Aerial Lidar Data Classification using AdaBoost

Suresh K. Lodha Darren M. Fitzpatrick David P. Helmbold
School of Engineering
University of California, Santa Cruz, CA USA
{lodha,darrenf,dph}@soe.ucsc.edu

Abstract

We use the AdaBoost algorithm to classify 3D aerial lidar scattered height data into four categories: road, grass, buildings, and trees. To do so we use five features: height, height variation, normal variation, lidar return intensity, and image intensity. We also use only lidar-derived features to organize the data into three classes (the road and grass classes are merged). We apply and test our results using ten regions taken from lidar data collected over an area of approximately eight square miles, obtaining higher than 92% accuracy. We also apply our classifier to our entire dataset, and present visual classification results both with and without uncertainty. We implement and experiment with several variations within the AdaBoost family of algorithms. We observe that our results are robust and stable over all the various tests and algorithmic variations. We also investigate features and values that are most critical in distinguishing between the classes. This insight is important in extending the results from one geographic region to another.

keywords: lidar data, classification, terrain, AdaBoost, uncertainty, visualization.

1. Introduction

Aerial and ground-based lidar data is being used to create virtual cities [9, 7, 14], terrain models [17], and to classify different vegetation types [3]. Typically, these datasets are quite large and require some sort of automatic processing. The standard technique is to first normalize the height data (subtract a ground model), then use a threshold to classify data into into low- and high-height data. In relatively flat regions which contain few trees this may yield reasonable results, *e.g*. the USC campus [18]; however in areas which are forested or highly-sloped, manual input and correction is essential with current methods in order to obtain a useful classification.

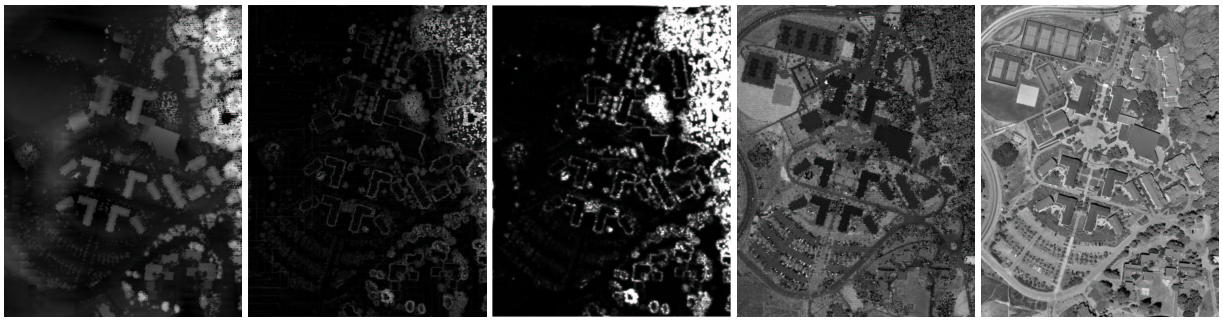
This work presents an algorithm for automatic classification of aerial lidar data into 4 groups – buildings, trees, roads, and grass – using the lidar data registered with aerial imagery. When aerial imagery is not available, our algorithm classifies aerial lidar data automatically into 3 classes: buildings, trees, and road-grass. We use 5 features: height, height variation, normal variation, lidar return intensity, and image intensity. We do not use the last feature for 3-way classification.

We use the well-known machine learning algorithm AdaBoost for this classification. AdaBoost is an elegant algorithm which automatically combines rough guesses or *weak hypotheses* into a stronger, general classifier by training on some data set with known classification. AdaBoost is at heart a binary (2-class) algorithm, however there are several extensions (such as AdaBoost.M2, AdaBoost.MH, or error-correcting codes, see Section 4) which allow for multiclass categorization. Further variations can arise based on the choice of weak hypothesis generation routine used in the specific Adaboost multi-class extension. In this work, we have experimented with several variations of weak hypothesis generation routines. These include All-Pairs Single Attribute Hypotheses (Section 4.2), expert-based hypothesis construction, and confidence-rated hypotheses with varying levels of complexity (Section 4.3). We apply our algorithm to aerial lidar data collected over a region which has highly undulating terrain and is well-forested. We present our results in Section 5.

2. Related Work

Several algorithms for classifying data into terrain and non-terrain points have been presented, including those by Kraus and Pfeifer [10] using an iterative linear prediction scheme, by Vosselman *et al*. [17] using slope-based techniques, and by Axelsson [2] using adaptive irregular triangular networks. Sithole and Vosselman [16] present a comparison of these algorithms. We have used a variation of these to compute our normalized height values.

