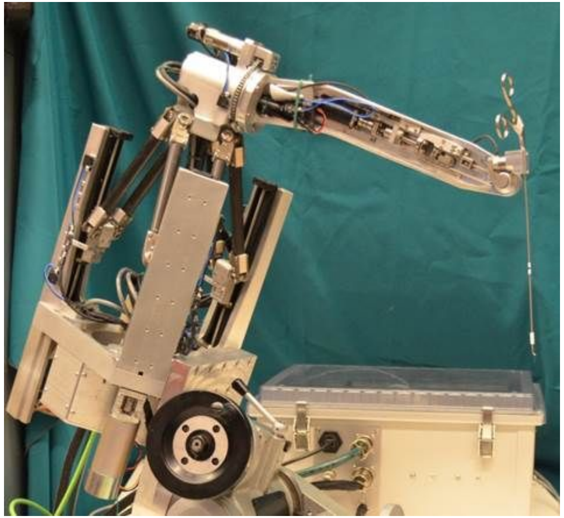
“Adaptive Simplification of Point Cloud Using *k*-means Clustering”

Authors: Bao-Quan Shi, Jin Liang, Qing Liu

Joseph Min - Group 9

Mentors: Russell Taylor, Yunus Sivimli, Bernhard Fuerst

Project Recap



Introduction | Initialization | Boundary | Subdivision | Refinement | Conclusion

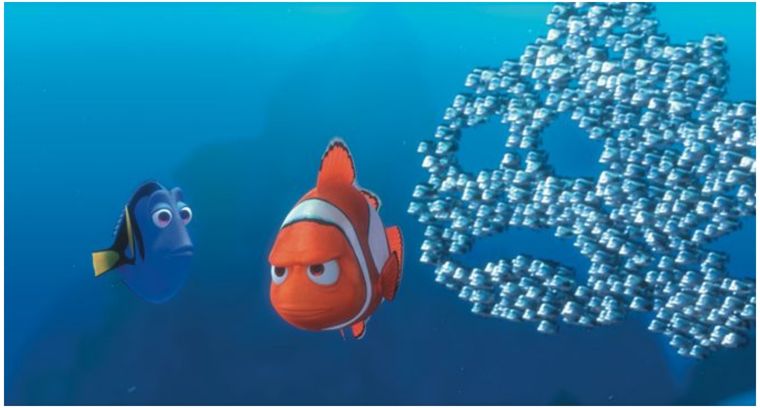
Paper Introduction

Problem: 3D clouds can be extremely dense

- Large storage space
- Long post-processing times

Solution: Remove unnecessary points

- Hard to correctly identify representative points and still keep small features and boundaries

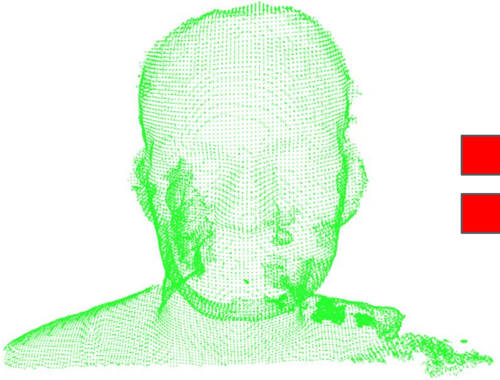


https://ohmy.disney.com/wp-content/uploads/sites/25/2013/09/MarlinAndDory.jpg

Relevance to Project

Clouds have >200,000 points

Need to preserve features - performing surgery!!



≠

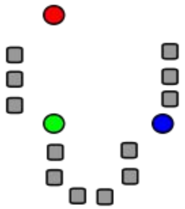
http://pointclouds.org/blog/_images/sub2_front.png

http://fundza.com/rman_helper/basics2/pyfig1.png

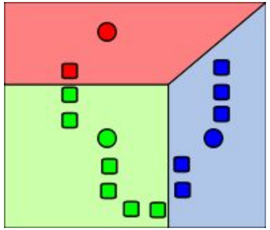
Simplification Steps

1. Create clusters using *k*-means clustering algorithm
2. Check original boundary integrity
3. Partition clusters into subclusters recursively
4. Refine clusters to balance density distribution of points

