2D TREE DETECTION IN LARGE URBAN LANDSCAPES USING AERIAL LIDAR DATA

George Chen and Avideh Zakhor

Video and Image Processing Lab, EECS Department, University of California, Berkeley

ABSTRACT

We present a scalable approach to tree detection in large urban landscapes using aerial LiDAR data. Similar to our previous work in 2006, our current method consists of segmentation followed by classification. However, unlike our previous work, the current approach does not use color information or aerial imagery, and hence is more generally applicable. Also, our current approach has been successfully tested on two very large datasets, which are many orders of magnitude larger than the dataset used in 2006. Specifically, we use a North American dataset, containing 125 million LiDAR returns over 3 km^2, and a European dataset, containing 200 million LiDAR returns over 7 km^2. For both datasets, we report precision and recall rates of over 95%.

Index Terms— Image classification, image segmentation, object detection, laser radar

1. INTRODUCTION

There has been a great deal of interest in the construction of 3D 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 process the data. One such approach involves segmenting aerial LiDAR data, and applying RANSAC-like polygonization technique to delineate roofs of individual buildings [2]. 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 [2].

Broadly speaking, aerial LiDAR classification approaches can be divided into two classes: pointwise classification [3, 4], and segmentation followed by classification [5, 6, 7]. Charaniya and Lodha perform pixel-wise, four-category classification using expectation-maximization with features such as height variation and return intensity, computed over a 2.5D height map [3]. Lodha et al. improve on this work by introducing new features and using AdaBoost [4]. They report results with approximately 90% accuracy over their Santa Cruz dataset. Segmentation followed by classification is generally considered to be superior to point-wise classification since it enforces spatial coherency through the segmentation process. Along these lines, Secord and Zakhor identify trees in aerial LiDAR point clouds over the city of Berkeley using normalized cut segmentation as a pre-processing step to supervised segment-wise classification [7]. Forlani et al. use two region growing segmentations followed by rule-based, segment-wise classification to identify three classes of LiDAR returns [5]. Their work also remains in the 2.5D domain, reporting results also in the 90% range over the city of Pavia, Italy. Zingaretti et al. [6] present techniques for automatically extracting the rules used by Forlani et al. [5].

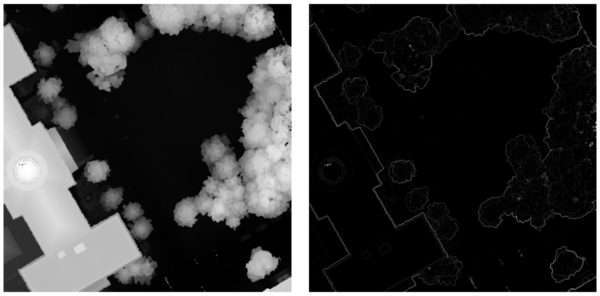

(a) (b)

Fig. 1. Example 2.5D depth images for a pixel size of 20cm^2: (a) z_{max}; (b) z_{edge}.

In this paper, we propose a two step approach based on segmentation followed by classification to tree detection in aerial LiDAR data. Our approach is more general than the existing ones in that it does not take advantage of color imagery or colored LiDAR returns, or image intensity of any sort. In addition, the precision recall performance of our approach has been shown to be considerable higher than existing approaches, i.e. over 95%. This is particularly significant considering that the performance of our approach has been characterized over two large datasets each of which are one to two orders of magnitude larger than the largest datasets reported in the literature. Finally, we have shown our approach to be general in that when trained on a separate dataset than the one being tested, it still results in reasonably low error rates of around 15-18%.

The outline of this paper is as follows. Sections 2 and 3 describe our approach to segmentation and classification respectively. We present results and conclude in Section 4.

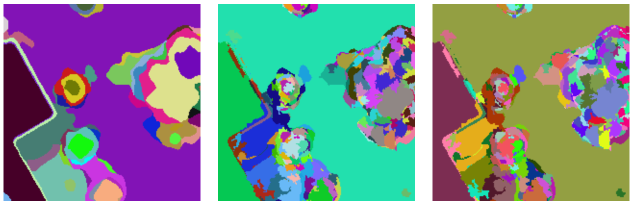
2. SEGMENTATION

We begin by projecting our 3D point cloud onto a 2.5D depth image with fixed-size pixels in the (x, y) plane. For our data, we use 20cm^2 pixels. In doing so, many LiDAR points might land onto a particular pixel. We compute the average and maximum heights of the LiDAR points in each pixel in order to form two 2.5D depth images z_{mean} and z_{max} respectively. An example z_{max} depth image is shown in Fig. 1(a), whereby the trees can be easily detected by the human eye.

