

- For [NuScene](#) dataset access, you may need to register on that website. To save time, you can download only the Full dataset/Mini set.

Mini ▾

Subset of trainval, 10 scenes, used to explore the data without downloading the whole dataset.

Metadata and sensor file blobs [US, Asia] 3.88 GB (4167696325 Bytes) md5: 791dd9ced556cfa1b425682f177b5d9b

- I download Full NuScene dataset/Mini set from the “[NuScene](#)” website into my documents and created a separate folder for all NuScene datasets.

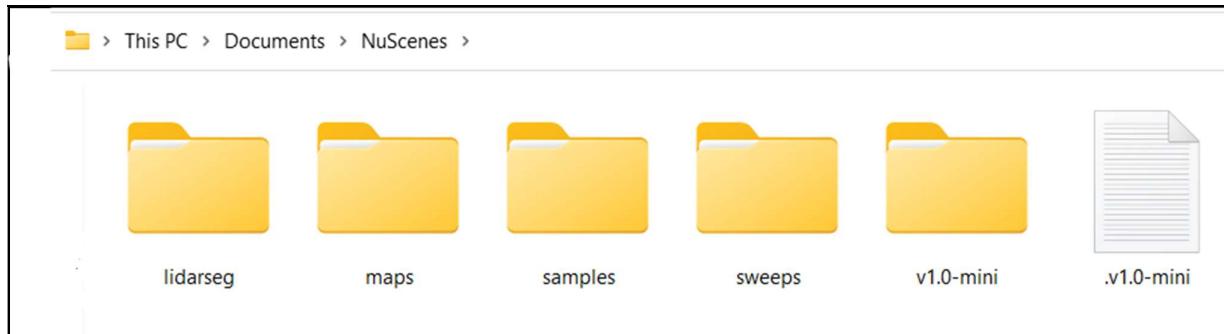


Figure 1

Path-Link: C:\Users\siddh\OneDrive\Pictures\Screenshots\Documents\NuScenes

- If you use Python, set up the NuScene develop kit locally, you may need to install Anaconda and Jupyter notebook; If you use Matlab, setup your Matlab for this data process.

- I installed Anaconda and jupyter Notebook in my laptop to setup the NuScene develop kit locally.

