Point Cloud Lane Marking Detection

Geospatial Vision and Visualisation

Haoyu Wei
Tzu-jui Liu
Yuchao Wang

Overview

Objective
Detect lane marking from 3D Point Cloud

Code:
https://github.com/whywww/Point-Cloud-Lane-Marking-Detection/tree/master/code

Execution:
Requirements (tested on):

- pandas == 1.0.3
- numpy == 1.18
- matplotlib == 3.2.1
- scikit-learn == 0.23
- opencv-python == 4.2.0

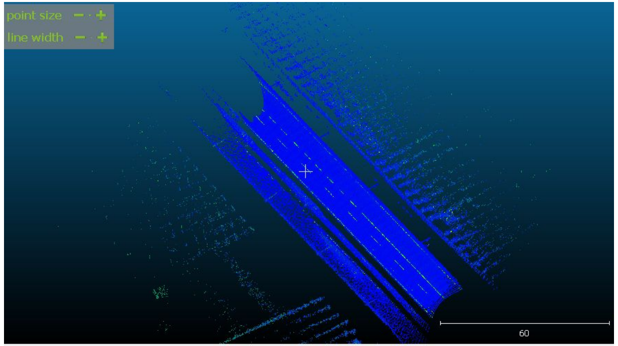
Run:
```
python3 code/load_data.py
python3 code/filtering.py
python3 code/link_edges.py
python3 code/visualize.py
python3 code/detect.py
```

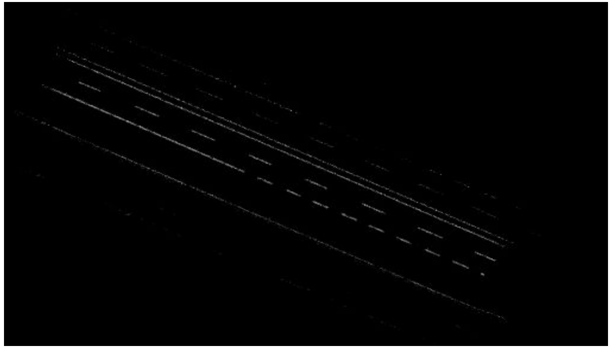
Methodology

- Visualization & Point Cloud Coordinate Conversion
- Point Cloud Filtering
- Point Cloud Lane Marking Detection
  - Hough Transform for line segments
  - DBSCAN Clustering

Visualization

- We tried several ways in visualization of the 3D point clouds.
  - Open Source Softwares:
    - CloudCompare
  - Projection into image:
    - we projected them on an image from a bird's eye view. With intensity as pixel values, and X, Y coordinates as pixel locations.
    - The benefit of this method is that it's fast and has little distortions, and it enables us to compare and check our results easily.





Points Coordinate Conversion

- Convert the latitude and longitude into UTM Coordinates

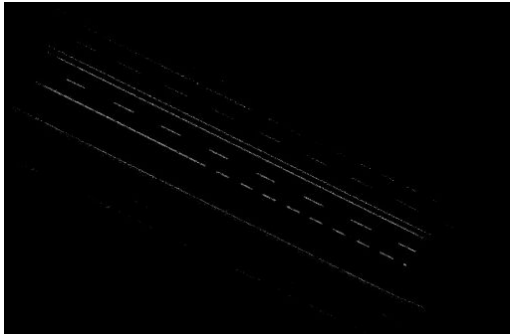
45.90388340	11.02841352	232.4648	10
45.90368343	11.02822054	234.4706	5
45.90368259	11.02822200	234.4459	7
45.90368197	11.02822381	234.4307	7
45.90368131	11.02822576	234.4232	7
45.90386824	11.02855032	225.3104	14
45.90386606	11.02855384	225.1985	14
45.90386725	11.02855644	223.9948	10
45.90384611	11.02853136	225.1121	12
45.90384208	11.02852814	225.0700	23
45.90384231	11.02853083	225.0273	24
45.90383643	11.02852828	226.0132	12
45.90383630	11.02853041	225.9743	12
45.90383566	11.02853178	225.9344	12
45.90383447	11.02853167	225.9041	24
45.90383347	11.02853195	225.8721	12
45.90383806	11.02854444	225.7867	13
45.90383699	11.02854478	225.7515	13
45.90367422	11.02824389	233.8122	7