Similar to [5], we perform two separate segmentations, one on z_{max} and the other on z_{edge}. The image corresponding to edge density, z_{edge}, is obtained from z_{max} by applying the Roberts cross gradient operator [8]:

$$\begin{aligned} z_{edge}(i,j) &= |z_{max}(i,j) - z_{max}(i+1,j+1)| \\ &\quad + |z_{max}(i+1,j) - z_{max}(i,j+1)| \,. \end{aligned}$$

An example z_{edge} image is shown in Fig. 1(b). We have empirically found that segmentation based on z_{max}, shown in Fig. 2(a), results



(a) (b) (c)

Fig. 2. Typical segmentation using (a) z_{max}; (b) z_{edge}; (c) intersection of (a) and (b).

in under-segmentation, while segmentation based on z_{edge}, shown in Fig. 2(b), results in over-segmentation. In the former case, objects of similar heights such as a tree next to a building are segmented together; in the latter case, a building or a tree is fragmented into too many segments. In [5], a rule-based scheme is used to merge tiny segments which are then declared to be a tree; this results in misclassification of many building edges as trees. In contrast, we apply statistical machine learning techniques, to be described shortly, to classify our segments as tree and non-tree.

In order to make our approach scale to large regions, we use seed-based region growing segmentation and merge segments smaller than a pre-specified threshold with neighboring segments. After both segmentations are completed, pixel (i, j) in each 2.5D depth image belongs to two segments, one from the z_{max} segmentation and the other from the z_{edge} segmentation. Next, we intersect the two segmentations as shown in Fig. 2(c) in order to arrive at one set of segments to be used in the classification process. Specifically, each segment σ resulting from the intersection process belongs to a unique segment σ_{max} in z_{max} segmentation and a unique segment σ_{edge} in the z_{edge} segmentation.

3. CLASSIFICATION

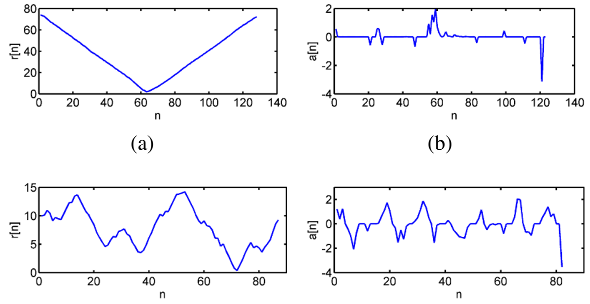
We use a random forest classifier [9] to classify segments from the above double segmentation process. For each segment σ, we use the following feature vector:

1. Height variation of σ_{max}
2. Height variation of σ_{edge}
3. Edge density of σ_{max}
4. Edge density of σ_{edge}
5. Contour non-linearity of σ_{max}
6. Contour non-linearity of σ_{edge}

To compute height variation feature for each segment, we first begin by removing noise in z_{max} and z_{edge} via median filtering. Next we compute a new signal z_{std}, which is the result of applying a 3 × 3 local standard-deviation filter to z_{mean}. We use z_{mean} here rather than z_{max} since z_{mean} captures information about more LiDAR returns in a particular pixel than z_{max} does. The height variation of a segment is then simply the average z_{std} value of pixels in that segment. Likewise, the edge density of a segment is defined to be the average z_{edge} value of pixels in that segment.

The contour non-linearity feature is used in order to differentiate between building edges and trees. Similar to trees, building edges have large height variation and edge density. However, building edges have contours that are generally rectangular or polygonal in shape while trees have round contours.

To define contour non-linearity, consider first the function $r[n]$ that measures the Euclidean distance from each pixel along a segment's contour to the segment centroid. For trees, we have empirically found $r[n]$ to be a relatively smooth function, as shown in Fig. 3(c), while for building segments, it tends to be piecewise linear with sharp corners, as shown in Fig. 3(a). To amplify the effect of these manmade, sharp corners, we estimate the second derivative $a[n]$ of $r[n]$ as follows:



(a) (b)

(c) (d)

Fig. 3. Example contour functions: (a) building contour $r[n]$; (b) building contour $a[n]$; (c) tree contour $r[n]$; (d) tree contour $a[n]$.