Previous multiclass classification efforts include research by Axelsson [2], Maas [13], Filin [5], and Haala and Brenner [8]. Most of these approaches are ad-hoc and based on heuristics. In addition, in most cases, results are



(a) Normalized Height (H) (b) Height Variation (HV) (c) Normal Variation (NV) (d) Lidar Return Intensity (LRI) (e) Image Intensity (I)

Figure 1. The five feature images computed for College 8 Region of the UCSC Campus.

presented for small regions without much discussion on the overall quality of results obtained. Finally, these approaches often require substantial manual input. The most relevant previous work uses Expectation-Maximization (EM) and Support Vector Machine (SVM) algorithms to classify lidar data [11, 12]. Accuracies obtained using Adaboost algorithm in this work appear to be similar to the ones obtained in the previous work. In Section 5, we list possible advantages of using Adaboost over the SVM and EM algorithms.

3. Data Processing

3.1. Data Collection and Preparation

Our lidar dataset was acquired by Airborne1 Inc. The data was collected for approximately eight square miles of the UCSC Campus using a 1064 nm laser at a pulse rate of 25 KHz. The raw data consists of about 36 million points, with an average point spacing of 0.26 meters. We resampled this irregular lidar point cloud onto a regular grid with a spacing of 0.5m using nearest-neighbor interpolation.

In addition, we use high-resolution (0.5ft/pixel) orthorectified gray-scale aerial imagery. We downsampled the aerial imagery to the same 0.5m/pixel resolution as the lidar data and registered the two.

3.2. Features

We have identified five features to be used for data classification purposes:

- *Normalized Height (H)*: We computed the terrain elevation data automatically from the aerial lidar data using a variation of the standard DEM extraction algorithms [16]. The lidar data is normalized by subtracting terrain elevations from the lidar data.

- *Height Variation (HV)*: Height variation is measured within a 3×3 pixel $(2.25m^2)$ window and is calculated as the absolute difference between the min and max values of the normalized height within the window.

- *Normal Variation (NV)*: We first compute the normal at each grid point using finite differences. Normal variation is the average dot product of each normal with other normals within a 10×10 pixel $(25m^2)$ window. This value gives a measure of planarity within the window.

- *Lidar Return Intensity (LRI)*: is the amplitude of the response reflected back to the laser scanner.

- *Image Intensity (I)*: corresponds to the response of the terrain and non-terrain surfaces to visible light. This is obtained from the gray-scale aerial images.

All the five features have been normalized to lie between 0 and 255. Figure 1 shows images of these five features computed on the College 8 region of the UCSC Campus.

3.3. Classes and Training

We classify the dataset into four groups: buildings (rooftops), trees or high vegetation (includes coniferous and deciduous trees), grass or low vegetation (includes green and dry grass), and road (asphalt roads, concrete pathways and soil). We also combine the road and grass classes to perform a 3-class labeling using only lidar-derived features. Ten different regions of the original dataset were segmented and subregions of these were manually labeled for training and validation. The sizes of these ten regions vary from 100,000 to 150,000 points; together they comprise about 7% of our entire dataset. Roughly 25-30% of each regions was manually labeled using a graphical user interface. Although we attempted to cover the four classes adequately,

the relative proportions of the classes vary from region to region.

4. The AdaBoost Algorithm

AdaBoost [6] is a generic iterative supervised learning algorithm that combines weak hypotheses into a much more accurate master hypothesis. This master hypothesis H is a weighted linear combination of these hypotheses. The master hypothesis typically performs much better than any of the weak hypotheses alone, and thus is likely to predict better on new examples as well.

Although the original AdaBoost algorithm creates binary classifiers, there are a number of variants designed for multiple classes, as in our problem. The most straightforward extension of AdaBoost to multiple classes, AdaBoost.M1, is not very popular because it requires weak hypotheses to have accuracy better than 50% (which generally is not a simple task when there are many classes). Popular multiclass extensions which make less stringent assumptions include AdaBoost.M2 [6] and AdaBoost.MH [15], as well as error-correcting output code [4] and other ways of encoding the multiclass problem as a sequence of binary classification problems [1]. In order to obtain a concrete algorithm, one must not only select a boosting method but also pick a method for generating weak hypotheses. We have experimented with several weak hypothesis generation routines (see Sections 4.2 and 4.3). Since we have obtained very accurate results with AdaBoost.M2, we have not considered other multiclass boosting methods thus far.

4.1. AdaBoost.M2

AdaBoost.M2 [6] requires a training set $\{x_1,y_1\}\ldots\{x_N,y_N\}$ of N *examples*, or *instances* x_i and correct labelings y_i, where each instance x_i is a vector of measurements (feature values). We first describe three important features of AdaBoost.M2 and then present the algorithm. First, each weak hypothesis $h(x)$ used in AdaBoost.M2 outputs confidences in the range $[0,1]$ for each possible classification. So, for a 4-way classification, $h(x)$ maps instance x into the vector (c_1,c_2,c_3,c_4), where each $c_i \in [0,1]$. Note that the c_i values are not probabilities, and do not need to sum to 1. We use the notation $h(x)[y]$ to represent the confidence assigned to class y by hypothesis h on instance x.

