hat{P} , i.e. maximize J , where

$$J = \sum_{i \in I} \frac{1}{|I|} \sum_{j \in I} \hat{P}_{ij} \log P_{ij}. \quad (3)$$

We use a standard gradient descent method to maximize J with respect to the weight parameters λ . The gradient is calculated by,

$$\frac{\partial J}{\partial \lambda_n} = \frac{1}{|I|} \left(\sum_{ij} \hat{P}_{ij} \mathbf{fv}_{ij}^n - \sum_{ij} P_{ij} \mathbf{fv}_{ij}^n \right) \quad (4)$$

where λ_n and \mathbf{fv}_{ij}^n are the n^{th} elements of λ and $\mathbf{fv}_i - \mathbf{fv}_j$ respectively.

4 Classification

4.1 The SVM Algorithm

To classify the resulting segments from the region growing algorithm, we use the SVM algorithm. The SVM was originally proposed by Boser, et. al. [6], and has been a centerpiece in much work on classification and regression. Since the SVM algorithm requires a feature vector for each item to be classified, a new feature vector needs to be defined for each segment obtained in the segmentation step.

The specific SVM algorithm used is weighted SVM. Weighted SVM makes it is possible to traverse a receiver operating characteristic (ROC) curve, exploring the trade-off between the false and true positives.

4.2 Feature Selection

To use SVM to classify, we need to define the features for each segment; segment i being assigned feature vector \mathbf{x}_i . Even though we have already defined a feature vector for each data point in a segment, we now need to define a feature vector for each segment. Our approach is to use the statistics of the point-wise features over a segment to arrive at the features for that segment. Examples of segment features include the variance of the height z , or the mean of the hue h . Since, the segment sizes range from two to over a thousand data points, we divide the segments into four bins as follows: first bin has segments with 2 - 4 points, second with 5 - 10 points, third with 11 - 30 points, and fourth with 31 or more points.

Also, it is possible to compute averages of a given feature over either an entire segment or to first compute it over an $n \times n$ window followed by averaging over an entire segment. For bins 1 and 2 we have empirically found the later approach, with $n = 3$, to outperform the former approach, as it results in better spatial separation in our segment feature space. For bins 3 and 4 however, we only compute the average over each point in a segment.

Our feature vector for each segment has a total of five features, given by the mean of the hue μ_h , saturation μ_s , value μ_v , and height variation μ_{hv} , and the variance of the height var_z . Intuitively speaking, (x, y) location of the points in a segment are not useful as trees and buildings are nearly uniformly distributed over the ground. We have also empirically found the normal vector data not to be useful in the classification process.

4.3 SVM Implementation

We implement the SVM algorithm with the LIBSVM software [7]. We use a soft margin classifier, and optimize a standard cost function in which a parameter C is used to trade off between margin maximization and training error minimization. We select the radial basis function, or Gaussian kernel for SVM:

$$K(x_i, x_j) = e^{-\gamma \|x_i - x_j\|^2} \quad (5)$$

To obtain optimal values of C and γ we perform a grid search of 10×10 in which we compute the cross-validation accuracy for each point in the grid. The software is run in parallel on a four processor server. Cross-validation accuracy and training, over the complete (C, γ) grid is calculated within two hours. For data sets of roughly 700,000 points, run nearly 100 times to generate ROC curves, the classification is completed in less than an hour.

In a n -fold cross-validation, the training data is first divided into n equally sized subsets. The classifier is trained on $n - 1$ subsets and then tested on the remaining subset. This is done over all combinations such that every subset is tested once by a classifier trained on the other $n - 1$ subsets. The cross-validation accuracy (CVA) is the percentage of points that are correctly classified over all the subsets when they were used as the testing subset. In the binary case, one can break down the cross-validation accuracy into its two components: correct classification of class -1 and correct classification of class +1. Let $y_i \in \{\pm 1\}$ be the true class of training data point i and $\hat{y}_i \in \{\pm 1\}$ be the class assigned to the point by the SVM. Then the CVA can be written as follows:

$$\text{CVA} = \left(\frac{\sum_{\hat{y}_i \text{ s.t. } y_i = -1} 1(\hat{y}_i = y_i)}{\sum_{y_i} 1(y_i = -1)} \right) \left(\frac{\sum_{y_i} 1(y_i = -1)}{\text{Total \# of points}} \right) + \left(\frac{\sum_{\hat{y}_i \text{ s.t. } y_i = 1} 1(\hat{y}_i = y_i)}{\sum_{y_i} 1(y_i = 1)} \right) \left(\frac{\sum_{y_i} 1(y_i = 1)}{\text{Total \# of points}} \right) \quad (6)$$

