Analysis of LIDAR Point Cloud Data Using Poisson Surface Reconstruction

Ruoyang Song and Jianfei Ma

Dept. of Computing Science, University of Alberta, Canada

Abstract. Tree volumes play a significant role in analyzing the bioinformation of a forest. It provides biologists ground truth to analyze the change in biodiversity according to the number of species and their distribution. Light Detection and Ranging (LiDAR) scanning can be used to scan large multi-hectare areas of forests with thousands of trees. During the past decades, computer scientist has been working on calculating In this paper, we investigate into several point cloud segmentation algorithms for dividing the irregular shaped buttress into regular shaped object for calculation, and we promote a new method for analyzing the volume using Poisson Surface Reconstruction algorithm. Our proposed method has four main steps: 1) Removing outliers using a Statistical Outlier Removal filter, 2) Computing normals of the point clouds, 3) Apply the Poisson Surface Reconstruction algorithm to the point cloud, 4) Calculate the mesh volume. We examinated Euclidean Cluster Extraction algorithm and Cylinder segmentation algorithm for irregular shaped buttresses.

1 Introduction

For decades, scientists have been dedicated for calculating the accurate value of tree volume. The volume of tree can contribute significantly in multiple fields of studies. For example, biologists can analyze the basic bioinformation of this forest and species may live inside it. In addition to that, the detailed tree specie data can be used for biologists to analyze the change in biodiversity according to the number of species and their distribution. Scientists from different background have been approaching this problem using all kinds of technologies. First of all, a vast majority of research effort in tree species recognition has been devoted to retrieve the tree texture and how to match to the acknowledged database.In 2004, Norbert Haala and his fellow coauthors derived geometric quantities like position and diameter of trees from the IMAGER 5003 LIDAR system, and texture parameters from the high resolution panoramic imagery shoot by a EYSCAN camera [1]. From the result, we can see that since the texture differs between stems, using only texture to identify tree species is a novel approach but has limited correct rate. And in 2013, Othmani improved this methodology by using the 3D geometric texture of the bark, rather than 2D texture from imagery. Othmani computed the texture features using a combination of the Complex

Wavelet Transforms (CWT) and the Contourlet Transform (CT), and classification was done using the Random Forest (RF) classifier. After testing using a dataset composed of 230 samples, even the worst average overall accuracy has increased to 80.62% (with a standard deviation of 5.82%) [2]. Secondly, hyperspectral data is also widely used in analyzing tree species. In 2010, Puttonen collected a database contains the shape parameters of three boreal tree species which is merged from two dataset: One dataset is actively scanned hyperspectral point cloud, and the second dataset is collected with a spatially accurate LIDAR [3]. And the result shows a huge increase in correct rate compared to previous methodology: with a combination of 2 features from each dataset, the average classification accuracy was over 85% for all species. And in 2013, Vauhkonen improved the tree species classification methodology by analyzing the hyperspectral reflectance that had penetrated through the foliage [4]. In this study, he focused on the species-specific differences as recorded by an active hyperspectral LiDAR in a laboratory measurement of pine and spruce trees. He divided the reflectance values into several categories based on Normalized Difference Vegetation Index (NDVI) and reflectance losses. And by analyzing the pulse penetration and reflectance, he got a 78% to 97% of correct rate on 18 spruce and pine trees. Lastly, in the latest study done by Åkerblom in 2017, not like any other study done before, he did not start from using the 3D point cloud data directly, trees are first reconstructed as quantitative structure models (QSM) [5]. For each tree, the QSM reconstruction process creates another partition of the tree point cloud into small patches, and then segments them into the stem and branches. Then he computed a scalar value for each QSM model, and match them with a table of tree features as shown in Figure 1. In this way, based on applying this methodology to 1200 trees in different species, it shows a quite convincing result of an average classification accuracy above 93%. However, all these methodologies mentioned share one hypothesis: the assumption that all the trees' bottom part look like cylinder. However, there are still many other trees that have buttress root lives on earth. Applying above methodologies to trees that have buttress root may lead to a biased result. As a result, we will focus on how to improve the classification accuracy to trees with buttress root. divide an irregular shaped buttress into several cylinder objects or Euclidean Cluster. We assumed that it must be approachable for calculating the accurate volume of these segmented regular shaped objects using convex hull algorithm. However, the result we got has a significant deviation with reality and this biased result does not support us in the next step of calculating the accurate value of the buttress. As a result, we moved to a new approach using Poisson Surface Reconstruction. Our goal is to calculating the result as accurate as possible. The general steps of the proposed method includes removing outliers using a Statistical Outlier Removal filter, computing normals of the point clouds, apply the Poisson Surface Reconstruction algorithm to the point cloud, and calculate the mesh volume. The remaining sections of this paper are structured as follows: Section 2 and 3 covers materials and methods, which provides details on the dataset, preprocessing and the proposed method for calculating the volume of an irregular shaped buttress

