A COMPRESSED SENSING METHOD WITH ANALYTICAL RESULTS FOR LIDAR FEATURE CLASSIFICATION

Josef D. Allen
Oak Ridge National Laboratory
1 Bethel Valley Road
Oak Ridge, TN 37831
Email: allenjd@ornl.gov

Jiangbo Yuan and Xiwen Liu
Dept. of Computer Science
Florida State University
Tallahassee, FL 32306-4530
Email: {jyuan, liux}@cs.fsu.edu

Mark Rahmes
Harris Corporation, GCSD
Melbourne, Florida 32904
Email: mrahmes@harris.com

ABSTRACT

We present an innovative way to autonomously classify LiDAR points into bare earth, building, vegetation, and other categories. One desirable product of LiDAR data is the automatic classification of the points in the scene. Our algorithm automatically classifies scene points using Compressed Sensing Methods via Orthogonal Matching Pursuit algorithms utilizing a generalized K-Means clustering algorithm to extract buildings and foliage from a Digital Surface Models (DSM). This technology reduces manual editing while being cost effective for large scale automated global scene modeling. Quantitative analyses are provided using Receiver Operating Characteristics (ROC) curves to show Probability of Detection and False Alarm of buildings vs. vegetation classification. Histograms are shown with sample size metrics. Our inpainting algorithms then fill the voids where buildings and vegetation were removed, utilizing Computational Fluid Dynamics (CFD) techniques and Partial Differential Equations (PDE) to create an accurate Digital Terrain Model (DTM) [6]. Inpainting preserves building height contour consistency and edge sharpness of identified inpainted regions. Qualitative results illustrate other benefits such as Terrain Inpainting’s unique ability to minimize or eliminate undesirable terrain data artifacts.

Keywords: Compressed Sensing, Sparsity, Data Dictionary, LiDAR, ROC, K-Means, Clustering, K-SVD, Orthogonal Matching Pursuit

I. INTRODUCTION

We propose a new method for constructing local features from “height-based images”, which can be used to recognize different objects with distinctive height properties. The features are designed to classify different regions from images to belong to different objects as predefined in a dictionary. We introduce the compressive sensing to deal with the classification issue. Meanwhile, a dictionary is constructed by randomly choosing partial features from the training samples. Our approach to recognition can robustly identify objects based on “height-based images”.

II. METHODOLOGY

A central idea is that we use the notion of compressed sensing (CS). CS is the process of acquiring a signal utilizing prior knowledge, i.e. data dictionaries, that it is sparse or compressible. A straightforward way to solve for sparsity is Orthogonal Match Pursuit (OMP). Here OMP is used to create the data dictionaries used for matching. The genesis of CS can be found in trying to find the $l_0$ norm [1]. The $l_0$ norm is a measure of sparsity. Image processing, multi-dimensional signal processing, is well suited for the utilization of the sparsity. Consider a full rank matrix $A \in \mathbb{R}^{n \times m}$ with $n < m$, and define the underdetermined linear system of equations $Ax = b$. This system has infinitely many solutions. If we want to narrow the solution we would need additional criteria. Hence, we define the general optimization problem.

\[
\begin{align*}
(P_j): \min_x J(x) \text{ subject to } b = Ax.
\end{align*}
\]

(a familiar strictly convex $J(x)$ is the squared $l_2$ norm i.e. $\|x\|_2^2$). While the squared $l_2$ norm is the measure of energy the $l_0$ is a measure of sparsity. We define
Intuitively a vector $\mathbf{x}$ is sparse if there are few non-zeroes among possible entries in $\mathbf{x}$. In this paper we present a fundamental approach for LiDAR processing using compressive sensing.

Figure 1 shows a processing flow diagram for training and extracting features. Orthogonal Matching Pursuit (OMP) is used to create the Data Dictionary and the coefficients. The Data Dictionary uses iterative K-SVD for training and compressed decomposition. K-SVD maps each signal to the closest atom. Each signal is a linear combination of atoms or code words [3]. The data dictionary is updated by K-SVD. The first Data Dictionary element is the DC value and all others are zero mean [4].

