Pole Detection

Amar Shah, Randall Harris, Sonia Nigam

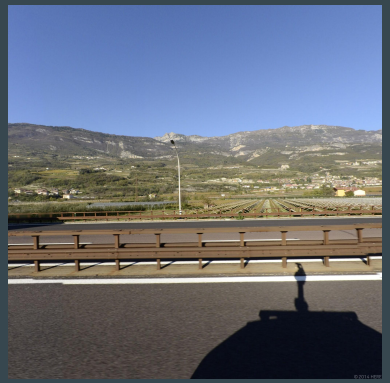
Background Research

Texts:

- Shape-Based Recognition of 3D Point Clouds in Urban Environments
  - Guided our step-by-step methodology (outlined in following slides)
  - http://www.cs.princeton.edu/~funk/iccv09.pdf
- Documentation for Python bindings for PCL
  - Library that handles point cloud processing
  - Optimized functions for segmentation and filtering
  - http://pointclouds.org/news/2013/02/07/python-bindings-for-the-point-cloud-library/

Raw Images of the Scene





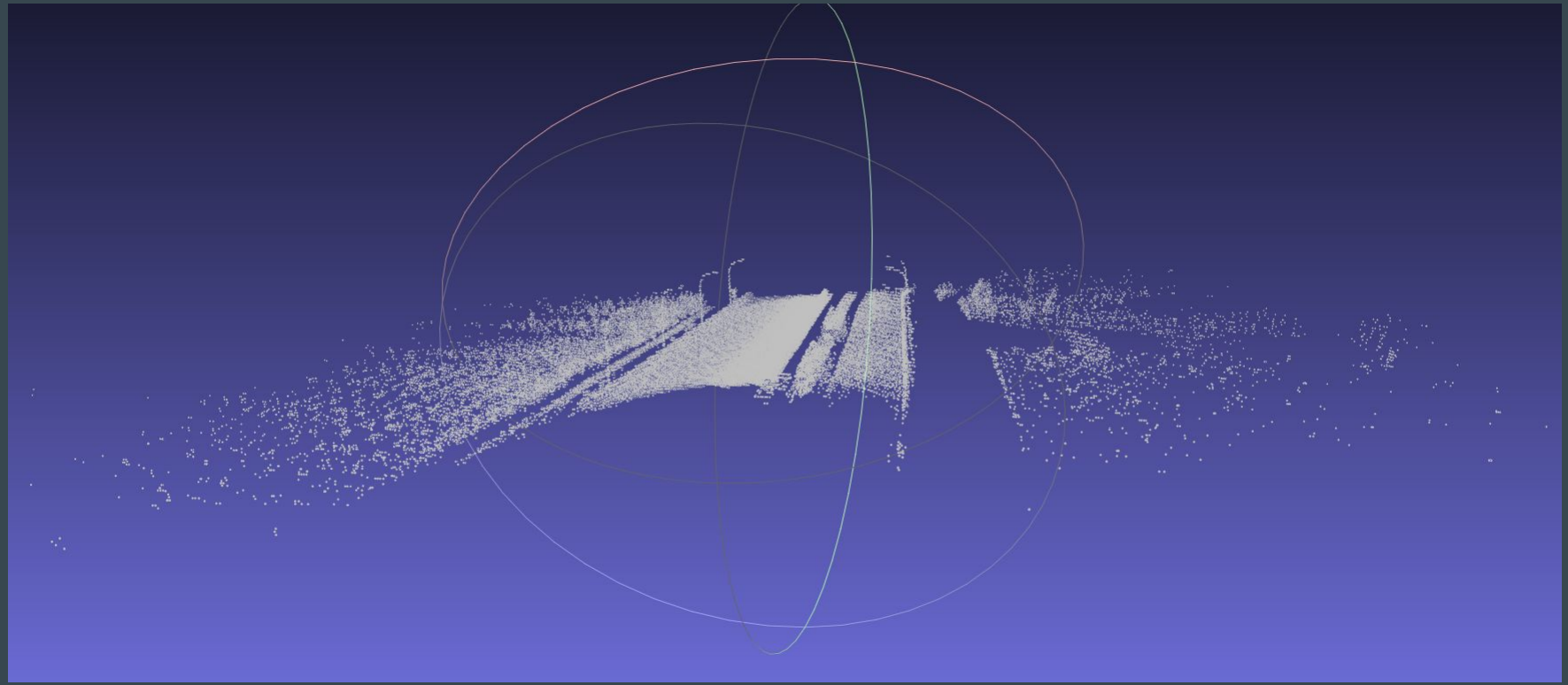

Industry Approach

- In the real world, object detection relies on advanced algorithms trained on extensively labelled datasets
- The training set would be comprised of point cloud data derived from images of different angles
    - Road scenes with several objects labelled, including poles
- We would select an ML algorithm, and train it on a labelled dataset such that it learned the typical composition of a pole within point cloud data
- Our testing set would be the inputted point cloud
    - The algorithm would detect poles based on its understanding of objects from the training data

Process: First Step - Visualization

- Input: file_project_point_cloud.fuse
  - Format: latitude longitude elevation intensity
- Convert each point into (x, y, z) coordinates
  - Function: cartesian(latitude, longitude, elevation)
- Save the converted data as a .obj file
  - Format: v x y z
  - File name: “pointcloud.obj”
- Meshlab
  - Open source project for rendering 3D data
  - Opened the saved file utilizing this software
  - Enabled quick, quality visualization of our original dataset
  - http://www.meshlab.net