using Poisson Surface Reconstruction. Experimental results and discussion are covered in Section 4. Finally, Section 5 presents the conclusion.

2 Data Set

The point cloud data used in the preparation of this article was collected from scanning two tree samples using LiDar camera. As shown in the Fig.1, the first tree sample has a straight brunch which allow us to reconstruct it into a cylinder easily. On the controversy, the another tree sample has a complicated irregular shaped buttress structure as shown as Fig.2.

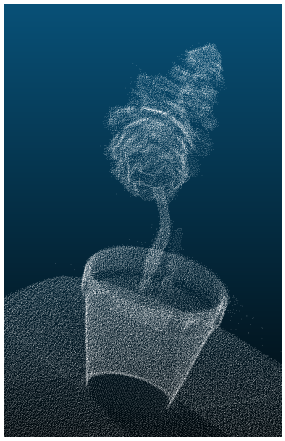

Fig. 1: Regular shaped brunch.

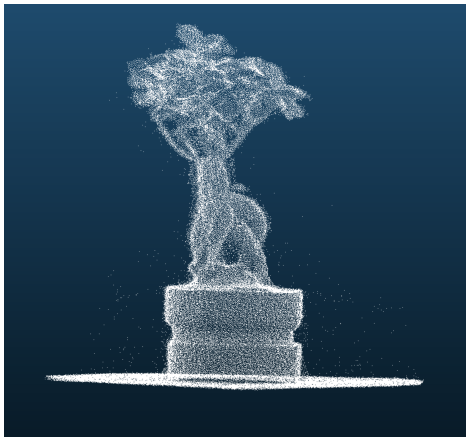

Fig. 2: Irregular shaped buttress.

3 Abortive Experiments

Since we focused on analyzing the point cloud of irregular shaped data, our first approach is to segment the irregular shaped part into smaller objects with regular shapes.

3.1 Euclidean Cluster Extraction algorithm

Euclidean clustering essentially groups points that are close together. We set a "closeness" threshold, such that points within this threshold are considered to be part of the same cluster. We also set a minimum and a maximum number of points that a cluster can contain. This helps us to filter out noise (isolated single points) or parts of the surface that weren't segmented out [6]. The outline of this algorithm is:

1. Create a Kd-tree representation for the input point cloud dataset P.
2. Set up an empty list of clusters C , and a queue of the points that need to be checked Q.
3. For every point p in P , perform the following steps:
   (a) Add p to the current queue Q.
   (b) For every point p in Q do:
      i. Search for the set of point n neighbors of p in a sphere with radius r.
      ii. For every neighbour n , check if the point has already been processed, and if not add it to Q.
   (c) When the list of all points in Q has been processed, add Q to the list of clusters C, and reset Q to an empty list
4. The algorithm terminates when all points p have been processed and are now part of the list of point clusters C

From the Fig.3, we can see that the tree sample was segmented into several Euclidean Cluster. However, we have to manually set up parameters for threshold and maximum number of points, and these two parameters vary from one tree to another, which indicates that this approach can not be generalized to all buttress without manual operation.