Second, the probability distributions D_t used at each iteration t by AdaBoost.M2 are defined over the cross product of training examples x_i and the set of possible class labels Y. The value $D_t(x_i,y)$ represents the *badness* of predicting class $y \in Y$ on training instance x_i. Therefore, since y_i is the correct class label for x_i, $D_t(x_i,y_i)$ is always set to zero.

Third, rather than using the probability of misclassification (as in binary AdaBoost and AdaBoost.M1), a new metric, *pseudo error*, is used to measure the badness of weak hypotheses. Recall that using probability of misclassification as a metric requires weak hypotheses that have better than 50% accuracy, a non-trivial task when there are many classes. The pseudo error metric rewards high confidence on the correct class while penalizing high confidence on incorrect classes. Formally, the pseudo error of weak hypothesis h_t with respect to distribution D_t is:

$$e_t = \frac{1}{2}\sum_{i=1}^{N}\sum_{y\in Y} D_t(i,y)(1+h_t(x_i)[y]-h_t(x_i)[y_i]). \quad (1)$$

Note that if $h_t(x_i)$ assigns the same confidence to all classes then $h_t(x_i)[y]-h_t(x_i)[y_i]=0$. If this happens for all the instances in the training set then $e_t = 1/2$ since $\sum_{i,y} D_t(i,y) = 1$. Thus, a random hypothesis has expected pseudo error 1/2 but the probability of error is much larger, $1-\frac{1}{|Y|}$.

Algorithm 1 The AdaBoost.M2 algorithm

Require: Training set $(x_1,y_1)\ldots(x_N,y_N)$ where each $x_i \in X$ is a vector of measurements; each $y_i \in Y$ is a label with values from 1 to the number of classes, $|Y|$.

INITIALIZE $H_0(x)[y]=0$ for all $x \in X$ and $y \in Y$, and

$$D_1(x_i,y) \Leftarrow \begin{cases} \frac{1}{N*(|Y|-1)} & i=1\ldots N, y\neq y_i \\ 0 & i=1\ldots N, y=y_i \end{cases}$$

for $t=1\ldots T$ **do**

Use weak hypothesis generator on training set weighted by distribution D_t to obtain weak hypothesis $h_t: X \to [0,1]^{|Y|}$ with pseudo error (as given in Equation 1) $e_t < \frac{1}{2}$

Set the weight of h_t to $a_t = \frac{1}{2}\ln(\frac{1-e_t}{e_t})$

Set $H_t(x) = H_{t-1}(x) + a_t h_t(x)$

Set $D_{t+1}(i,y) = D_t(i,y)e^{(-a_t(1-h_t(i)[y_i]+h_t(i)[y]))}$

Normalize D_{t+1} so $\sum_{i=1}^{N}\sum_{y\in Y} D_{t+1}(i,y)=1$

end for

Output final hypothesis $H(x) = \arg\max_{y\in Y}\{H_T(x)[y]\}$

An outline of AdaBoost.M2 is given in Algorithm 1. Since master hypothesis H_t is a linear combination of weak hypotheses, $H_t(x)$ is also a vector of confidences; though each component may lie within the range $[0,\infty]$. Single-class prediction from H_t is achieved by taking the class with the greatest confidence. Let $H_T(x) = (c_1(x)\ldots c_{|Y|}(x))$. Furthermore, without loss of generality, assume that c_1 is the maximum and c_2 is the next largest, then the confidence c (or uncertainty, $1-c$) of the prediction is taken to be the ratio $c = \frac{c_1-c_2}{c_1} \in [0,1]$. This value is used in our visual results (see Section 5).

AdaBoost.M2 has strong performance guarantees like those of binary AdaBoost. In particular, if weak hypotheses have pseudo errors $e_t < 1/2$, then H_T will be perfect on the training sample within $O(\log N)$ iterations. AdaBoost.M2 can be run for a fixed number of iterations, until convergence, or until the pseudo errors of the weak hypotheses become too large, that is, $e_t \geq 1/2$.

4.2. Generating Weak Hypotheses using All-Pairs

There are several strategies for constraining the infinite universe of potential weak hypotheses to a more tractable set [6]. For most of our experiments, we have considered only hypotheses which consider a pair (y_1, y_2) of classes and test a single feature against a threshold, predicting either class y_1 with full confidence $(h(x)[y_1] = 1, h(x)[y \neq y_1] = 0)$ or class y_2 with full confidence $(h(x)[y_2] = 1,\ h(x)[y \neq y_2] = 0)$. We refer to this approach as All-Pairs Single Attribute Hypotheses. Of course, by constraining ourselves so, there is now a risk of overlooking the hypothesis with the absolute lowest possible pseudo error. Therefore we also consider other variations, including some which assign nonzero confidence to all of the classes (Section 4.3).