Original Point Cloud: 430,736 points

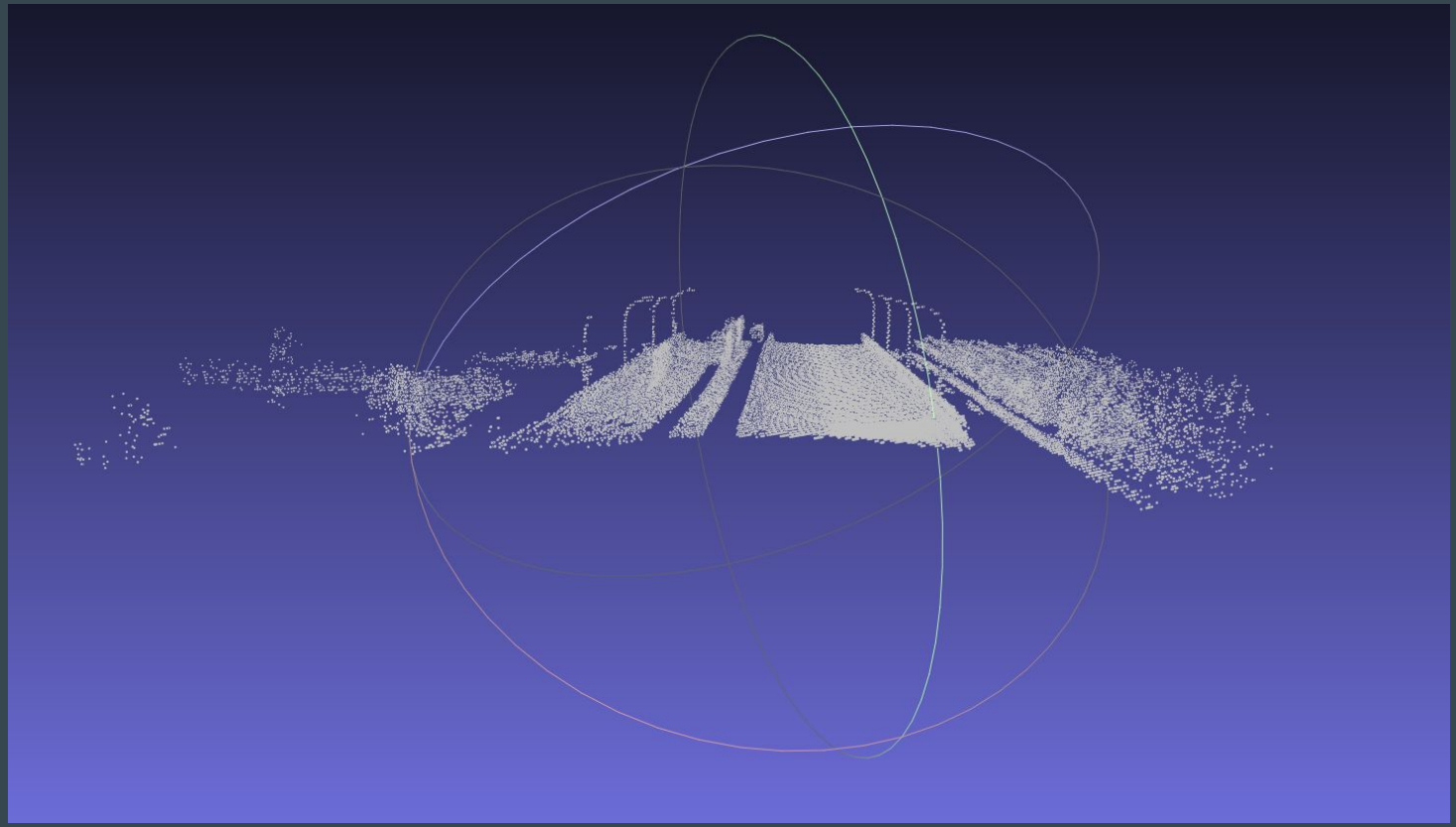


Process: Second Step - Filtering Outliers

“Remove isolated points.”

- We started our filtering process by removing outliers, not relevant to the data
  - Smoothes out noisy original data set
- PCL: Statistical Outlier Filter
  - Uses KNN to trim data points in sparsely dense areas
  - K: 50
    - Number of neighboring points analyzed
  - Standard Deviation Threshold: 5
    - Points with a distance greater than 5 standard deviations away from mean distance from the query point are removed
- Extract indices found in the filter, and return remaining point cloud data

