

Leaf Counting and Area Estimation using 3D Point Cloud Data

TAU02E - Tomi Maijala (LUT), Vinh Van (TAU)

Problem

- ▶ Goal: Estimate total leaf area and number of leaves from lidar point cloud.
 - ▶ Input: 3D point cloud measured from lidar at position $[0, 0, 1.5]$.
 - ▶ Lidar records only the closest object along each beam.
 - ▶ Points may correspond to leaves, branches, trunk, or ground.
 - ▶ Assume no obstacles between lidar and measured points.
 - ▶ Task: Use modelling to compute leaf area and estimate leaf count.
 - ▶ Also assess accuracy and reliability of chosen methods.
 - ▶ Data simulated using HELIOS++ (Winiwarter et al., 2022).

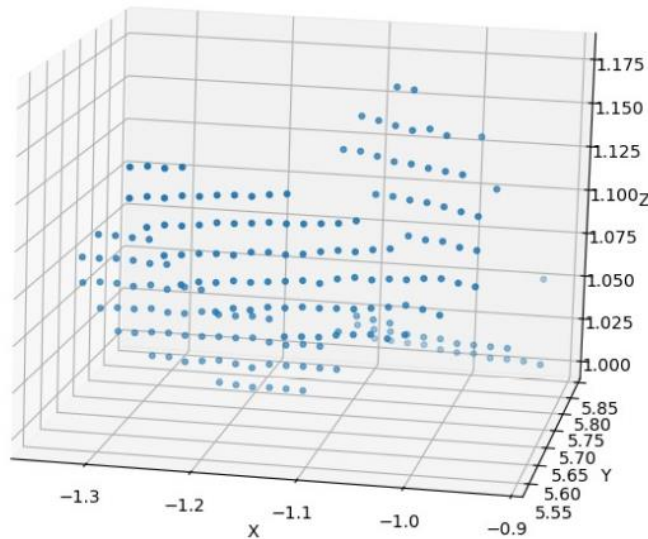


Agenda

- Problem description
- Assumptions made
- Model
 - Leaf identification
 - Objective function
- Model Optimization
 - DBSCAN
 - RANSAC
- Evaluation
 - Visual inspection
 - Simulated data
- Final estimation
- Conclusion

Assumptions

1. **Point Density:** The point cloud density is adequate to capture the essential structure of the leaves. Its resolution is finer than the most leaves' size.
2. **Leaf Shape:** Leaves are mostly planar and can be approximated as flat surfaces.
3. **Environmental Conditions:** The data was obtained in still air conditions, minimizing motion blur or distortion in the point cloud.



Model - Leaf Identification

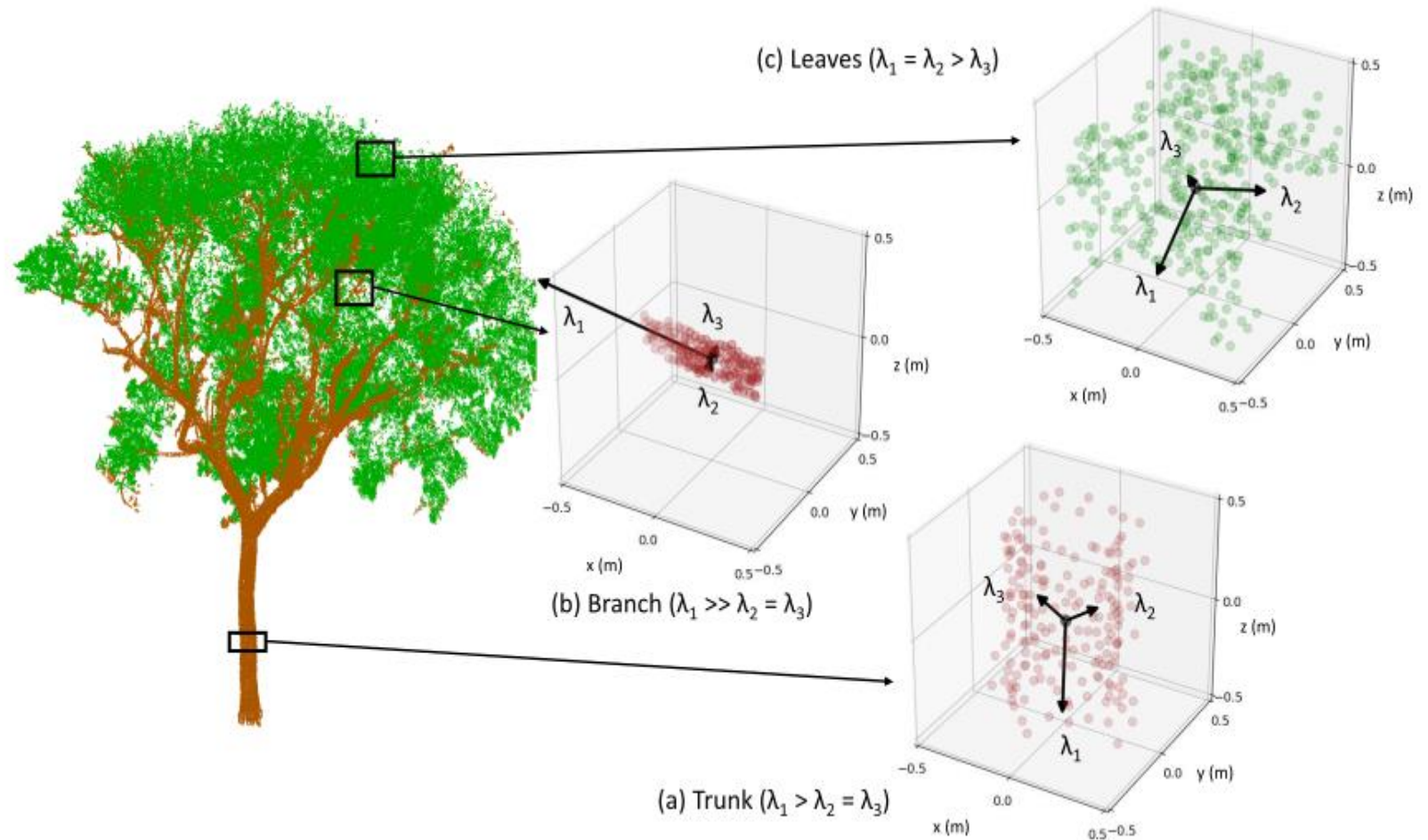
Planarity: Leaf points approximately form a flat surface.

Separability & Density: clusters separated by sparse regions.

Leaf Size: Cluster area and dimensions within leaf-size limits.

function `is_leaf_cluster(points)`

1. planarity via `PCA(points)`
2. all *points* lies in a plane
3. *#points* within some range
4. return TRUE if satisfy all



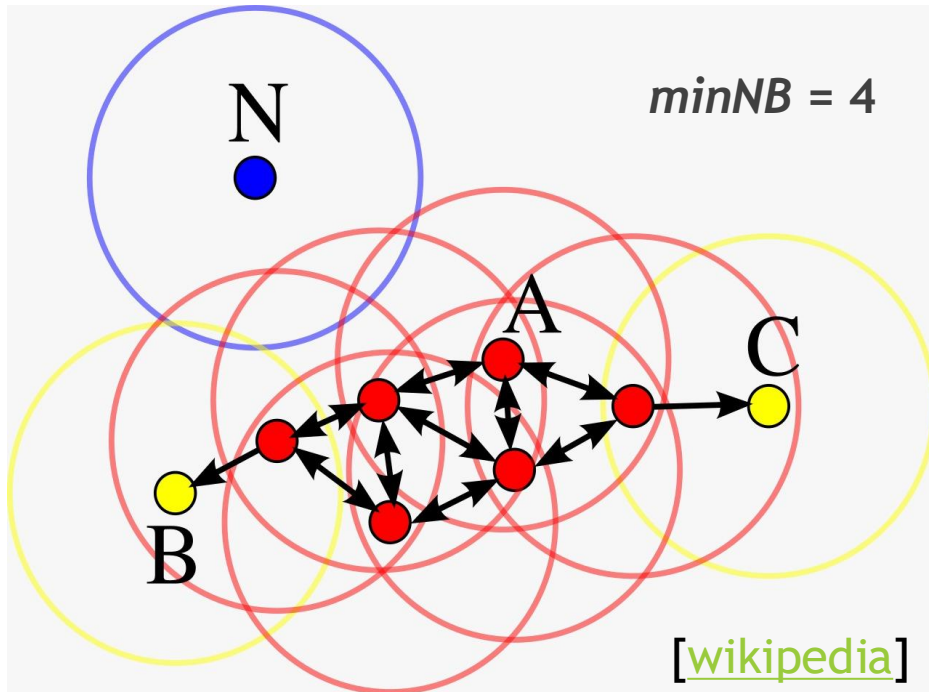
[[Moorthy et al, 2020](#)]

