Object detection in Point Cloud: Lane Marking

Final Project | Geospatial Vision and Visualization | Spring 20

Chirag Khandhar | Akshay Kulkarni | Megha Tatti

A20438926 | A20448255 | A20427027

Introduction

- The goal of this project is to detect the Lane marking for a small LIDAR point cloud.

- We've built a system which is able to identify the lane markings by analyzing the intensity value within the point cloud.

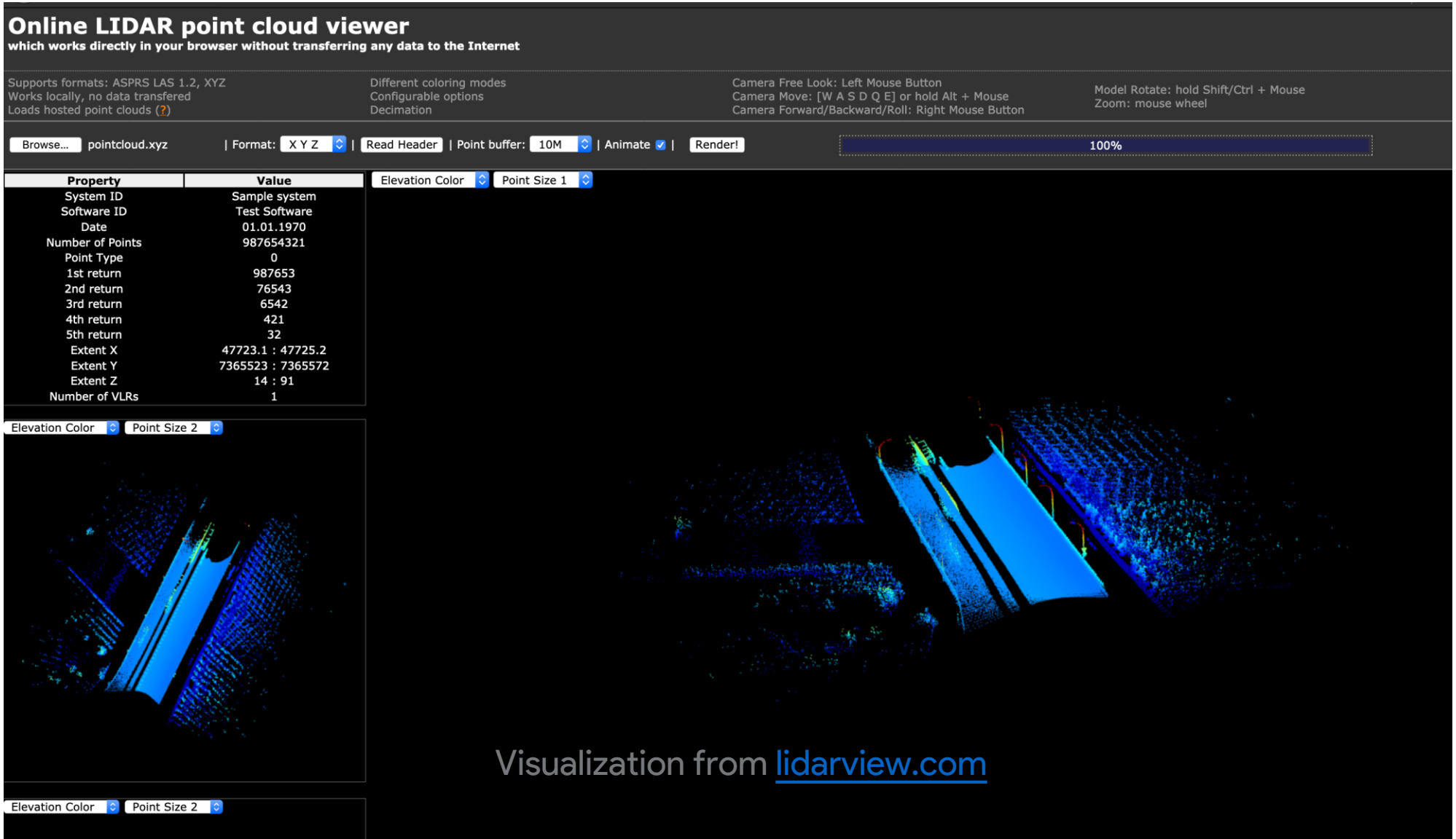
Methodology

- Visualization of Point cloud
- Filtering of Point Cloud
- Detection of Lane Marking
- Results