Point Cloud Without Outliers: 428,148 points



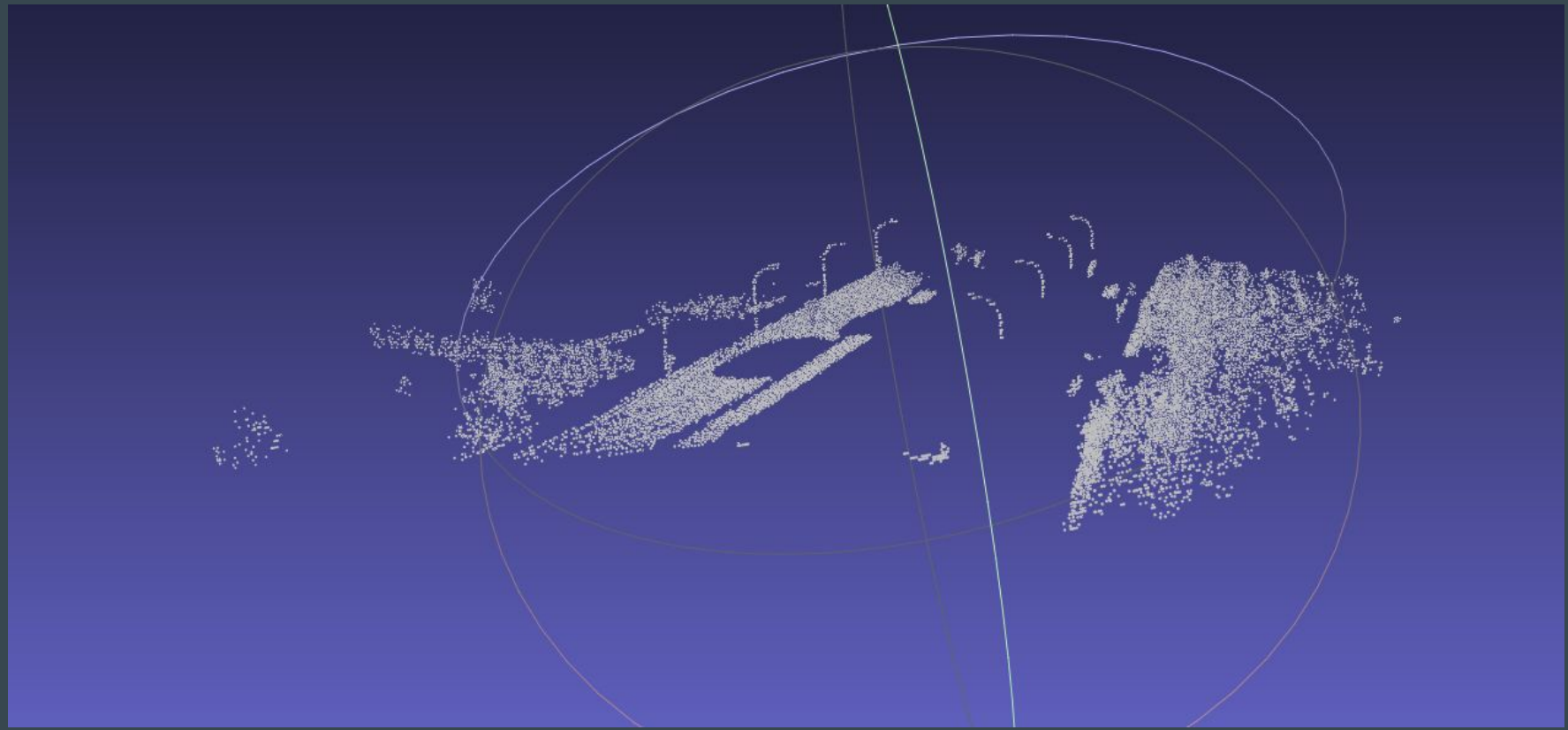
Process: Third Step - Filtering Large Components

*“We filter out points likely to belong to *buildings by removing very large connected components.”*

- Remove the points associated with the road
- Remove the points associated to the surrounding hills
  - The large point groups to the left and right of the center road plane
- KD Tree
  - For each point, found 1000 closest points (a tentative component) and their squared distances
    - PCL: kd.nearest_k_search_for_cloud(point_cloud, 1000)
  - Summed squared distances per tentative component
  - Threshold: if summed squared distance < 5000, removed this component
    - Assessed density: only removed large chunks of data that were closely linked

*in our case, buildings were not the large objects targeted for removal, but this was still an applicable step within the process

Point Cloud Without Large Components: 42,135 points



Process: Fourth Step - Cylindrical Segmentation (Paper)

“Latitude a simple approach to finding potential object locations is to generate a 2D scalar image representing the “height” of the point cloud, and performing image processing operations to find local maxima”