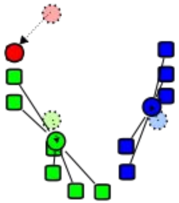
General *K*-Means Clustering



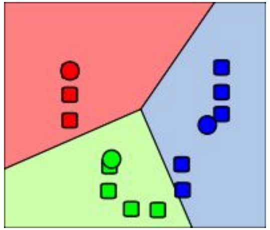
Select initial centroids



Partition points



Move centroids



Partition points again

Cluster Initialization Algorithm

Build a k–d tree using the N input points in R3

For i = 1 to N do

    If Pi is non-marked, search the fixed radius neighbors of Pi, [fixed radius specified by user]

        Mark those fixed radius neighbors

    End if

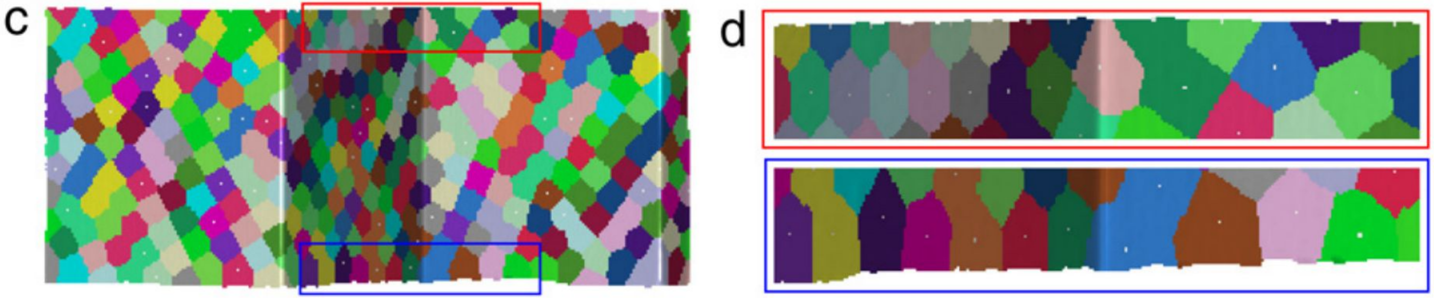
End for

Select non-marked points as initial cluster centroids

Cluster Initialization Example

Preserve Boundary Integrity

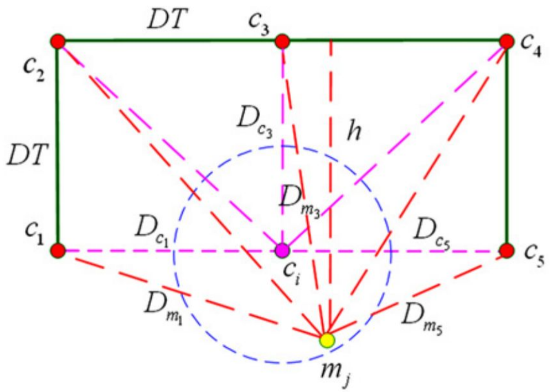
Possible for centroids to be far from actual boundary



Fixing Boundary Clusters

Split boundary clusters that have centroids far from true boundary

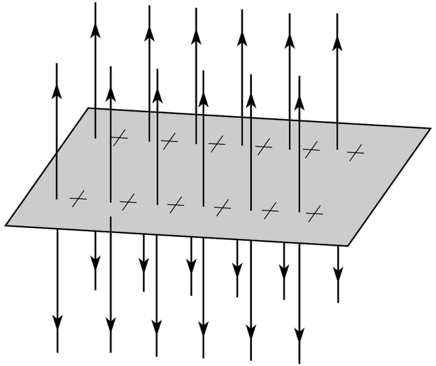
Two new centroids: the original, and a point furthest from centroids of neighboring clusters



