OBJECT DETECTION IN POINT CLOUD: POLE

GEOSPATIAL VISION/ VISUALISATION
CS - 513

Submitted by:
Bharatiben Kyada (A20353636)
Sharul Saxena (A20380920)
Tanmay Pradhan (A20376280)
Vickyben Patel (A20370450)

Introduction

- A point cloud is a set of data points in some coordinate system. In a three-dimensional coordinate system, these points are usually defined by X, Y, and Z coordinates, and often are intended to represent the external surface of an object.
- In this project, an algorithm is developed to automatically locate, segment and classify small objects (in our case, pole) in 3D scan of urban environment.
- Object detection in point cloud is popular in HD Map and Sensor-Based autonomous driving. There are basically four types of object which can be obtained from daily scenarios like: lane, guardrail, pole, road surface etc.

Approach

- Object detection based on point cloud relies on labelled datasets obtained from point cloud data, which requires advance algorithms.
- Python’s Point Cloud Library functions are used for segmentation and filtering of the point cloud data.
- The point cloud data which is taken as input consists of latitude, longitude, altitude and intensity. It is derived from images obtained from different angles including poles.
- The algorithm build will be able to detect the typical composition of a pole within point cloud.
- The output generated will be in 3D and saved as .obj format which can be viewed using Meshlab.

Algorithm

Original Point Cloud Data

- Input: final_project_point_cloud.fuse
  - Format: [latitude] [longitude] [altitude] [intensity]
- Data file is then read and each point is converted into (x,y,z) coordinates using cartesian(latitude, longitude, altitude).
- Converted data is saved as “originalData.obj” file
  - Format: v x y z
  - Meshlab will show the original point cloud 3D view with 430,736 points - (Result/Fig.1)

Removing Anomalous Points

- Isolated points which are not relevant to data in point cloud are removed, reducing noise in original data.

- Statistical Outlier Removal is used to filter outliers.
  - This uses point neighborhood statistics to filter outliers from data.
  - This will compute the average distance that each point has to its k-nearest neighbors. The value of k is set to be 50 using setMeanK().
  - Next, mean and standard deviation are computed for all the distances.
  - The standard deviation multiplier threshold is set to 5 using set_std_dev_mul_thresh.
  - The distance threshold will be equal to: mean + stddev_mult * stddev.
  - Thus, points with distance greater than 5 standard deviations away from mean distance from isolated point are removed.
- The filtered point cloud data with reduced noise is then obtained for further processing.
  - Meshlab will show the filtered point cloud 3D view with 428,148 points - (Result/Fig.2)

Removing Large Objects

- Points related to mountains, buildings and road are filtered out.

- kD - Tree :
  - kD-Tree is built upon the filtered data obtained after reducing noise in previous step.
  - For each point, find closest points (1000) and squared distances.
  - Squared distances obtained are summed using numpy..
  - Taking 5000 as threshold, if summed squared distance is less than 5000, it is removed
- Thus, point cloud data is now free of large components.
  - Meshlab will show the filtered point cloud 3D view with 42,135 points - (Result/Fig.3)

First Segmentation

- At this step, the connected components with area below threshold are extracted. This is useful in finding small objects. Thus, it is effective at finding poles.
  - SACMODEL_CYLINDER segmentation is used to isolate poles.
  - The segments thus obtained were cylindrical in shape with radius between 0 & 10.
  - Distance threshold is set to 20 between the points.
- Therefore, cylindrical objects are segmented out and other points are discarded.

Second Segmentation

- Points close to ground which is a uniformly spaced plane of points are filtered out.

