Spatial Clustering in Python: Processing LiDAR points into objects

Shane Grigsby
University of California, Santa Barbara
shane@geog.ucsb.edu
http://geog.ucsb.edu/~shane

Science Motivation

Estimation of vegetation biophysical parameters is needed for ecosystem models, and for retrieval of biochemical parameters using remote sensing and spectroscopy. Retrieving biophysical parameters like Leaf Area Index (LAI) is time consuming, destructive, and often inaccurate if not directly sampling leaves. Other biophysical parameters, such as Leaf Angle Distribution (LAD), are often impossible to field sample in practice. One solution that is emerging is the use of field based Light Detection And Ranging (LiDAR) to retrieve structural information, using the point data to fit biophysical parameters.

Figure 1 – *Dense vegetation where determination of biophysical parameters is difficult. Measuring leaf angles without disturbing the leaves is almost impossible with manual methods.*

Figure 2 – *Riegl Z420i terrestrial laser scanner, with co-mounted RGB camera. One standard scan produces 3 million points; scans are generally repeated and linked to achieve multiple look angles.*

Figure 3 – *Data Acquisition in the field. Scanner is mounted un top of a car above dense agricultural vegetation in the San Joaquin Valley.*

1.) Find the Objects: Acquire LiDAR → Filter Data → Segment Points to Objects

2.) Estimate Parameter Distribution: Fit parameter to Objects → Build Distribution

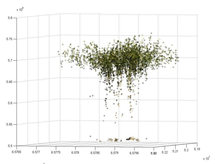


Figure 4 – *Filtered LiDAR points for a portion of a grape vine. Filter was based on HSV vales and height to pre-filter data to likely leaves.*

Figure 5 – *Unfiltered LiDAR points, combined with RGB values from mounted camera.*

Segmentation of Points to Objects: K-means

K-means Clustering: K-means clustering has three problems, all of which make it unsuitable. First, K-means assumes that we know the number of objects that we are looking for; if we are trying to segment leaf-level objects, this is false. Second, breaks in the data are not respected when constructing clusters. Third, all points are assigned to some cluster, which in turn requires that the preprocessing filter step is robust.

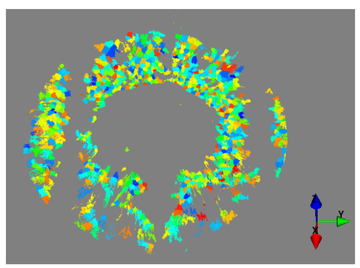


Figure 6 – *K-means clustering for 1000 clusters, 5.5 x 10^5 points.*

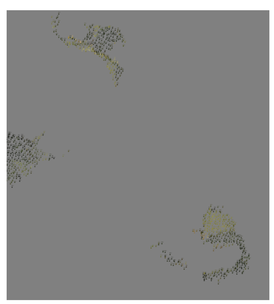
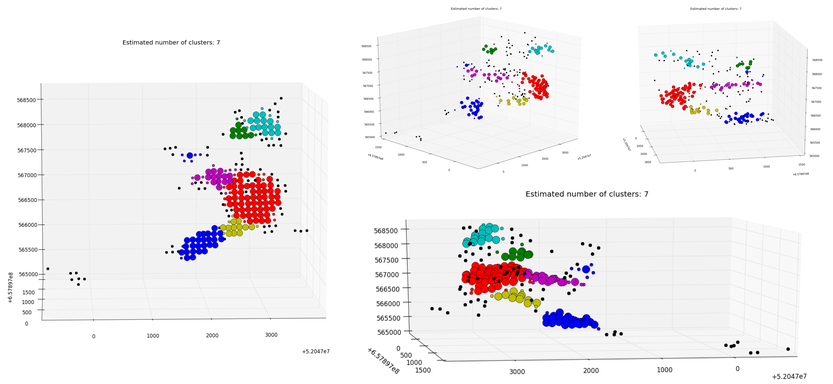


Figure 7 – *Zoomed view of three clusters produced from k-means, displayed with RGB values. Note the odd shapes, and clear presence of sub-clusters*

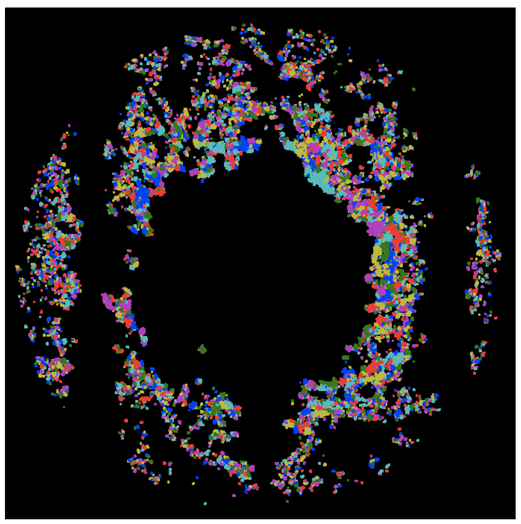
Segmentation of Points to Objects: Density Clustering

Density Based Hierarchal Clustering (DBSCAN and OPTICS): The DBSCAN and OPTICS algorithms discard noise, work with numerous geometries, and select either child clusters or parent clusters when clustering parameters are not optimal for desired object size. Both algorithms designate core points, cluster points, and noise points. The algorithms utilize a minimum number of points (MinPts) within a distance epsilon for core point membership, and determine cluster membership by adjacency to core points. Below shows several views of the same subset of points, using different look angles; black is noise.