The first term represents the true negative percentage and the second term represents the false negative term. These are the actual terms we are interested in as we pose our problem as a binary detection problem. For the grid search over C and γ we calculate the cross-validation accuracy as the true negative and true positive terms. The (C, γ) grid is composed of exponentially growing values of C and γ , e.g., $C = 2^{-5}, 2^{-3}, \dots, 2^{15}$, $\gamma = 2^{-15}, 2^{-13}, \dots, 2^3$. The grid search is necessary because the CVA over (C, γ) set is not convex. To reduce computation time, the grid search is done in parallel. Furthermore, a course grid is explored first, followed by a finer grid in regions that have better cross-validation accuracy. Once C and γ for each bin have been selected, they are used in conjunction with the entire training set, to determine the optimal hyperplane for each bin to be used as the classifier. The resulting model parameters for each bin are then used to classify all the segments.

5 Results and Discussion

We run our algorithm as follows: the LiDAR is pre-processed by Airborne 1 Inc. to define data points as ground or structure. Two data sets are constructed. The

first is a $600\text{m} \times 700\text{m}$ area over a suburban region of Berkeley, California, and the second is a $700\text{m} \times 680\text{m}$ area over the University of California, Berkeley campus. For these two data sets, a ground truth is constructed by hand for performance characterization purposes. Only structure points are considered in our algorithm. For each structure point, i , a feature vector fv_i is created. Then the segmentation is carried out using optimal weights given by $z = .16, \text{hue} = 0.00005, \text{saturation} = .1, \text{value} = .0001, \text{heightvariation} = .52, \text{normal}_x = .1, \text{normal}_y = .10$ determined from the learning spectral clustering algorithm described in Section 3. As seen, height variation and height are the most influential features, while hue and value are least important. DSM and the segmentation results for residential data are shown in Fig. 1. Large segments corresponding to building structures are easily discernable in Fig. 1(b). These segments are then sorted into bins, new features are determined for each segment, and the SVM algorithm is used to classify the segments as described in Section 4.

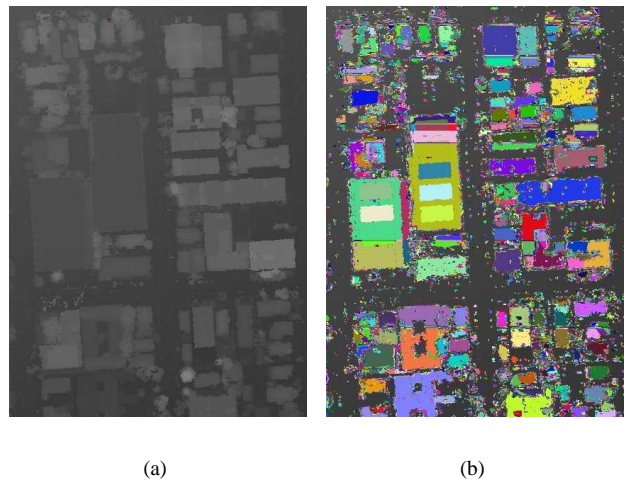


Figure 1. Residential data set DSM visualization (a) original and (b) after segmentation.

The segment data for each bin is shown in Table 1. The training data is taken from the ground truth constructed for each data set. As seen in the fourth and last rows of Table 1, the training segments represent at most 4.6% of the total number of segments. Since we are doing a binary classification, we present it as a binary detection problem and calculate the ROC curves. A false positive is considered a non-tree point being classified as a tree point, and the true positive is a tree point being classified as a tree point. The computation of the ROC curves is done by adjusting the parameters in the weighted SVM algorithm [11]. Each point

Data Set 1	All Segs	Bin 1	Bin 2	Bin 3	Bin 4
Total Segs	83331	60242	15257	5653	2179
Non-tree Segs	53871	37535	10430	4160	1746
Tree Segs	29460	22707	4827	1493	433
Training Segs	1400	1000	200	100	100
Data Set 2	All Segs	Bin 1	Bin 2	Bin 3	Bin 4
Total Segs	113137	84479	19235	6844	2579
Non-tree Segs	42416	28174	8741	3805	1696
Tree Segs	70721	56305	10494	3039	883
Training Segs	1500	1000	300	100	100

Table 1. Segment data for both data sets

on the ROC curve can be calculated independent of the others, reducing computation time by parallel processing.

The ROC curves for each bin in the residential and the campus data set are shown in Figs. 2(a) and 2(b), respectively. As seen, the classification mostly performance improves with bin size. This is to be expected since the smaller segments use fewer data points for segment feature calculations, resulting in noisier features. In addition, when the false positive rate is low, the larger bins for the campus data outperform the residential. This can be explained by considering that the campus data consists of buildings that are far from each other and from trees, and hence results in more reliable segmentation than residential data, particularly for larger bin sizes. For comparison, a point-wise technique similar to the one described in [8], is examined by applying SVM classifiers directly to features of individual data points in our data set on a point-wise basis, bypassing the segmentation step. The ROC curves comparing the segmentation followed by classification method, and the point-wise method are shown in Figs. 3(a) and 3(b), for the residential and campus data sets respectively. It can clearly be seen from these figures that the segmentation followed by classification outperforms the point-wise method.