Model - Objective function

- ▶ **Goal:** Evaluate how well a clustering separates leaf-like point clusters while avoiding over-segmentation.
- ▶ Scoring rule for the leaf-like clusters (**score_function**):
 - ▶ Score = sum of (number of points in each leaf-like cluster)²
 - ▶ Encourages large, coherent leaf clusters
 - ▶ Penalizes splitting a single leaf into many small clusters
 - ▶ Integrates leaf-likeness (semantic + instance segmentation)
- ▶ Objective function optimization was done by simple exhaustive search for different hyperparameters and clustering methods.

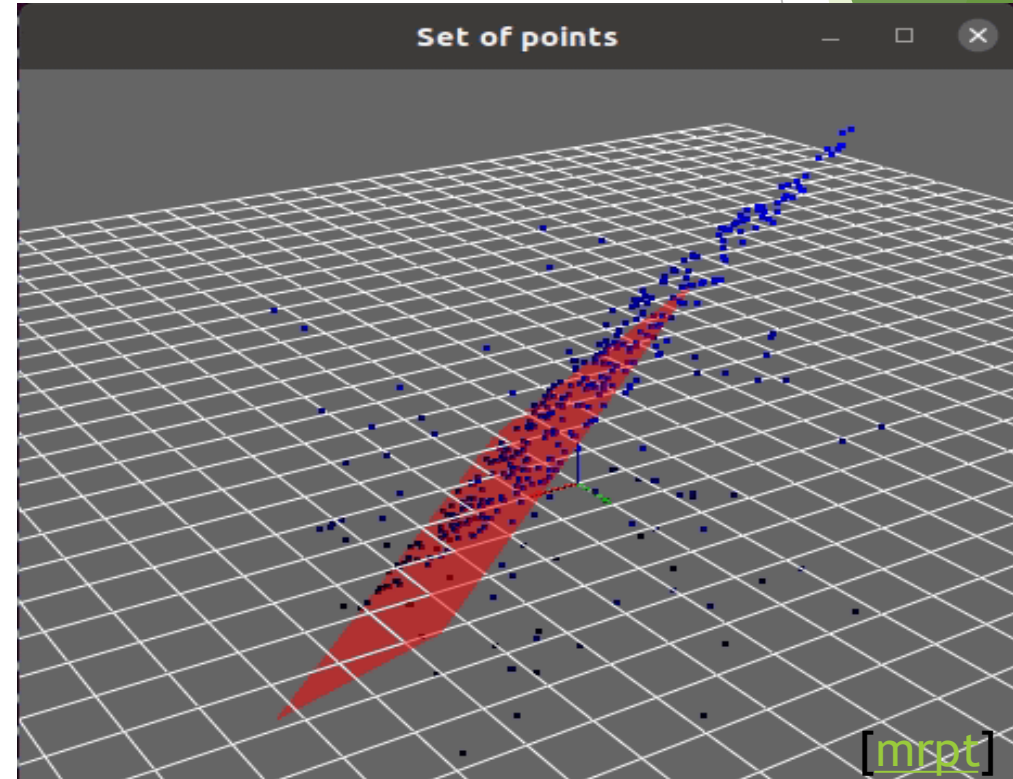
Model - Tools

DBSCAN: Density based clustering



red (A): core points, $\#NB \geq 4$ (included)
yellow (B,C): neighbour of a core (included)
blue (N): no core neighbour, $\#NB < 4$ (excluded)

► **RANSAC:** Outlier-proof linear regression

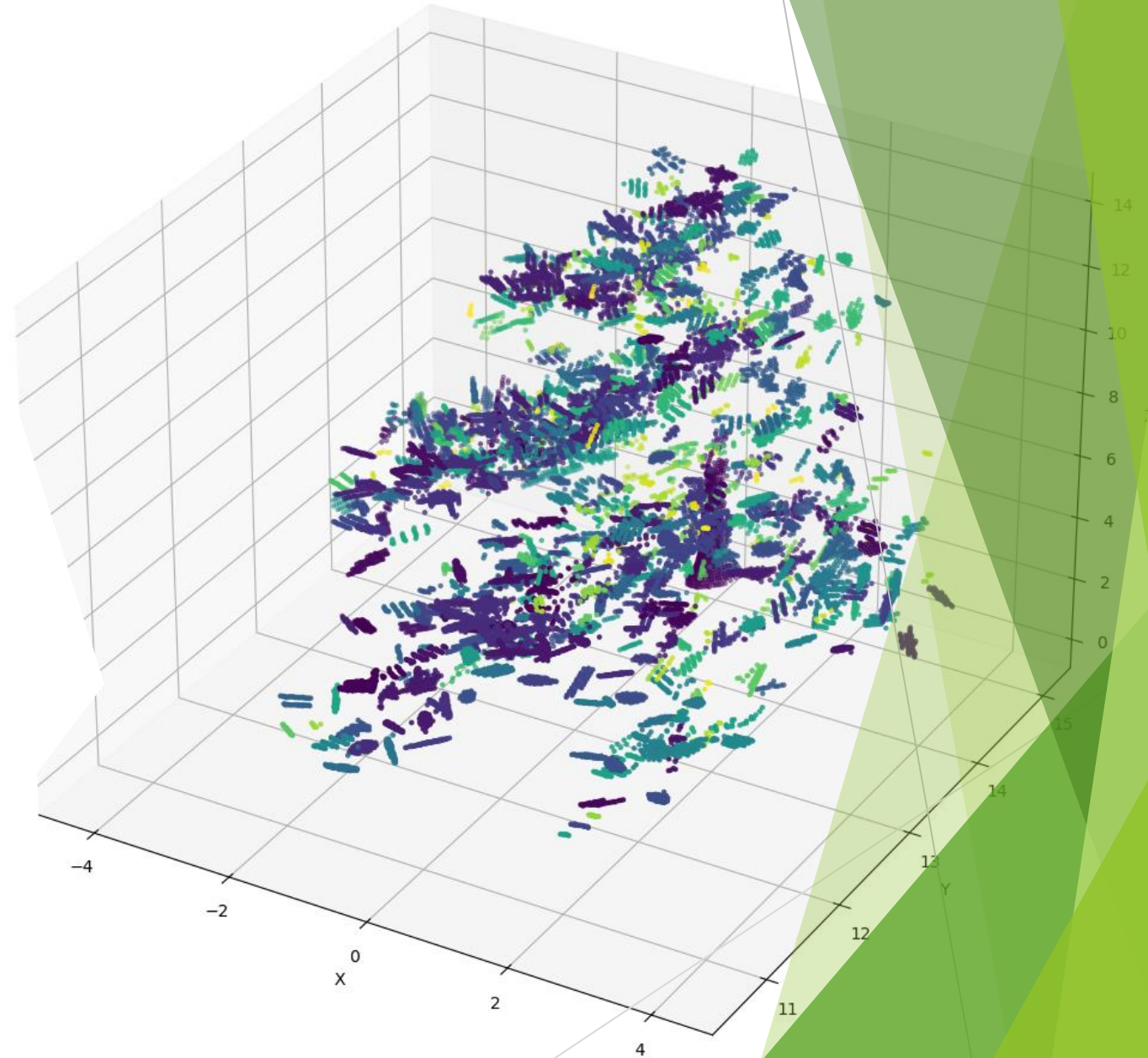


maximize number of points
within error tolerant; stochastic

Model - Optimization

Initial Clustering

- Cut the whole cloud points into smaller initial clusters.
- These clusters are large enough such that no leaf got fragmented by this step.
- Make it easier and faster for later optimization.
- Use **DBSCAN** with large ϵ and small $minNB$, as it respect the density assumption

