# Segmenting Individual Trees from Single-Scan LiDAR Point Clouds

Darrell Hoffman
*Student Number: 210305771*
*ec211057@qmul.ac.uk*

Supervisors: M. Hansard, J. Paredes

***Abstract*—Several studies have produced algorithms to segment individual trees from LiDAR point clouds. Many of these algorithms assign eigenvector values to points to discover relatively flat surfaces, representing tree stems in the data. This technique fails when the point cloud is not sufficiently dense, so running these algorithms will likely require pre-processing to merge multiple point clouds. This lead us to develop a segmentation algorithm that can effectively segment single scan LiDAR point clouds. Our segmentation algorithm assigns densities to each point in the 0-2m height range, based on the number of points within a 25cm radius of that point and filters out points below a user defined density threshold. Stems are then clustered using K-means clustering, and points above 2m in height are then clustered to the stems based on Euclidean Distance in 5cm vertical slices. Eighty-six of 101 target trees were correctly segmented and 17 non-tree objects were mistakenly segmented in the seven point clouds captured. Twenty-five of 26 trees were correctly segmented from a more dense point cloud, outperforming an existing eigenvector based algorithm. F-scores calculated for 6 individual trees based on the number of correctly segmented points ranged from 0.544 to 0.971.**

***Index Terms*—sensors, segmentation**

## I. Introduction

The world's forests provide numerous essential ecosystem services. Forests sequester large amounts of carbon annually [1], benefit human health by improving air [2] and water quality [3], and sustain biodiversity, which in turn ensures more robust and healthy forest ecosystems [4]. Managing forest resources to enhance carbon sequestration, support biodiversity, harvest sustainably, control pests, and prevent destructive forest fires will benefit from ever better appraisal of the state of those resources through improved monitoring techniques in the face of increasing stress on forest health globally.

Gauging forest dynamics manually requires measurements (properties typically collected in forest surveys such as diameter at breast height (DBH) [5], tree height [6], crown width [7] [8], and estimates of woody biomass [9]) of trees in a sample plot, producing sparse representations of each tree in the dataset [10]. These surveys also require a significant investment of person-hours to collect the data. Remote-sensing techniques, including the use of light detection and ranging (LiDAR) can increase the volume and speed of spatial information capture in forests. Terrestrial laser scanning (TLS) is an implementation of LiDAR scanning at ground level, in contrast to Aerial Laser Scanning, which captures LiDAR point clouds from above. TLS can capture tree and understory structures at a greater level of detail [11], producing a point cloud that can accurately represent the physical structure of the forest. However, the individual tree metrics such as DBH, height, and crown width can only be extracted from the point cloud once the challenging task of tree segmentation is completed.

Clustering algorithms struggle to properly segment forest point clouds. DBSCAN, for example, tends to over-segment the data [12]. Accurate 3D reconstructions of forest plots from photographs are possible [12], but require upwards of 100 photographs per plot and intensive processing to create point clouds from the data. Though the results may provide a better representation of a forest stand than the metrics recorded during a forest survey, the time demands on the researchers to collect the data is not significantly reduced. Several applications have been developed to extract tree parameters from TLS point clouds including: SimpleTree [13], CompuTree [14], LiForest [15], AutoStem [16], 3D Forest [17], and TreeSeg [18]. A back propogation neural network (BPNN) was successfully used to classify objects, including trees, from LiDAR data [19] to inform an unmanned ground vehicle in an urban setting. The streets of a city, where objects tend to be spaced apart, differ greatly from a complex forest stand. Other research has focused on optimizing algorithms to automatically segment trees within complex natural forests. Liang et al. [20] used the Z-axis component of the normal vector for each point to indicate which points in the cloud belonged to vertically oriented planes and could be identified as belonging to the trunk of a tree with 73% accuracy. Using a similar method, Olofsson and Holmgren [21] established a 3D-voxel grid populated with their point cloud data, using eigenvectors of points to find flat objects and connect grid cells containing aligned, flat objects. These methods are able to successfully distinguish tree stems from branches and foliage, but in natural forests, tree canopies overlap and it can be difficult to distinguish individuals. However, 3D forest [10] and methods developed by Liu et al. [11] use similar approaches to better segment more complex forest point clouds into individual stems and then build out from the stems to include branches and foliage. For example, the method used by Liu et al. [11] involved establishing a vertical boundary within the point cloud between the understory and canopy and assigning the remaining points above the boundary, nearest to

already classified points, to those clusters until all points were classified. This method correctly segmented stems with an F-score of 0.96.