A natural question that arises is whether or not point-wise SVM data can be used to correct misclassified segments, thereby improving overall performance. Another possibility is to use point-wise SVM classification to trigger misclassified segments. We have empirically found that in practice, point-wise SVM data cannot be used to significantly improve overall performance of our segmentation based scheme [11].

References

- [1] D. Frère, J. Vandekerckhove, T. Moons, and L. V. Gool, "Automatic modeling and 3d reconstruction of urban buildings from aerial imagery" In *IEEE International Geoscience and Remote Sensing Symposium Proceedings*, pp. 2593–2596, Seattle, WA, 1998.
- [2] F. Rottensteiner, J. Trinder, S. Clode, and K. Kubik, "Using the Dempster Shafer Method for the fusion of LiDAR Data and Multi-spectral Images for Building Detection" *Information Fusion*, 2004, Vol. 6, No. 4, pp. 283-300.
- [3] H. Arefi, M. Hahn, and J. Lindenberger, "LiDAR Data Classification with Remote Sensing Tools" *ISPRS Joint Workshop on Challenges in Geospatial Analysis, Integration and Visualization II*, pp. 131-136, Stuttgart, Germany, 2003.
- [4] D. Tovarí, T. Vogtle "Object Classification in Laser Scanning Data" *Proceedings of the ISPRS working group VIII/2 on Laser-Scanners for Forest and Landscape Assessment* Freiburg, Germany October 2004.

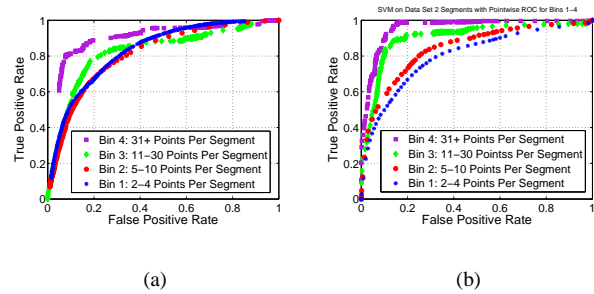


Figure 2. ROC curves obtained by segmentation followed by classification for different bin sizes for (a) the residential data set and (b) the campus data set.

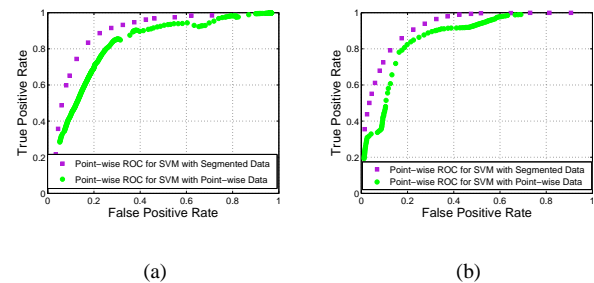


Figure 3. ROC curves comparing segmentation followed by classification versus point-wise classification for all bins combined for (a) the residential data set and (b) the campus data set.

- [5] F. R. Bach and M. I. Jordan, "Learning spectral clustering" *Advances in Neural Information Processing Systems (NIPS) 16*, 2003.
- [6] B. E. Boser, I. Guyon, and V. Vapnik, "A training algorithm for optimal margin classifiers" *Computational Learning Theory*, pages 144–152, 1992.
- [7] C.-C. Chang and C.-J. Lin, *LIBSVM: a library for support vector machines*, 2001. Software available at <http://www.csie.ntu.edu.tw/~cjlin/libsvm>.
- [8] A. P. Charaniya, R. Manduchi, and S. K. Lodha, "Supervised parametric classification of aerial lidar data" *IEEE Workshop on Real-Time 3D Sensors*, pages 25–32, 2004.
- [9] C. Frueh and A. Zakhor, "Constructing 3d city models by merging ground-based and airborne views" *Conference on Computer Vision and Pattern Recognition*, pages 562–569, 2003.
- [10] M. Mailla and J. Shi, "Learning segmentation with random walk" *Neural Information Processing Systems (NIPS) 2001*, pages 873–879, 2001.
- [11] J. M. Secord, "Tree detection in aerial lidar and image data. Master's thesis, University of California at Berkeley, Berkeley, CA, 2005.
- [12] J. Shi and J. Malik, "Normalized cuts and image segmentation" *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:888–905, Aug. 2000.