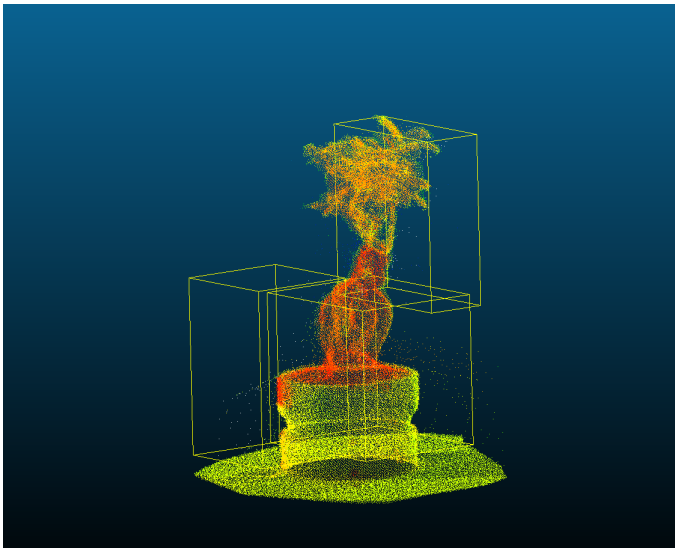

Fig. 3: Result for applying the Euclidean Cluster Extraction algorithm to the complex tree sample.

3.2 Cylinder segmentation algorithm

In 2014, Trung-Thien Tran published his Sample Consensus Segmentation algorithm for extraction of cylinders and estimation of their parameters from point clouds [7]. Since this algorithm is widely used in multiple fields of point cloud analyzing industry, we can use this algorithm from PCL library as following:

```
// Estimate point normals
ne.setSearchMethod (tree);
ne.setInputCloud (cloud_filtered);
ne.setKSearch (50);
ne.compute (*cloud_normals);

// Create the segmentation object for the planar model and set all the para
seg.setOptimizeCoefficients (true);
seg.setModelType (pcl::SACMODEL_NORMAL_PLANE);
seg.setNormalDistanceWeight (0.1);
seg.setMethodType (pcl::SAC_RANSAC);
seg.setMaxIterations (100);
seg.setDistanceThreshold (0.03);
seg.setInputCloud (cloud_filtered);
seg.setInputNormals (cloud_normals);
```

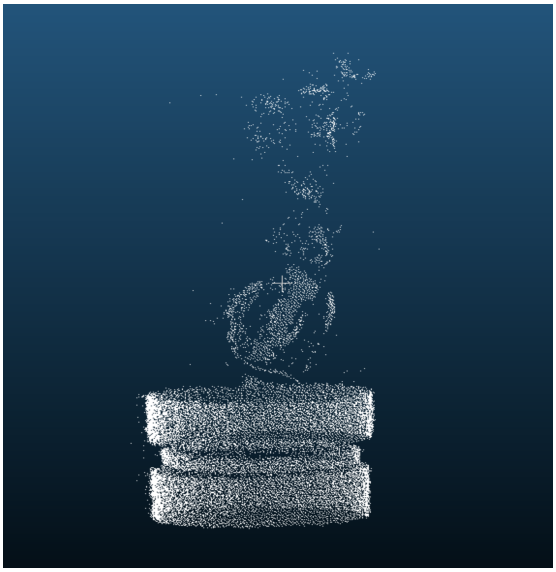

Fig. 4: Result from Sample Consensus Segmentation algorithm

However, from Fig. 4, we can see this algorithm does not perform very well. The reason behind this is that the buttress part of the point cloud data we collected was intertwined into together, and it does not contains a relatively big empty space between each buttress. This Sample Consensus Segmentation algorithm should have better performance for a tree like Fig. 5.

Fig. 5: Tree with widespread buttress

4 Proposed Method

There are three general steps included in out proposed methods:

The overview of our proposed method is presented in Fig.**??** which has 3 general steps including: 1) Preprossing, 2) Computing normals of the point clouds, 3) Apply the Poisson Surface Reconstruction algorithm to the point cloud, 4) Calculate the mesh volume. Next, each step is explained in detail.

4.1 Preprossessing

Preprocessing is a fundamental step of calculating the accurate volume of a tree point cloud. In this paper, for reconstructing a meticulous mesh using the point cloud, several prepossessing steps are needed.