$$a[n] = r[n] - 2r[n-1] + r[n-2] .$$

Examples of $r[n]$ and $a[n]$ for a building contour and a tree contour are shown in Figs. 3(b) and 3(d) respectively. We find that $a[n]$ has a few tall spikes for a building contour and numerous tall spikes for a tree. The numerous spikes in $a[n]$ for trees have to do with tree contours often being random in shape. Moreover, we have found that even for a circular tree contour, spatial quantization of pixels results in $a[n]$ having numerous spikes. Rectangular or polygonal contours typically do not exhibit this effect. The segment contour non-linearity measure is then defined to be

$$\frac{1}{|C|} \times \sum_{n \in C} \mathbb{I}\left(|a[n]| > \tau_{linearity}\right),$$

where C is the set of pixels belonging to the segment's contour and $\mathbb{I}$ is an indicator function that is 1 when $|a[n]|$ is larger than some pre-specified threshold $\tau_{linearity}$ and 0 otherwise.

4. RESULTS AND CONCLUSION

We use two different datasets, D1 and D2 to validate our approach experimentally. D1 is a public domain dataset [10] and captures range information over a North American city. It contains approximately 125 million LiDAR returns with an average spatial density of 65 returns/m^2. D2 captures range information over a European city and contains approximately 200 million LiDAR returns with an average spatial density of 25 returns/m^2. Both datasets are split into 100m × 100m tiles of data.

To create training data for our random forest classifier, we label 2.5D z_{max} depth images that have been generated from a subset of our aerial LiDAR data. This subset is chosen by taking tiles from many different parts of the cities. Most tiles are from urban areas, which are more challenging to classify than rural areas. Then, using photo editing software, we identify each pixel in the ground truth image as tree or non-tree. The label for each segment is based on a majority vote of labels of pixels in the segment.

For each dataset, we train a separate random forest classifier. Our training set for D1 consists of 21,170 segments resulting from the intersection of our double segmentation. Of these, 12,787 segments are identified as trees in the ground truth. These segments comprise approximately $\frac{1}{12}$ or 0.2km^2 of the D1 dataset. Our training set for D2 is composed of 22,700 segments, of which 12,763 are

	Precision	Recall	Overall Error
D1 Cross Validation	95.0%	96.5%	5.2%
D2 Cross Validation	97.4%	97.2%	3.0%
Train on D1, test on D2	78.9%	92.6%	18.1%
Train on D2, test on D1	88.0%	86.2%	15.4%

Table 1. Classifier accuracy rates.

	Predicted:	Tree	Non-tree
Truth:	Tree	12,342	445
	Non-tree	652	7,734

Table 2. Confusion matrix for D1 cross validation.

	Predicted:	Tree	Non-tree
Truth:	Tree	12,413	355
	Non-tree	329	9,607

Table 3. Confusion matrix for D2 cross validation.

	Predicted:	Tree	Non-tree
Truth:	Tree	11,821	946
	Non-tree	3,161	6,776

Table 4. Confusion matrix for training on D1 and testing on D2.

	Predicted:	Tree	Non-tree
Truth:	Tree	11,024	1,763
	Non-tree	1,503	6,883

Table 5. Confusion matrix for training on D2 and testing on D1.

identified as trees in the ground truth. These segments comprise approximately $\frac{1}{20}$ or 0.3km^2 of the D2 dataset. For both training sets, we build a random forest with 200 decision trees. For each dataset, we split the training segments into 10 equal-sized bins and perform 10-fold cross validation. We also train on D1 and test on D2, and vice versa.

Precision, accuracy, and overall error for the ground truth portion of both datasets are reported in Table 1. Confusion matrices are reported in Tables 2, 3, 4, and 5. We find that when we train and test on disjoint sets either both from D1, or both from D2, our classifier has a low error of about 3-5%. When we train on one dataset and test on the other, we find that the classifier generalizes reasonably well and achieves error rates of about 15-18%. This is significant since the datasets are from two different continents, each with very different terrain and architecture characteristics. In particular, the North American dataset features lightly urban, hilly terrain with river valleys, high rises, and height variation of 30 meters. The European dataset is mainly flat, densely urban, and contains architecture dating from the 1800's with few high rises. Furthermore, the two datasets were obtained using different acquisition systems: range data from D1 has higher resolution and is substantially less noisy than that of D2.