Visualization of Point cloud

- The given LIDAR point cloud has latitude and longitude information and needs to be converted to the Cartesian system.
- We used the latitude and longitude info to calculate UTM coordinates for each point.
- Hence, we got values for Easting(X) and Northing(Y)
- The altitude value is reflected as the Z coordinate in our new coordinate system.
- So now to visualize our newly calculated data we used http://lidarview.com

Point cloud Visualization

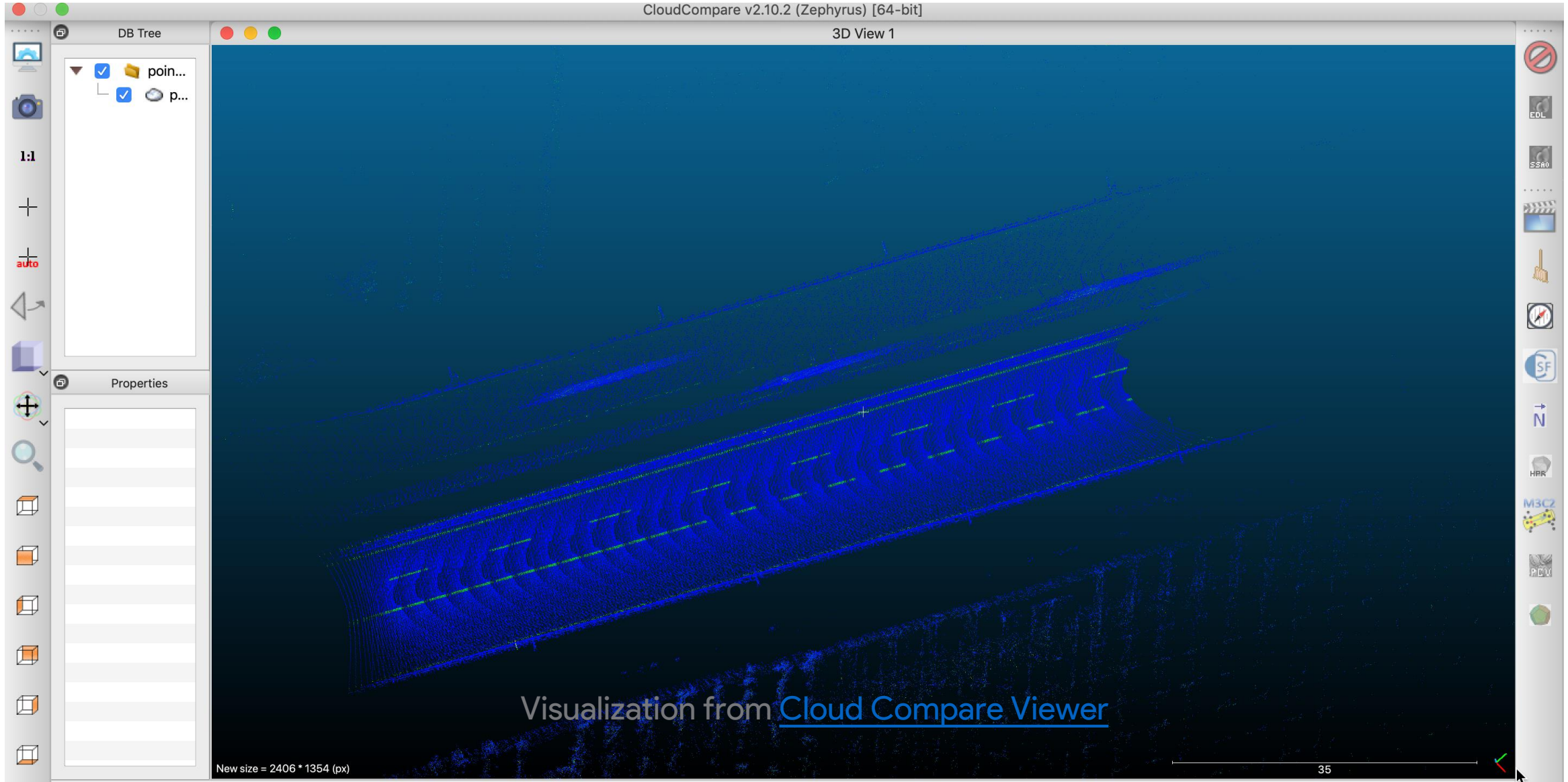


Visualization from lidarview.com

Visualization of Point cloud

- The idea to recover the lane marking from a flat surface is to extract it's intensity.
- Because the lane markings have a higher intensity compared to the plain asphalt or concrete of the street.
- However, the tool we used did not work very well, because it uses different colors to denote different levels of elevation.
- Thus, to solve this problem we used Cloud Compare Viewer that uses different colors to denote different levels of intensity.