Convert To PCD Format Since the point cloud data we collected was initially stored in TXT format, we need to convert our data into PCD format for the following advantages over other file format [8]:

1. the ability to store and process organized point cloud datasets – this is of extreme importance for real time applications, and research areas such as augmented reality, robotics, etc;

2. binary mmap/munmap data types are the fastest possible way of loading and saving data to disk.
3. storing different data types (all primitives supported: char, short, int, float, double) allows the point cloud data to be flexible and efficient with respect to storage and processing. Invalid point dimensions are usually stored as NAN types.
4. n-D histograms for feature descriptors – very important for 3D perception/-computer vision applications

And we added the following header to create a PCD file:

```
'''# .PCD v0.7 - Point Cloud Data file format
VERSION 0.7
FIELDS x y z
SIZE 4 4 4
TYPE F F F
COUNT 1 1 1
WIDTH {0}
HEIGHT {1}
VIEWPOINT 0 0 0 1 0 0 0
POINTS {2}
DATA ascii\n'
```

Removing outliers using a Statistical Outlier Removal filter After we convert our point cloud into PCD format, we need to remove the noise in the point cloud for generating a meticulous mesh in the next step. We used the following PCL library in this step [9]:

```
using namespace pcl;

StatisticalOutlierRemoval<PointXYZ> sor;
sor.setInputCloud (cloud);
sor.setMeanK (50);
sor.setStddevMulThresh (1.0);
sor.filter (*cloud_filtered);
```

We can see the comparison before and after removing the outliers from Fig. 6 All the outliers which are floating outside of the main object is removed and the result is good start point for generating the mesh.

4.2 Computing normals of the point clouds:

There are many approaches for estimating the normals of the point cloud. In 1992, Hugues published his work about surface reconstruction from unorganized points [10], in this article, he estimates normals approximating a tangent plane with a regression that is computed efficiently by principal component analysis (PCA). And Alexandre presents a new method based on a robust version of the

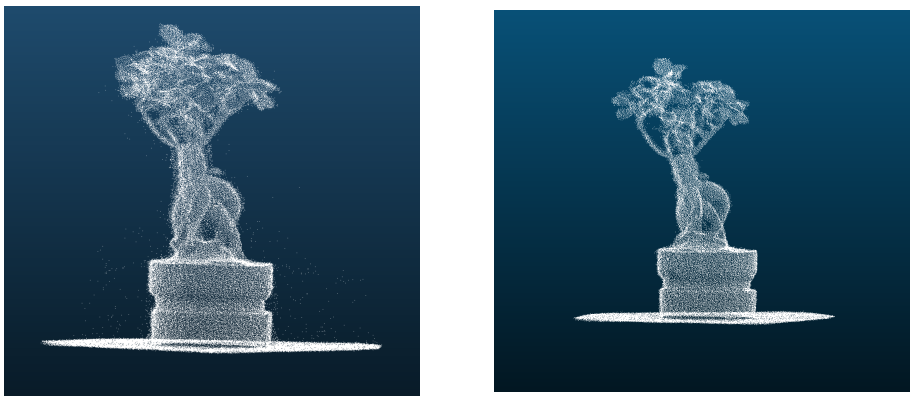

Fig. 6: Comparison between the point cloud before and after removing the outliers

Randomized Hough Transform (RHT) which consider the filled Hough transform accumulator as an image of the discrete probability distribution of possible normals. In the following code, we used the normal estimation function provided by PCL which is based on Hugues implementation.

```
using namespace pcl;
NormalEstimation<PointXYZ, Normal> ne;
ne.setInputCloud (cloud_filtered);
search::KdTree<PointXYZ>::Ptr tree (new search::KdTree<PointXYZ> ());
ne.setSearchMethod (tree);
PointCloud<Normal>::Ptr cloud_normals (new PointCloud<Normal>);
ne.compute (*cloud_normals);
```

Fig. 7 shows the result of estimating the normals.

4.3 Using Poisson Reconstruction to generate a mesh:

In 2006, Michael published his work of reconstructing a surface using Poisson formulation which considers all the points at once, without resorting to heuristic spatial partitioning or blending, and is therefore highly resilient to data noise. We used the Poisson Reconstruction provided by the PCL library as following:

```
Poisson<PointNormal> poisson;
poisson.setDepth(9);
poisson.setDegree(2);
poisson.setSamplesPerNode(1);
poisson.setScale(1.25);
poisson.setIsoDivide(8);
poisson.setConfidence(0);
poisson.setManifold(0);
```

