

FAST APPROXIMATION FOR GEOMETRIC CLASSIFICATION OF LIDAR RETURNS

Xiaozhe Shi and Avideh Zakhor

Department of Electrical Engineering and Computer Science, University of California, Berkeley

ABSTRACT

Current LiDAR classification methods are excessively slow to be used in real-time navigation systems, even though they are useful for human perception. These methods typically analyze curvature by applying Principal Component Analysis (PCA) to each point in a point cloud. For variable-density aerial LiDAR obtained by at a shallow angle with respect to the ground rather than in a top-down fashion, the variations in density pose special challenges in terms of choosing the appropriate PCA parameters. In this paper we use gridded approximate nearest neighbor searches for fast classification of geometric features in large LiDAR point clouds. The underlying algorithm exploits spatial hashes and the forgiving nature of PCA as a part of geometric classification. We show a factor of 10-20 speed up for both actual and simulated point clouds with little or no loss in classification performance. Our approach is applicable to both uniform and variable-density aerial LiDAR datasets.

Index Terms—Aerial LiDAR, LiDAR Segmentation, 3D LiDAR Classification, PCA, Curvature Analysis

1. INTRODUCTION

Historically, LiDAR systems have been used to provide important navigational information. Aerial LiDAR range scanning can help pilots avoid collisions and identify salient terrain features, e.g. water for emergency landings or relatively flat areas for landing zones by generating a point cloud. The challenge with visualizing point clouds of raw LiDAR returns is that while they provide spatial cues to 3D structures, they often discard or add noise to other visual cues that humans normally use to differentiate objects.

Carlberg et al. addressed this problem for urban landscapes by applying 3D shape analysis to the raw point cloud [1]. More specifically, they label each point in a point cloud sequentially by curvature analysis via Principal Component Analysis (PCA). This results in a set of eigenvectors corresponding to the principal directions of the points, and a set of eigenvalues corresponding to their size and shape. With this information, it is possible to infer whether the region is planar or scatter, and use that to classify the point as an urban feature, e.g. a building, vegetation, or ground, as showcased in Figure 1.

One problem with the approach in [1] is that it is tailored to constant-density LiDAR scans, typical of overhead aerial mapping of a region. The terrain map point cloud is typically represented in 2.5D, with many more LiDAR returns concentrated in the planes parallel to the ground plane than those perpendicular to the horizon. These scans have a similar point density in most terrain features of similar materials because almost all of the features are of equal distance to the sensor. However, returns from aerial navigational LiDAR systems are different in that they are typically aimed at an angle towards the ground. Thus, the resulting point clouds can have highly variable densities: features that are close to the sensor typically result in many more returns than those that are far away.

PCA requires enough representative points of a terrain feature to result in a meaningful set of eigenvalues for feature analysis. This is especially true of LiDAR returns because the scanners incur natural noise and as such, a certain number of points are needed to overcome the noise. This can create a problem with the classifier in [1] because it is designed to only deal with static densities. By restricting the radius of points collected for PCA to be of constant size throughout a point cloud, it is necessary to set the PCA radius to a large enough value to encompass enough points for features with the lowest density. A large PCA radius can, however, lead to another problem: in high-density areas, it is necessary to run PCA on many more

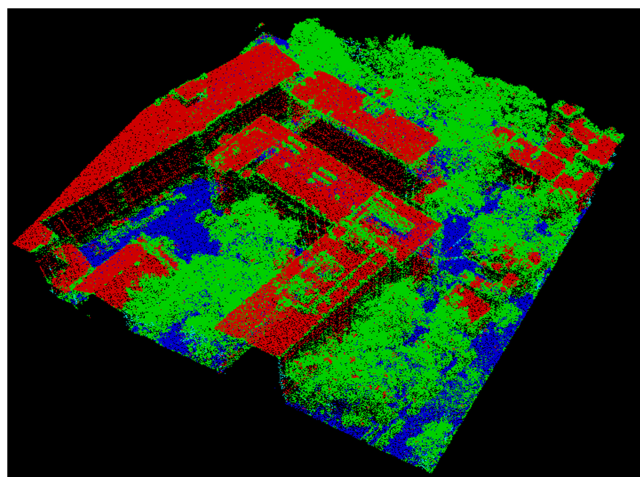


Figure 1 An example of the output from a pointwise PCA classifier performed on real LiDAR data.

points than needed, which can slow processing to a crawl.

A natural extension to the method in [1] is to start with a small radius and only increase it whenever there are not enough points for PCA. Even though we have empirically shown this to dramatically improve performance when dealing with variable-density point clouds, it is still excessively slow.

