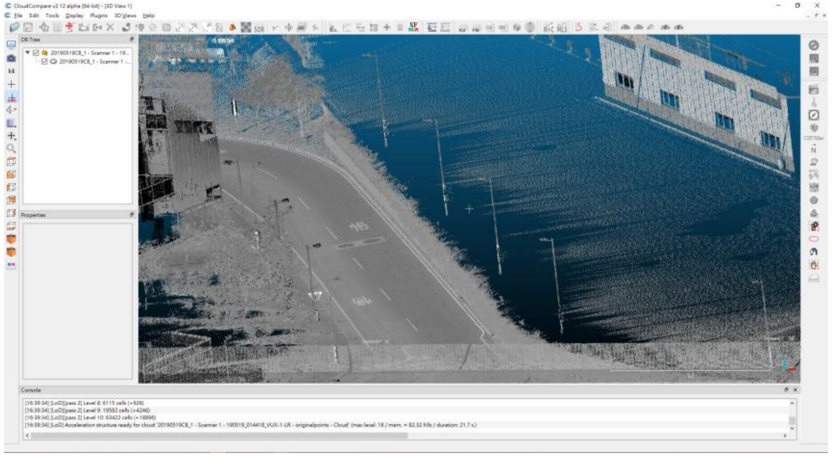
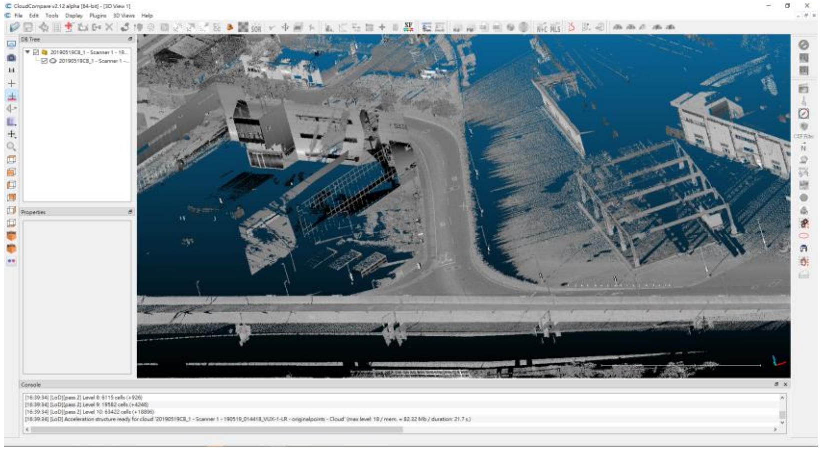
Point Cloud Dataset – las file

- Las files store info from lidar and radar
- Existing outliers and blurrings in Raw data
- Greyscale



CloudCompare view



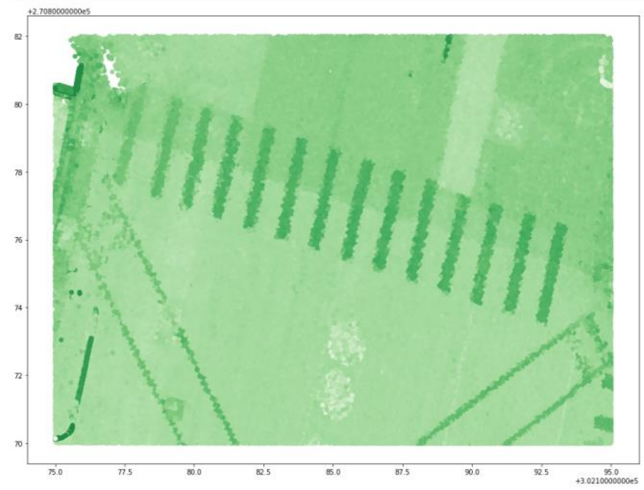
CloudCompare view

Point Cloud Dataset – las file

- Load .las file with python module laspy
- las_file.x : latitude, epsg3826
- las_file.y : longitude, epsg3826
- las_file.intensity : strength of lidar signal
- Sequential data instead of grid (image)

```
fig = plt.figure(figsize=(16, 12))
ax = plt.axes()
ax.scatter(las_file.x[dense_idx], las_file.y[dense_idx], c=las_file.intensity[dense_idx], cmap='Greens')
```

```
<matplotlib.collections.PathCollection at 0x7f4ab8242be0>
```




```
las_file = laspy.file.File('../dataset/BS2001_shalun/ground/groundpt09.las', mode='rw')
```

```
las_file.x.shape, las_file.y.shape, las_file.intensity.shape
```

```
((29989809,), (29989809,), (29989809,))
```