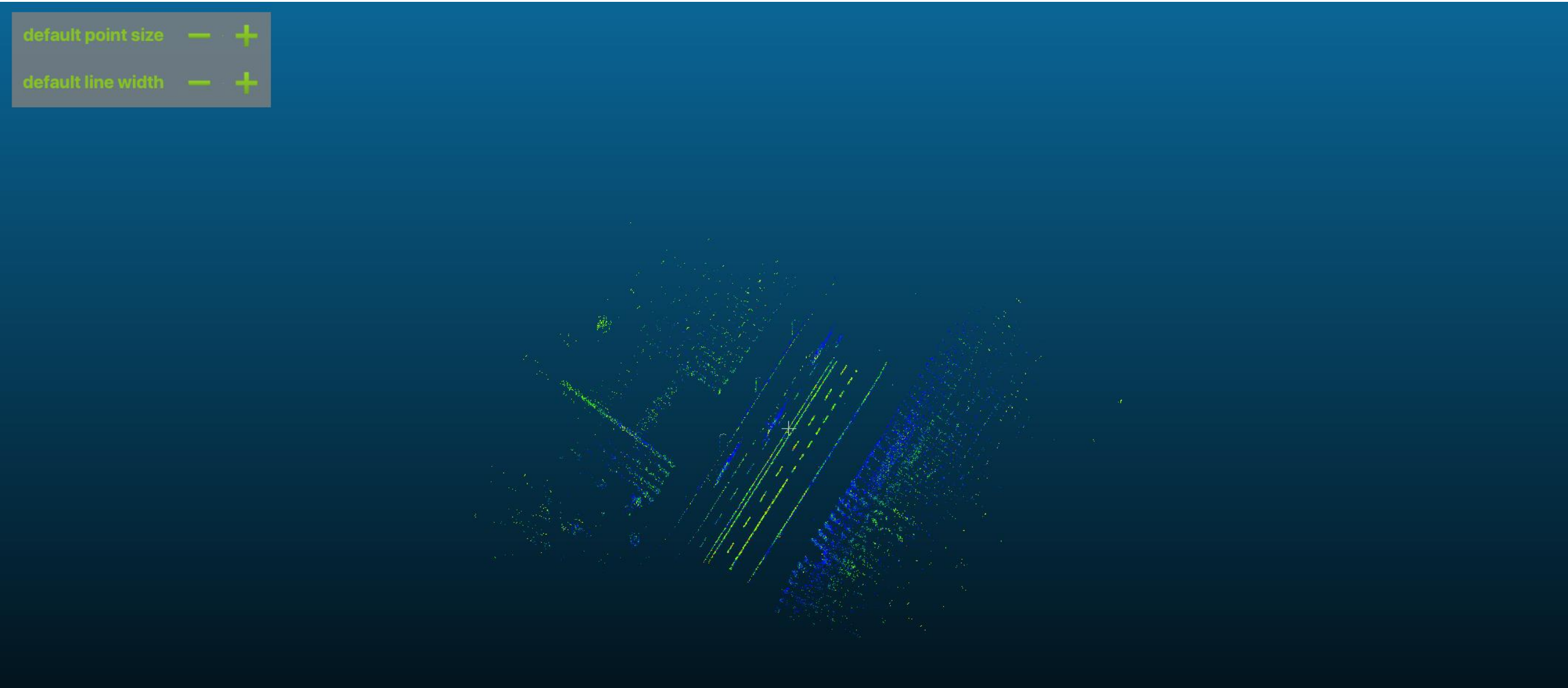
Visualization of Point cloud



Filtering of Point cloud

- The idea is to remove all the noise from the data, i.e. to remove all those points that have extreme intensities (too high/too low).
- As, different lighting conditions (day-night) might affect the intensity values, we thus calculated the mean and standard deviation intensity for all points.
- So our assumption is intensity of lane marking is greater than mean of all points.
- Also, we could have also set an adaptive threshold per point cloud slice. However, since there was no slice information left we had to use this approach.