Existing methods using eigenvectors to discover flat surfaces [20] or differentiate curvature [22] in the data require high-quality, dense point clouds. This requires either a very high-quality sensor that can capture such densely populated point clouds, or pre-processing must be done to merge several point clouds. If a forestry professional without access to the needed level of equipment or a substantive coding background wished to use tree segmentation software, this could present a barrier to using these methods. Furthermore, with LiDAR technology becoming more widely available (even being included in the latest iPhone model) there is value in producing a segmentation methodology that can successfully segment trees from lower density point clouds produced from a single scan.

None of the algorithms listed were developed in Python. With Python being the dominant language used in machine learning applications, developing methods to work with forest TLS data in Python is desirable to open doors to the application of machine learning techniques that could predict tree features such as species and biomass. It is ideal that developers fluent in Python can edit or improve the code as they wish before extracting whatever further value they wish from the data. With the insight that the generally vertical stems of trees would produce high point densities on a horizontal X,Y grid, after the vertical Z-dimension was removed from the data, an alternative tree segmentation method will be demonstrated using point densities in the X,Y dimensions to locate tree stems and then adding the branches and leaves by clustering points step-wise from the bottom up.

Our aim is to produce a tree segmentation algorithm that can be used by anyone with a basic understanding of the plot where their data was captured, is developed in Python, and that can process relatively sparse point clouds compared to existing methods.

*Contributions*

M. Hansard suggested the novel component of the segmentation algorithm, which involves finding stems by removing the vertical Z-dimension and searching for the highest point densities in the X,Y dimensions of the point cloud. He also obtained all of the equipment described in the Data Capture Section, assisted with site selection and data capture, provided feedback on the algorithm and report, and suggested using Gaussian Mixture Models (GMMs) as an alternative clustering method for comparison. J. Paredes guided me through the process of operating the sensor, assisted with site selection and data capture, provided feedback on the algorithm, and suggested using Hierarchical DBSCAN (HDBSCAN) as an alternative clustering method for comparison. H. Huen provided guidance on the equipment setup and P. Goddard soldered a voltage regulator into the wire between the external power source and the sensor to ensure it received the correct voltage.

## II. Methods

The tree segmentation algorithm was developed in Python, making use of the NumPy, pandas, Seaborn, Matplotlib, Plotly, SciPy, and scikit-learn libraries.

*Segmentation Steps*

1) Select plot to be segmented
2) Remove terrain (ground points) from dataset
3) Split dataset into below 2m height (stems) and above 2m height (crown)
4) Remove points below user defined density threshold from stem dataset to discover stems
5) Cluster stems using K-means clustering
6) Slice bottom 5cm of crown dataset and cluster points to nearest stem based on Euclidean Distance (1)
7) Take next 5cm slice of crown dataset and cluster to nearest cluster, repeating until the maximum height of the dataset is reached

*Euclidean Distance Formula*

$$d(p,q) = \sqrt{\sum_{i}^{n} (q_i - p_i)^2} \tag{1}$$

The distance formula used to cluster points and to determine metrics of the segmented trees is the Euclidean Distance $d$ (1), where $p$ and $q$ represent any two points in the point cloud, $n$ represents the number of dimensions involved in the distance calculation, and $i$ represents the current dimension in which the points are being compared.

### A. Data Capture

Point clouds were captured using an Ouster OS1-64 portable LiDAR scanner, which captures 64 vertical channels over a 45° vertical field and a 360° horizontal field, with a range of 120m. Captured points can be expected to have a precision between ±0.7cm to ±5cm. Data was captured at two location in London, Carlton Square and Gardens, Bethnal Green (51.524162, -0.044519), and Tower Hamlets Cemetery Park, Southern Grove (51.523171, -0.024998). The LiDAR scanner was connected to a laptop and an external power source and scans recorded using the Ouster Sensor SDK (https://static.ouster.dev/sdk-docs/), which provides code allowing users to interact with the scanner. LiDAR scans were taken at approximately 1.3m above the ground surface for 10 seconds at a rate of 10 frames per second and recorded as *.pcap* files on the laptop.