We now describe a tractable search to identify candidate All-Pairs Single Attribute hypotheses: one for each unordered pair of classes (6 or 3) and feature (5 or 4) for a total of 30 or 12 potential hypotheses in each iteration. Observe that, assuming 256 integer values for feature values, in the most naive implementation, the total number of all hypotheses to be considered at each iteration is 15,360 (or 4608). The tractable search is achieved by identifying exactly one threshold value v for a given class pair and feature. To make things concrete, and without loss of generality, assume that we are finding the threshold value v for the candidate hypothesis that tests the height feature and predicts building (B) below the threshold and tree (T) above it. Our heuristic is to find the threshold v that minimizes the badness of the incorrect B and T predictions. We first compute the arrays A_B and A_T, where $A_B[j]$ (respectively $A_T[j]$) is the total badness of predicting building (respectively tree) summed over all the examples having height j, that is, $A_T[j] = \sum_{i|height(x_i)=j} D_t(x_i, T)$. Next we compute the prefix sums $S_T[j] = \sum_{k=0}^{j} A_T[k]$ and $S_B[j] = \sum_{k=0}^{j} A_B[k]$. Observe that the badness of predicting building when the height is $\leq v$ is $S_B[v]$. Similarly, $S_T[255] - S_T[j]$ is the total badness of predicting tree summed over those examples having height greater than j. Therefore, $S_{BT}[v] = S_B[v] + S_T[255] - S_T[v]$ is the total badness of predicting building when the height is at most v and tree when the height is greater than v. By symmetry, there is another desirable hypothesis for which the threshold v minimizes the tree-building badness $S_{TB}[v] = S_T[v] + S_B[255] - S_B[v]$. However, they are related as follows:

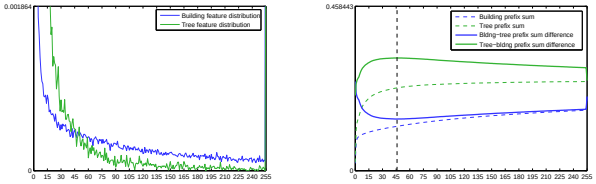


Figure 2. Automatic value selection for hypothesis generation: (left) A_T (green) and A_B (blue) : badness of predicting tree and grass respectively for different values of the feature normal variation; (right) prefix sums S_T and S_B (dashed lines) obtained by integrating the graphs on the left, superimposed with S_{TB} (solid green line) and S_{BT} (solid blue line), note that S_{TB} and S_{BT} are symmetric with respect to $(S_T[255] + S_B[255])/2$. The vertical dashed line indicates the chosen optimum value to be used in the weak hypothesis.

$$S_{BT}[v] = S_T[255] + S_B[255] - S_{TB}[v]. \qquad (2)$$

This shows that the building-tree badness is minimized exactly where the tree-building objective is maximized. Therefore we can select from both the building-tree and tree-building hypotheses at once by computing just the building-tree badness for each threshold v and finding the value that makes this badness closest to zero or closest to $S_T[255] + S_B[255]$. This is illustrated in Figure 2 by using an example from our experiments.

4.3. Alternative algorithms

Recall that since we pick a hypothesis from a small set of 30 candidates during each iteration, we might not find the weak hypothesis with smallest pseudo error from the larger set of possible weak hypotheses. To investigate whether one could do better by expanding our search to a larger space, we considered alternative weak hypothesis generators.

In our first variation, we selected the weak hypotheses from a set of 14 hand-crafted prototypes. Each hypothesis type was paired with 10 randomly selected, uniformly distributed threshold values. Although this implementation tested 140 hypothesis per feature, results were comparable (within 3%) to the All-Pairs approach reported later in Section 5.

In another version, we considered more complex hypotheses: if feature $< v$, then each class y_i has confidence a_i, else each class y_i has confidence b_i, where both a_i and b_i were computed using information from D_t. Despite this generality, the accuracy results did not improve much and this complexity did not seem justified.

We found the classification accuracy to be similar (within 2-3%) over all the variations we implemented. We also found that the simple hypotheses generated by the All-

Pairs (Section 4.2) approach have the advantage of producing an intuitively understandable master hypotheses. We present the accuracy results using All Pairs hypotheses in the next section.

5. Results and Analysis

We first describe two types of training, three different types of tests, and how we measure classification accuracy. We then present and analyze our classification results.

We use two methods to sample training data: (i) *class-weighted training*: the examples are distributed uniformly across all the classes by downsampling the abundant categories (typically trees and grass) [1], and (ii) *sample-weighted training*: each class is represented in the training sample in the same proportion as it occurs in the labeled data. In either case, the 25-30% of each region which had been manually labeled was downsampled to 10% of the original size and the training samples were taken to be these points.

We used three different types of tests: (i) leave-one-out, (ii) train-half-test-half, and (iii) train-all-test-all. In the leave-one-out test, 9 regions are used for training and the remaining region is tested. For train-half-test-half, half the regions were used for training and the remaining 5 were tested. Finally, in train-all-test-all, all ten were used both to train and test. Results from train-all-test-all were marginally better than both leave-one-out and train-half-test-half tests, which are similar to 10-fold and 2-fold testing typically used in machine learning. Results from both of these tests were comparable – here we report only the results from leave-one-out-test unless otherwise stated.