The objective of bare earth processing is to autonomously reconstruct the bare earth in places where buildings, trees, and other non-bare earth objects have been removed or where data is missing while maintaining continuous height contours. This allows our technique to generate high resolution bare earth DEMs from high frequency terrain [2]. Once the bare earth LiDAR points are identified, the building and tree points need to be separated and classified.

One of the more common applications of Terrain Inpainting is in the creation of a DTM of an input scene as either a final product or as an intermediate input for further processing (e.g., 3D site model creation or orthomosaic production). During this process, our algorithm automatically classifies and removes culture and vegetation from the input Digital Surface Model (DSM). Our algorithm is designed to process DSMs created from multiple sources. Primary examples are surface models created from photogrammetry, LiDAR, or IFSAR. Compressed sensing also serves as an algorithm for the restoration of 2D signals in inpainting problems for medical imaging [5].

Figure 2a shows a DSM generated from LAS point data. After the completion of the automated bare earth process, we output a model containing only those points that fall on the terrain surface, see Figure 2b. All other points in the input belonging to cultural or vegetation features have been removed. These void areas introduced during bare earth
processing must be filled to create a complete DTM. Inpainting attempts to accurately propagate information from extracted building and tree boundaries to void interiors as shown in Figure 2c [6]. Figure 2d shows the automatic classification of culture and vegetation as a separate output.

The software automates the creation of geospatial products including bare earth Digital Terrain Models (DTM) and image-textured 3D site models [2]. Terrain inpainting provides void fill processing for geospatial data production in areas where information is incomplete. Geospatial products created through digital processing can introduce visible artifacts from void fill and other associated processing. Terrain inpainting produces minimal artifacts, and at the same time provides a representation designed to be as accurate as possible with quantifiable accuracy assessment. Quantitative assessment and built-in error estimation are vital for the robustness and applicability of terrain inpainting.

III. LIDAR CLASSIFICATION

We perform an initial automated classification of the LAS file to a gridded space producing a classified DTM. Our algorithm ingests the original LAS file and DTM, and then outputs a new LAS file with the point classification field set by comparing points to DTM appropriate height values. For each point in the input LAS file, we classify its feature label. Geospatial coordinates are left unaltered. Automated scoring through comparison of the classifications of the input and output is performed. The first step is recording the original classification of the input file. Then each point in the DTM with valid height (i.e., not NaN) has its feature type label classified. The closest index point for each valid post is located
in the DTM as the ground point. If the point is within the specified tolerance of the ground surface DTM, we classify it as ground. This threshold is called the ground surface tolerance.

If we are labeling buildings, we make sure the point is non-null in the building DEM and then ensure that the height of the current point (from the LAS file) is above the ground by a specified tolerance. If so, we classify it as building. If we are labeling vegetation, the same process described for buildings is used for vegetation. We identify points that are not in any of the previously checked categories and label them with the other label. An output comparison between the original LAS and reclassified LAS points is performed.

Figure 3 below shows the LiDAR classification processing flow diagram. The process begins with ingest of LAS formatted LiDAR points, but the process is able to use other formats. The unordered points are gridded and small gaps are filled with a simple nearest neighbor interpolation for a fast void fill. In the case of LiDAR inputs, this process is performed for both the first and last return DSMs. The larger voids are then filled with our PDE inpainting technology for greater accuracy [6]. Building and tree points are further separated through the use of the height difference between first and last DSMs along with a set of additional post-processing classification steps. A slope calculation is performed to help distinguish buildings and trees for inputs where the input DSM only has one reflective surface available. Using the ground point DTM, the building post DSM and the vegetation post DSM our algorithm assigns a classification label to the original points and outputs a labeled LAS formatted file [2].