### B. Data Preparation

Data was converted from the *.pcap* file type recorded by the sensor to *.csv* using the Ouster SDK. The segmentation algorithm also works with the *.pcd* files used by 3DForest

Fig. 1. LiDAR Sensor setup at one of the Tower Hamlets Cemetery Park plots.

and most software dealing with point cloud data. After importing and visualizing the data, the entire scan can be processed or a smaller area within the scan can be selected before proceeding. The terrain is then removed from the dataset. Points with the smallest values in the Z-dimension are determined for each square meter ($1m^2$ in X,Y dimensions) of the plot and any points within a range of 5cm vertically (Z-dimension) are removed from the dataset. Finally, the new minimum value in the Z-dimension is determined and the dataset is split into the stems (up to 2m height) and crowns (above 2m height).

### C. Tree Segmentation

The algorithm requires some user inputs, most importantly, specifying a density at which tree stems are extracted and then identifying the number of tree stems to be clustered. Each point in the stem point cloud is assigned a density attribute based on the number of other points within a two-dimensional (X,Y), 25cm radius of that point using the SciPy KDTree class. A minimum density threshold, equivalent to the mean density for all points in the stem dataset, is suggested, but the user may need to adjust the threshold to determine the optimal density where only stems are visible in 2D and 3D visualizations of the data (other vertical objects may also be captured at the appropriate density, but these can be removed in a later review of the segmentation results). When the user is satisfied with the density threshold, points below that density are removed from the stems point cloud. The 2D (X,Y) plotted data should then appear as several groupings of points, representing the stems from a birds-eye-view, which the user must count to determine the number of clusters to form. The scikit-learn KMeans class is then used to create a cluster for each stem in the dataset based on the number of stems observed in the plots in the previous step. After forming clusters in the stems dataset the crowns point cloud is clustered to the existing clusters step-wise in horizontal slices, from bottom (directly above 2m height) to the top of the point cloud (maximum value in the Z-Dimension) in 5cm increments. Each point in a given slice is clustered with the existing cluster containing the nearest point according to Euclidean distance (scipy.spatial.distance.cdist.min()). If a point is not within 1m of any existing cluster, it is not clustered with any existing cluster and removed from the dataset. Any object with a height below 2m is also removed before proceeding with calculations of individual tree metrics.

### D. DBH, Height, Crown Width

After segmenting the trees, individual tree metrics can be obtained automatically and the segmented trees can be reviewed and any non-tree objects or trees not meeting minimum DBH or height requirements can be removed from the dataset. The estimation of DBH requires first finding the nearest point in the Z-dimension to the 1.3m height (standard breast height) mark, measured from the bottom of each tree. DBH was calculated as the largest Euclidean Distance (scipy.spatial.distance.cdist.max()) in the X,Y dimensions between points within a $\pm$1cm range of that point. Tree height was calculated as the difference between the maximum and minimum values in the Z-dimension for each tree. Crown width was calculated as the largest Euclidean Distance (scipy.spatial.distance.cdist.max()) in the X,Y dimensions for each tree.

### E. Accuracy Assessment

Liu et al. [11] evaluated the performance of their segmentation algorithm using an F-score (2), based on the number of stems in each plot vs. the number of stems segmented by their algorithm. In our assessment of accuracy, we use this metric to determine the accuracy of the segmentation algorithm when applied to our plots, with comparison to clustering algorithms (HDBScan, KMeans, and Gaussian Mixture Models) and pre-existing methods (3D Forest). We also calculate F-scores for individual segmented trees to determine the accuracy of individual tree segmentations compared to manually segmented point clouds, and by pre-existing methods (3DForest). Individual tree F-scores are calculated based on the number of points correctly attributed to an individual tree vs. the total number of points belonging to that tree.

*F-Score Formula*

$$F_1 = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{2}$$

Evaluation of the accuracy of the segmentation algorithm is determined using the F-Score $F_1$ (2), where $TP$ is the

number of true positives (either the number of stems correctly segmented, or the number of points correctly clustered with a particular tree), $FP$ is the number of false positives (either the number of non-tree objects segmented as trees, or the number of points incorrectly clustered with a particular tree), and $FN$ is the number of false negatives (either the number of trees in the plot that were not segmented, or the number of points that should have been clustered with a particular tree, but were not).

## III. Results

The results presented here, with the exception of the Accuracy Assessment section, primarily describe the segmentation of the test plot (13).

### A. Data Preparation

Figures 1 and 2 show the results of the terrain/ground point removal from the test plot. The curved lines in Figure 1 represent the terrain, the more dense clusters of points are tree canopies, and the straight lines include a fence at the edge of the park and objects (cars, houses) in the adjacent street.