- This final step isolates the ground plane and keeps everything but isolated segments.
  - SACMODEL_PLANE segmentation is used to isolate ground plane.
  - After this, all indices are extracted which are found within the segmented plane.
  - A coordinate-based filter is applied to remove all points below threshold. Limits are thus set on x coordinate i.e. 0, 4364071.0
- Meshlab will show the filtered point cloud 3D view with 25,341 points - (Result/Fig.4)

Result

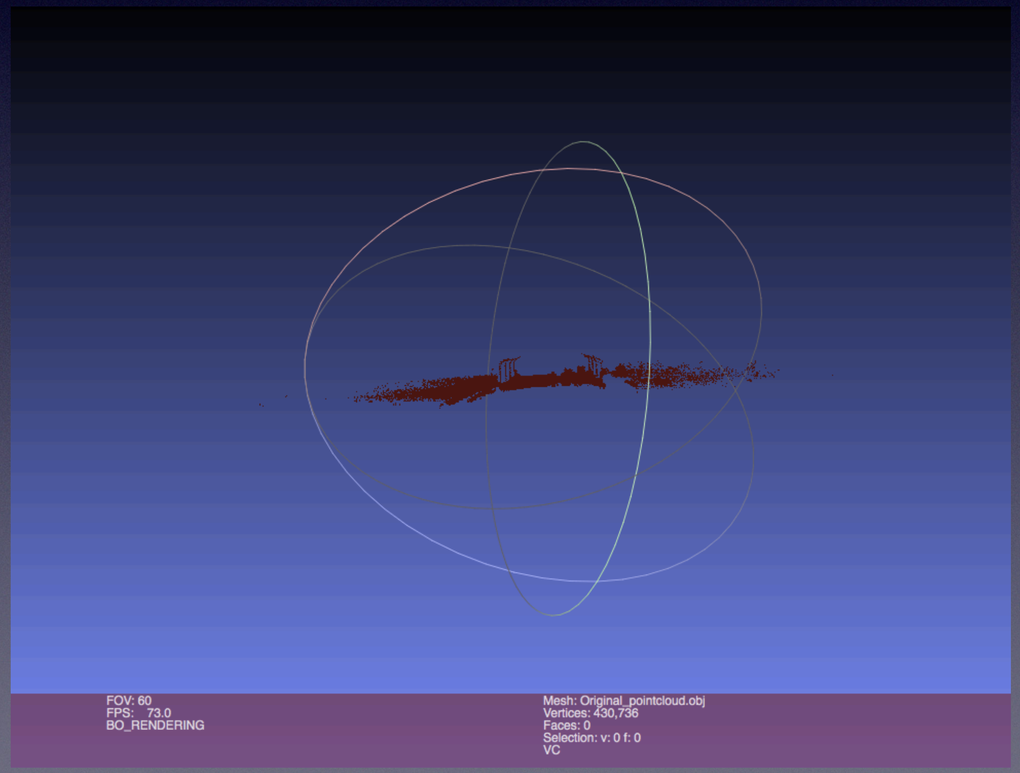

Fig.1

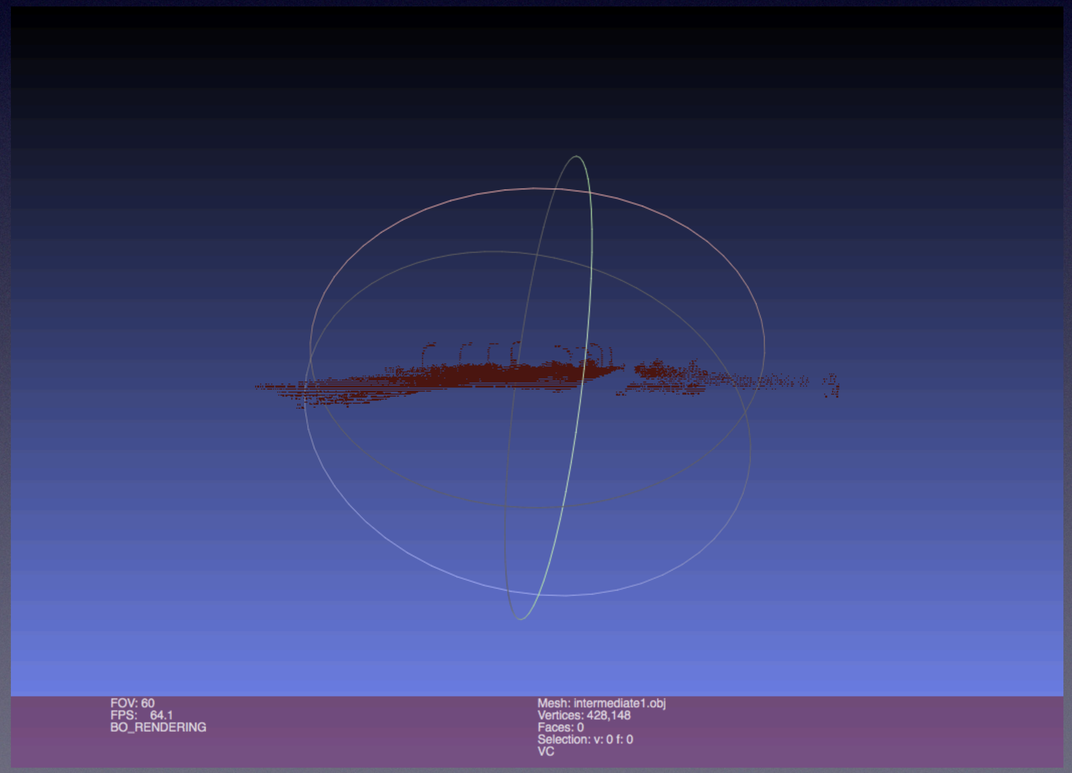

Fig.2

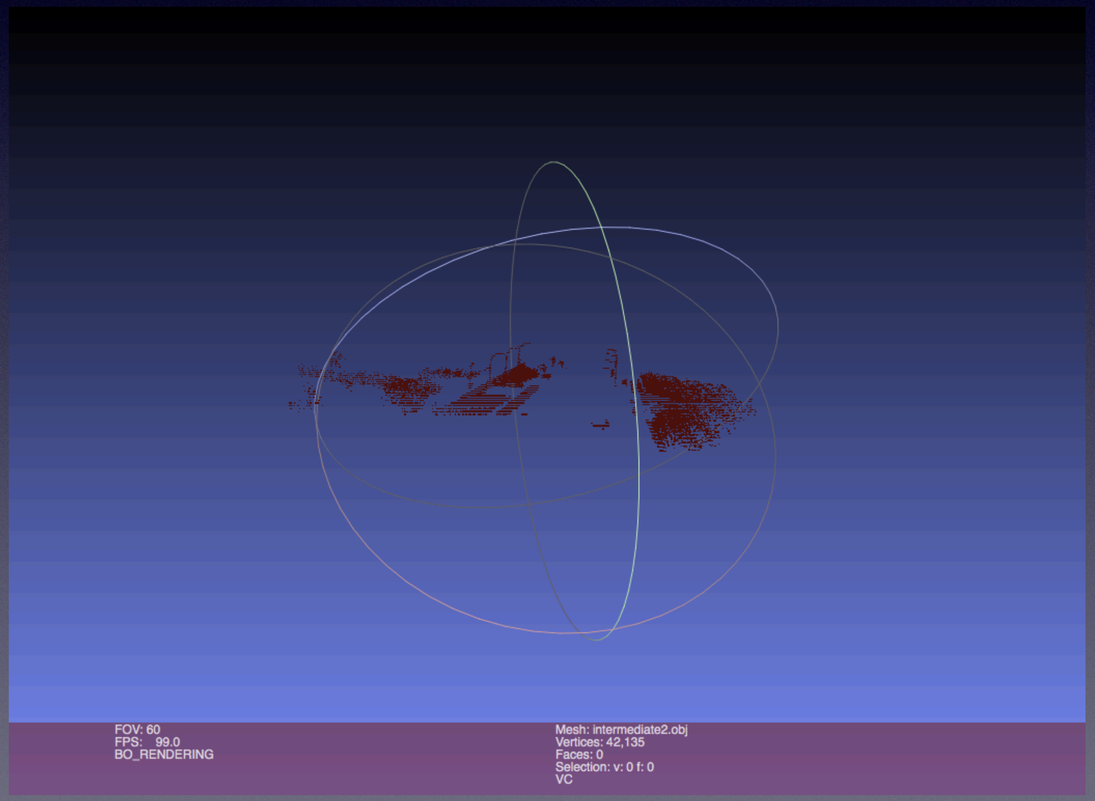
FOV: 60
FPS: 99.0
BO_RENDERING
Mesh: intermediate2.obj
Vertices: 42,135
Faces: 0
Selection: v: 0 f: 0
VC