![Figure 3. LiDAR Classification Processing Flow Diagram](image)

### IV. ANALYSIS & RESULTS

Classifying and separating building and tree LiDAR points is accomplished using Compressed Sensing. The major steps include:

1. Extract local windows: we scan full images and extract as many interesting local windows as we can; the scanning widow size \( m \) should be chosen properly;
2. Calculate local descriptors: we treat each extracted local window as a unit and calculate height-gradients for all sample points in each such window; then calculate the gradient density descriptor relying on each window; thus, each interesting local window is represented as a height-gradient density descriptor;
3. Set up a dictionary: we randomly chose equal numbers of the descriptors from each class and to form a dictionary;
4. Finally, we use the remaining descriptors to test and adjust the dictionary size and other parameters.

The concrete operations and detail equations are given as following. For each location \((x, y)\) in an image \(I\), the gradient magnitude \(g(x, y)\) is given by:

\[
g(x, y) = \sqrt{(I(x+1,y) - I(x-1,y))^2 + (I(x,y+1) - I(x,y-1))^2}
\]  

Therefore, we calculate gradient vectors in the length of \((m \times m)\) for each interesting local window. After that, we transform each gradient vector to a final descriptor by estimating its probability density.
Our classification measure is based on a way of Compressive Sensing (or Sparse Representation). The idea is mainly borrowed from [7], which shows that sparse representation can be utilized to handle with face recognition. In this method, we construct a dictionary, $\mathcal{A}$, by directly using training samples. Suppose that the dictionary size is $n$ and the dimension of the descriptors is $d$, then $\mathcal{A}$ is a $n \times d$ matrix as:

$$
\mathcal{A} = [a_1, a_2, ..., a_d] \in R^{n \times d}
$$

(4)

Given a new descriptor $b$ (an unclassified local window), we represent it as $b = \mathcal{A}\hat{x}_1$, where $\hat{x}_1$ is a coefficient vector with a property of sparsest representation and subject to $l^1$-minimization,

$$
\hat{x}_1 = \text{arg min}_x \|x\|_1
$$

(5)

The energy of coefficients from the same class is more likely to be larger than those from other classes. Finally, we identify the descriptor $b$ as the class $i$ by

$$
\text{classify}(b) = \text{arg max}_i \|x_i\|_1
$$

(6)

**Experiment 1:** Here is an illustration for our method, which shows that the features (descriptors) we proposed are highly distinctive for “height-based images”. Figure 4 shows the 3D view of the truth data. Figure 5 shows two original height-based images which present height information of buildings and trees separately. The red regions in Figure 6&7 show the detected interesting regions. The Table1 shows how many local windows were detected and the correctly recognized rate we got for each class.

![Figure 4. Truth Data](image-url)
Table 1. Considering the dictionary is randomly chosen we separately run 20 times of our algorithm by local window sizes of 7*7 and 5*5 on the images “Trees” and “Buildings”, and presented the best, worst, average correctly recognized rates and also the variances as below. For this typical two class problem, the base elements of the dictionary are chosen by half and
half from the two classes.

<table>
<thead>
<tr>
<th>Image</th>
<th>Window Size</th>
<th>Detected Local Window</th>
<th>Dictionary Size</th>
<th>Best</th>
<th>Worst</th>
<th>Average</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trees</td>
<td>7*7</td>
<td>23518 (42 for training)</td>
<td>84</td>
<td>0.9993</td>
<td>0.9929</td>
<td>0.9984</td>
<td>0.0015</td>
</tr>
<tr>
<td>Buildings</td>
<td>7*7</td>
<td>84 (42 for training)</td>
<td>84</td>
<td>1.0000</td>
<td>0.9762</td>
<td>0.9988</td>
<td>0.0053</td>
</tr>
<tr>
<td>Trees</td>
<td>5*5</td>
<td>42374 (100 for training)</td>
<td>200</td>
<td>0.9969</td>
<td>0.9835</td>
<td>0.9993</td>
<td>0.0034</td>
</tr>
<tr>
<td>Buildings</td>
<td>5*5</td>
<td>284 (100 for training)</td>
<td>200</td>
<td>1.0000</td>
<td>0.9946</td>
<td>0.9997</td>
<td>0.0012</td>
</tr>
</tbody>
</table>