Additionally, using our training set for D1, we have tested the resulting random forest classifier on the entire D1 dataset excluding the training set. Since this much larger test set—spanning over 2 km^2—is unlabeled, we opt to use visual inspection to assess classification quality. Specifically, we split up the test set into 100m × 100m tiles, and for each tile, we categorize the tile's classification results on an integer scale from 0 to 3. A score of 3 refers to an accurate classification, where nearly every segment in the tile is correctly classified. The classifier successfully captures the trees in the urban scene containing complex buildings, cars, and trees directly abutting buildings. Minor false negatives may occur at the center of select trees due to the flat tree canopies. A typical tile classification given a score of 3 is shown in Fig. 4. A score of 2 refers to a fair classification where the tile is mostly error-free but may contain a select few larger building edge misclassifications or perhaps a large tree was negatively labeled. A typical example of a tile classification given a score of 2 is shown in Fig. 5. A score of 1 refers to an inaccurate classification where there are numerous errors, such as many building chunks being misclassified. A typical example of a tile classification given a score of 1 is shown in Fig. 6. A score of 0 refers to a classification with a very severe error such as a large building being misclassified. A typical example of this is shown in Fig. 7. We have repeated this qualitative analysis for the over 7 km^2 of dataset D2 as well. The results for both datasets are summarized in Table 6.

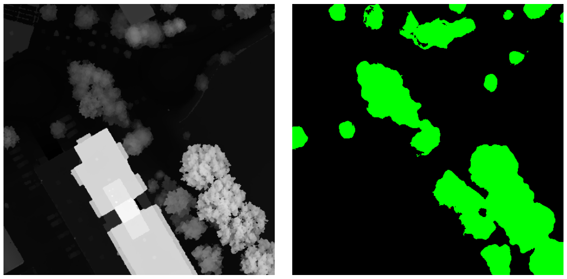

(a) (b)

Fig. 4. Typical tile classification with qualitative score 3: (a) grayscale z_{max} image generated after projecting a raw point cloud from D1 to 2.5D; (b) resulting classification of (a) where green pixels denote predicted tree points and black pixels denote predicted non-tree points.

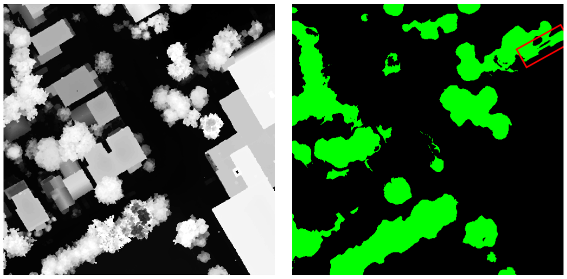

(a) (b)

Fig. 5. Typical tile classification with qualitative score 2: (a) grayscale z_{max} image; (b) resulting classification of (a). Although most of the tile is correctly classified, a large building edge boxed in red is misclassified.

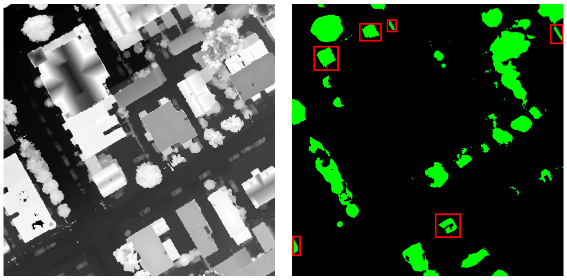
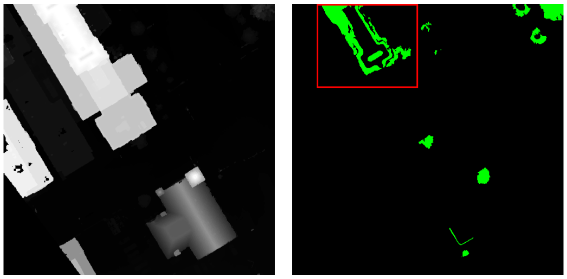

(a) (b)

Fig. 6. Typical tile classification with qualitative score 1: (a) grayscale z_{max} image; (b) resulting classification of (a). Many building fragments boxed in red are misclassified.

Nearly all of the more severe errors involve building edges. This is due to building edges being noisy and often times being segmented into too many fragments such that segment contour non-linearity is ineffective. In particular, roofs with large height variation are prone to severe misclassification as in the Fig. 7 case. However, from Table



(a) (b)

Fig. 7. Typical tile classification with qualitative score 0: (a) grayscale z_{max} image; (b) resulting classification of (a). A huge chunk of a building boxed in red is misclassified.