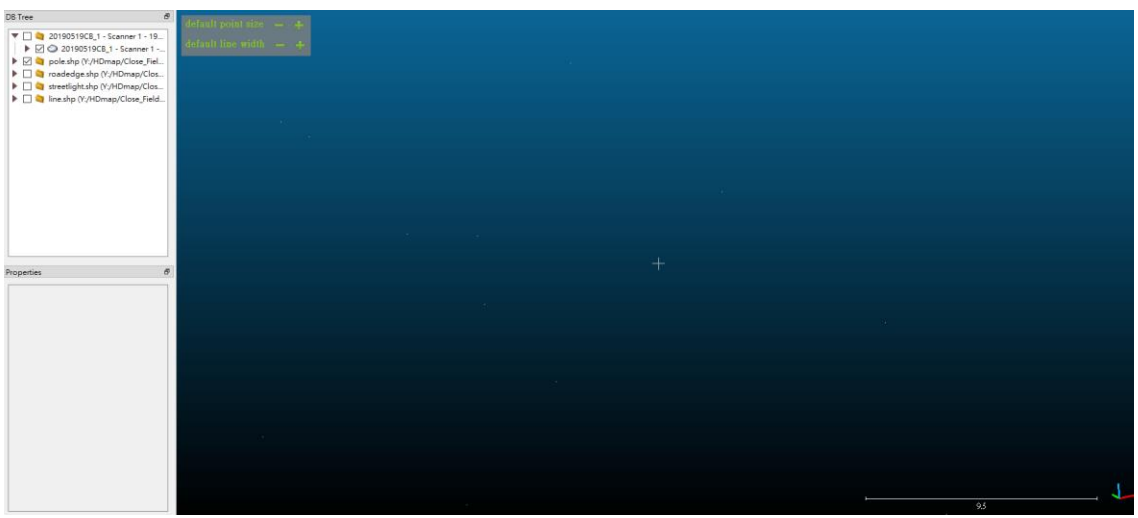
```
las_file.x, las_file.y, las_file.intensity
```

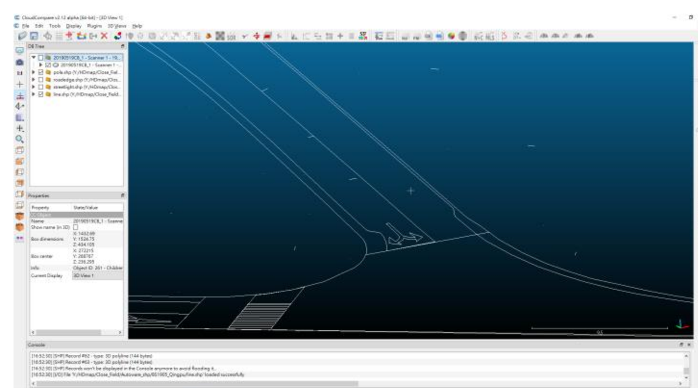
```
(array([177033.541, 177033.552, 177033.547, ..., 177032.998, 177033.304,
        177033.222]),
 array([2535879.01 , 2535879.042, 2535879.49 , ..., 2535880.205,
        2535880.042, 2535880.125]),
 array([ 8748,  9699, 14941, ..., 27754, 27754,     0], dtype=uint16))
```