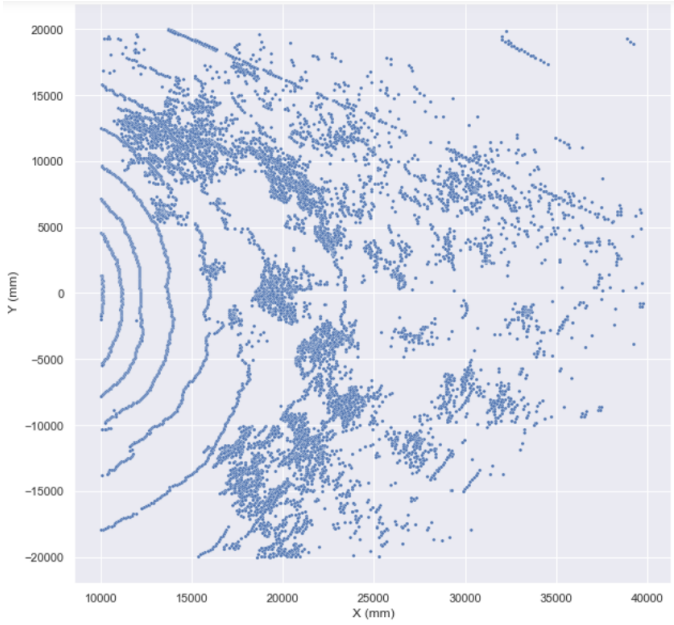


Fig. 2. Target point cloud from above (X and Y dimensions) with terrain points included (Test Plot 13).

### B. Tree Segmentation

Figure 3 shows a view from above of the segmented stems after the dataset is split into separate stems and crowns point clouds, densities are assigned to each point in the stems point cloud using the KDTree class, a minimum density threshold of 10 points per 25cm radius is applied to the points, and 19 clusters were formed using KMeans clustering, based on user feedback after counting 19 groupings of points in the plotted data.


Fig. 3. Target point cloud from above (X and Y dimensions) with ground points removed (Test Plot 13).

The step-wise clustering of points in 5cm vertical slices from the crown data set ($>$ 2m) resulted in 16 crowns being clustered with 16 of the 19 'stem' clusters. Two stumps and a section of fence formed the remaining three clusters and were automatically removed from the data due to being less than 2m in height. The resulting segmented trees included four immature trees with DBH $<$ 5cm and 12 mature trees of the 14 total mature trees on the plot.

### C. DBH, Height, and Crown Width

Table 1 shows the results of the DBH, Height, and Crown Width estimations from the data, including the non-tree clusters. DBH for tree 10 was measured manually as 39.1cm in comparison to the computed value of 43.3cm. Height measurements for four trees segmented from the 3D Forest test data, 21.04m, 20.17m, 18.9m, and 21.36m respectively, were compared to the actual heights of those trees, 20.73m, 19.69m, 18.55m, and 21.36m. All the heights were within 1m of the actual values and 1 tree was correct to 1mm.

### D. Accuracy Assessment

The accuracy of the segmentation algorithm varied between plots and between individual trees. Table 2 provides a summary of the F-scores obtained for the 7 plots segmented using the algorithm. The target number of stems to be segmented for each plot was selected as the number of stems visible from the perspective of the scanner and falling within the plot area. The test plot, plot 13, obtained an F-score of 0.875, as two of the 14 mature tree stems were not segmented, and the algorithm

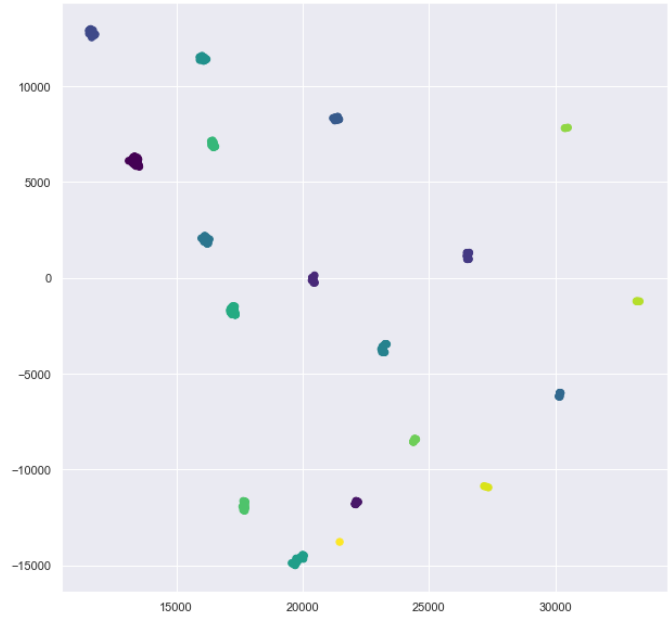


Fig. 4. Clustering results. Total of 18 stems including 2 stumps and 4 small trees at the front. There is also a small cluster from part of the fence behind the trees