To compute accuracy, we use the manualy labeled data for which the truth is known. For sample-weighted training, the accuracy is tested on all the labeled data. For class-weighted training, the accuracy is tested on the downsampled labeled data where the sample size in each class is the same.

All of the following results used the weak hypothesis generation routine described in Section 4.2. Table 1 presents the classification accuracy for each of the 10 regions' leave-one-out tests. The average accuracy obtained is better than 92% for 4-way classification and better than 95% for 3-way classification. Table 2 presents the class-weighted accuracy results organized by class. This table shows that, for 4-way classification, Type I error (rejecting points belonging to a class) is highest for roads. This occurs because road is the least-populated category and training is somewhat deficient for this class. Type II error (accepting points belonging to a different class) is highest for grass, which typically collects points from road, followed by road. For 3-way classification, the confusion between the remaining classes is comparable to 4-way, with the new road-grass class being confused much less than before, as separate classes. This shows that most of the confusion is between buildings-trees and grass-road for 4-way classification and between buildings-trees for 3-way classification. This highlights the need for improving the algorithm to better distinguish between these classes.

[1] Since the number of labeled data points for road is far fewer than other categories, in some cases, road was still undersampled compared to other classes.

Region	5 features, 4 classes		3 features, 3 classes	
	class-weighted	sample-weighted	class-weighted	sample-weighted
Region 1	86.45 %	93.12 %	92.61 %	93.98 %
Region 2	92.62 %	93.25 %	98.26 %	96.63 %
Region 3	96.56 %	96.28 %	95.91 %	94.38 %
Region 4	93.28 %	95.14 %	96.29 %	98.07 %
Region 5	92.32 %	93.10 %	96.26 %	93.82 %
Region 6	93.60 %	95.55 %	96.23 %	98.85 %
Region 7	90.20 %	85.82 %	94.58 %	97.11 %
Region 8	94.20 %	91.34 %	97.86 %	96.40 %
Region 9	90.21 %	88.29 %	91.21 %	94.52 %
Region 10	92.51 %	90.31 %	98.62 %	98.19 %
Overall	92.20 %	92.22 %	95.78 %	96.20 %

Table 1. Leave-one-out test accuracy: average and for each of the 10 regions.

	bldng	tree	grass	road	Type I
bldng	94.06 %	5.13 %	0.19 %	0.62 %	5.94 %
tree	2.66 %	96.09 %	0.56 %	0.72 %	3.93 %
grass	3.94 %	1.12 %	87.54 %	7.41 %	12.46 %
road	0.24 %	1.03 %	13.14 %	85.61 %	14.41 %
Type II	6.84 %	7.27 %	13.88 %	8.75 %	

	bldng	tree	road grass		Type I
bldng	93.92 %	5.68 %	0.36 %		6.04 %
tree	1.92 %	96.90 %	1.17 %		3.09 %
road grass	4.06 %	1.09 %	94.85 %		5.16 %
Type II	5.98 %	6.77 %	1.53 %		

Table 2. Classification accuracy and Type I /Type II error results organized by classes for 4 classes/5 features and 3 classes/3 features using leave-one-out-test. The reported accuracy numbers are for class-weighted training averaged over all 10 regions.

We now present the visual classification results with and without uncertainty. Figure 6 shows the classification results without uncertainty for the College 8 region. The buildings, trees, road and grass classes are colored blue, green, brown and yellow respectively for 4-way classification; for 3-way output the road-grass class is colored yellow (this coloring scheme is continued for our other visual results). Recall that only 10% of the manually labeled data (about 2-3% of all the points within the 10 regions) was used for training. Buildings and trees are evident in these images. Sometimes there is no 'correct' labeling for small areas, *e.g.* the tennis courts in the upper-left which are labeled as road. Further research is needed to handle these cases elegantly.

We have applied the master hypothesis to our entire dataset and computed the associated uncertainty (as described in Section 4.1). Figure 7 shows these results. Con-

fidence has been mapped to brightness (darker values are to be trusted less). A zoomed-in view of the area (within the red box of Figure 7) is shown in Figure 8. From these we see that we have predicted interiors of buildings and trees with high confidence, while being less commital near boundaries. We also noticed some spatio-temporal uncertainty such as at the top-left of the region in Figure 7; certain structures such as trailers which existed during lidar collection were not present during aerial image collection.

Accuracies reported in some of the previous work using the SVM [11] and the EM [12] algorithms are in the similar range of 90% to 94% accuracy. However, without the detailed knowledge of the exact data characteristics, and the training and sample size, it is not possible to carry out a strict comparison between these algorithms. The biggest significant advantage of Adaboost over the other machine learning algorithms such as the SVM and the EM algorithm appears to be its interpretability, which we now discuss in detail. Neither SVM nor the EM algorithms can provide such detailed insights into the role that different features play in discriminating between the different classes.