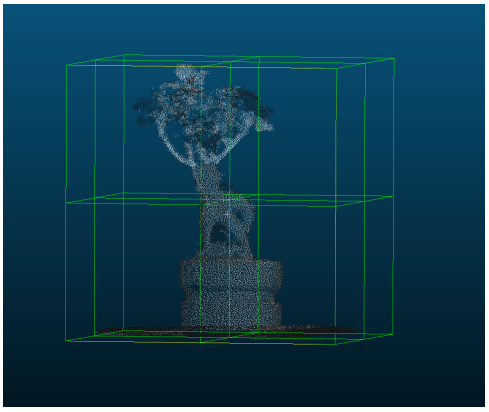

Fig. 7: Result of estimating normals

```
poisson.setOutputPolygons(0);
poisson.setSolverDivide(8);
poisson.setInputCloud(cloud_filtered_normals);
```

The result of poisson reconstruction can be shown in Fig. 8.

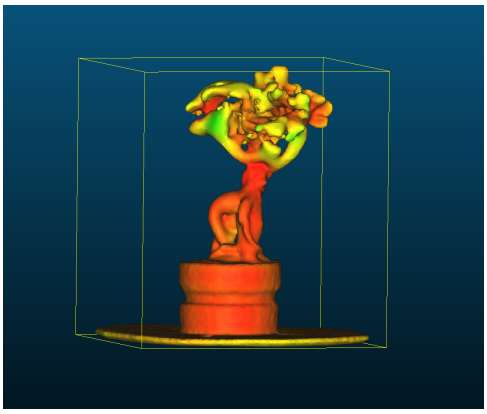

Fig. 8: Result of poisson reconstruction

4.4 Segment the buttress part

If we only want to calculate the volume of the buttress part we can use the segmentation tool in CloudCompare, we segment the buttress part from the original mesh. Notice that there may be redundant part included, we can apply another segmentation on it. The comparison between before and after adjustment can

be seen from Fig. 9. As noticed in Fig. 10, There are two significant holes on our mesh after segmentation. For our algorithm, there will be a large error if the mesh has holes. We will use Meshlab[v2016.12] to close these them. We import our mesh in the Meshlab, run the Close Holes feature under the Filter menu. We can adjust the thresholds of the max size for Close Holes; the holes should be closed properly.