Point Cloud Dataset – shp file

- Consist of points
- Latitude and longitude
- High precision

LaneCenterLine.shp	2020/9/22 上午 02:23	SHP 檔案	11,138 KB
LaneLine.shp	2020/9/22 上午 02:06	SHP 檔案	653 KB
MarkArea.shp	2020/9/22 上午 12:27	SHP 檔案	76 KB
MarkGraph.shp	2020/9/22 上午 12:26	SHP 檔案	1,140 KB
MarkLine.shp	2020/9/22 上午 02:16	SHP 檔案	110 KB
Node.shp	2020/9/23 上午 12:08	SHP 檔案	58 KB
Object.shp	2020/9/22 下午 11:59	SHP 檔案	989 KB
Parking.shp	2020/9/3 上午 11:56	SHP 檔案	7 KB
Pole.shp	2020/9/22 上午 12:15	SHP 檔案	13 KB
ReferenceLine.shp	2020/9/23 上午 12:08	SHP 檔案	301 KB
RoadEdge.shp	2020/9/22 上午 01:57	SHP 檔案	175 KB
sign.shp	2020/9/22 上午 12:17	SHP 檔案	8 KB
Signal.shp	2020/9/22 上午 02:35	SHP 檔案	5 KB
SignalData.shp	2020/9/24 上午 12:19	SHP 檔案	31 KB
StopLine.shp	2020/9/22 上午 02:35	SHP 檔案	9 KB
WayPoint.shp	2020/9/22 上午 02:23	SHP 檔案	2,723 KB