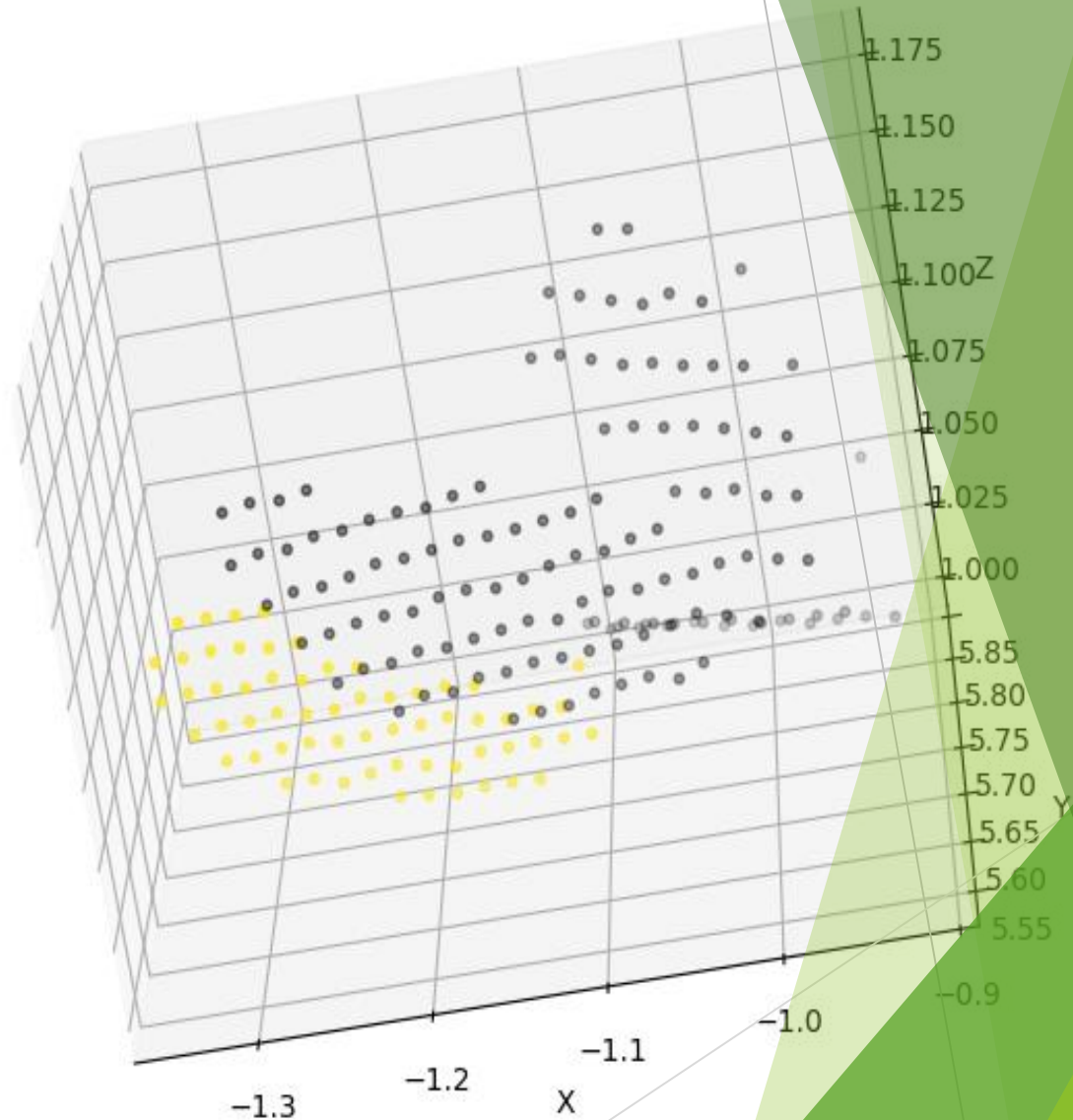


Model - Optimization

DBSCAN

function **DBSCAN_optim**(points):

1. chose a pair (ϵ , *minNB*)
2. *clusters* = **DBSCAN**(points, ϵ , *minNB*)
3. for each cluster, remove noise using **RANSAC**
4. *score* = **score_function**(*clusters*)
5. return best *clusters* in all pair (ϵ , *minNB*)



Model - Optimization

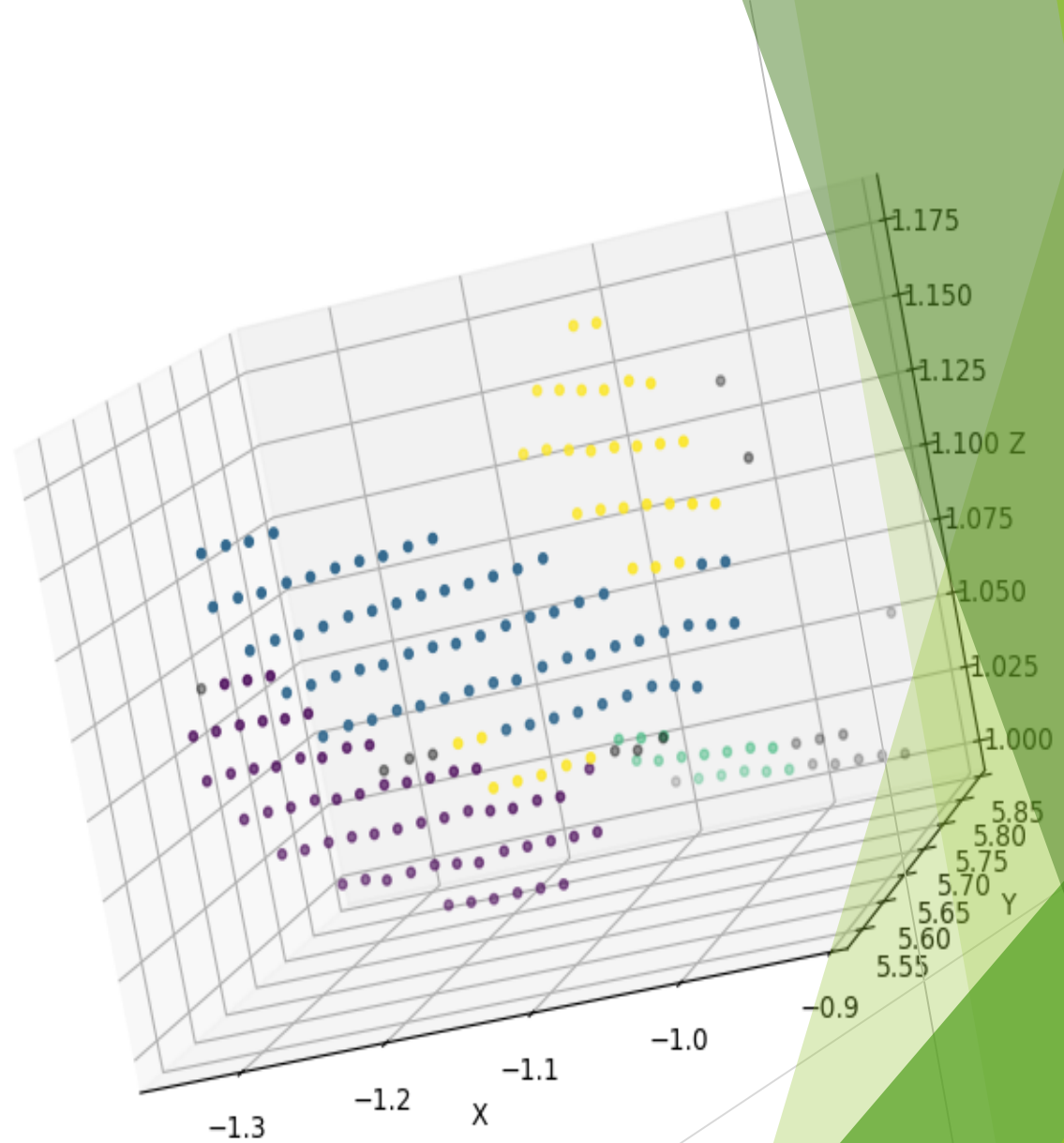
Sequential RANSAC

function **seq_RANSAC_optim**(points):