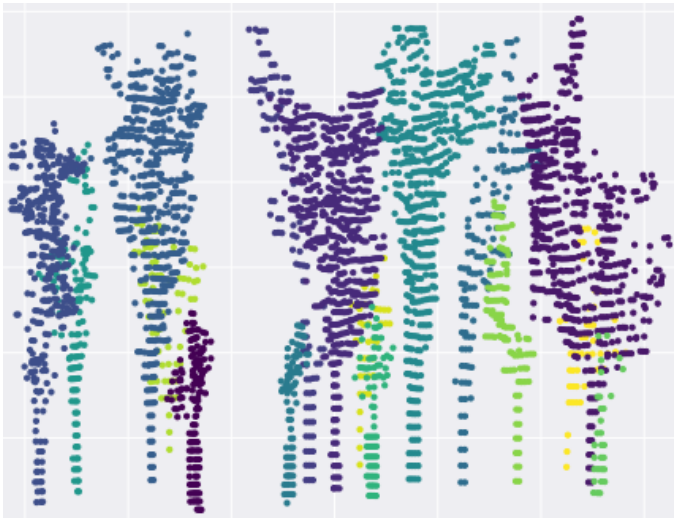

Fig. 5. Segmentation results. Total of 16 stems including 4 immature trees at the front.

also segmented four immature trees (< 5cm DBH). These immature trees were surrounded by fencing and posts, making them appear larger than they actually were. The remaining plots obtained F-scores between 0.421 and 0.941. The lowest score occurred for plot 8, which was taken along a path with many tombstones (in the Tower Hamlets Cemetery Park), and the majority of trees in this plot were mistakenly clustered with the tombstones rather than with their respective stems or both the stems and tombstones were clustered together. The highest score occurred for plot 14, which was a smaller plot within the area segmented in plot 13, but scanned from a different angle. In plot 14, the tree furthest from the scanner was not

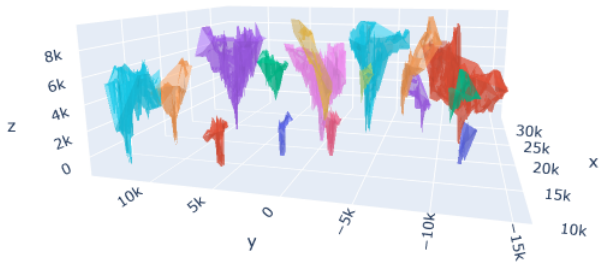


Fig. 6. 3D Representation of segmentation results.

TABLE I
DBH, HEIGHT, AND CROWN WIDTH

| Tree | *DBH (cm)* | *Height (m)* | *Crown Width (m)* |
|---|---|---|---|
| 0 | 45.4 | 4.19 | 2.08 |
| 1 | 15.4 | 10.25 | 9.21 |
| 2 | 12.4 | 8.74 | 5.24 |
| 3 | 32.6 | 10.14 | 4.04 |
| 4 | 40.5 | 8.03 | 5.79 |
| 5 | 14.1 | 9.88 | 7.88 |
| 6 | 18.9 | 10.23 | 6.13 |
| 7 | 10.0 | 3.85 | 14.02 |
| 8 | 43.2 | 10.0 | 6.05 |
| 9 | 12.3 | 7.51 | 5.09 |
| 10 | 43.3 | 4.13 | 1.70 |
| 11 | 16.9 | 3.43 | 1.37 |
| 12 | 16.6 | 6.27 | 4.00 |
| 13 | 79.4 | 5.83 | 4.83 |
| 14 | 0.0 | 4.83 | 2.22 |
| 15 | 54.8 | 5.41 | 3.17 |