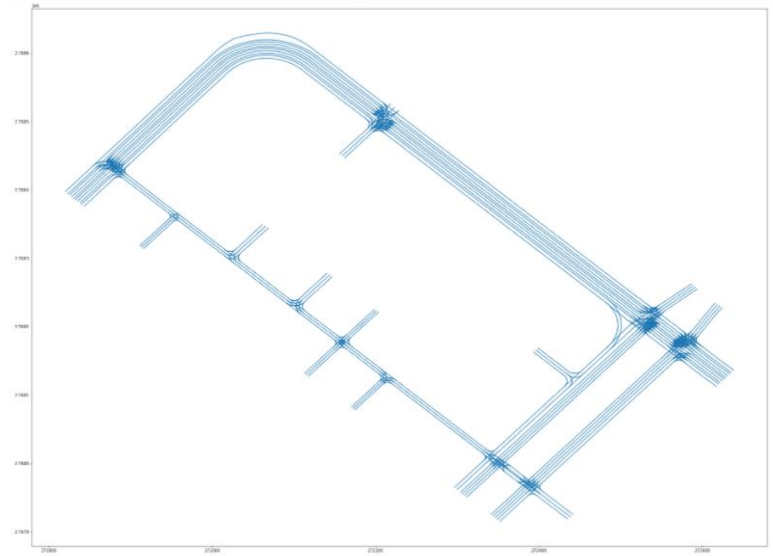




Point Cloud Dataset – shp file

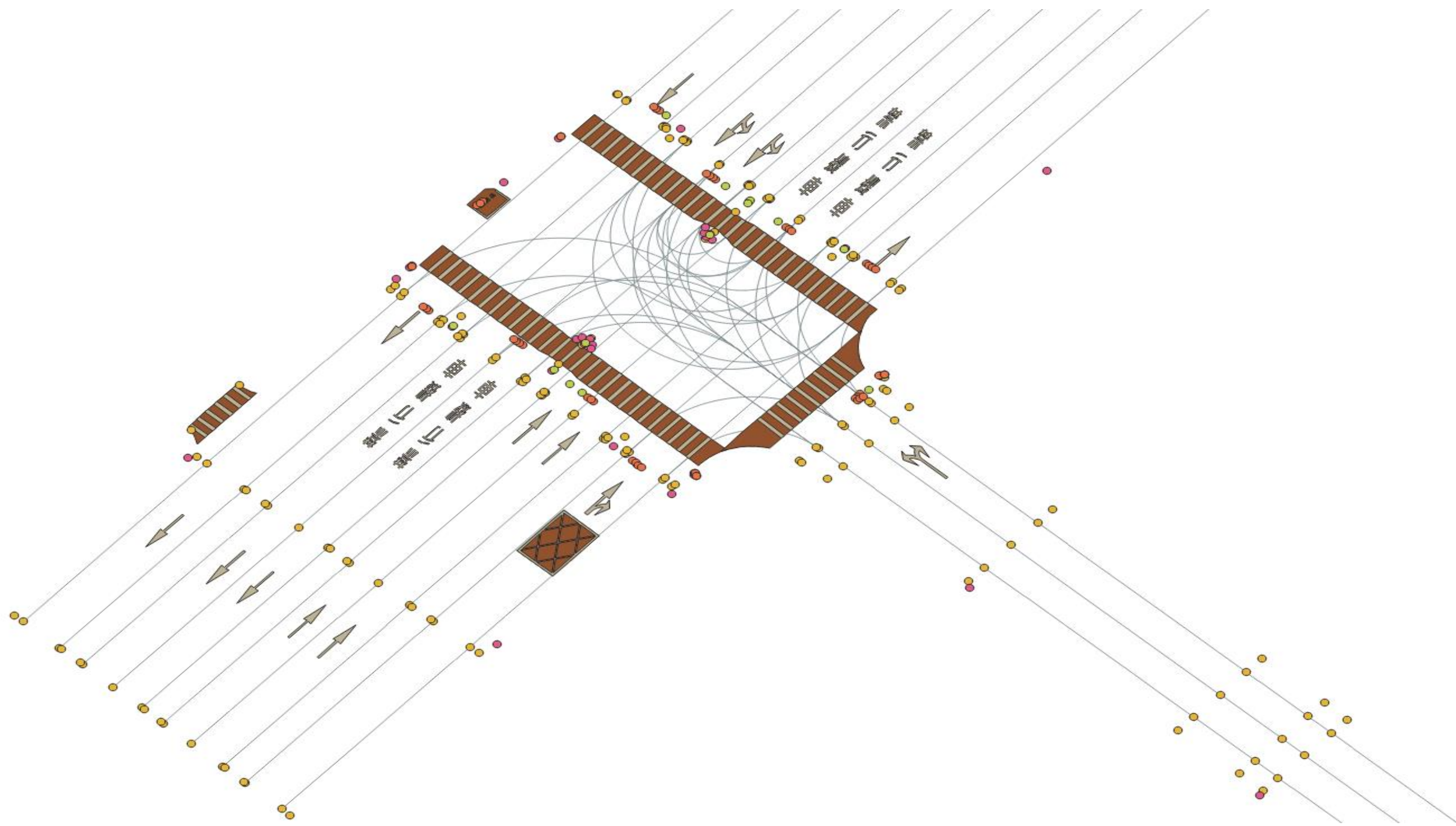
- You could have full code at SHP/demo.ipynb

```
In [3]: line_shp = fiona.open("LaneLine.shp")
        fig = plt.figure(figsize=(32, 24))
        ax = plt.axes()
        ax.plot()
        for shp in line_shp:
            if shp is None or shp['geometry'] is None:
                continue

            # shp['geometry']['coordinates'] might get multiple points
            targets = shp['geometry']['coordinates']
            for idx in range(len(targets)):
                if idx == len(targets) - 1:
                    continue
                point1, point2 = targets[idx], targets[idx + 1]

                ax.add_line(Line2D((point1[0], point2[0]), (point1[1], point2[1])))
```