For a real time aerial LiDAR system, a much faster approach than [1] is required. The scheme must also be robust enough to deal with the density variations that result from aerial LiDAR scans at an angle, while still be tunable for speed and precision. The major issue is that Fixed Radius Near Neighbors (FRNN) is not the exact problem we need to solve for 3D shape analysis when dealing with variable-density point cloud data. This is because the numbers of points that can be considered close enough to correspond to the same geometric feature can fluctuate significantly.

In this paper, we propose a two-pronged approach on a uniform 3D grid to address this issue. Our goal is to achieve similar classification accuracy to [1] on multiple datasets of varying densities while being an order of magnitude faster for most LiDAR data. The outline of the paper is as follows: Section 2 is an overview of existing methods; in Section 3, we describe our proposed approach. Section 4 includes experimental results and conclusions are in Section 5.

2. OVERVIEW OF EXISTING METHODS

2.1. The Extended Carlberg Method

The Extended Carlberg Method (ECM) is the algorithm in [1] designed to solve the problem of point-wise fixed-radius PCA with the enhancement of increasing the radius when there are not enough points for PCA. There are four steps to this approach: (1) *Gridding*: The point cloud is first hashed into a 2-dimensional x-y grid with an arbitrary grid granularity to speed up nearest neighbor searches. (2) *FRNN Searches*: The searches are performed pointwise on the 2D grid. For each point to be classified, additional points are found in enough grid cells to encompass the minimum PCA radius defined by the dataset and added to a list of points for PCA. (3) *Radius Growing*: If the number of points collected in the cells is below the minimum threshold set for PCA, the radius is increased and more cells are added to the search list until either the number of points is sufficient for PCA or the radius reaches a maximum parameter set by the user. (4) *PCA*: The eigenpairs are found by general eigenvalue decomposition on the covariance matrix.

2.2. The General 3D Gridding Algorithm

The general gridding algorithm is designed to solve the problem of K Nearest Neighbor (KNN) with a 3D grid [2]. It consists of two steps: (1) *Gridding*: The point cloud is

hashed into a 3D grid. (2) *Nearest Neighbor Searches*: For each point to be searched (the reference point) the point is hashed again and a starting cell is found. From there the search region is expanded by adding adjacent cells to the search region. After a minimum of k points are added, the search area is increased by a preset number of layers of cells for more accuracy. Finally, after the potential nearest neighbor points list is complete, the k nearest neighbors are returned by computing the kth farthest point from the reference point, and returning all points that are closer than that.

3. PROPOSED METHOD

Our proposed method takes the general 3D grid-based FRNN scheme and modifies it to approximate PCA. We propose to use a FRNN search for high-density areas so as to encompass the entire geometric feature, while using a typical ANN search for low-density areas so as to obtain enough points to perform PCA on.

We start with the same gridding approach described in Section 2.2. However, when we do an initial search of the minimum radius, we make a decision based on how many points we find. If we have more points than we require for PCA, we simply cull off all the points above a certain distance to the original reference point. This approximates a FRNN search for high-density areas. If there are insufficient points after the distance culling, we then expand the search radius until we find enough points in the grid to meet the minimum number of points required for PCA. This is effectively an ANN search for low-density areas.

In [1], the PCA radius must be large enough to encompass enough points to overcome the noise in the LiDAR scanner for all relevant terrain features. The problem with that method is that in point clouds with large variations in density, the PCA radius required is much larger than is necessary for the higher-density regions.

With our hybrid method, the PCA radius automatically expands to include relevant points in areas of low density, where the scanner noise is most apparent. This provides a reasonable classification approximation of the method in [1], while resulting in a better run time as well as some parallel scalability [4]. There are 4 major parts to the algorithm: grid construction, range search, distance culling, and principal component analysis (PCA), which are described below.

3.1. Parameters

Grid granularity: Grid granularity is the length of each voxel on the grid. A lower granularity can increase PCA times because excess points are considered for range culling. However, a very high granularity increases the time required for range searches and grid construction. This parameter should be set to a value close to the PCA radius.

Minimum PCA radius: This is the minimum radius around each point to be considered in performing PCA. This parameter varies with the density of the point cloud. A larger radius is more accurate in classifying dense noisy LiDAR captures, but it also slows down the classification algorithm because more points are considered on average.

Minimum Points for PCA: This is the minimum number of points that the range search must find in order to perform PCA. This parameter only becomes important if the density of the point cloud varies significantly between regions.

Maximum Points for PCA: This is the maximum number of points the PCA classifier considers per point. For portions of point clouds that are dense, it is desirable to limit the number of points considered. A higher value here results in more accuracy, but also increases runtime. Generally, this parameter is directly proportional to the density of the point cloud.