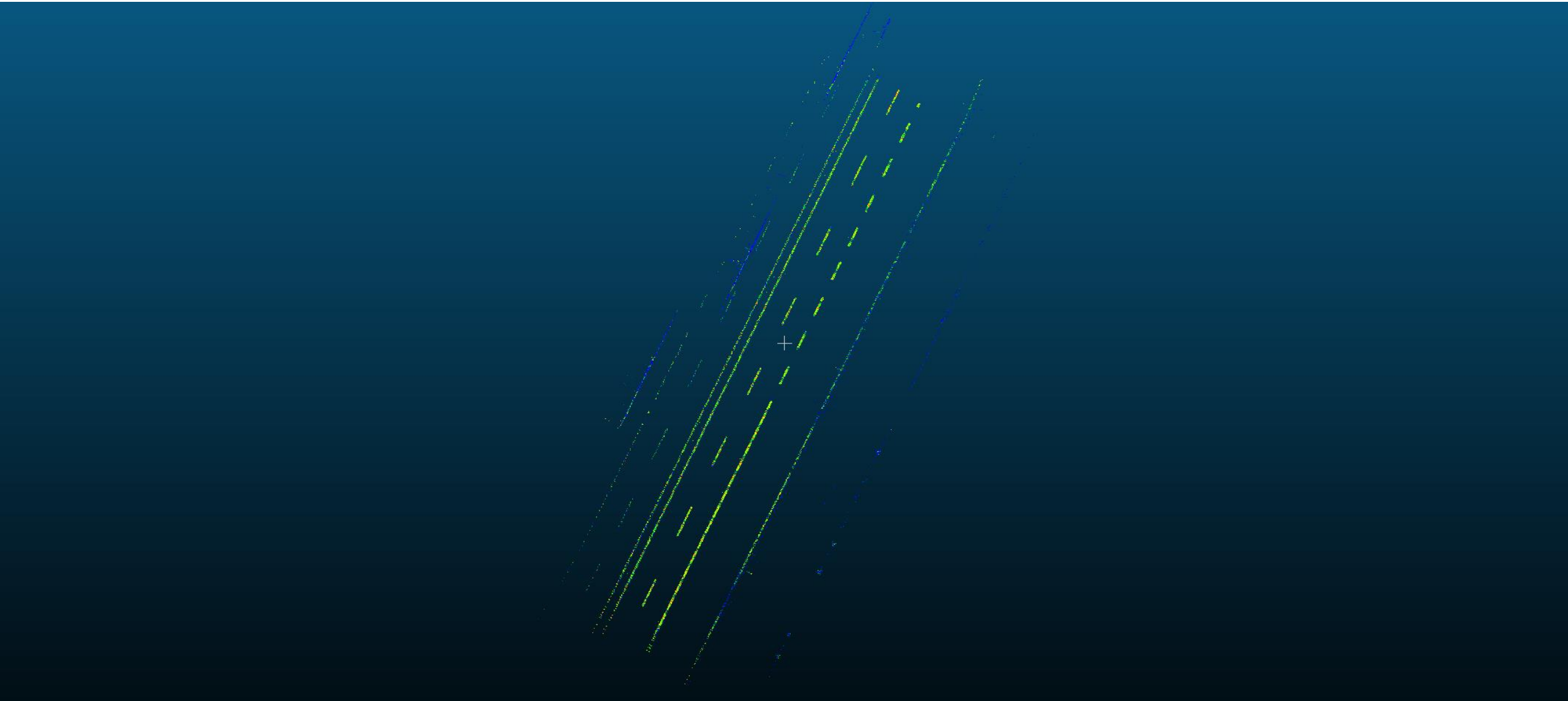
Filtering of Point cloud



Filtering of Point cloud

- So we can see that we have significantly removed a lot of noise, however there are a lot of points of the road.
- To reduce this, we used the actual trajectory information by removing points that are too far away from the actual trajectory line.
- We used a threshold value of 20m which reduced the number of points from 32191 to 18496.
- The results can be seen clearly in the next slide.