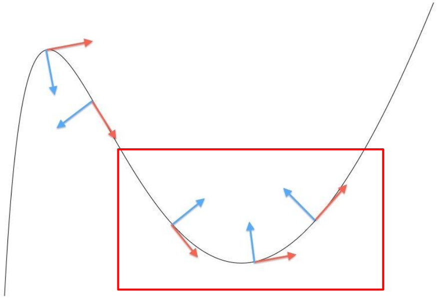
Picking Clusters to Subdivide

Use max normal vector deviation

Least square plane-fitting to find normal vectors during pre-processing



http://www.physics.brocku.ca/PPLATO/h-flap/phys3_1f_13.png



http://mathwiki.ucdavis.edu/@api/deki/files/71/line_1.jpg

Creating Subclusters

If max normal deviation of a cluster is higher than threshold

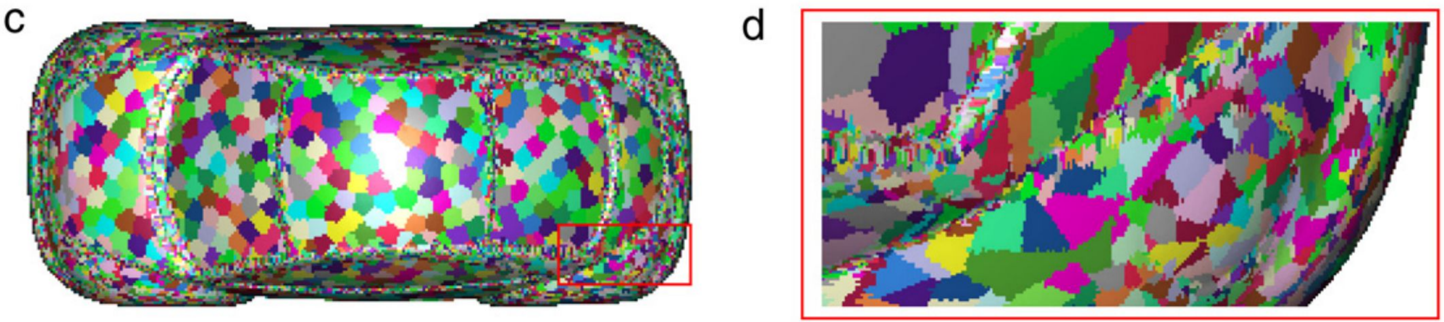
Pick the two points with the highest normal deviations as new centroids for two subclusters

Use same *k*-means algorithm to reassign points to new subclusters

Recursively repeat

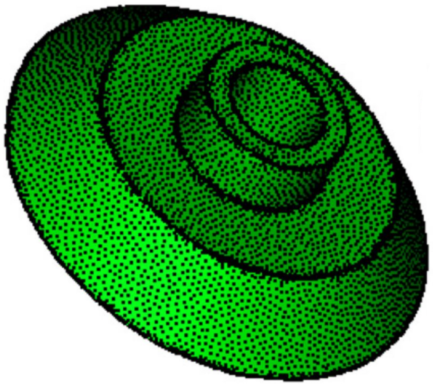
End if

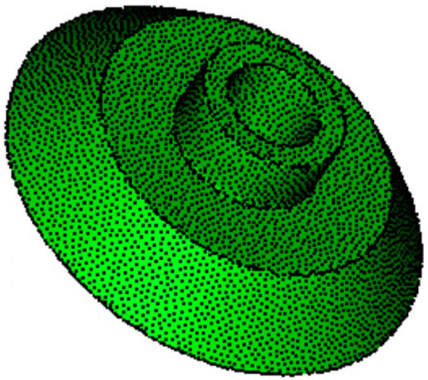
Subcluster Example



Cluster Refinement

Areas of surface change have way more subclusters





Refinement Algorithm

Summary:

Initial clusters should have <4 subclusters

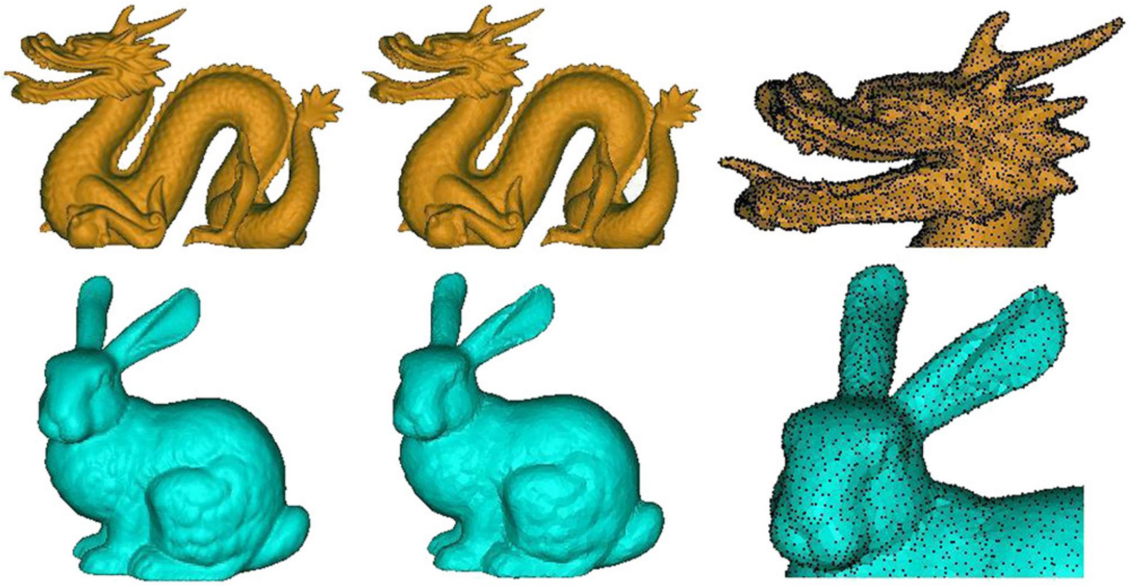
Subcluster centroids should not be close to each other