1. use **RANSAC** to detect good planar
2. planes may cut through objects → apply **DBSCAN_optim** within the plane to separate different objects
3. collect/remove leaflike clusters found by **DBSCAN_optim**
4. repeat leaflike cluster extraction until too few points remain or no new inliers are found.

Strength: separates leaves in dense regions where DBSCAN fail to split.

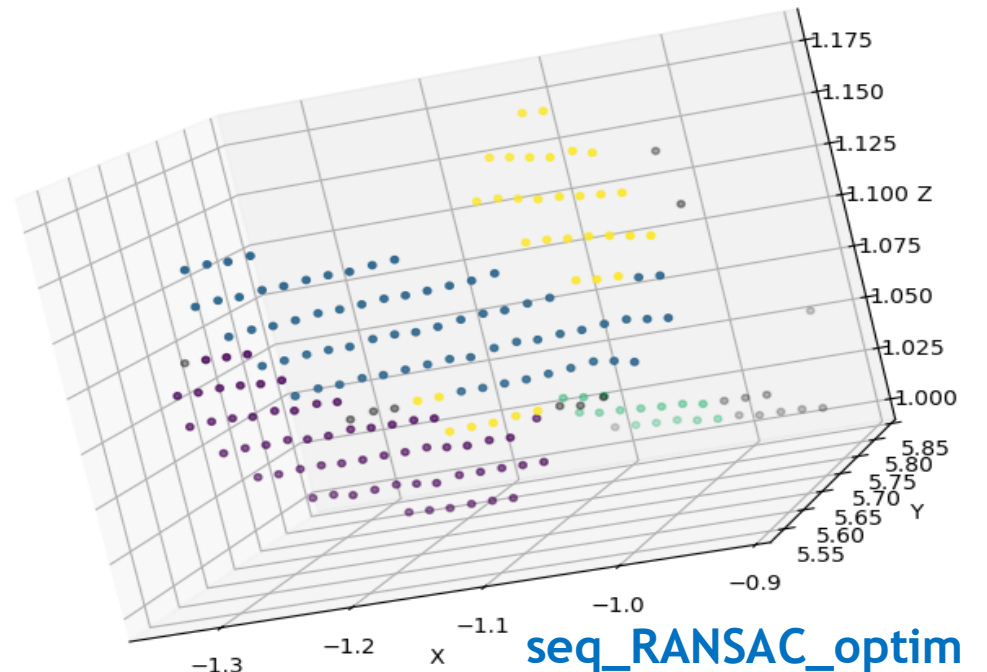
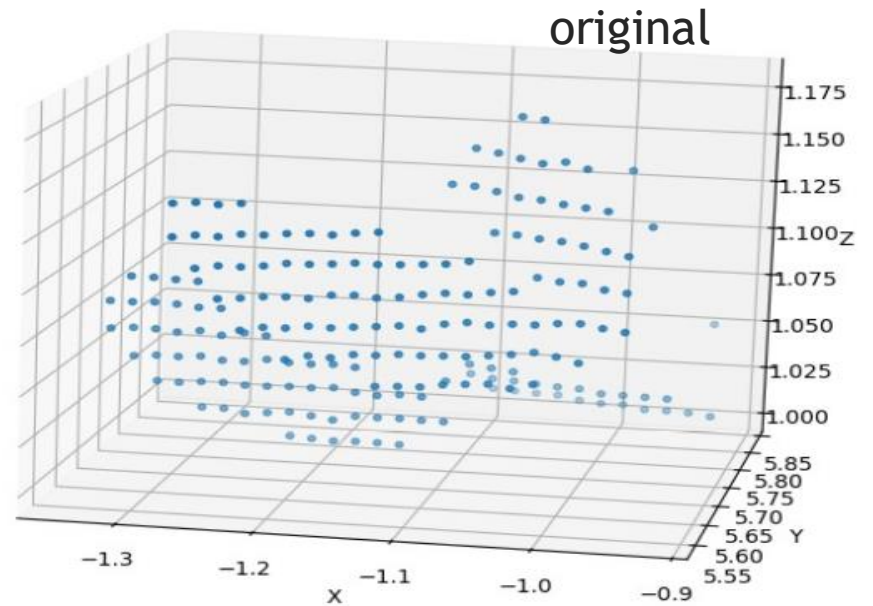
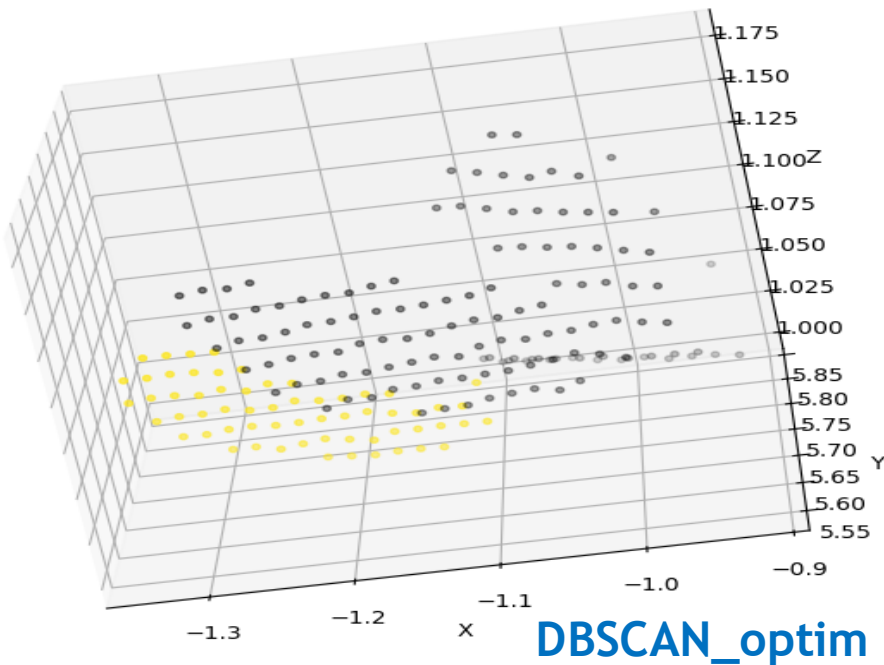
Weakness: stochastic → multiple runs



Model - Optimization Final

function **clustering_optim**(points)

