TREE DETECTION IN AERIAL LIDAR AND IMAGE DATA

John Secord and Avideh Zakhor

University of California, Berkeley
Electrical Engineering and Computer Science Department
{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 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 are 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 using aerial LiDAR data is beginning to emerge due to the higher achievable accuracy and the increased number of algorithms to processes 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[2]. This approach involves segmenting aerial LiDAR data, and applying RANSAC-like polygonization technique 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.

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 [3].

The segments resulting from the region growing algorithm are then classified using support vector machines (SVM). 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 same features calculated for segments that contain two points and those that contain hundreds of points are usually quite different, segments are separated 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, obtained by Airborne 1 Incorporated, represents a large 3.5km $\times$ 3.5km area of Berkeley, California, including residential, commercial, and University of California, Berkeley campus areas. 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 large objects, such as buildings, but too sparse to model trees and irregular geometries.

The format of the data acquired by Airborne 1, is an unstructured point cloud, and each point represented by a simple coordinate system, (x, y, z). The resulting point cloud is then processed as follows.First, a grid of 0.5m $\times$ 0.5m 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 the z value from its nearest neighbor.

The next step in processing the LiDAR data is texture mapping the DSM using aerial photography. The aerial photographs are shot from a helicopter using a Nikon digital camera. The LiDAR data and aerial photography acquisitions were carried out at different times, 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 points, p, in the data that has not been previously assigned to a segment, and assigns a new segment identifier to it. The algorithm then analyzes the eight

neighboring points of p. If a neighbor point to p is already in a segment, the algorithm moves on to the next neighboring point. When it identifes 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 [3]:

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

where $\mathbf{fv}_i$ is a vector of features for point i, and $diag(\boldsymbol{\lambda})$ is a matrix whose diagonal entries are the weights in the vector $\boldsymbol{\lambda}$ 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[4]. 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,

$$\mathbf{fv} = [z \;\; h \;\; s \;\; v \;\; hv \;\; n_x \;\; n_y]^T \quad (2)$$

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[3, 5]. The learning method detailed in [3], has been applied to a specific spectral clustering algorithm known as normalized cuts [6]. We have chosen the learning method in [3] 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[3]: 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,

$$S = [S_{ij}]. \quad (3)$$

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:

$$\hat{P}_{ij} = \begin{cases} 0, & j \notin A \\ \frac{1}{|A|}, & j \in A \end{cases} \quad (4)$$

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. $\max\ J$, where

$$J = \sum_{i \in I} \frac{1}{|I|} \sum_{j \in I} \hat{P}_{ij} \log P_{ij}. \quad (5)$$

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

$$\frac{\partial J}{\partial \lambda_n} = \frac{1}{|I|} \left(\sum_{ij} \hat{P}_{ij} \mathbf{fv}_{ij}^n - \sum_{ij} P_{ij} \mathbf{fv}_{ij}^n \right) \quad (6)$$

where λ_n and $\mathbf{fv}_{ij}^n$ are the n^{th} elements of $\boldsymbol{\lambda}$ 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 support vector machine (SVM) algorithm. The SVM was originally proposed by Boser, et. al. [7], and has been a centerpiece in much work on classification and regression. Since the SVM algorithm also 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.

Specifically, weighted SVM, with a class specific misclassification parameters $C_{\pm 1}$, is used to classify the segments. By changing the weights of $C_{\pm 1}$ it is possible to traverse a receiver operating characteristic (ROC) curve, exploring the tradeoff 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

Bin Num.	Num. of Data Pts.
1	2-4
2	5-10
3	11-30
4	31+

Table 1. Binning of Data Segments

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 shown in Table 1. 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,

$$x_i = [\mu_h \;\; \mu_s \;\; \mu_v \;\; \mu_{hv} \;\; var_z]^T. \tag{7}$$

Intuitively speaking, (x, y) location of the points in a segment are not useful as trees and building 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[8]. We select the radial basis function, or Gaussian kernel for SVM:

$$K(x_i, x_j) = e^{-\gamma||x_i - x_j||^2} \tag{8}$$

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 $y_i \, \epsilon \, \{\pm 1\}$ be the true class of training data point i and $\hat{y}_i \, \epsilon \, \{\pm 1\}$ be the class assigned to the point by the SVM. Then the CVA can be written as follows:

$$\text{CVA} = \left(\frac{\sum\limits_{\hat{y}_i \; s.t. \; y_i = -1} 1(\hat{y}_i = y_i)}{\sum\limits_{y_i} 1(y_i = -1)} \right) \left(\frac{\sum\limits_{y_i} 1(y_i = -1)}{\text{Total \# of points}} \right) + \left(\frac{\sum\limits_{\hat{y}_i \; s.t. \; y_i = 1} 1(\hat{y}_i = y_i)}{\sum\limits_{y_i} 1(y_i = 1)} \right) \left(\frac{\sum\limits_{y_i} 1(y_i = 1)}{\text{Total \# of points}} \right) \tag{9}$$