segmented, although the immature trees were also not falsely recognized as mature trees. KMeans, GMM, and HDBSCAN clustering was performed on the dataset after terrain removal on 2D (X,Y) data and 3D (X,Y,Z) data. In some cases, when applied to the test data, each of these algorithms clustered multiple stems together or clustered stems to non-tree objects. For this reason, results are presented as the number of stems correctly segmented, without being clustered with other trees or objects. KMeans in 2D correctly segmented 8 stems, and in 3D correctly segmented 5 stems. GMM in 2D correctly segmented 8 stems, and in 3D correctly segmented 6 stems. HDBSCAN in 2D correctly segmented 4 stems, and in 3D correclty segmented 7 stems. Although HDBSCAN segmented fewer stems than the other algorithms, it also produced fewer false positives and produced clearer segmentations of the individual trees. Application of the algorithm to the 3D Forest test data correctly segmented 25 stems in the point cloud correctly, with one false positive, obtaining an F-score of 0.962, very similar to the average results for Liu et al. [11]. Twenty four of the 26 trees in the 3D Forest test dataset were correctly segmented with 1 false positive for an F-score of 0.940. Segmentation of the seven scans in our dataset using 3D Forest resulted in 6 true positives, 1 false positive, and 94 false negatives for an F-score of 0.112.

Table 3 provides a summary of the F-scores obtained for

TABLE II
PLOT LEVEL F-SCORES BY SCAN

| Plot | *TP* | *FP* | *FN* | $F_1$ *Score* |
|---|---|---|---|---|
| 13 (test plot) | 12 | 4 | 2 | 0.875 |
| 5 | 3 | 1 | 2 | 0.667 |
| 7 | 8 | 2 | 0 | 0.889 |
| 8 (tombstones) | 4 | 6 | 5 | 0.421 |
| 9 | 22 | 2 | 2 | 0.917 |
| 10 | 6 | 1 | 0 | 0.923 |
| 14 | 8 | 0 | 1 | 0.941 |
| 3D Forest | 25 | 1 | 1 | 0.962 |

TP = True Positive (correctly segmented point)
FP = False Positive (incorrectly segmented point)
FN = False Negative (non-segmented point)

TABLE III
TREE LEVEL F-SCORES BY INDIVIDUAL TREE POINT CLOUD

| Tree | *TP* | *FP* | *FN* | $F_1$ *Score* | *3DF* $F_1$ |
|---|---|---|---|---|---|
| ms 1 | 586 | 14 | 77 | 0.928 | 0 |
| ms 2 | 504 | 6 | 24 | 0.971 | 0 |
| 3F 14 | 4399 | 422 | 5030 | 0.617 | 0.630 |
| 3F 17 | 7268 | 1126 | 2428 | 0.804 | 0.671 |
| 3F 19 | 6539 | 3672 | 7295 | 0.544 | 0 |
| 3F 20 | 4233 | 1018 | 2114 | 0.730 | 0.484 |

TP = True Positive (correctly segmented point)
FP = False Positive (incorrectly segmented point)
FN = False Negative (non-segmented point)
3DF = 3D Forest

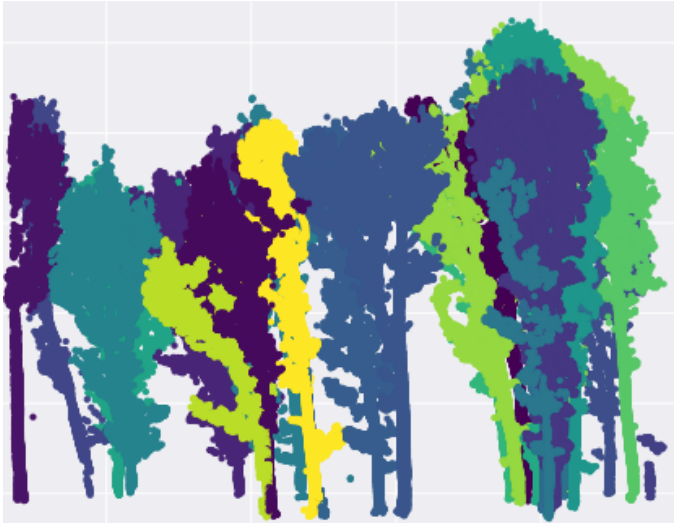

Fig. 7. Segmentation results for 3D Forest test dataset.

six individual trees selected from the segmentation results for two separate plots. The first two rows show comparisons between the point clouds obtained by manually segmenting two trees in the test plot and those obtained for the same trees using the segmentation algorithm. The remaining rows show comparisons between the point clouds for four tree point clouds included as references in the 3D Forest test dataset, and those obtained for the same trees using the segmentation algorithm. The trees selected for the test plot were selected because they could be manually segmented accurately with reference to plots of the dataset, photos of the site, and google earth imagery, which may have also made them easier for the algorithm to segment. In light of this, the F-scores obtained for these trees were high, at 0.928 and 0.971. The accuracy for trees resulting from segmentation of the 3D Forest dataset was lower and more variable when compared to the 3D Forest individual tree point clouds, at 0.544, 0.617, 0.730, and 0.804. The algorithm generally outperformed 3D forest in segmenting individual trees in both our dataset (F-scores of 0 and 0 for the same two trees) and the 3D Forest test dataset (F-scores of 0, 0.630, 0.484, and 0.671 for the same four trees).