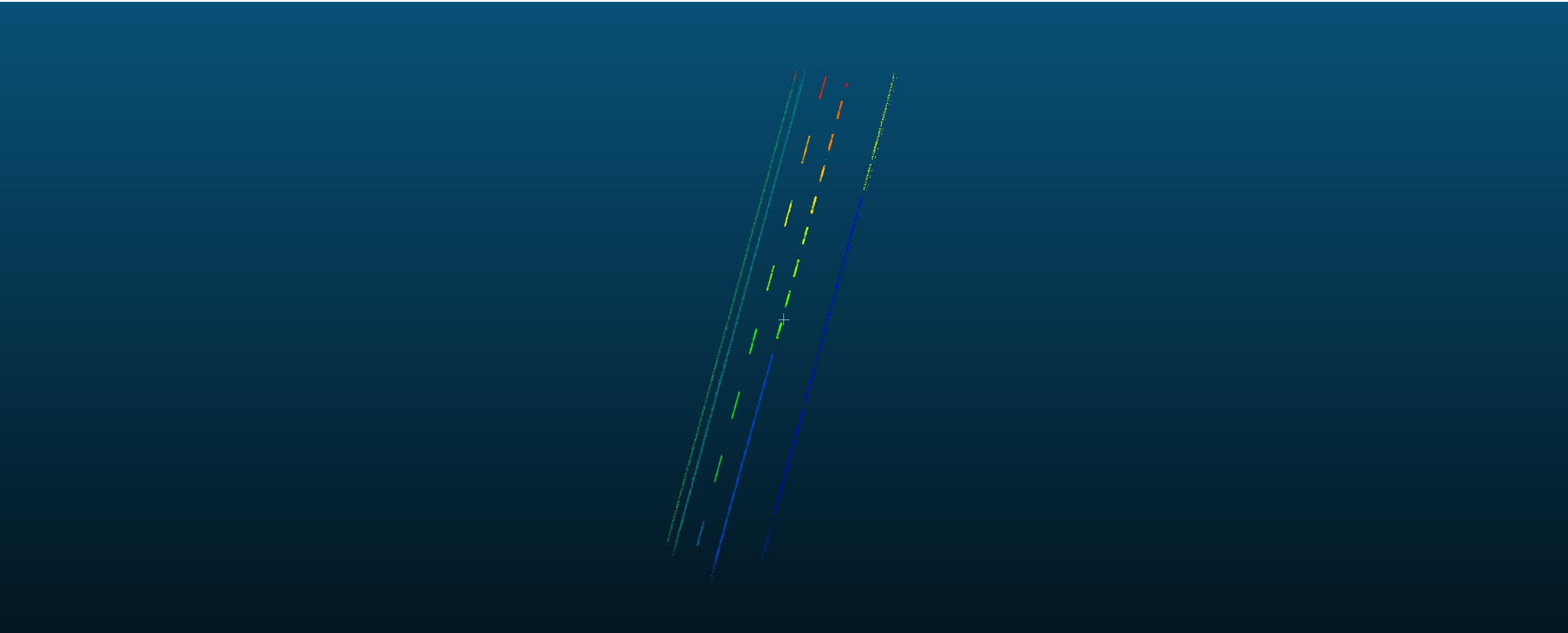
Filtering of Point cloud



Detection of Lane Marking

- So after filtering our pointcloud with Trajectory and Mean value filter, we can see the lane markings to a very well extent.
- However, the markings on the opposite side of the lane is also visible.
- So, in order to get rid of this problem, we used clustering using DBSCAN algorithm.
- This algorithm allows us to specify a number of points required to form a cluster and the distance separating them.
- Thus, we formed clusters of 40 points and the eps = 0.05
- Hence, 4073 more points we discarded leaving back 14423 points.