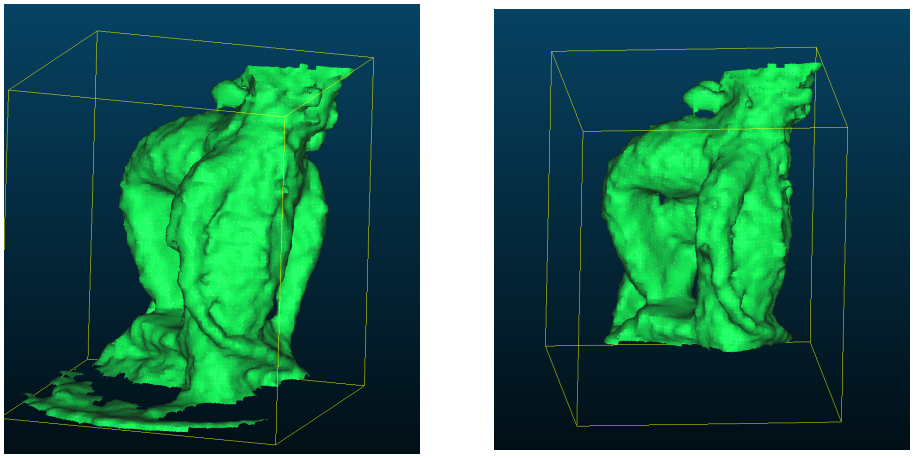

Fig. 9: Comparison between the mesh before and after segmentation

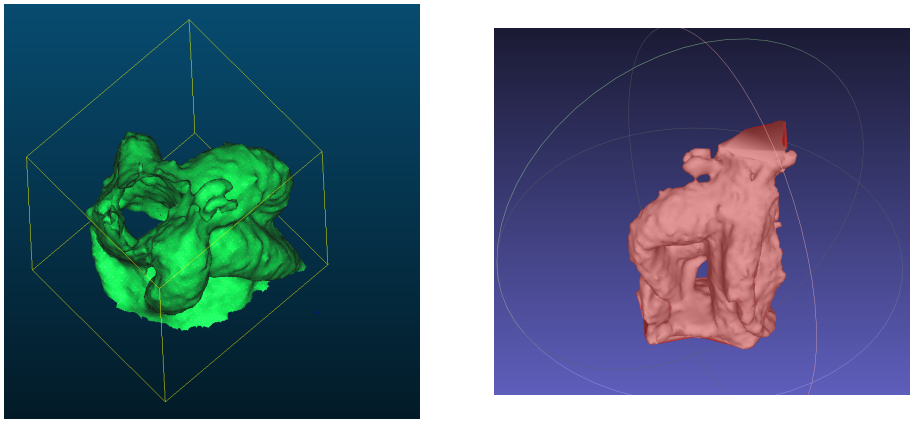

Fig. 10: Comparison between the mesh with and without hole

4.5 Calculating the volume from the buttress mesh

Since we have the accurate mesh presentation of the buttress, we use the following algorithm to calculate the volume:

```
using namespace pcl
float volumeOfMesh(PolygonMesh mesh) {
float vols = 0.0;
PointCloud<PointXYZ>::Ptr cloud (new PointCloud<PointXYZ>); fromPointCloud2(m
for(int triangle=0; triangle<mesh.polygons.size(); triangle++) {
PointXYZ pt1 = cloud->points[mesh.polygons[triangle].vertices[0]];
PointXYZ pt2 = cloud->points[mesh.polygons[triangle].vertices[1]];
:PointXYZ pt3 = cloud->points[mesh.polygons[triangle].vertices[2]];
vols += signedVolumeOfTriangle(pt1, pt2, pt3);
}
return Math.Abs(vols.Sum());
}
```

The final result we get from this algorithm is shown in Fig. 11.

Fig. 11: Result of calculating the mesh volume

5 Conclusion

We presented a point cloud analysis method for calculating the volume of irregular-shaped buttress. This advantage of this method over other volume estimation methods is that this method has high accuracy all different shaped buttress. Since this method is based on Poisson Reconstruction, we can keep a high accuracy when applying this method to an irregular shaped buttress just like we shown in the article. In future work, we can focus on developing an automative workflow of the buttress segmentation and hole closing.

References

1. Norbert Haala, Ralf Reulke, Michael Thies, Tobias Aschoff (2004). Combination of terrestrial Laser Scanning with high resolution panoramic Images for Investigations in Forest Applications and tree species recognition.
2. Othmani, A., Voon, L. F., Stolz, C., Piboule, A. (2013). Single tree species classification from Terrestrial Laser Scanning data for forest inventory. Pattern Recognition Letters, 34(16), 2144-2150. doi:10.1016/j.patrec.2013.08.004.
3. Puttonen, E., Suomalainen, J., Hakala, T., Räikkönen, E., Kaartinen, H., Kaasalainen, S., Litkey, P. (2010). Tree species classification from fused active hyperspectral reflectance and LIDAR measurements. Forest Ecology and Management, 260(10), 1843-1852. doi:10.1016/j.foreco.2010.08.031.

4. Vauhkonen, J., Hakala, T., Suomalainen, J., Kaasalainen, S., Nevalainen, O., Vastaranta, M., . . . Hyyppa, J. (2013). Classification of Spruce and Pine Trees Using Active Hyperspectral LiDAR. IEEE Geoscience and Remote Sensing Letters, 10(5), 1138-1141. doi:10.1109/lgrs.2012.2232278.
5. Åkerblom, M., Raumonen, P., Mäkipää, R., Kaasalainen, M. (2017). Automatic tree species recognition with quantitative structure models. Remote Sensing of Environment, 191, 1-12. doi:10.1016/j.rse.2016.12.002.
6. Euclidean Cluster Extraction Documentation. Retrieved from http://pointclouds.org/documentation/tutorials/cluster_extraction/.
7. Tran, T., Cao, V., Laurendeau, D. (2015). Extraction of cylinders and estimation of their parameters from point clouds. Computers & Graphics, 46, 345-357. doi:10.1016/j.cag.2014.09.027.
8. PCD Documentation. Retrieved from http://pointclouds.org/documentation/tutorials/pcd_file_format.php/.
9. Filter Documentation. Retrieved from http://pointclouds.org/documentation/tutorials/statistical_outlier.php/.
10. Hoppe, H. (1995). Surface reconstruction from unorganized points. Ann Arbor, Mi: University Microfilms International.