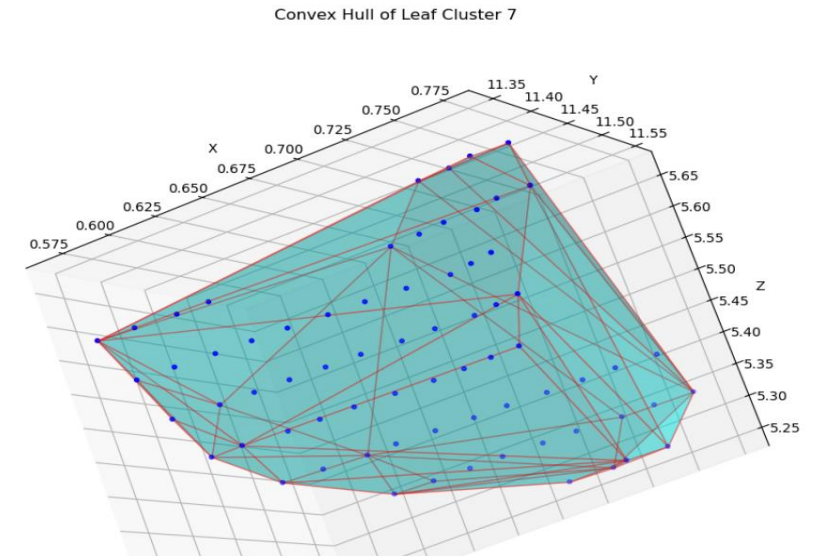
1. *score1, clus1* = **DBSCAN_optim**(points)
2. *score2, clus2* = **seq_RANSAC_optim**(points)
3. return the highest score clustering



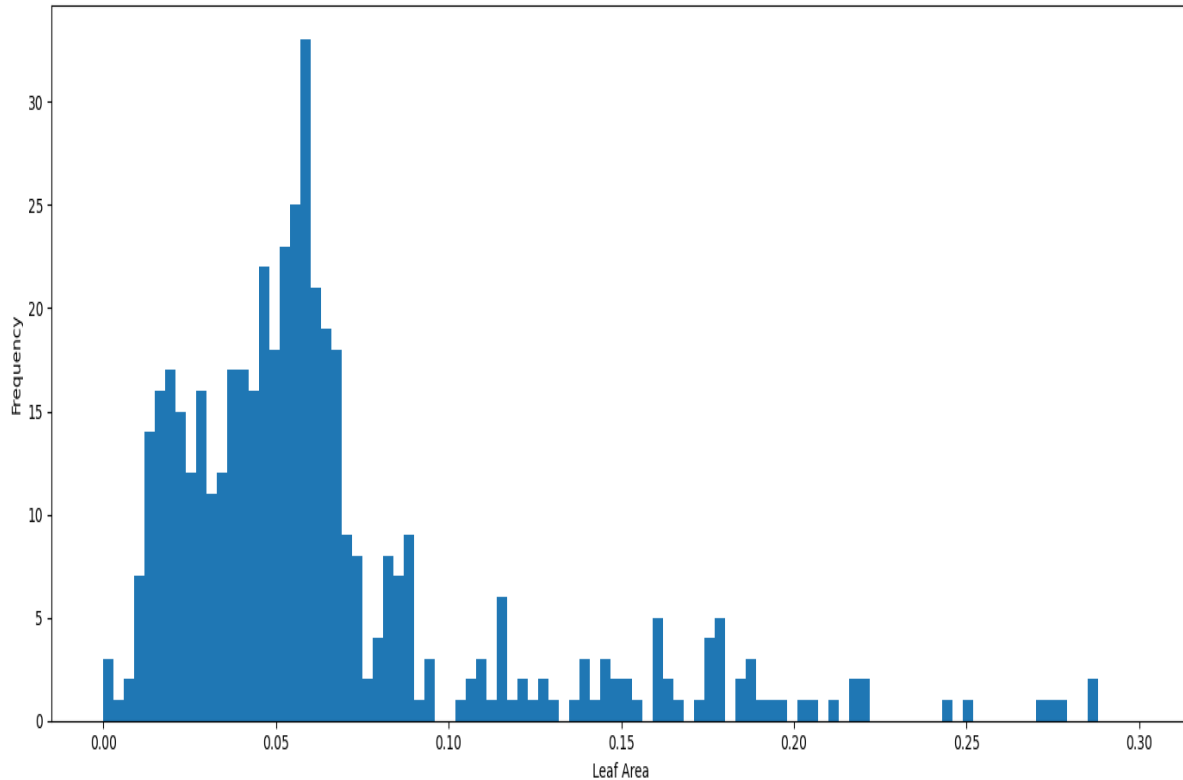
Model - Leaf Size Constrain

The size of leaf clusters should be in some reasonable range → constrain by area and, largest distance between 2 points in the clusters.

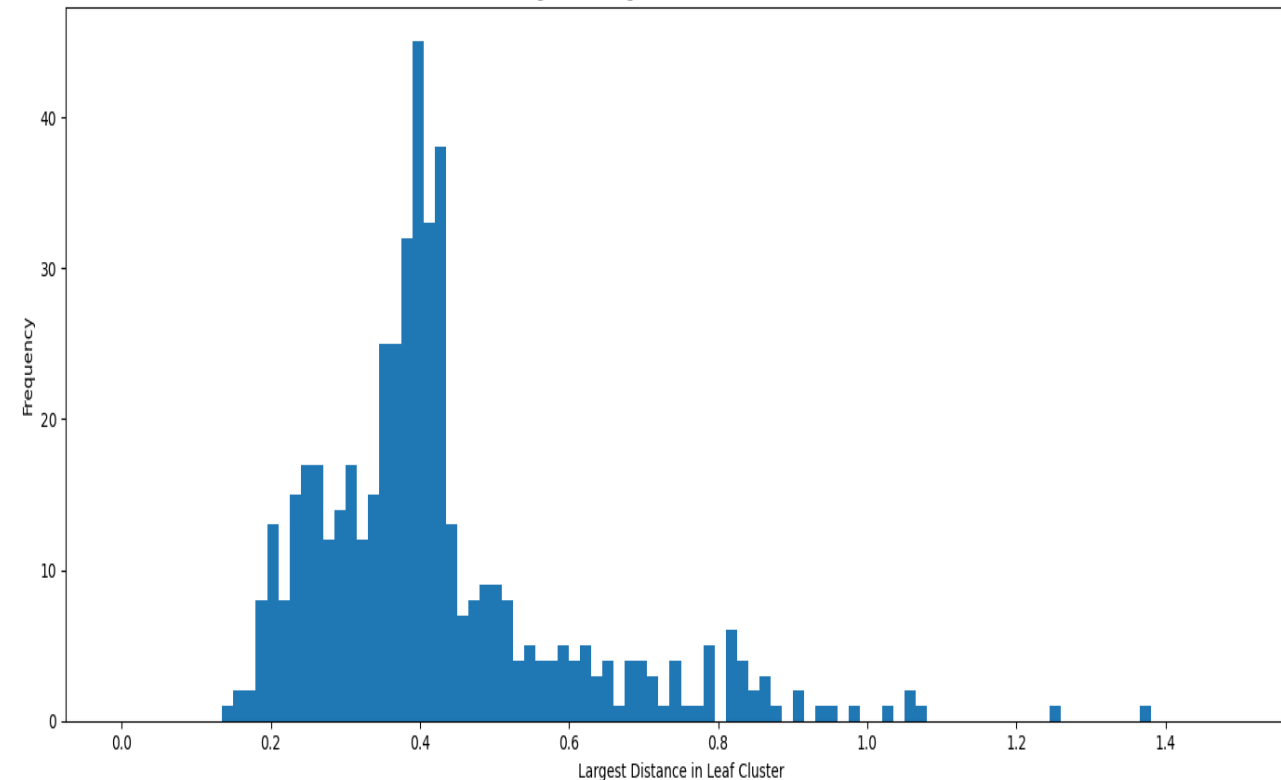
$leaf_area = 0.5 * surface_area(convex_hull(cluster))$