Through graphs, tables, and images we now analyze our training methods and the roles of our features and hypotheses in discriminating between classes. Figure 3 shows the evolution of accuracy during our train-all-test-all testing for each classification setup. Typically, the accuracy rate stabilizes reasonably quickly. We observe that although overall accuracy is about the same for both class- and sample-weighted training, the accuracy is more uniform across classes for class-weighted training in both 4-way and 3-way classification. This suggests that if higher accuracy is desired for a particular class then additional training examples for said class relative to others will be useful.

Recall that the master hypothesis is a weighted vote of weak hypotheses, and that each weak hypothesis tests only a single feature. Given a class pair and a feature, Table 3 presents the total weight percentage of all weak hypotheses which used said feature in the final master hypothesis to discern between the given class pair. This table reveals that height (H), lidar return intensity (LRI), normal variation (NV), and image intensity (I) played important roles. Height (H) and normal variation (NV) were also important in 3-way classification. It also became clear that height variation (HV) was the least useful feature. It played a limited role and perhaps should be replaced with an improved height texture feature rather than a simple difference between maximum and minimum height within a window.

We can gain further insight by examining the threshold values used by important weak hypotheses. Figure 4 presents the weight by threshold value of the five highest-weighted hypotheses in the 4-way classification. These graphs reveal that some of the most important hypotheses in discriminating between the four classes are: height feature

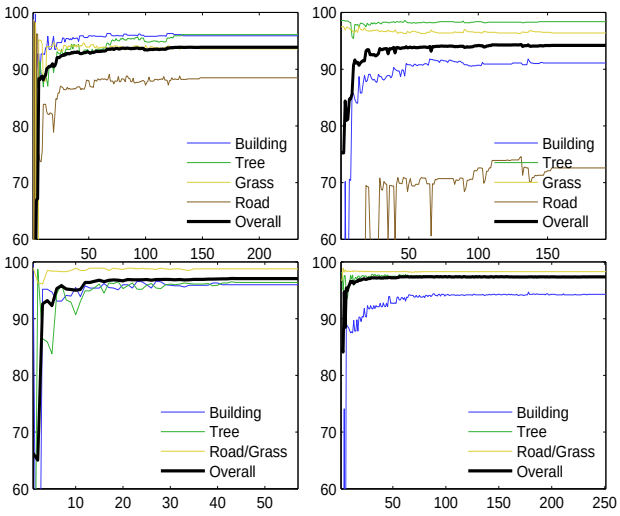


Figure 3. Classification accuracy (train-all-test-all) as training proceeds: (top row, left to right): 4-way classification with class-weighted training (233 iterations) and sample-weighted training (191 iterations) (bottom row, left to right): 3-way classification with class-weighted training (56 iterations) and sample-weighted training (250 iterations). Success rate is more uniformly distributed with class-weighted training.

	I	LRI	H	HV	NV
bldng-tree	0.04	0.03	0.11	0.03	0.08
bldng-grass	0.03	0.02	0.11	0.00	0.00
bldng-road	0.01	0.01	0.04	0.00	0.00
tree-grass	0.00	0.00	0.00	0.00	0.03
tree-road	0.02	0.04	0.08	0.00	0.04
grass-road	0.06	0.09	0.05	0.06	0.02
overall	0.16	0.19	0.39	0.09	0.17

	LRI	H	HV	NV
bldng-tree	0.13	0.16	0.04	0.13
bldng-rd/gs	0.00	0.16	0.00	0.00
tree-rd/gs	0.01	0.20	0.00	0.16
overall	0.14	0.52	0.04	0.29

Table 3. Percentages of the total weight by feature and class pair for 4-way and 3-way classification after 233 and 56 iterations respectively using train-all-test-all with class-weighted training.

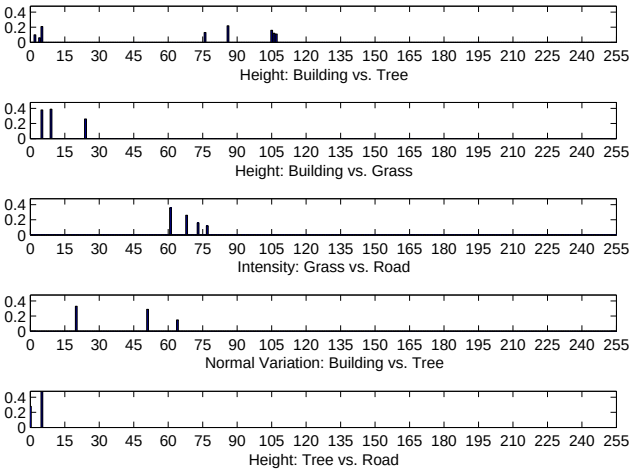


Figure 4. Weight distribution by threshold value for five important feature-class pair combinations for 4-way classification obtained from the master hypothesis.