Point Cloud Dataset – visualization of shp file

- Visualize shp files with QGis

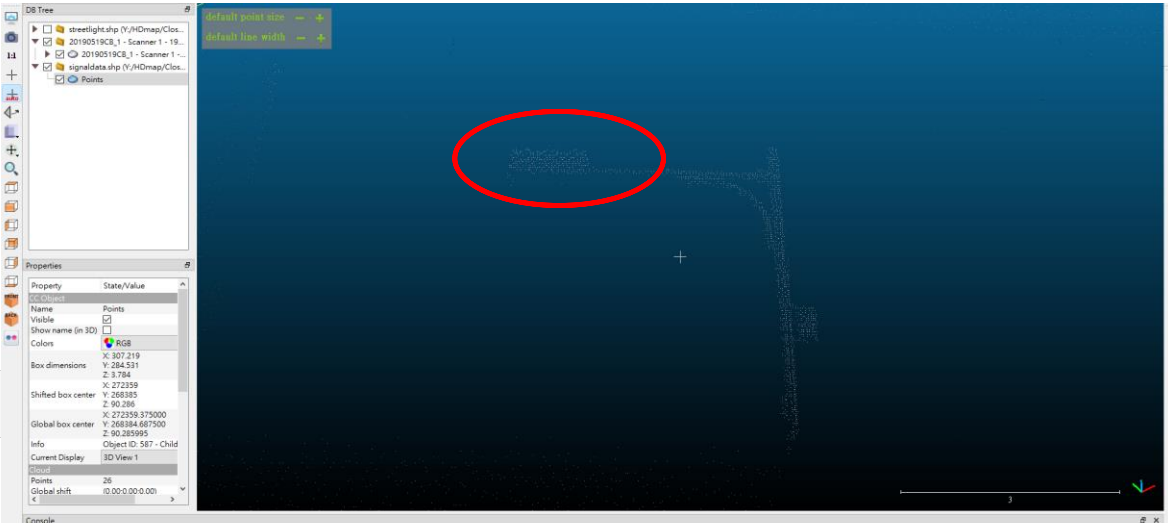


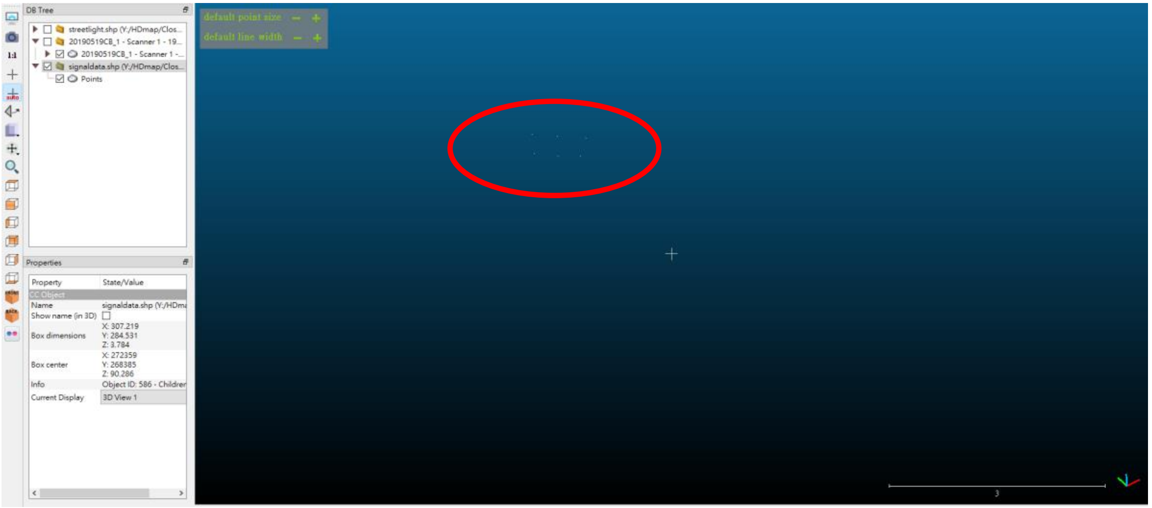
Point Cloud Dataset – Qingpu shp example

- Here is a reference with shp geometric type and its name

Term	Chinese Description	Object	SHP type	Autoware SHP Name	Necessary for Autoware
1	車道線	3 標線	line	whiteline	
2	標誌、號誌、燈面	X	point	vector	
3	停止線	3 標線	line	stopline	
4	燈面	9 燈面	point	signaldata	
5	標誌	7 標誌	point	roadsign	
6	道路邊界(路緣石)	1 道路	line	roadedge	
7	道路標線	3 標線	area	road_surface_mark	
8	桿(標誌、號誌)	10 桿	point	pole	
9	所有點	X	point	point	V
10	車道中心節點(ＩＤ)	2 車道	point	node	V
11	所有線	X	line	line	V
12	車道中心線	2 車道	line	lane	V
13	車道中心節點(空間)	2 車道	point	dtlane	V
14	停等區	4 物體	area	driveon_portion	
15	行人穿越道	3 標線	area	crosswalk	
16	所有面	X	area	area	