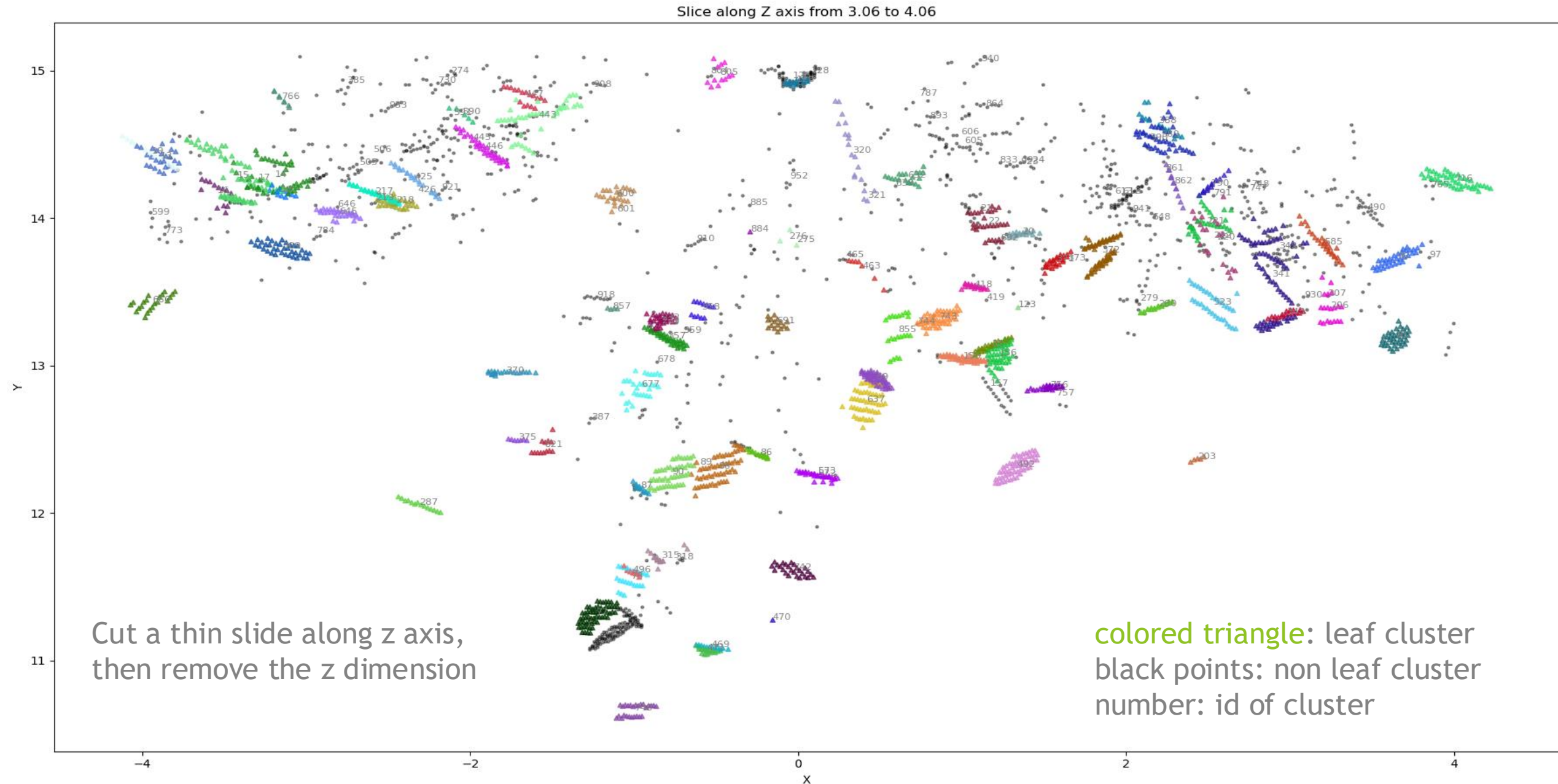
Histogram of Leaf Areas



Histogram of Largest Distances in Leaf Clusters



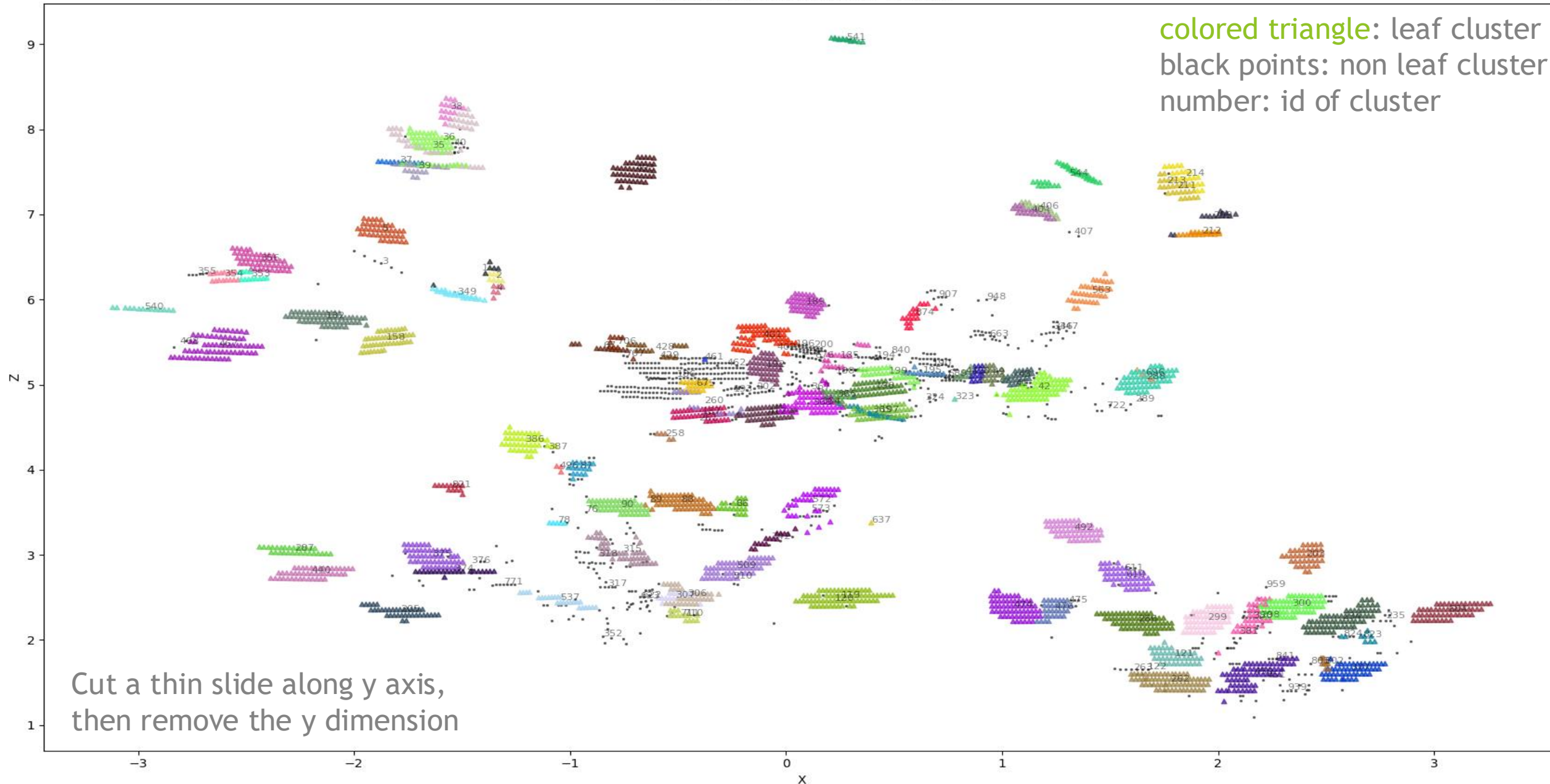
Evaluation - Visual Inspection - CT Scan