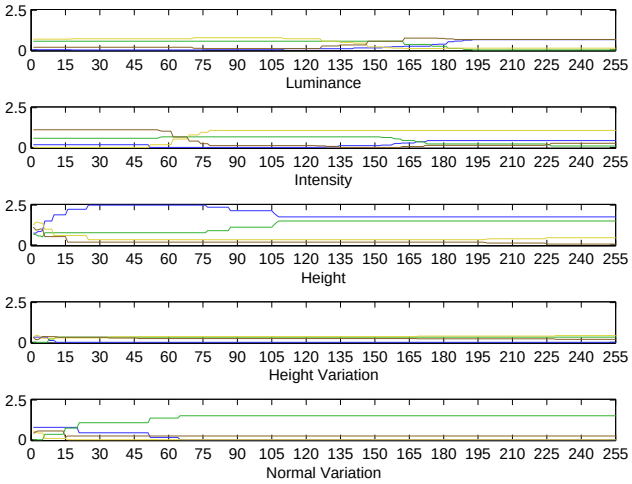


Figure 5. Class prediction weight by feature value obtained from the master hypothesis for 4-way classification: buildings (blue), trees (green), grass (yellow), and road (brown).

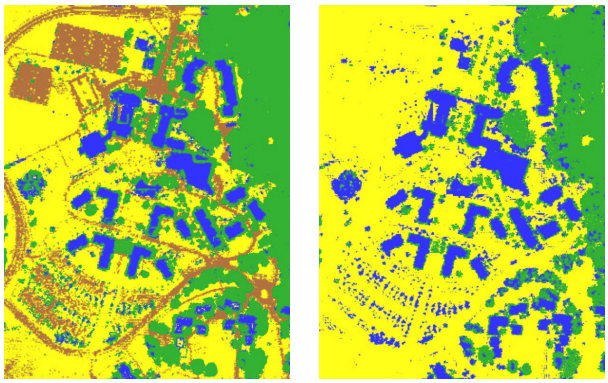


Figure 6. Leave-one-out classification results for Region 2 using sample-weighted training: (left) 4-way classification using 5 features, and (right) 3-way classification using 4 features.

with values 5, 5 or 9, and 86 to distinguish between trees and road, buildings and grass, and buildings and trees, respectively; intensity values of 61 and 68 to distinguish between grass and road; and normal variation with values 20 or 51 to distinguish between buildings and trees. Please notice the changes in the graphs in Figure 5 corresponding to these threshold values.

The plots in Figure 5 can be used to help understand how the master hypothesis is making its predictions. For each feature, the class prediction is shown as a function of feature value. The class prediction for a feature value is then determined by summing for each possible class the votes of each weak classifier which tests the given feature (the master hypothesis would do this for each feature, then output the class with highest total vote as the predicted label). For example, the large vote for buildings on instances with height 30-75 will tend to cause the master hypothesis to predict the buildings class (blue). However, the class trees can out-vote buildings if the instance has a low luminance value, intensity below 160 or so, and high normal variation.

The flat, low-lying graphs of height variation again show the relative insignificance of this feature. In contrast, important features such as height and normal variation have high-lying and expressive graphs. These graphs also bring out which features are important in identifying a particular class. For example, height is important in classifying buildings and height and normal variation is important in classifying trees for 4-way classification.

6. Conclusions and Future Directions

We have developed an algorithm which automatically classifies aerial lidar data into buildings, trees, roads, and grass using AdaBoost. The results of this classification are stable and have higher than 92% accuracy. Our experimentation with variations on weak hypothesis generation and successful application of the classifier to a relatively large untrained dataset underscore its robustness. Our analysis through both tables and graphs and confidence-rated visual output help us understand problem areas and potential weaknesses of the classifier. This understanding is important for future extensions of the algorithm to classify aerial lidar data from different geographic regions and for finer classification into sub-classes such as vehicles, telecommunication wires, or different types of vegetation.

Acknowledgements: This research is partially supported by the Multi-disciplinary Research Initiative (MURI) grant by U.S. Army Research Office under Agreement Number DAAD19-00-1-0352, the NSF grant ACI-0222900, and the NSF-REU grant supplement CCF-0222900.