3.2. Details of the Algorithm

Grid construction: Before actually querying points, we need to construct a data structure in memory that organizes the points in a spatially coherent manner. We use a spatial hash function similar to [3] in order to organize the points into a voxel grid.

Range Search: We do the range search the same way as the approach described in Section 2.2. During the range search, we keep track of all non-empty cells within the radius of the search. For each point, we search the cell that contains the point and the cells around it until we find enough points required for PCA.

Distance Culling: If the number of points needed for PCA is exceeded, we cull off all the points that are farther than the PCA radius. This simulates a FRNN search for areas with a high point-density. However, if after culling the points, there are fewer than the minimum points required for PCA, we simply use all available points to simulate an ANN search. In this case, the density around the specific point must be low. Low-density areas in the point cloud require a less exact body of points to represent the geometric features in the area; the only requirement becomes to have a collection of points that encompasses a wide enough area to characterize the feature. We take advantage of these two properties and use the points we have found already in the initial range search when encountering low density areas. This provides resilience towards datasets with variable density.

Curvature Analysis via PCA: In this step, we first compute the 3×3 symmetric covariance matrixes of the points and then perform singular value decomposition on this matrix by using the QR algorithm [4]. After PCA, we are left with 3 eigenpairs. The geometric classifications are

then decided the same way as [1]: (a) Planar, if $\lambda_1 \approx \lambda_2 \gg \lambda_3$, (b) Scatter if $\lambda_1 \approx \lambda_2 \approx \lambda_3$, (c) Linear if $\lambda_1 \gg \lambda_2 \approx \lambda_3$, (d) Ground if $\lambda_1 \approx \lambda_2 \gg \lambda_3$ and the normal vector's z value is above a certain threshold, (e) Other if all other tests fail.

4. EXPERIMENTAL RESULTS

Our proposed classifier is tested on various real and synthetic datasets. These point clouds include 3 datasets from a real LiDAR capture with generally uniform point density, 3 aerial LiDAR simulations with variable density, and 14 synthetic uniform random point clouds.

The accuracy results are obtained by assuming results from [1] to be the ground truth. The tests are run on a PC running Windows XP with an Intel Xeon X5355 CPU at 2.66 GHz and 4 GB of RAM. The results are shown in Tables 1 and 2.

Dataset	Time	Points	ECM Time	Speed up	Accuracy
Real1	7.9s	492989	138.3s	17.46	95.97%
Real2	9.0s	596996	177.0s	19.77	95.87%
Real3	9.4s	637573	209.2s	22.06	95.80%

Table 1 Runtimes, speed-up rates, and accuracy data for constant-density aerial LiDAR returns. Minimum (maximum) number of points for PCA is 15 (200). Grid length is 0.5.

Dataset	Time	Points	ECM Time	Speed up	Accuracy
Aerial1	55.0s	3762830	814s	14.80	99.21%
Aerial2	119.4s	7226259	1652s	13.84	99.40%
Aerial3	14.8s	704162	200s	13.54	97.63%

Table 2 Runtimes, speed-up rates, and accuracy data for variable-density synthetic aerial LiDAR. Minimum (maximum) number of points for PCA is 15 (1000). Grid length is 0.5.

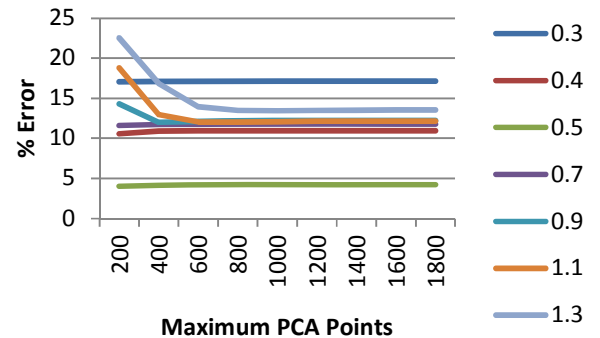


Figure 2 Percent error for the Real 2 dataset for various PCA radii as a function of maximum PCA points.

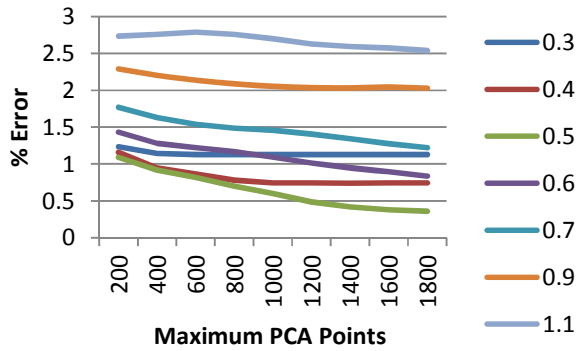


Figure 3 Percent error for variable-density synthetic aerial dataset 2 for various PCA radii as a function of maximum PCA points.