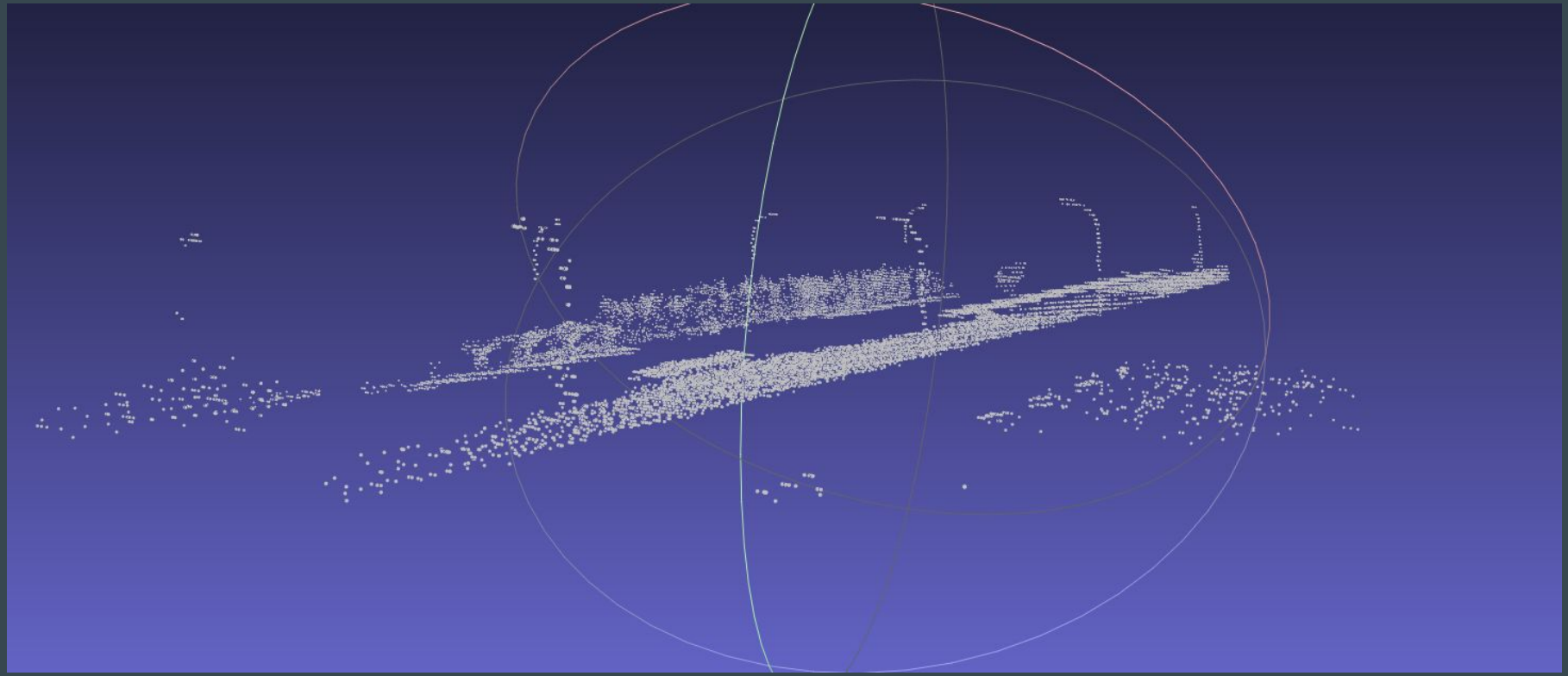
- The paper recommended to make a 2D image of the point cloud
- Due to time constraints, we did not believe that creating this 2D image and filtering by cloud was significantly better compared to our own cylindrical segmentation
- Both methods has similar cons

Process: Fourth Step - Cylindrical Segmentation Our Method

“Extract connected components with area under a threshold to find small objects. This method is effective at finding isolated poles, but performs worse for cars and objects amongst clutter.”

- SACModel = cylinder
  - Segment for points that were in a cylinder with radius between bounds 0 and 10.
  - Distance threshold between points was 20
  - Our goal was to segment out cylindrical objects and discard all other points
- Weakness: long part of the pole is a cylinder, but the top is not
- Radius : Had to increase radius to account for top part of lamp pole, but this allowed other points to come in