- signaldata.shp





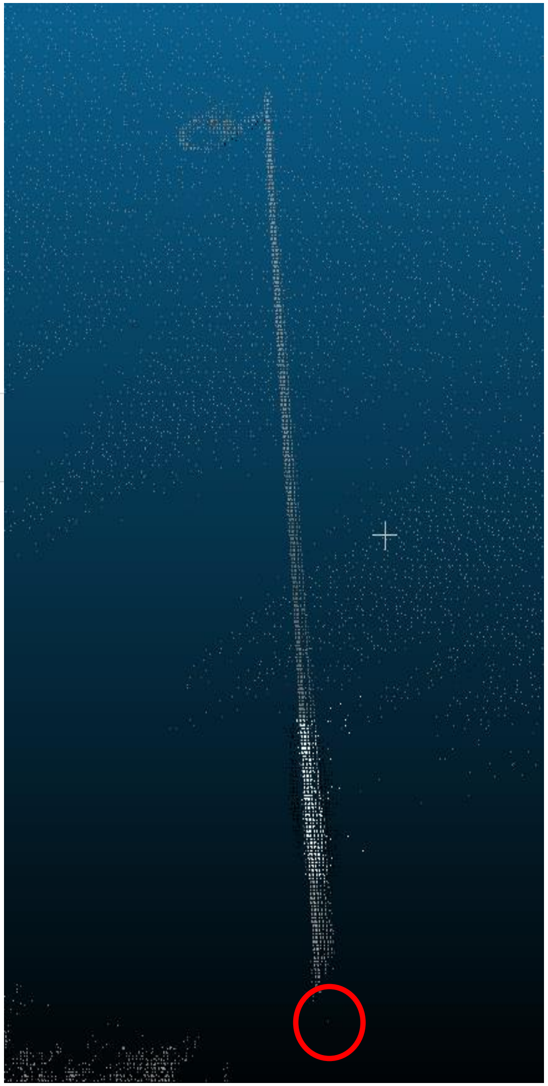
```
In [13]: signal_shp = Reader('../../Databox/HDmap/Close_Field/Autoware_shp/BS1905_Qingpu/signaldata.shp')
         print('Number of signal: ', len(signal_shp))
         for shp in signal_shp.shapes():
             print(shp.__dict__)
```