As shown in Tables 1 and 2, our proposed method is faster than [1] by about a factor of 20 and 14 for real and synthetic data respectively. Moreover, it is able to reach a high degree of fidelity as compared to the original ground truth method in [1] given a proper PCA radius. In addition to the data confirmation, the results are visually indistinguishable as well. There is a considerable amount of tolerance to the maximum PCA points as shown in Figures 2 and 3. In contrast, there seems to be a lack of tolerance to the PCA radius. In practice however, we have found that the optimum PCA radius for our scheme is proportional to that used for generating the ground truth via ECM, and as such, is not scene-dependent.

We run the synthetic datasets with fixed PCA radii and grid granularity, but variable data size as shown in Figure 4. Since the overall dimensions are constant, the density rises with more points. As shown in Figure 4, our proposed method shows a linear relationship between the number of points and runtimes while the ECM rapidly degrades in speed.

The speed up factor of our algorithm on LiDAR data varies between 13.5 to 22 times. Our method performs just as well, if not better on the uniform-density datasets as the variable-density ones. One reason might be that the real dataset has higher point density for each terrain feature than the synthetic aerial LiDAR data, and the improvements of our proposed method allow us to use a smaller PCA radius than in [1].

Qiu et al. also use point-wise PCA on a point cloud for 3D registration [5]. The difference is that they only use a KNN of 50 points to do PCA. Also, they use a kd-tree rather than a grid. It is interesting to note that their sequential CPU processing times for a 68k point dataset are actually slower than our times for a 704k point dataset; i.e. approximately 27 seconds versus our 14 seconds. Furthermore, the average number of points used in our case is considerably larger. Note that 50 nearest points used in [5] are too few to result in sufficient classification accuracy for a variable-density

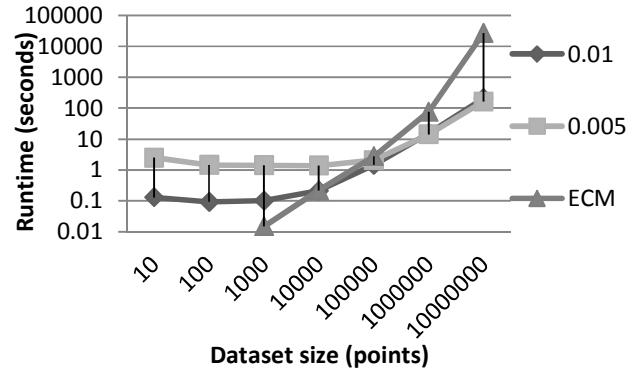


Figure 4 Synthetic benchmark tests on various numbers of points. There are two sets of results with the proposed method: one with a grid granularity of 0.005, and one with a grid size of 0.01.

point cloud. While the approaches are different, this shows that our method is competitive in regards to more complex data structures.

5. CONCLUSIONS

Our proposed algorithm is considerably faster than the existing approaches while providing high fidelity to the results obtained from the ECM. In addition, it seems to scale well with point density. The approach in [1] on the other hand degrades rapidly when both density and points are increased. Future work involves improving tolerance to parameters and multicore or GPU implementations.

6. REFERENCES

- [1] Carlberg, M.; Gao, P.; Chen, G.; Zakhor, A.; , "Classifying urban landscape in aerial LiDAR using 3D shape analysis," Image Processing (ICIP), 2009 16th IEEE International Conference on , vol., no., pp.1701-1704, 7-10 Nov. 2009
- [2] Weber, R.; Schek, H.; Blott, S.; "A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Spaces," International Conference on Very Large Data Bases, 1998. VLDB '98. Proceedings of the 24rd International Conference on Very Large Data Bases, Pp. 194-205.
- [3] Leite, P.; Teixeira, J.; de Farias, T.; Teichrieb, V.; Kelner, J.; , "Massively Parallel Nearest Neighbor Queries for Dynamic Point Clouds on the GPU," Computer Architecture and High Performance Computing, 2009. SBAC-PAD '09. 21st International Symposium on , vol., no., pp.19-25, 28-31 Oct.
- [4] Watkins, D. S.; "Understanding the QR Algorithm," Society for Industrial and Applied Mathematics. SIAM Review, Vol. 24, No. 4 (Oct., 1982), pp. 427-440
- [5] Qiu D., May S., and Nüchter A.; "GPU-accelerated nearest neighbor search for 3D registration," International Conference on Computer Vision Systems, Liège, Belgium, 2009.