Scaling to Large Datasets

The scikit-learn library has an implementation of DBSCAN that uses a distance matrix to compute the clustering structure. This in turn requires a N-by-N floating point matrix to execute. For 55,000 points, 11.27 GB of memory is needed; this scales to 1.126 TB for the 550,000 points in the data set to left and below.



To scale to arbitrary large datasets, distances calculations to neighbors within a reasonable distance need only occur for the point being processed. While calculating all distances to all points simultaneously in a distance matrix is memory inefficient, looping through the whole dataset per point is time inefficient. One approach, implemented here, is to use a tree structure for fast neighbor distance lookups. This allows focal rather than global distance lookups, and scales to arbitrarily large data sets.

Figure 8 – *Density clustering for 5.5 x 10^5 points, using 15 cm epsilon and 5 minimum point object threshold; 7022 clusters where identified.*

Several large clusters are present at the right and front of the scan, indicating where the truck holding the scanner was parked and pushed leaves together

References

- Ankerst, Mihael, et al. "OPTICS: ordering points to identify the clustering structure." ACM SIGMOD Record 28.2 (1999): 49-60.
- Ester, Martin, et al. "A density-based algorithm for discovering clusters in large spatial databases with noise." Proceedings of the 2nd International Conference on Knowledge Discovery and Data mining. Vol. 1996. AAAI Press, 1996.

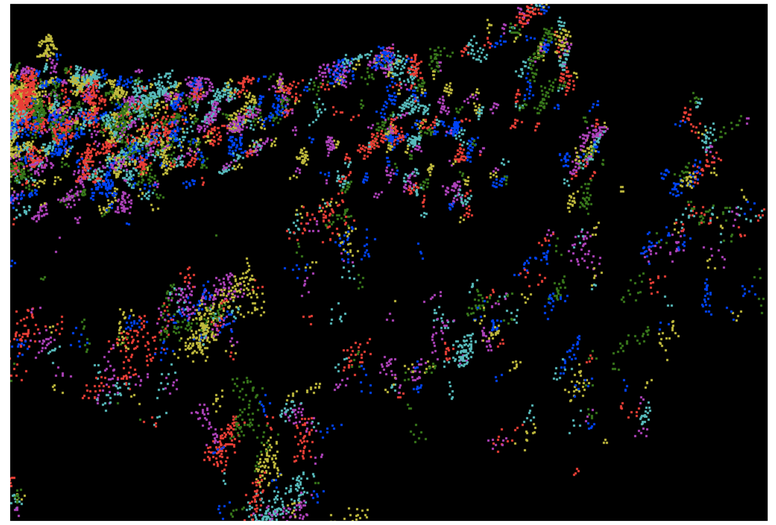


Figure 9 – *Density clustering for 5.5 x 10^5 points, using 15 cm epsilon and 5 minimum point object threshold; 7022 clusters where identified. (Magnification of a portion of figure 8)*

Most clusters appear to approximate a plane, as is expected for leaves. The distribution of horizontal to vertical leaves is theorized to be a spherical distribution for this canopy.

Future Applications

The density based segmentation explored here works at multiple scales, generalizing up from leaf segmentation to tree crown identification. Further use on new heterogeneous datasets will provide opportunities to explore the hierarchical nature of the clusters. In the future, it may be possible to identify trees, branches, and leaves within a forest from the same remote sensing data set.

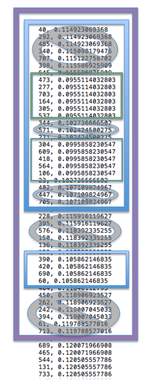


Figure 10 – *Implicit clustering structure found from OPTICS. Purple box represents the largest cluster at the greatest epsilon, grey indicates noise relative to adjacent boxes. Green boxes are siblings, as are blue boxes.*

Clusters are extracted by looking for break points within a list of distances that decrease monotonically within a terminal cluster.

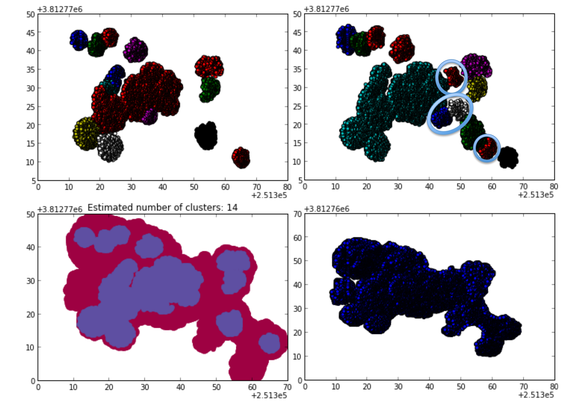


Figure 11 – *OPTICS density-based clustering applied to aerial LiDAR data of tree crowns in Alameda Park, Santa Barbara.*

- ***Top Left*** *indicates segmented clusters at 2.5 meter epsilon.*
- ***Bottom Left*** *shows clusters overlaid on points designated noise from the input data set (epsilon 2.5 meters).*
- ***Bottom Right*** *displays the original un-segmented data set (70,000 points)*
- ***Top Right*** *shows the impact of increasing epsilon, in this case to 3 meters. The yellow, white, blue, and red clusters have merged into a larger cluster (cyan), but we have also gained several new canopy clusters (circled in blue) by relaxing the density requirement imposed by the lower epsilon.*

Acknowledgements

- The entire staff at NSERC and NASA SARP for making much of this data possible
- Bodo Bookhagen for the use of his LiDAR scanner; Mike Alonso for the Alameda data
- Austin Hopkins for his help and collaboration