Ignore initial clusters with few subclusters

1. If $Dep_{Ci} > 2$, where Dep_{Ci} is the depth value of the initial cluster c_i.
2. Sort the sub-clusters in descending order according to their weights.
3. If $d(subc_j, c_j) < \delta DT$ /// in our implementation $\delta = 1/3$.
4. Remove the sub-cluster $subc_j$.
5. End if.
6. If $W_{subc_j} == W_{subc_1}$, $j \in [2, N]$, where N is the number of the sub-clusters.
7. Push $subc_j$ into the queue $\mathbf{Q}$.
8. End if.
9. Iteratively compute $d(subc_j, subc_k)$, $j \in [1, \mathbf{Q}.size()]$, $k \in [1, \mathbf{Q}.size()]$, $j \neq k$.
10. If $Maximum(d(subc_j, subc_k)) > \delta DT$
11. Preserve $subc_j$ and $subc_k$. ///Keep two sub-clusters.
12. If $d(subc_j, subc_l) > \delta DT$ && $d(subc_k, subc_l) > \delta DT$
13. Preserve $subc_l$. /// $l \in [1, \mathbf{Q}.size()]$
14. End if. // Keep another important sub-cluster.
15. If $d(subc_j, subc_m) > \delta DT$ && $d(subc_k, subc_m) > \delta DT$
16. Preserve $subc_m$. /// $m \in [\mathbf{Q}.size(), N]$
17. End if. // Keep a sub-cluster in the flat area.
18. Else
19. Preserve $subc_1$
20. End if.
21. End if.

Results

Impressive compression

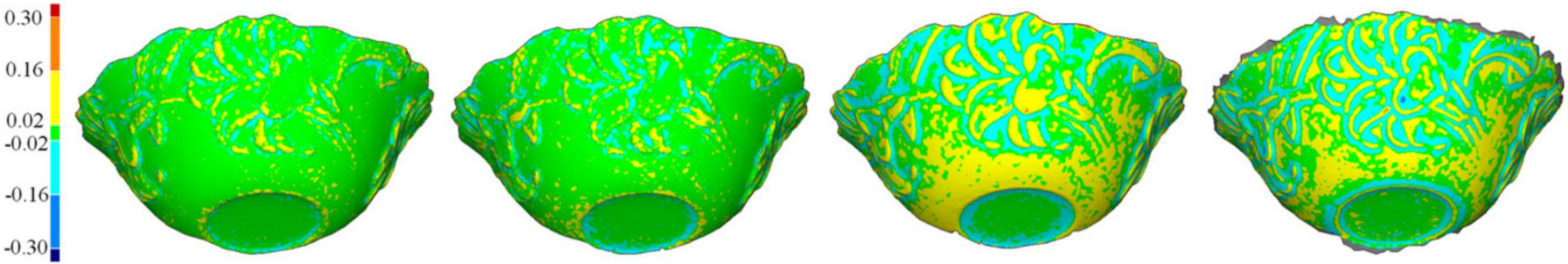


Dragon: 7.6% (435,545 to 32,925)

Bunny: 12.8% (34,834 to 6,500)

Results Comparison

Error measured as Euclidian distance between a sampled point and its projection on simplified surface



k-means | 3D Grid Method | Curvature-Aware Method | MLS-based Method

Analysis

Impressive feature integrity and data compression

Deals very poorly with noisy data (because it uses max normal deviation)

Not suitable for our project

Sources

Shi, Bao-Quan, Jin Liang, and Qing Liu. "Adaptive Simplification of Point Cloud Using Kk-means Clustering." Computer-Aided Design 43.8 (2011): 910-22. Science Direct. Computer-Aided Design, 9 Apr. 2011. Web. 21 Apr. 2016.

https://en.wikipedia.org/wiki/K-means_clustering

Questions?