		Tile Qualitative Score			
		3	2	1	0
Number of tiles (and % of dataset)	D1	132 (49.6%)	87 (32.7%)	28 (10.5%)	19 (7.1%)
	D2	509 (71.7%)	162 (22.8%)	28 (3.9%)	11 (1.5%)

Table 6. Tree classifier qualitative results.

	Precision	Recall	Overall Error
D1 Cross Validation	93.6%	94.6%	7.2%
D2 Cross Validation	95.5%	94.1%	5.8%
Train on D1, test on D2	73.6%	73.6%	29.7%
Train on D2, test on D1	86.9%	75.4%	21.7%

Table 7. Classifier accuracy rates when using only features 1 and 6.

	Predicted:	Tree	Non-tree
Truth:	Tree	12,094	693
	Non-tree	825	7,561

Table 8. Confusion matrix for D1 cross validation when using only features 1 and 6.

	Predicted:	Tree	Non-tree
Truth:	Tree	12,009	756
	Non-tree	567	9,372

Table 9. Confusion matrix for D2 cross validation when using only features 1 and 6.

	Predicted:	Tree	Non-tree
Truth:	Tree	9,395	3,372
	Non-tree	3,378	6,559

Table 10. Confusion matrix for training on D1 and testing on D2 when using only features 1 and 6.

	Predicted:	Tree	Non-tree
Truth:	Tree	9,637	3,150
	Non-tree	1,447	6,939

Table 11. Confusion matrix for training on D2 and testing on D1 when using only features 1 and 6.

6, we notice that by far most tiles do not have severe errors and are well-classified.

We further analyzed the importance of the features used for training. Using the random forest's permutation-based variable importance measure [9], we found training features 1 and 6 from Section 3 to be the most indicative of tree presence. Using just these two features for training, we repeated our initial quantitative tests. Precision, recall, and overall error rates are summarized in Table 7. Confusion matrices are reported in Tables 8, 9, 10, and 11.

Even though there is clearly a drop in performance compared to using all six training features, it is clear that using only training features 1 and 6 still allows for high precision and recall rates of over 93% when restricting ourselves to working with either dataset. However, when we try to generalize across datasets, using only training features 1 and 6 results in a significant increase in overall error rate. This suggests that the other training features are important in characterizing trees well, providing a more robust classification.

Future work involves finding better features to discriminate between building edges and trees. Furthermore, our 2D approach can potentially be combined with a 3D approach to classify individual LiDAR returns rather than pixels [11].

5. REFERENCES

[1] D. Frére, J. Vandekerckhove, T. Moons, and L. Van Gool, "Automatic modeling and 3d reconstruction of urban buildings from aerial imagery," in *IEEE International Geoscience and Remote Sensing Symposium Proceedings*, Seattle, WA, USA, 1998, pp. 2593–2596.

[2] C. Früh and A. Zakhor, "Constructing 3d city models by merging aerial and ground views," *IEEE Comput. Graph. Appl.*, vol. 23, no. 6, pp. 52–61, 2003.

[3] A. P. Charaniya, R. M., and S. K. Lodha, "Supervised parametric classification of aerial lidar data," in *IEEE Conference on Computer Vision and Pattern Recognition Workshop*, 2004, pp. 25–32.

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

[5] G. Forlani, C. Nardinocchi, M. Scaioni, and P. Zingaretti, "Complete classification of raw lidar data and 3d reconstruction of buildings," *Pattern Analysis & Applications*, vol. 8, no. 4, pp. 357–374, February 2006.

[6] P. Zingaretti, E. Frontoni, G. Forlani, and C. Nardinocchi, "Automatic extraction of lidar data classification rules," in *ICIAP '07: Proceedings of the 14th International Conference on Image Analysis and Processing*, Washington, DC, USA, 2007, pp. 273–278, IEEE Computer Society.

[7] J. Secord and A. Zakhor, "Tree Detection in Urban Regions Using Aerial Lidar and Image Data," *IEEE Geoscience and Remote Sensing Letters, vol. 4, issue 2, pp. 196-200*, vol. 4, pp. 196–200, Apr. 2007.

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

[9] L. Breiman, "Random forests," in *Machine Learning*, 2001, pp. 5–32.

[10] Ohio Wright Center for Data, "The wright state 100," 2008, `http://www.daytaohio.com/Wright_State100.php`.

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