Fig.3

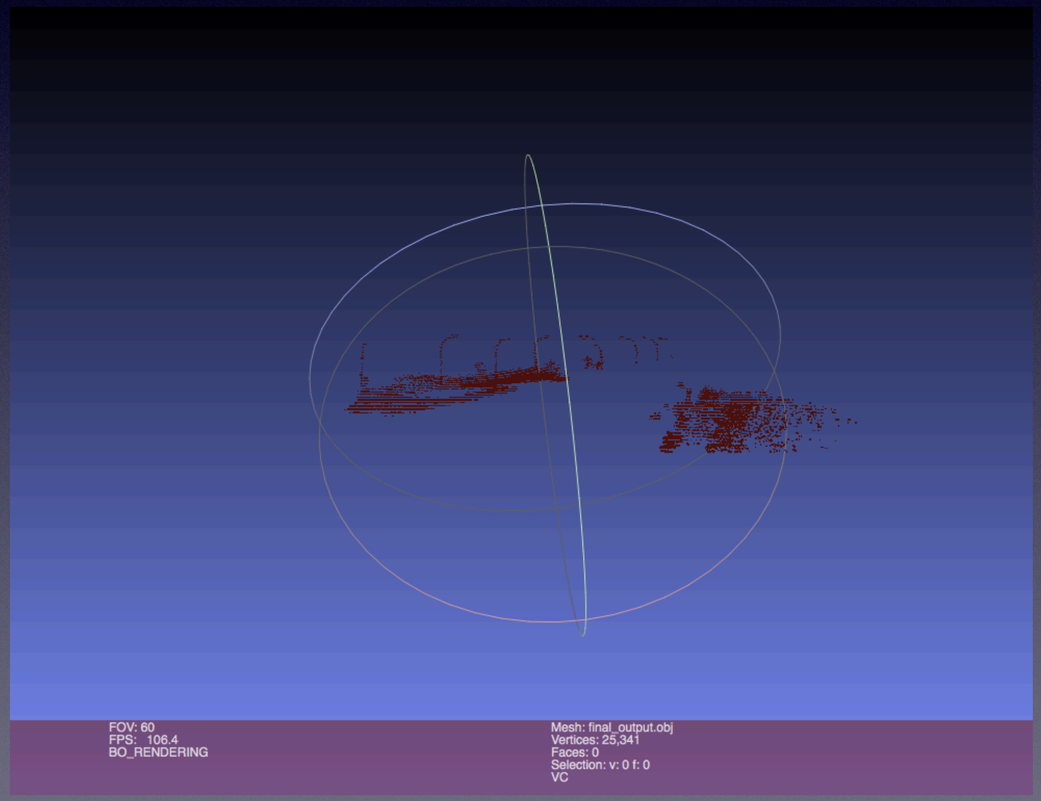
FOV: 60
FPS: 106.4
BO_RENDERING
Mesh: final_output.obj
Vertices: 25,341
Faces: 0
Selection: v: 0 f: 0
VC

Fig.4

Conclusion

- The final image obtained shows the output after performing filtering and segmentation of the point cloud data.
- These final data points can be seen in final.obj file. Meshlab will show the 3D view with isolated and intact poles.
- This process removed approximately 405,395 data points.

Future Scope

- Implementation can further be improved by developing anomaly filters because this did not remove a significant number of points.
- The long parts of pole which was cylindrical in shape were easily detected but not the top of the pole which can further be improved.
- Using ML algorithms with well defined and labelled images will lead to a more efficient algorithm

References

- Paper Reference:
    - http://www.cs.princeton.edu/~funk/iccv09.pdf
- Reference link for the function which converts latitude, longitude and elevation to x y z.
    - http://stackoverflow.com/questions/8981943/lat-long-to-x-y-z-position-in-js-not-working