Convert

108.226320862188	65.04354738630354	232.4648	10
93.82292302499991	42.445589639246464	234.4706	5
93.93854165216908	42.35514342505485	234.4459	7
94.08068640588317	42.28983030188829	234.4307	7
94.2338033040287	42.220349254086614	234.4232	7
118.8800314438995	63.62913292646408	225.3104	14
119.15922072681133	63.39387383311987	225.1985	14
119.35752612259239	63.53121621068567	223.9948	10
117.47194509580731	61.13301740027964	225.1121	12
117.23357563104946	60.67891965061426	225.07	23
117.44157510250807	60.709780778735876	225.0273	24
117.26040245627519	60.05146465357393	226.0132	12
117.42598303535488	60.04122384916991	225.9743	12
117.53405565023422	59.97282107267529	225.9344	12
117.52888663718477	59.84039162285626	225.9041	24
117.55343095678836	59.729841203428805	225.8721	12
118.50924414210021	60.26444671489298	225.7867	13
118.53864020260517	60.14623748045415	225.7515	13
95.66009291191585	41.46839587204158	233.8122	7

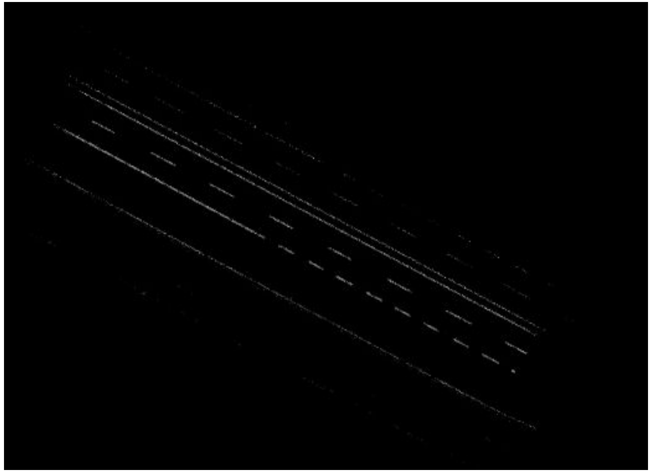
Points Filtering

1. By observing the dataset, we noticed that we can exploit some useful information to remove undesired points and improve the accuracy of our algorithm.
2. We noticed that lanes on the road have higher intensity because they are usually painted with light color like white or yellow. Hence, for the first filter, we use the mean and standard deviation of the intensity as filter, and we remove points with intensity lower than mean+1*std.
3. For the second filter, we use distance to trajectory as filter. After experiment, we found that removing points that are more than 20 meters away from the trajectory can give us decent results. Therefore, we set the threshold equal to 20 meters.
4. For the third filter, we use the elevation as filter, because lanes are on road surface, and thus will not have very high elevation. We calculate the mean of elevation of the remaining points and remove points that are above the mean.
5. We eventually get around 17,000 points.