## IV. CONCLUSIONS

Segmenting tree stems by filtering out points below a minimum density threshold, followed by KMeans clustering succeeded in segmenting the majority of stems across the eight plots tested. The success of stem segmentations varied between plots (F-scores between 0.421 and 0.962), with 86 of 101 target trees being correctly segmented and 17 non-tree objects or immature trees being mistakenly segmented. Generally, the fewer non-tree objects and less understory foliage, the better the algorithm performed. The quality of individual tree segmentations also varied among the six trees tested with F-scores between 0.544 and 0.971 when comparing the automatic segmentation of trees to manual segmentation of the point cloud. For all six trees, the algorithm was more likely to not cluster points that belonged to a tree (false negatives) than to cluster points that did not belong to a tree (false positives). The performance of the algorithm on our dataset greatly surpassed the performance of 3D Forest, which correctly segmented stems for 6 of the 101 trees, with 1 false positive, and correctly segmented 24 of 26 stems in the 3D Forest test dataset. Five of the 6 individual tree segmentations produced by our aglorithm were also more accurate than the those produced by 3D Forest.

The purpose of this segmentation algorithm was to function from single LiDAR scans, rather than requiring multiple scans and pre-processing to merge the scans into a dense point cloud. The drawback of this method is that only trees in full view of the sensor for that single scan will be segmented to produce point clouds of sufficient quality to obtain accurate measurements of individual tree metrics such as DBH, height, and crown width. Another issue is that the algorithm can mistake other vertically oriented objects (fences and tombstones in our case) as tree stems and cluster tree crowns with those objects rather than their respective stems. However, the algorithm also performed well on a larger, denser point cloud, suggesting that this method could function with a greater range of data quality and support projects requiring tree segmentation with variable point cloud density and user expertise in working with LiDAR data.

## V. Future Work

Several improvements can still be made to the segmentation algorithm and individual tree metrics calculations. Some initial experimentation with the radius used to calculate point densities in the test data produced good results at 25cm. Adjusting this radius could potentially prevent segmentation of non-tree objects such as fences, or better segment different sized stems. DBSCAN could be an optional substitute for KMeans clustering of stems, when larger stems or those nearer to the sensor form multiple clusters when using KMeans. Given the large numbers of false negatives for points in individual trees in the 3D Forest dataset segmented by the algorithm, exploring different clustering methods for segmenting the canopy should be explored, possibly adding an optional top down pass after the initial pass. Fitting cylinders to tree stems and branches with a model like Matlab's pcfitcylinder could allow better automatic DBH calculations and estimation of tree volume. A method to automatically differentiate between foliage and wood, potentially based on reflectivity and/or infrared values captured by the sensor. If a substantial enough dataset was captured it would be interesting to attempt predictions of tree features such as species or woody and foliage biomass based on the segmented point clouds.

## References


[1] Pan, Y., Birdsey, R.A., Fang, J., Houghton, R., Kauppi, P.E., Kurz, W.A., Phillips, O.L., Shvidenko, A., Lewis, S.L., Canadell, J.G., Ciais, P., Jackson, R.B., Pacala, S.W., McGuire, A.D., Piao, S., Rautiainen, A., Sitch, S., Hayes, D. 2011. A large and persistent carbon sink in the world's forests. Science. 333: 988-993.

[2] Nowak, D.J., Hirabayashi, S., Bodine, A., Greenfield, E. 2014. Tree and forest effects on air quality and human health in the United States, Environmental Pollution,Volume 193, Pages 119-129, ISSN 0269-7491.

[3] Duffy, C., O'Donoghue, C., Ryan, M., Kilcline, K., Upton, V., Spillane, C. 2020. The impact of forestry as a land use on water quality outcomes: An integrated analysis, Forest Policy and Economics, Volume 116, 102185, ISSN 1389-9341.