Evaluation - Visual Inspection - CT Scan

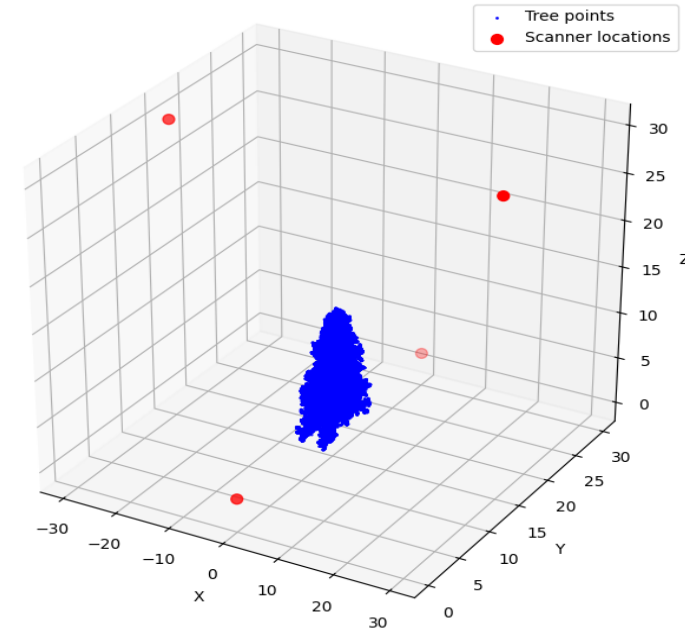
Slice along Y axis from 11.61 to 12.61

colored triangle: leaf cluster
black points: non leaf cluster
number: id of cluster

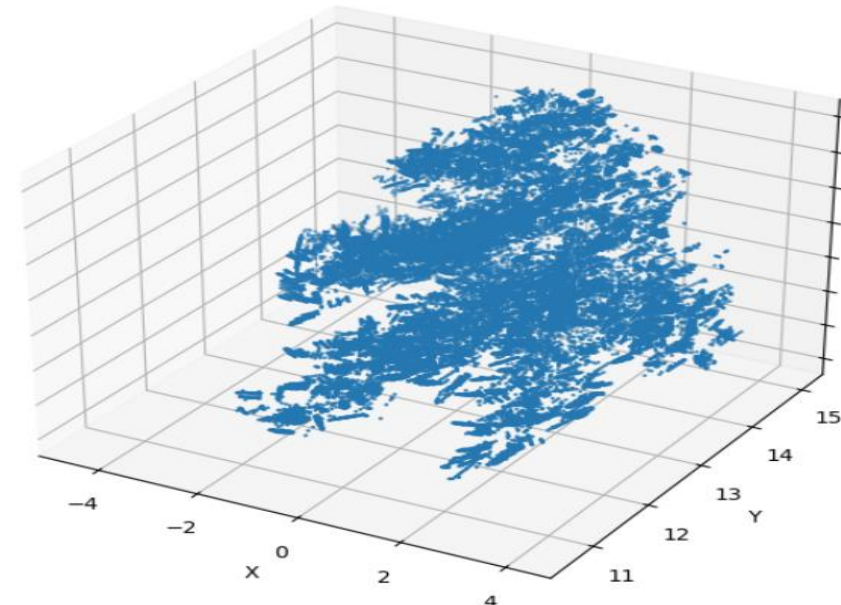
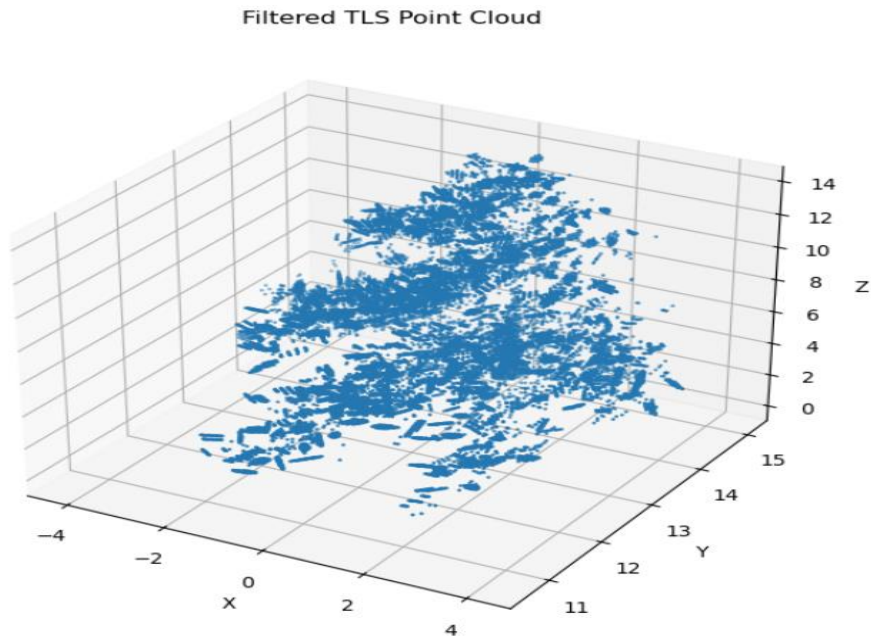


Evaluation - Simulated Data

- ▶ HELIOS++ was used to create artificial data, but we also found the same tree model as in the assignment
- ▶ With 3 added scanners more, previously occluded, leaves were found by using our model



Simulated Point Cloud



Final Estimation

Estimated on single-scan data:

- ▶ 916 leaves
- ▶ 55.59 m² leaf area

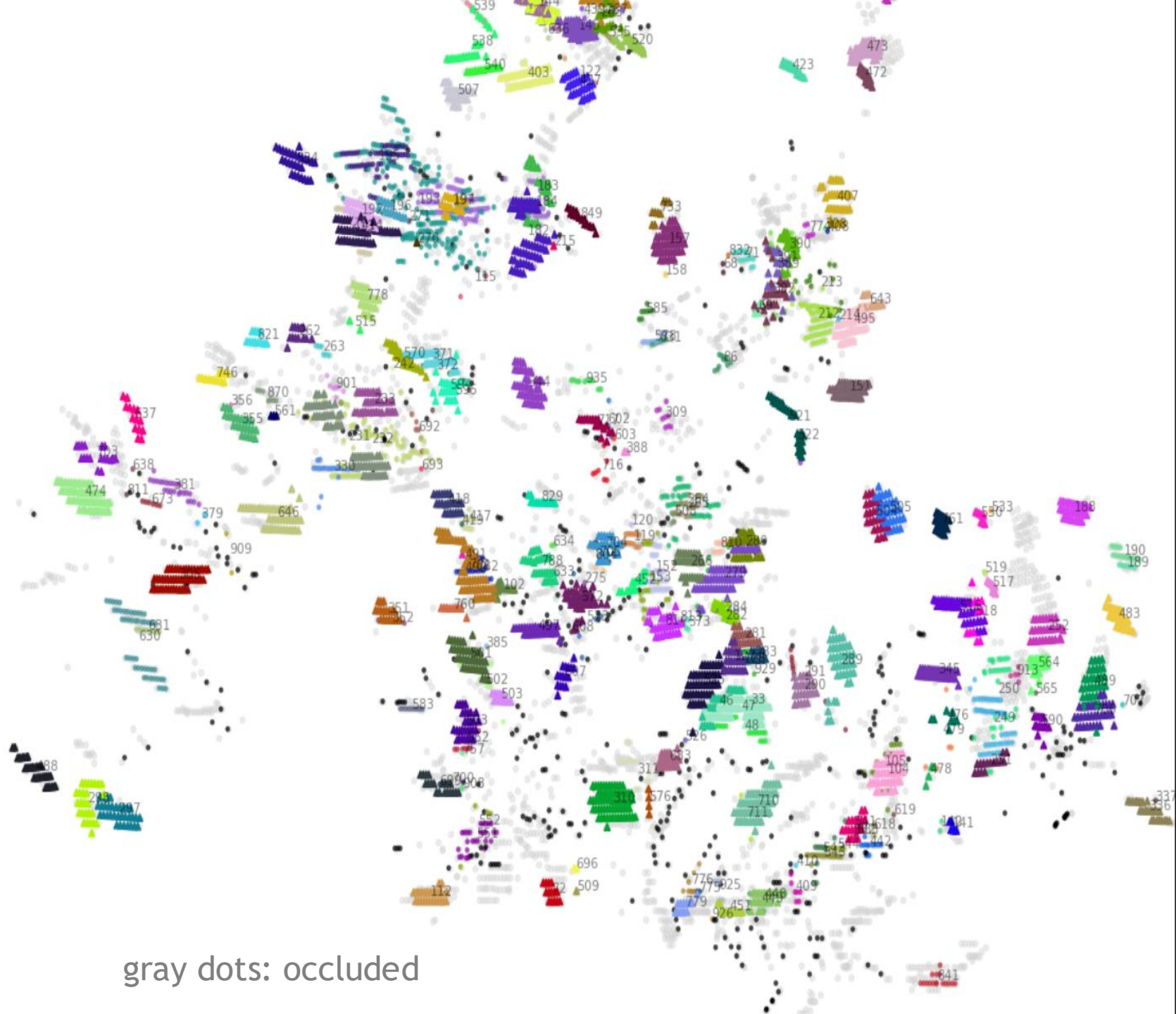
Estimated on multi-scan data:

1256 leaves
83.52 m² leaf area

Underestimation due to occlusion
in our single-scan data:

27% for leaf count
33% for leaf area

Maybe even more (~1400-1500
leaves) as there is still occlusion
in multi-scan data



Conclusion

Limitations:

1. Many hand tuned hyperparameters in the method. For new tree species, with different leaf shape, curvature and scan setting, these hyperparameters must be rechecked.
2. Subjective and manual: no labelled data, we relied on qualitative analysis.
3. No clear answer to the occlusion problem. Scanning from multiple angles is very helpful, but there is still occlusion. Some statistical model might be beneficial.

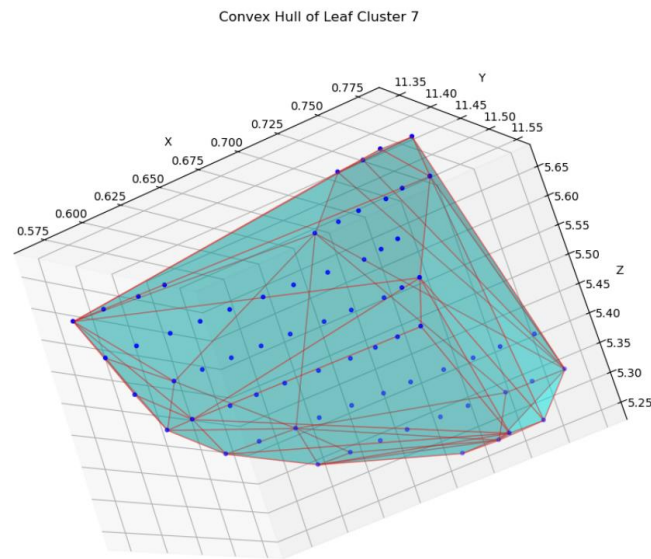
Advantages:

1. No labelled data needed. Which is the case in many real-world situations. Results can be evaluated qualitatively. Although, laborious and prone to error.
2. Explainable. We can clearly understand all the steps, parameters, results in the process.
3. The problem is casted as a constrain optimization problem. Better constrains and clustering algorithm could be added. So, it can be further generalized.

Thank you



Questions?



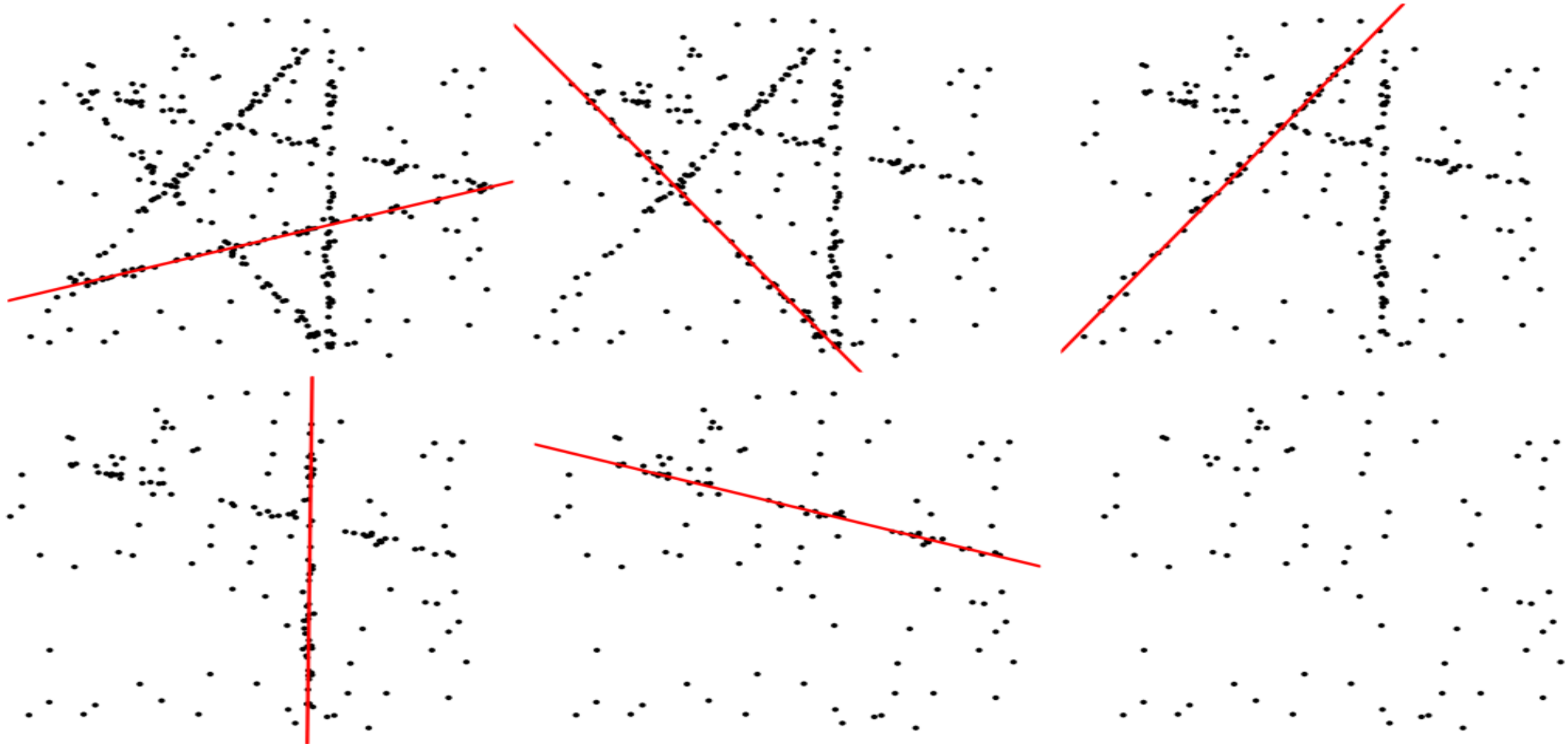


Summary

1. Keep only the tree half which faces the scanner.
2. Cluster tree into smaller clusters by **DBSCAN**, reduce the search space of next step.
3. For each smaller cluster, find the best clustering which gives the most leaf-like clusters. **DBSCAN_optim** and **seq_RANSAC_optim** are applied. The leaf-like clusters are identified by shape analysis (**is_leaf_cluster**). Exhaustive search over different parameters and clustering methods, to optimize **score_function**.
4. *Leaf area = half the surface area of convex hull of cluster*
5. Keep leaf clusters which have area and largest dimension in reasonable range.
6. **CT scans** along different axes, for qualitative evaluation.
7. Simulated data from HELIOS++ for estimate the effect of occlusion.

Why RANSAC stochastic?

[[Fouhey et al, 2011](#)]



RANSAC plane cut through objects