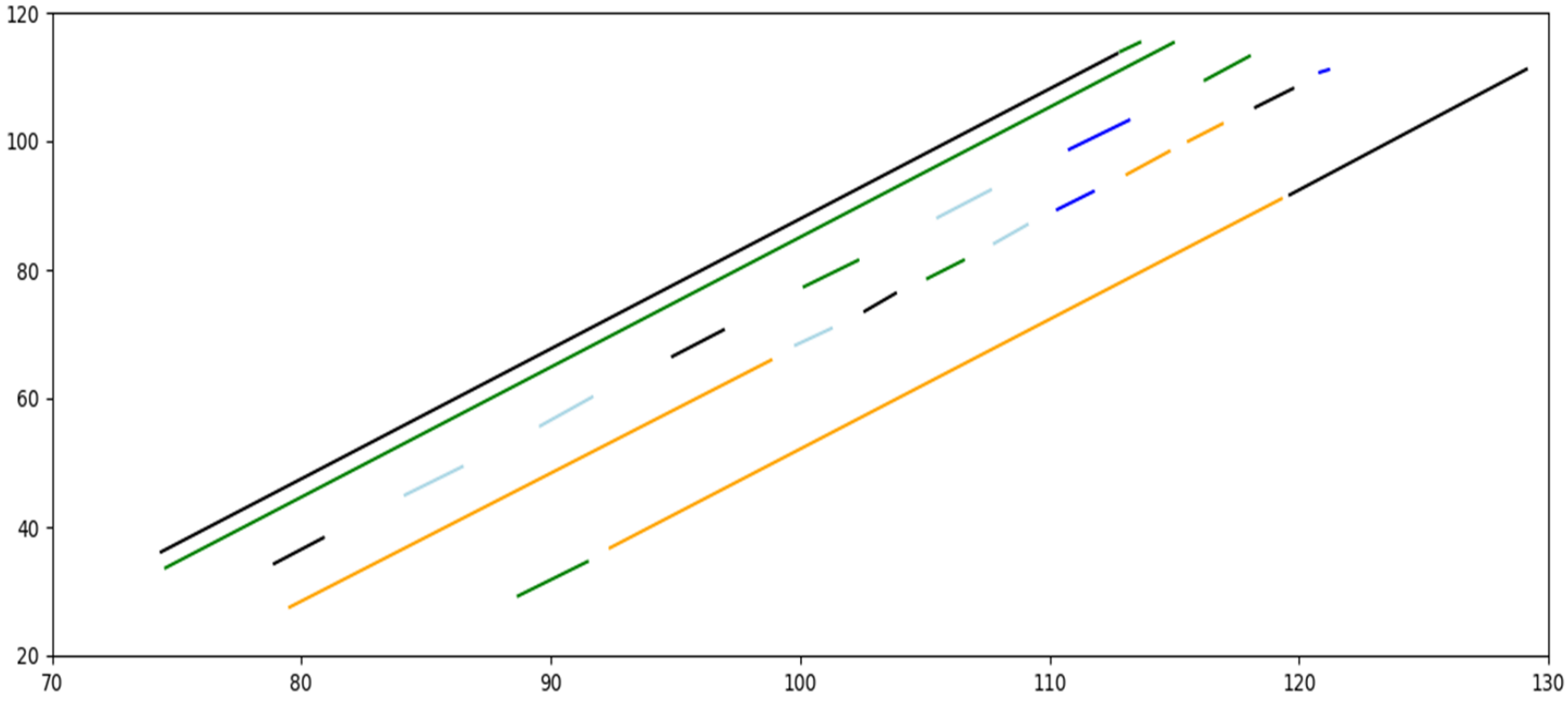
Detection of Lane Marking



Detection of Lane Marking

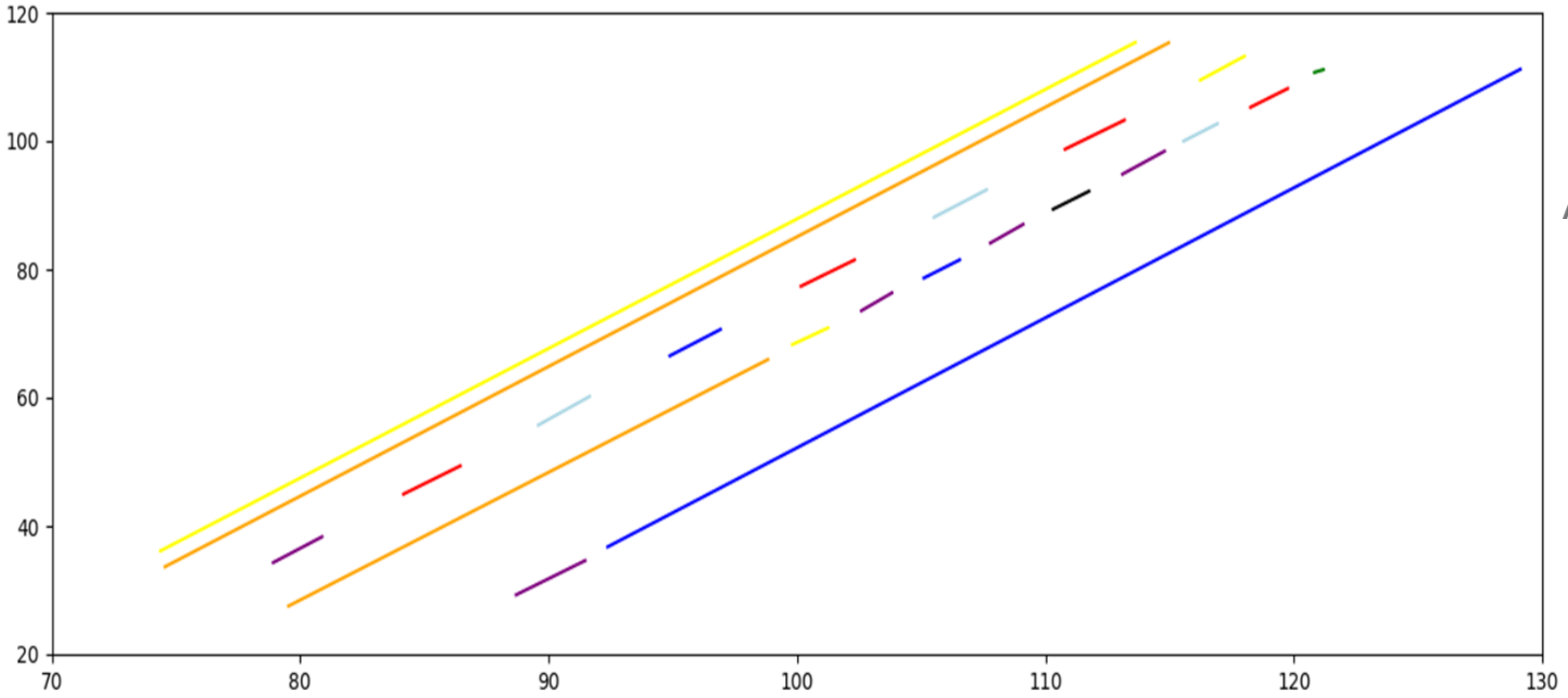
- The results from the clustering are quite impressive and we need to extract actual line segments out of it.
- To do so, we took 2 points that are furthest apart from each cluster (the starting and the end point)
- These two points serve as the starting and end point for a line segment.
- These line segments can easily be visualized using matplotlib.
- Also, we fused some of the lines that are quite close to each other.
- This allows us to increase the line length of some segments.

Detection of Lane Marking



Before fusing

Detection of Lane Marking



After fusing

Results

- The final step it to convert the UTM coordinates back to latitude and longitude coordinates.

- Also, we'd shifted all the coordinates closer to the centre of the coordinate system which also need to be shifted back before the conversion.

- Finally, the output is exported to a .csv file.

Results

Start_Latitude	Start_Longitude	Start_Z	End_Latitude	End_Longitude	End_Z
45.903632556743354	11.02820056281265	224.8436	45.90429371673364	11.02869840315925	224.8948
45.904334926738024	11.02850013301017	225.3589	45.90363105674842	11.027968482640023	225.2768
45.90361356674359	11.02818824280417	224.8351	45.90356614674433	11.028150622777892	224.8463