Point Cloud With Cylindrical Segmentation: 28,605 points

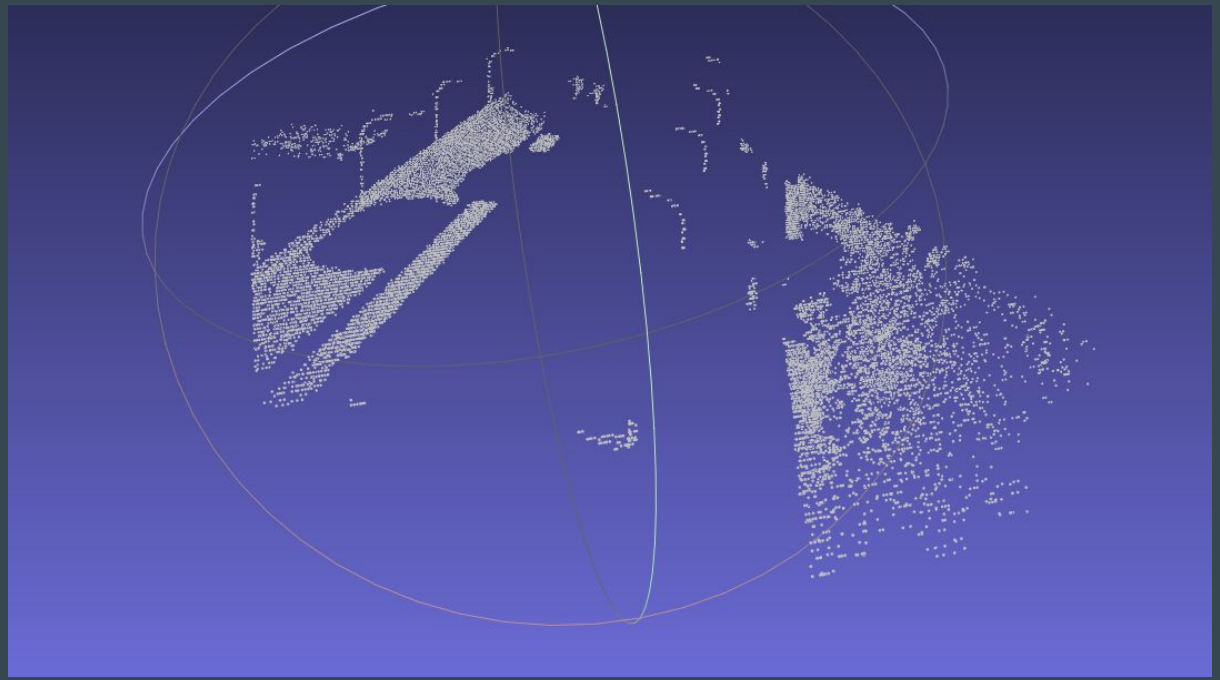


Process: Fifth Step - Plane Segmentation

“We filter out points close to the ground, which is estimated at uniformly spaced positions with iterative plane fitting.”

- The final step was isolating the ground plane, and then keeping everything but the isolated segment
  - PCL: SACMODEL_PLANE segmentation
  - Extracted all indices found within the segmented plane
- To reinforce the standard PCL segmentation, we also applied a coordinate-based filter
  - We wanted to set a height threshold, and dispose of all points below a given level
  - The point cloud was on a tilted axis, so this made this step a bit trickier
  - Empirical analysis supporting filtering based on x-coordinate
    - Filter bounds: 0, 4364071.0

Final Results: 25,341 points



The image on the left shows the output of the cloud points after filtering and segmentation. These data points are found in final.obj As you can see, the poles are intact and relatively isolated.

This process removed a net total of 405,395 data points

Errors/Improvements

- The primary weakness of our implementation is handling the ground plane
  - Implement a more accurate method for ground plane segmentation
  - Better define an altitude filter as a combination of coordinate parameters
- Utilize a training set
  - Leveraging labelled images would define a much more sophisticated approach
- We could also improve our implementation by further developing the outlier filter
  - This step did not remove a significant number of points → room for potential improvement
  - Instead of just relying on density, we could better define what “outlier” is in our scenario
  - As such, a lot of data could be removed upfront, perhaps clarifying the image for later segmentation steps