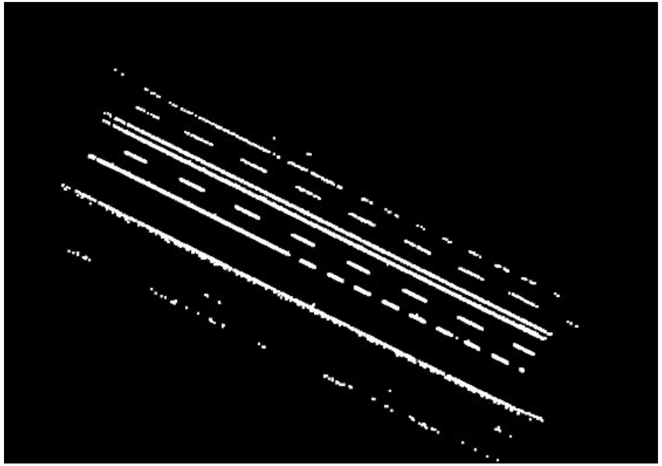


Point Cloud Lane Marking Detection

- Dilate the image of filtered point cloud

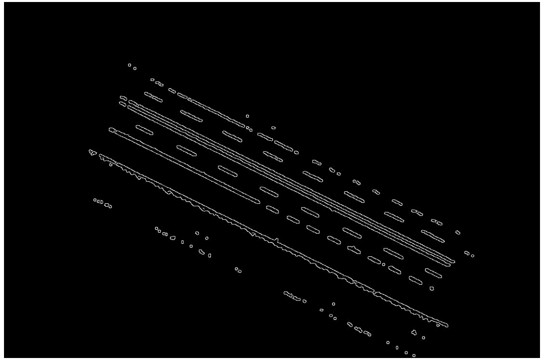


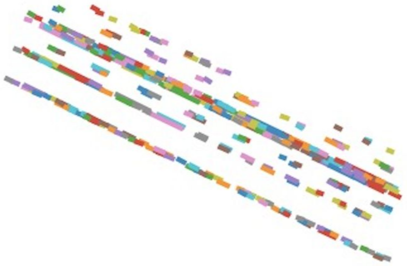
Dilate

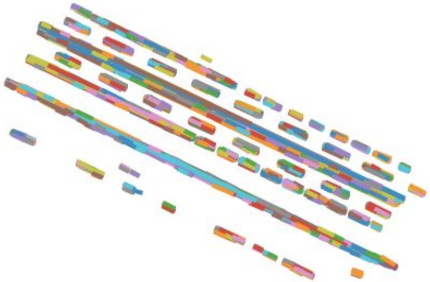


Point Cloud Lane Marking Detection

- Hough Transform:
  - Edge Detection
    - Since the point clouds are very fuzzy on the edge areas, we first used Gaussian Blurring and Median Blurring to smooth the edges, and then used Canny edge detector to detect the edges.
  - Hough Transform
    - We found that each lane marking has two edges due to its width. So we turned back to the original lane markings for detection and produced better result.
    - We used Hough Line Segment Detector instead of Line Detector for our task because line markings are often discontinuous even on the same line. But the cons are the transformation seems very sensitive to small discontinuity thus producing a lot of small segments.

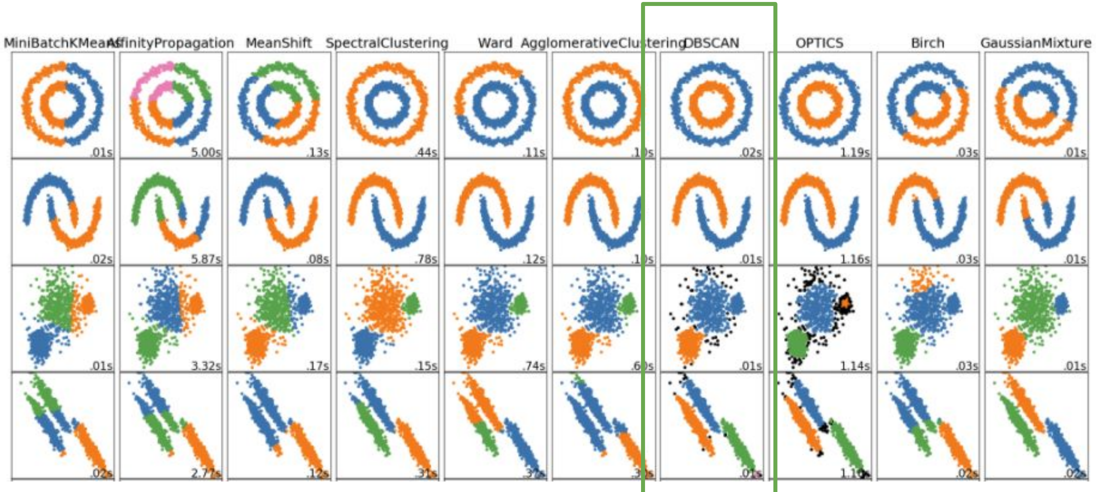




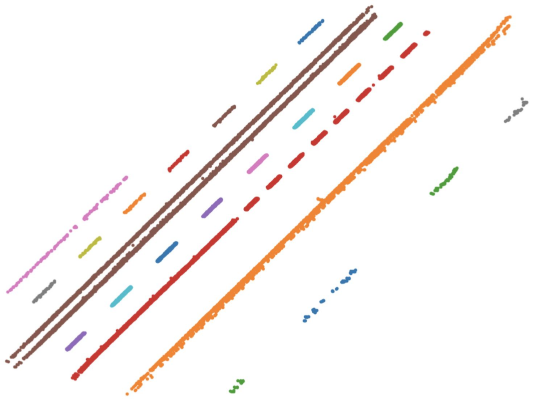


Point Cloud Lane Marking Detection

- DBSCAN Clustering:
  - Density-based spatial clustering of applications with noise (DBSCAN) is a clustering method that measures distance with nearest neighbors. So it's more suitable for lane marking clustering task.