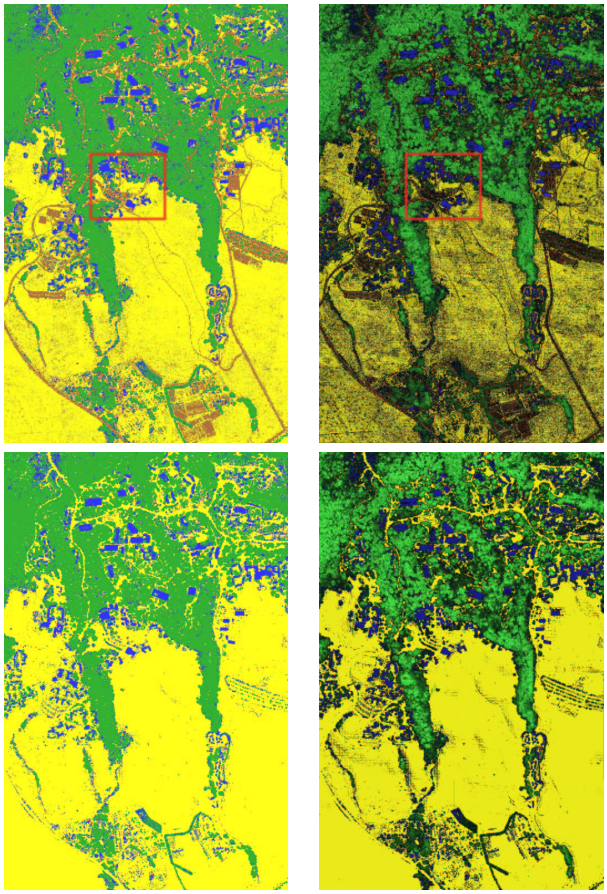


Figure 7. Top, left: 4-way classification of the entire dataset; top, right: 4-way confidence-weighted classification; bottom, left: 3-way classification of the entire dataset; bottom, right: 3-way confidence-weighted classification.

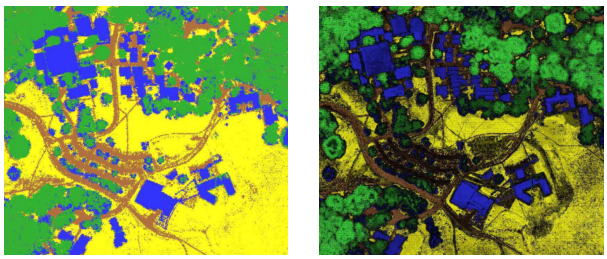


Figure 8. A zoomed-in view of the areas inside the red boxes shown in Figure 7.

References

[1] E. L. Allwein, R. E. Schapire, and Y. Singer. Reducing multiclass to binary: A unifying approach for margin classifiers. *Journal of Machine Learning Research*, 1:113–141, 2000. 3

[2] P. Axelsson. Processing of laser scanner data -algorithms and applications. *ISPRS Journal of Photogrammetry and Remote Sensing*, 54(2-3):138–147, 1999. 1, 1

[3] J. B. Blair, D. L. Rabine, and M. A. Hofton. The laser vegetation imaging sensor: a medium altitude, digitsation-only, airborne laser altimeter for mapping vegetaion and topography. *ISPRS Journal of Photogrammetry and Remote Sensing*, 54:115–122, 1999. 1

[4] T. G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995. 3

[5] S. Filin. Surface clustering from airborne laser scanning. In *ISPRS Commission III, Symposium 2002 September 9 - 13, 2002, Graz, Austria*, 2002. 1

[6] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Comput. Syst. Sci.*, 55(1):119–139, 1997. 3, 3, 3, 4

[7] C. Frueh and A. Zakhor. Constructing 3D city models by merging ground-based and airborne views. In *IEEE Conference on Computer Vision and Pattern Recognition*, June 2003. 1

[8] N. Haala and C. Brenner. Extraction of buildings and trees in urban environments. *ISPRS Journal of Photogrammetry and Remote Sensing*, 54(2-3):130–137, 1999. 1

[9] J. Hu, S. You, and U. Neumann. Approaches to large scale urban modeling. *IEEE Computer Graphics and Applications*, 23(6):62–69, November/December 2003. 1

[10] K. Kraus and N. Pfeifer. Determination of terrain models in wooded areas with airborne laser scanner data. *ISPRS Journal of Photogrammetry and Remote Sensing*, 53:193–203, 1998. 1

[11] S. K. Lodha, D. Fitzpatrick, and D. P. Helmbold. Aerial lidar data classification using support vector machines. In *Proceedings of the Conference on 3DPVT*, June 2006. 2, 6

[12] S. K. Lodha, D. Fitzpatrick, and D. P. Helmbold. Aerial lidar data classification using expectation-maximization. In *Proceedings of the SPIE Conference on Vision Geometry XIV*, volume 6499, pages L1–L11, January 2007. 2, 6

[13] H.-G. Maas. The potential of height texture measures for the segmentation of airborne laserscanner data. *Fourth International Airborne Remote Sensing Conference and Exhibition*, 21st Canadian Symposium on Remote Sensing:Ottawa, Ontario, Canada, 1999. 1

[14] W. Ribarsky, T. Wasilewski, and N. Faust. From urban terrain models to visible citites. *IEEE Computer Graphics and Applications*, 22(4):10–15, July 2002. 1

[15] R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, 1999. 3

[16] G. Sithole and G. Vosselman. Comparison of filtering algorithms. In *ISPRS Commission III, Symposium 2002 September 9 - 13, 2002, Graz, Austria*, 2002. 1, 2

[17] G. Vosselman. Slope based filtering of laser altimetry data. *International Archives of Photogrammetry and Remote Sensing*, XXXIII, Amsterdam, 2000, 2000. 1, 1

[18] S. You, J. Hu, U. Neumann, and P. Fox. Urban site modeling from lidar. In *Second International Workshop on Computer Graphics and Geometric Modeling*, 2003. 1