**ROC Curves:** To show the effectiveness of the randomly picking procedure, Fig. 8 shows the receiving operating characteristic (ROC) curves of two classes of the dataset 1. Through the ROC score we can further demonstrate the excellent properties of this technology. For each ROC curve, ROC score is defined simply as the area under the ROC curve. For example, if the decision is purely random, than the ROC score should be very close to 0.5; on the other hand, for a perfect classifier, its ROC curve consists of (0,0) to (0, 1) and then to (1, 1), and its ROC score is 1. Clearly, the higher the ROC score, the better the classifier performance.

![ROC Curves](image)

**Figure 8.** ROC curves of experiment 1

**Histograms:** Following sample data histograms in Figure 9 show distributions for “Buildings” and “Trees”. This figure is corresponding to the results from table 1 and ROC curves in figure 8. That means they have included all samples of the 20 times running thus the amount of samples in Fig. 9 left side is (23518+84)*20 and in Fig. 9 right side is (42374+284)*20. The horizontal axis of the histograms depicts the difference of the two coefficient absolute value summations of two classes. For this typical two class classification, we can define the difference of each sample $j$ as:

$$
\text{diff}(j) = \|x_j^1\|_1 - \|x_j^2\|_1
$$

In order to present the best distribution figures, we limited the range of $\text{diff}$ to be [-0.8, 0.8] which only hide very few samples in both experiment 1 and 2. Considering the huge difference between the amounts of two class samples, meanwhile, the vertical axis depicts each class’s “class conditional probability distribution” rather than using the number of samples directly. The equation 8 shows how can we calculate the class conditional probability for each unique $\text{diff}$.

$$
\text{prob}(k) = \frac{\text{amount}(\text{diff} = k)}{\text{amount}(\text{class}(j))} \quad \text{subject to } j = 1,2
$$
Experiment 2: We also provide a set of experimental results by applying our algorithms on a pair of images which make a much harder challenge. The reason is that a considerable area in the image “Buildings” is occupied by “Stadium seats”, which are unlike the usual building roofs. The data for Experiment 2 was provided by Oregon LiDAR Consortium website.
Table 2. We repeated the same procedure as above on the new images by local window size of 11*11 and 7*7. Different dictionary sizes were tested in experiment 2 in order to show that the impact of the amount of training samples.

<table>
<thead>
<tr>
<th>Image</th>
<th>Window Size</th>
<th>Detected Local Window</th>
<th>Dictionary Size</th>
<th>Best</th>
<th>Worst</th>
<th>Average</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trees</td>
<td>11*11</td>
<td>17509(100 for training)</td>
<td>200</td>
<td>0.9733</td>
<td>0.9474</td>
<td>0.9603</td>
<td>0.0264</td>
</tr>
<tr>
<td>Buildings</td>
<td>11*11</td>
<td>21230(100 for training)</td>
<td>200</td>
<td>0.9981</td>
<td>0.9607</td>
<td>0.9878</td>
<td>0.0309</td>
</tr>
<tr>
<td>Trees</td>
<td>11*11</td>
<td>17509(500 for training)</td>
<td>1000</td>
<td>0.9798</td>
<td>0.9654</td>
<td>0.9747</td>
<td>0.0040</td>
</tr>
<tr>
<td>Buildings</td>
<td>11*11</td>
<td>21230(500 for training)</td>
<td>1000</td>
<td>0.9966</td>
<td>0.9841</td>
<td>0.9909</td>
<td>0.0034</td>
</tr>
<tr>
<td>Trees</td>
<td>7*7</td>
<td>35522(100 for training)</td>
<td>200</td>
<td>0.9670</td>
<td>0.8715</td>
<td>0.9245</td>
<td>0.0287</td>
</tr>
<tr>
<td>Buildings</td>
<td>7*7</td>
<td>25199(100 for training)</td>
<td>200</td>
<td>0.9761</td>
<td>0.8795</td>
<td>0.9295</td>
<td>0.0260</td>
</tr>
<tr>
<td>Trees</td>
<td>7*7</td>
<td>35522(500 for training)</td>
<td>1000</td>
<td>0.9538</td>
<td>0.9287</td>
<td>0.9427</td>
<td>0.0081</td>
</tr>
<tr>
<td>Buildings</td>
<td>7*7</td>
<td>25199(500 for training)</td>
<td>1000</td>
<td>0.9564</td>
<td>0.9226</td>
<td>0.9405</td>
<td>0.0113</td>
</tr>
</tbody>
</table>