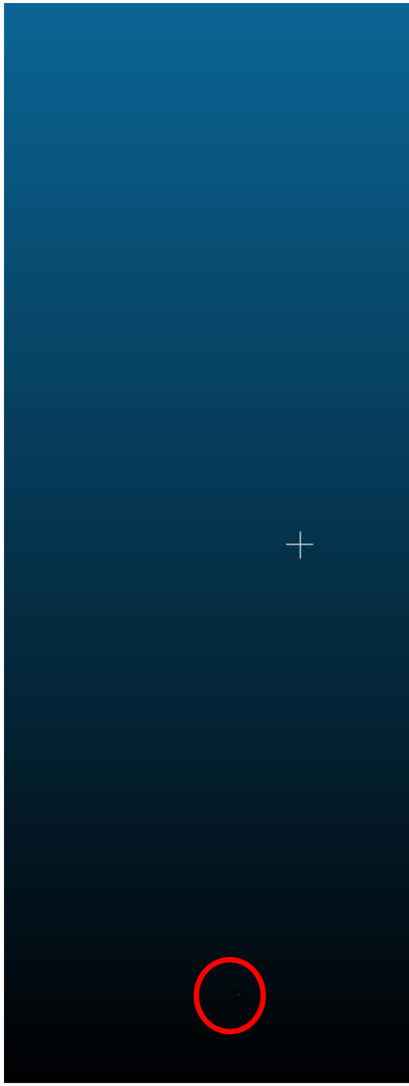
```
Number of signal:  26
{'shapeType': 11, 'points': [[272356.031, 268395.267]], 'parts': [], 'z': [90.259], 'm': [None]}
{'shapeType': 11, 'points': [[272356.283, 268395.016]], 'parts': [], 'z': [90.263], 'm': [None]}
{'shapeType': 11, 'points': [[272356.569, 268394.775]], 'parts': [], 'z': [90.269], 'm': [None]}
{'shapeType': 11, 'points': [[272352.453, 268392.538]], 'parts': [], 'z': [90.63], 'm': [None]}
{'shapeType': 11, 'points': [[272352.714, 268392.783]], 'parts': [], 'z': [90.63], 'm': [None]}
{'shapeType': 11, 'points': [[272353.005, 268393.058]], 'parts': [], 'z': [90.63], 'm': [None]}
{'shapeType': 11, 'points': [[272512.971, 268242.44]], 'parts': [], 'z': [92.178], 'm': [None]}
{'shapeType': 11, 'points': [[272512.971, 268242.44]], 'parts': [], 'z': [91.834], 'm': [None]}
{'shapeType': 11, 'points': [[272512.971, 268242.44]], 'parts': [], 'z': [91.458], 'm': [None]}
{'shapeType': 11, 'points': [[272233.362, 268500.728]], 'parts': [], 'z': [90.169], 'm': [None]}
{'shapeType': 11, 'points': [[272233.601, 268500.971]], 'parts': [], 'z': [90.169], 'm': [None]}
{'shapeType': 11, 'points': [[272233.867, 268501.241]], 'parts': [], 'z': [90.169], 'm': [None]}
{'shapeType': 11, 'points': [[272234.11, 268501.489]], 'parts': [], 'z': [90.169], 'm': [None]}
{'shapeType': 11, 'points': [[272205.742, 268526.207]], 'parts': [], 'z': [90.043], 'm': [None]}
{'shapeType': 11, 'points': [[272206.012, 268526.494]], 'parts': [], 'z': [90.043], 'm': [None]}
{'shapeType': 11, 'points': [[272206.255, 268526.751]], 'parts': [], 'z': [90.043], 'm': [None]}
{'shapeType': 11, 'points': [[272206.462, 268526.97]], 'parts': [], 'z': [90.043], 'm': [None]}
{'shapeType': 11, 'points': [[272227.589, 268519.943]], 'parts': [], 'z': [89.729], 'm': [None]}
{'shapeType': 11, 'points': [[272227.841, 268519.719]], 'parts': [], 'z': [89.734], 'm': [None]}
{'shapeType': 11, 'points': [[272228.103, 268519.497]], 'parts': [], 'z': [89.738], 'm': [None]}
{'shapeType': 11, 'points': [[272228.381, 268519.776]], 'parts': [], 'z': [89.738], 'm': [None]}
{'shapeType': 11, 'points': [[272228.093, 268520.048]], 'parts': [], 'z': [89.733], 'm': [None]}
{'shapeType': 11, 'points': [[272227.805, 268520.32]], 'parts': [], 'z': [89.729], 'm': [None]}
{'shapeType': 11, 'points': [[272350.774, 268391.015]], 'parts': [], 'z': [89.1], 'm': [None]}
{'shapeType': 11, 'points': [[272350.774, 268391.015]], 'parts': [], 'z': [88.744], 'm': [None]}
{'shapeType': 11, 'points': [[272350.774, 268391.015]], 'parts': [], 'z': [88.394], 'm': [None]}
```