45.90389376674177	11.028293362872251	225.0411	45.903552596746884	11.028032062690166	225.0846
45.9036493667466	11.028053032702267	225.1685	45.9036136367471	11.028026322683687	225.1858
45.904334136737646	11.028517383023036	225.3273	45.90360872674833	11.027969942641917	225.2916
45.90374751674511	11.028128402754792	225.1545	45.90370846674573	11.028097402733138	225.1699
45.903843606743756	11.028198982803834	225.1669	45.9038038567443	11.028170792784294	225.1555
45.903936466742344	11.028270332853566	225.1669	45.903899816742914	11.028242582834244	225.1762
45.90393753674115	11.028326112895039	225.0599	45.90391441674152	11.02830670288143	225.0598
45.903986066740465	11.028361052919287	225.0715	45.90396130674078	11.02834424290767	225.0668

Conclusion & Future Work

- We've extracted the lane markings out of a given point cloud.
- However, our current algorithm assumes that the LIDAR data only represents a short segment of 20-30m.
- Thus this approach is not able to detect lanes that have a curvature.
- We could solve this problem by picking a second or third degree polynomial regression algorithm.
- This would give our algorithm enough freedom to pick straight lane as well as those with a curvature.

Source Code is available at:

https://github.com/chiragkhandhar/Object-detection-in-Point-Cloud-Lane-Marking

Thank You.