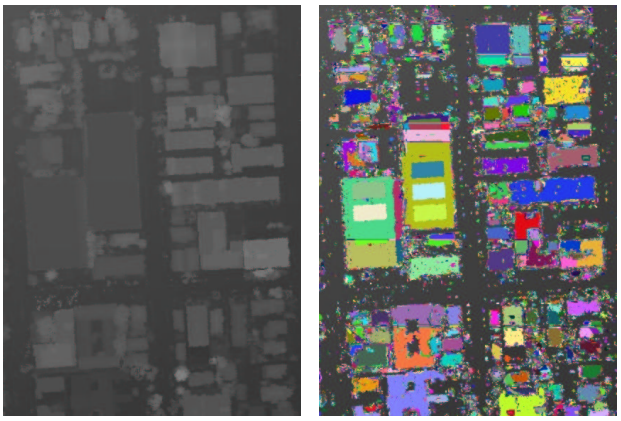
The first term represents the true negative percentage and the second term represents the false negative term. These are the actual terms we are interested in as we pose our problem as a binary detection problem. For the grid search over C and γ we calculate the cross-validation accuracy as the true negative and true positive terms. The (C, γ) grid is composed of exponentially growing values of C and γ, e.g., $C = 2^{-5}, 2^{-3}, ..., 2^{15}$, $\gamma = 2^{-15}, 2^{-13}, ..., 2^{3}$. Note, the grid search is necessary because the CVA over (C, γ) set is not convex. The grid search is done in parallel, reducing the computation time. Furthermore, to decrease computation time, 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 preprocessed by Airborne 1 Inc. to define data points as ground or structure. Two data sets are constructed. The first is a 600m $\times$ 700m area over a suburban region of Berkeley, California, and the second is a 700m $\times$ 680m 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 $\mathbf{fv}_i$ is created. Then the segmentation is carried out using optimal weights determined from the learning spectral clustering algorithm, as described in Section 3. 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.

The segment data for each bin is shown in Table 2. The training data is taken from the ground truth constructed for each data set. As seen in the fourth and last rows of Table 2, 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 C_{-1} and C_{+1} parameters centered around the optimal C found by the grid search over the (C, γ) parameter space as described in Section 4.3. Each point 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 are shown in Figs. 2(a) and 2(b), respectively. As seen, the classification performance improves with bin size. This is

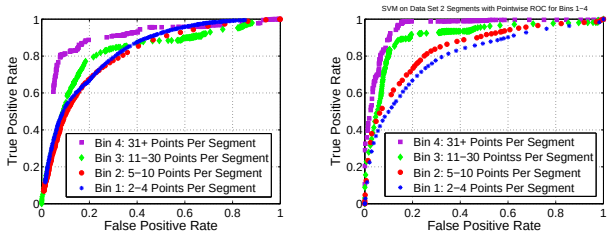


(a) (b)

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

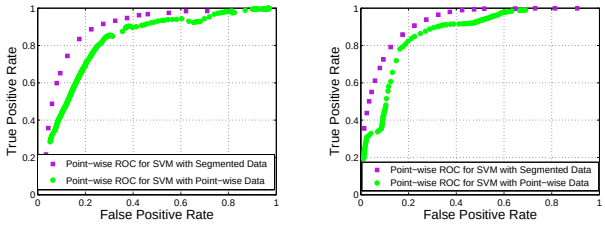
Data Set 1	All Segs	Bin 1	Bin 2	Bin 3	Bin 4
Total Segs	83331	60242	15257	5653	2179
Non-tree Segs	53871	37535	10430	4160	1746
Tree Segs	29460	22707	4827	1493	433
Training Segs	1400	1000	200	100	100
Data Set 2	**All Segs**	**Bin 1**	**Bin 2**	**Bin 3**	**Bin 4**
Total Segs	113137	84479	19235	6844	2579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

Table 2. Segment data for both data sets



(a) (b)

Fig. 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.



(a) (b)

Fig. 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.

to be expected since the smaller segments use fewer data points for segment feature calculations, resulting in noisier features. For comparison, a point-wise technique similar to the one described in [4], 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 [9].

6. REFERENCES

[1] David Frère, Jan Vandekerckhove, Theo Moons, and Luc Van Gool, "Automatic modeling and 3d reconstruction of urban buildings from aerial imagery," in *IEEE International Geoscience and Remote Sensing Symposium Proceedings*, Seattle, WA, 1998, pp. 2593–2596.

[2] Christian Frueh and Avideh Zakhor, "Constructing 3d city models by merging ground-based and airborne views," in *Conference on Computer Vision and Pattern Recognition*, 2003, pp. 562–569.

[3] Marina Maila and Jianbo Shi, "Learning segmentation with random walk," in *Neural Information Processing Systems (NIPS) 2001*, 2001.

[4] Amin P. Charaniya, Roberto Manduchi, and Suresh K. Lodha, "Supervised parametric classification of aerial lidar data," in *IEEE Workshop on Real-Time 3D Sensors*, 2004, pp. 25–32.

[5] Francis R. Bach and Michael I. Jordan, "Learning spectral clustering," in *Advances in Neural Information Processing Systems (NIPS)*, 2003.

[6] Jianbo Shi and Jitendra Malik, "Normalized cuts and image segmentation," vol. 22, pp. 888–905, Aug. 2000.

[7] Bernhard E. Boser, Isabelle Guyon, and Vladimir Vapnik, "A training algorithm for optimal margin classifiers," in *Computational Learing Theory*, 1992, pp. 144–152.

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

[9] John Michael Secord, "Tree detection in aerial lidar and image data," M.S. thesis, University of California at Berkeley, Berkeley, CA, 2005.