Note: In our method, good performance can be guaranteed by support from a very small portion of training samples. For example, accuracies of 96% and 98% can be reached when only 0.5% samples for training. In other words, the smaller size of the dictionary will speed up the classification process.

ROC Curves:
Figure 13. ROC curves of experiment 2.

The curves on the left figure show the ROC curves with window sizes of 11*11 and 7*7 while with the same dictionary size of 1000. The curves from the right side are with dictionary size of 200. The red curves are both with window size of 11*11 and with higher ROC scores than blue curves (with window size of 7*7). However, the Fig. 10 and Fig. 11 above have shown that the smaller scanning window size the more regions detected.

**Histograms:** As in experiment 1, sample data histograms are also provided here. The top left of Fig. 14 is the result histogram with scanning window size of 11*11 and dictionary size of 200; the top right is with window size of 11*11 and dictionary size of 1000; the bottom left is with window size of 7*7 and dictionary size of 200 and the bottom right is with same window size but dictionary size of 1000.

The amount of data samples involved in each histogram can be calculated in the same way of experiment 1, which is 20 times of the amount of samples in “trees” and “buildings” (refer to table 2). Moreover, the values on the horizontal axis of the histograms are still calculated by the equation 7. Instead of depicting class conditional probability, however, the vertical axis indicates the amount of data samples for each bin in experiment 2.
In this paper we discussed terrain inpainting for voids introduced during processing, such as for bare earth DTM generation. The production processing flow presented displays terrain inpainting’s ability to automatically fill voids using only the original source data at hand and in a way that both mitigates and quantifies error, and creates minimal processing artifacts. The application of this technology is beneficial in improving LiDAR classification of buildings vs. vegetation LiDAR points. We demonstrated a method for classifying LiDAR points into bare earth, building and vegetation points. The presented method demonstrates an automated scoring technique framework which may be used in making best feature classification decision.

FUTURE WORK

The compressed sensing method discussed in this paper presented a successful approach to classifying LiDAR data into bare earth, buildings and trees. Future work is required to further demonstrate robustness. In order to give this algorithm available advantage, the 3D .las files should be provided as input and Gieger-Mode LiDAR , rather than the 2.5D gridded DEMs. Due to the redundancy of native 3D data the intrinsic properties of compressed sensing should yield favorable success. Hence, this will allow the algorithm to use all the available information. Another addition to help further classify LiDAR data would be to investigate the use of fractals. Fractals with attention to symmetry may further distinguish between scene features in real world data.
ACKNOWLEDGEMENTS

We thank Wilson & Company, Inc., Engineers & Architects, for providing expert consulting and data for evaluation and testing for Experiment 1 and Oregon LiDAR Consortium (http://www.oregongeology.org/sub/projects/ole/default.htm) for Experiment 2. This research was supported in part by NSF grant DMS-0713012

REFERENCES