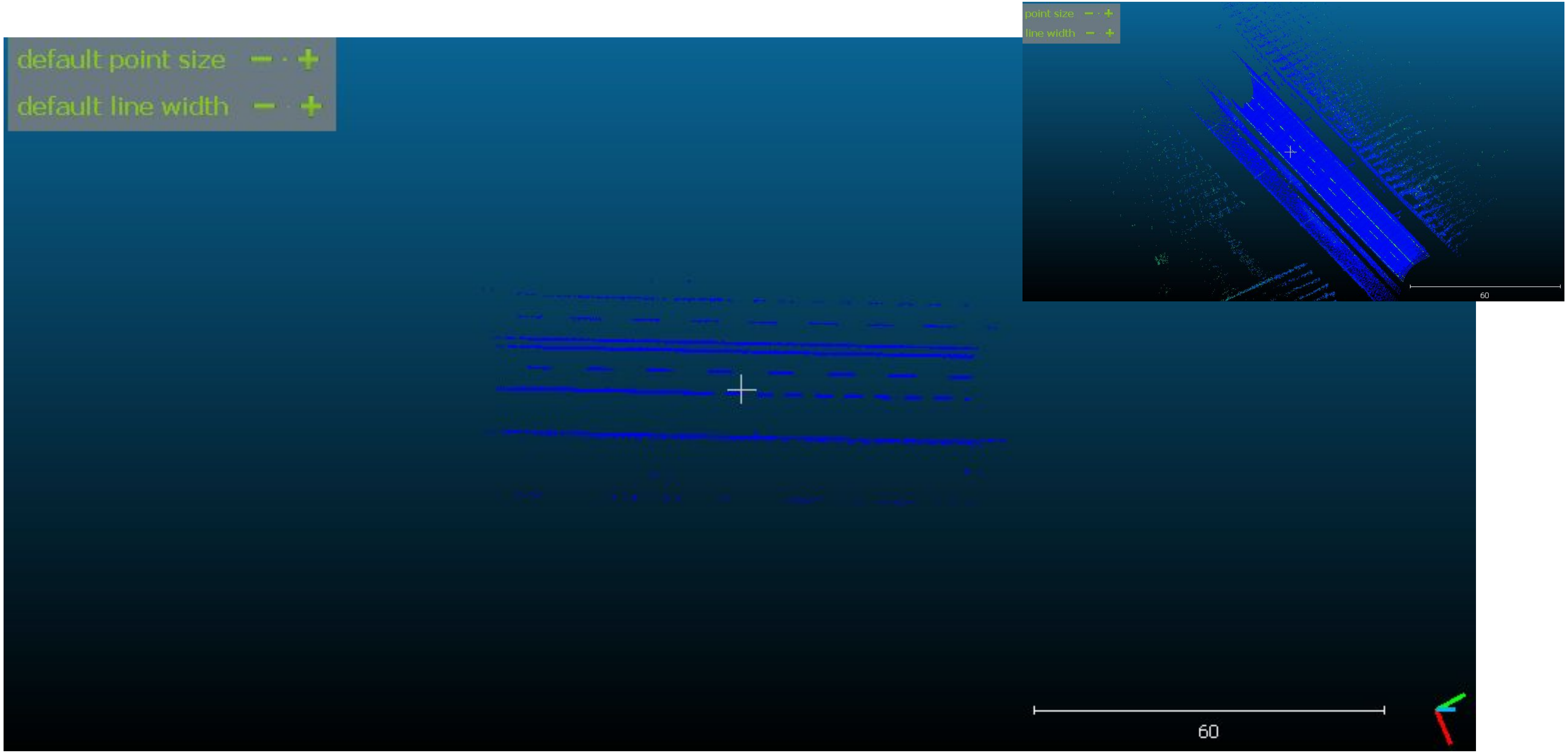


Comparison of different clustering methods



Our result

Point Cloud Results



Conclusion & Future Work

- **Conclusion**:

  We are able to detect highly accurate lane marking from points cloud. Also, considering the lane marking may not be a straight line, we use Hough transform and DBSCAN clustering to detect the curve from points cloud.

- **Future Work**:

  The clustering method now works well for both lines and curves. But in order to quantify our result, we further specified strip areas for each lane marking with line equations. However, this limits our method to lines only. So we want to generalize our approach to quantify more shapes in the future.

Reference

[1] https://scikit-learn.org/stable/modules/clustering.html

[2] https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_houghlines/py_houghlines.html