[4] Brockerhoff, E.G., Barbaro, L., Castagneyrol, B., Forrester, D.I., Gardiner, B., Gonzalez-Oblabarria, J.R., Lyver, P.O'B., Meurisse, N., Oxbrough, A., Hisatomo, T., Thompson, I.D., van det Plas, F., Jactel, H. 2017. Forest biodiversity, ecosystem functioning and the provision of ecosystem services. Biodivers Conserv 26, 3005–3035.

[5] Henning, J.G., Radtke, P.J. 2006. Detailed stem measurements of standing trees from ground-based scanning LiDAR. For Sci 52:67–80

[6] Maas, H.G., Bienert, A., Scheller, S., Keane, E. 2008. Automatic forest inventory parameter determination from terrestrial laser scanner data. Int J Remote Sens 29:1366–5901

[7] Unger, D.R., Hung, I.K., Brooks, R., Williams, H. 2014. Estimating number of trees, tree height and crown width using LiDAR data. Gisci Remote Sens 51:227–238

[8] Metz, J., Seidel, D., Schall, P., Scheffer, D., Schulze, E.-D., Ammer, C. 2013. Crown modeling by terrestrial laser scanning as an approach to assess the effect of aboveground intra- and interspecific competition on tree growth. For. Ecol. Manage. 310, 275–288.

[9] Harikumar, A., Bovolo, F., Bruzzone, L. 2017. An internal crown geometric model for conifer species classifcation with high-density LiDAR data. IEEE Trans Geoence Remote Sens 55:2924–2940.

[10] Trochta, J., Krůček, M., Vrs˘ka, T., Kra´l, K. 2017. 3D Forest: An application for descriptions of three-dimensional forest structures using terrestrial LiDAR. PLoS ONE 12(5): e0176871.

[11] Liu, Q., Ma, W., Zhang, J., Liu, Y., Xu, D., Wang, J. 2021. Point-cloud segmentation of individual trees in complex natural forest scenes based on a trunk-growth method. J. For. Res. 32, 2403–2414.

[12] Zhu, R., Guo, Z., Zhang, X. 2021. Forest 3D Reconstruction and Individual Tree Parameter Extraction Combining Close-Range Photo Enhancement and Feature Matching. Remote Sensing. 13(9):1633.

[13] Calders, K., Newnham, G., Burt, A., Murphy, S., Raumonen, P., Herold, M., Culvenor, D., Avitabile, V., Disney, M., Armston, J. and Kaasalainen, M. 2015. Nondestructive estimates of above-ground biomass using terrestrial laser scanning. Methods Ecol Evol, 6: 198-208.

[14] Hackenberg, J., Spiecker, H., Calders, K., Disney, M., Raumonen, P. 2015. SimpleTree-An Efficient Open Source Tool to Build Tree Models from TLS Clouds. Forests. 6(11):4245–94.

[15] Piboule, A., Krebs, M., Esclatine, L., Hervé, J. 2015. Computree: a collaborative platform for use of terrestrial LiDAR in dendrometry.

[16] True Reality Geospatial Solutions, 2015. LiForest -LiDAR software for forestry applications. 2 ed. http://www.liforest.com/2015.

[17] TreeMetrics, 2015. AutoStem Forest. http://www.treemetrics.com/2015.

[18] Burt, A., Disney, M., Calders, K., 2019. Extracting individual trees from lidar point clouds using treeseg. Methods Ecol Evol.; 10: 438– 445.

[19] Song, W., Zou, S., Tian, Y., Fong, S., Cho, K. 2018. Classifying 3D objects in LiDAR point clouds with a back-propagation neural network. Hum. Cent. Comput. Inf. Sci. 8, 29.

[20] Liang, X.L., Litkey, P., Hyyppä, J., Kaartinen, H., Vastaranta, M., Holopainen, M. 2012. Automatic stem mapping using single-scan terrestrial laser scanning. IEEE Trans Geosci Remote Sens.

[21] Olofsson, K., Holmgren, J. 2016. Single Tree Stem Profile Detection Using Terrestrial Laser Scanner Data, Flatness Saliency Features and Curvature Properties. Forests, 7, 207.

[22] Zhang, W., Wan, P., Wang, T., Cai, S., Chen, Y., Jin, X., Yan, G. 2019. A Novel Approach for the Detection of Standing Tree Stems from Plot-Level Terrestrial Laser Scanning Data. Remote Sensing 11, no. 2: 211.