- pole.shp

```
In [10]: pole_shp = Reader('../../Databox/HDmap/Close_Field/Autoware_shp/BS1905_Qingpu/pole.shp')
         print('Number of pole: ', len(pole_shp))
         for shp in pole_shp.shapes():
             print(shp.__dict__)
```

```
Number of pole:  171
{'shapeType': 11, 'points': [[272029.517, 268874.163]], 'parts': [], 'z': [87.625], 'm': [None]}
{'shapeType': 11, 'points': [[272230.28, 268572.693]], 'parts': [], 'z': [87.946], 'm': [None]}
{'shapeType': 11, 'points': [[272248.356, 268556.249]], 'parts': [], 'z': [87.708], 'm': [None]}
{'shapeType': 11, 'points': [[272263.455, 268530.752]], 'parts': [], 'z': [87.658], 'm': [None]}
{'shapeType': 11, 'points': [[272354.066, 268397.001]], 'parts': [], 'z': [87.448], 'm': [None]}
{'shapeType': 11, 'points': [[272400.452, 268355.159]], 'parts': [], 'z': [87.889], 'm': [None]}
{'shapeType': 11, 'points': [[272513.716, 268241.596]], 'parts': [], 'z': [89.092], 'm': [None]}
{'shapeType': 11, 'points': [[272540.577, 268263.705]], 'parts': [], 'z': [89.318], 'm': [None]}
{'shapeType': 11, 'points': [[272524.577, 268241.756]], 'parts': [], 'z': [89.765], 'm': [None]}
{'shapeType': 11, 'points': [[271972.156, 268867.459]], 'parts': [], 'z': [87.524], 'm': [None]}
{'shapeType': 11, 'points': [[272056.534, 268758.462]], 'parts': [], 'z': [87.493], 'm': [None]}
{'shapeType': 11, 'points': [[272358.823, 268393.167]], 'parts': [], 'z': [84.798], 'm': [None]}
{'shapeType': 11, 'points': [[272350.43, 268390.682]], 'parts': [], 'z': [85.0], 'm': [None]}
{'shapeType': 11, 'points': [[272512.365, 268242.738]], 'parts': [], 'z': [87.247], 'm': [None]}
{'shapeType': 11, 'points': [[272231.388, 268499.187]], 'parts': [], 'z': [84.471], 'm': [None]}
{'shapeType': 11, 'points': [[272203.939, 268524.599]], 'parts': [], 'z': [84.403], 'm': [None]}
{'shapeType': 11, 'points': [[272230.314, 268517.798]], 'parts': [], 'z': [84.237], 'm': [None]}
{'shapeType': 11, 'points': [[271964.401, 268993.813]], 'parts': [], 'z': [85.298], 'm': [None]}
{'shapeType': 11, 'points': [[271986.238, 268994.356]], 'parts': [], 'z': [85.298], 'm': [None]}
{'shapeType': 11, 'points': [[272001.836, 268977.491]], 'parts': [], 'z': [85.082], 'm': [None]}
{'shapeType': 11, 'points': [[272015.11, 268960.48]], 'parts': [], 'z': [85.058], 'm': [None]}
{'shapeType': 11, 'points': [[272042.268, 268926.021]], 'parts': [], 'z': [85.107], 'm': [None]}
{'shapeType': 11, 'points': [[272048.851, 268905.323]], 'parts': [], 'z': [85.12], 'm': [None]}
{'shapeType': 11, 'points': [[272030.618, 268887.365]], 'parts': [], 'z': [85.109], 'm': [None]}
{'shapeType': 11, 'points': [[272031.264, 268872.814]], 'parts': [], 'z': [85.272], 'm': [None]}
{'shapeType': 11, 'points': [[272045.529, 268853.958]], 'parts': [], 'z': [85.215], 'm': [None]}
{'shapeType': 11, 'points': [[272058.779, 268836.927]], 'parts': [], 'z': [85.376], 'm': [None]}
{'shapeType': 11, 'points': [[272070.913, 268818.932]], 'parts': [], 'z': [85.35], 'm': [None]}
{'shapeType': 11, 'points': [[272074.909, 268801.678]], 'parts': [], 'z': [86.197], 'm': [None]}
```





Point Cloud Dataset – task

- Given point cloud and corresponding shp files, find attributes of every points (classification, cluster etc.).

- You would have only shp files, no groundtruth (attributes of every points).

- Possible solutions:

  1. With certain algorithms, directly convert shp to attributes, and mapping to point clouds(deterministic).

  2. Use Machine Learning models, feed shp and point cloud to output attributes, like clustering (unsupervised).

  3. Train Deep Learning models, input point cloud and let shp files as semi-groundtruth(semi-supervised).