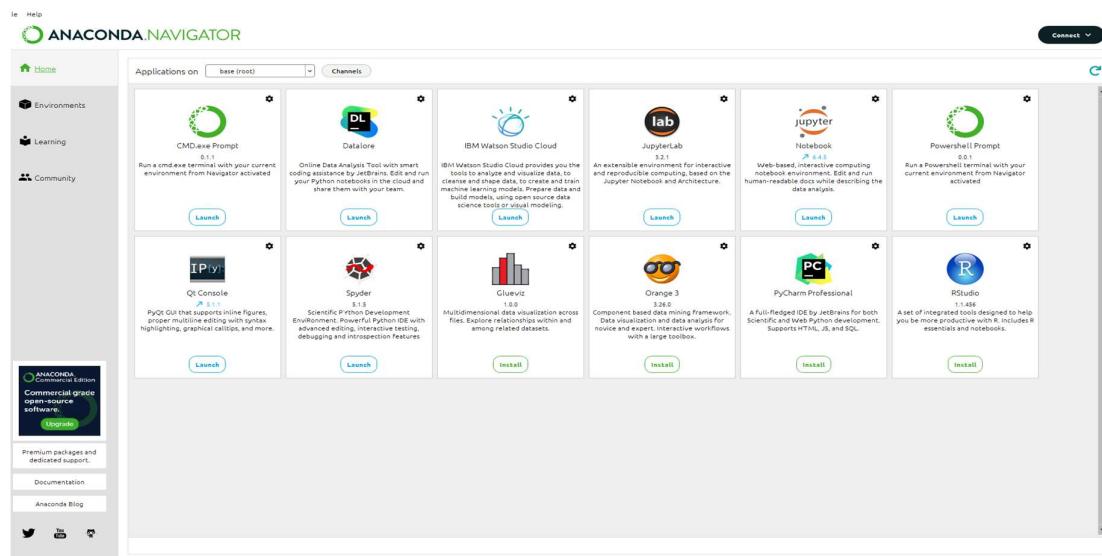


Figure 2

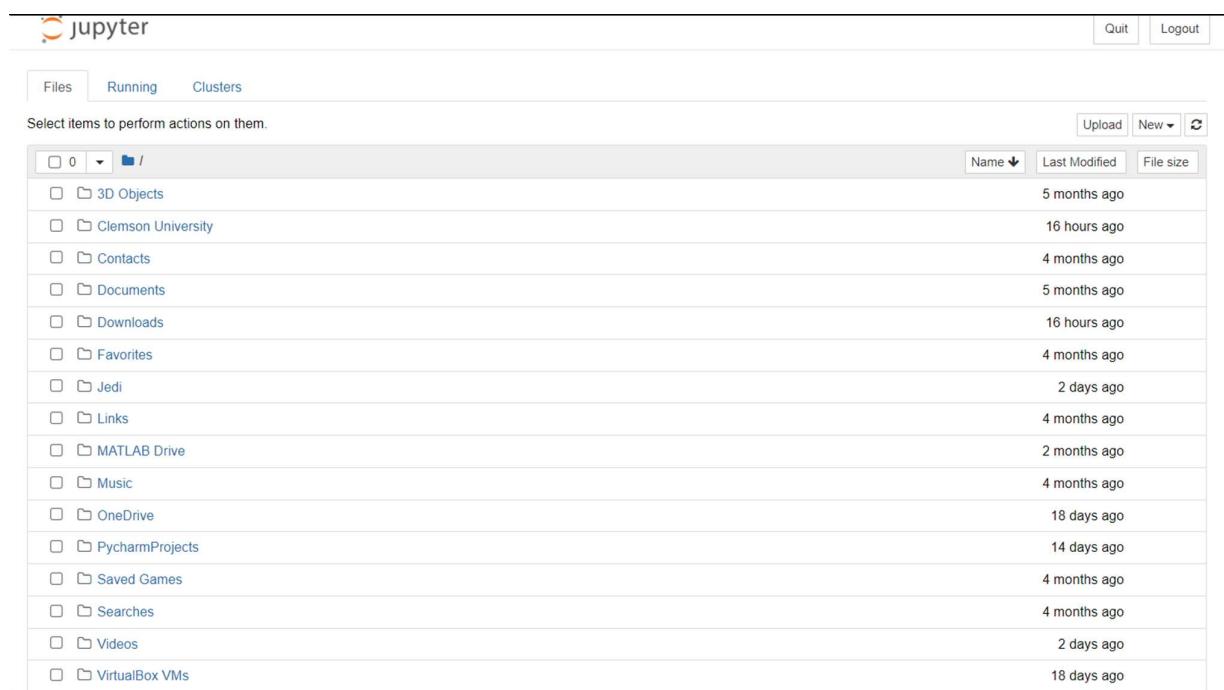


Figure 3

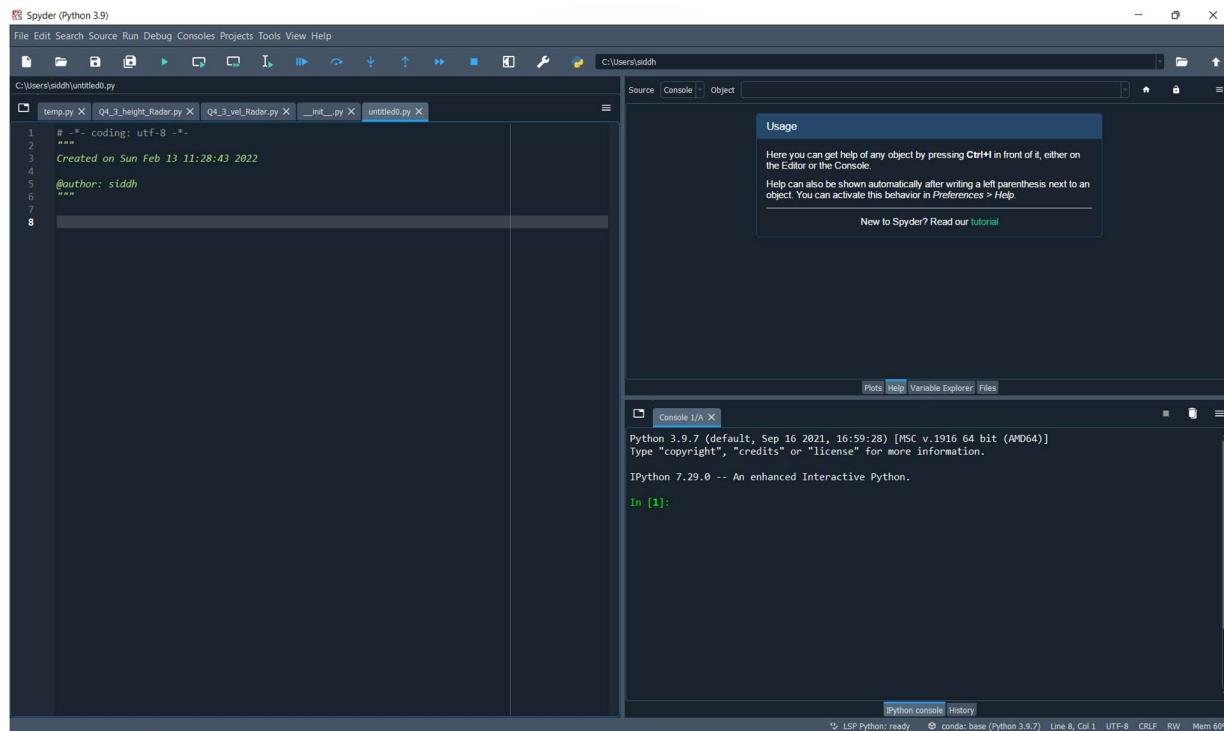
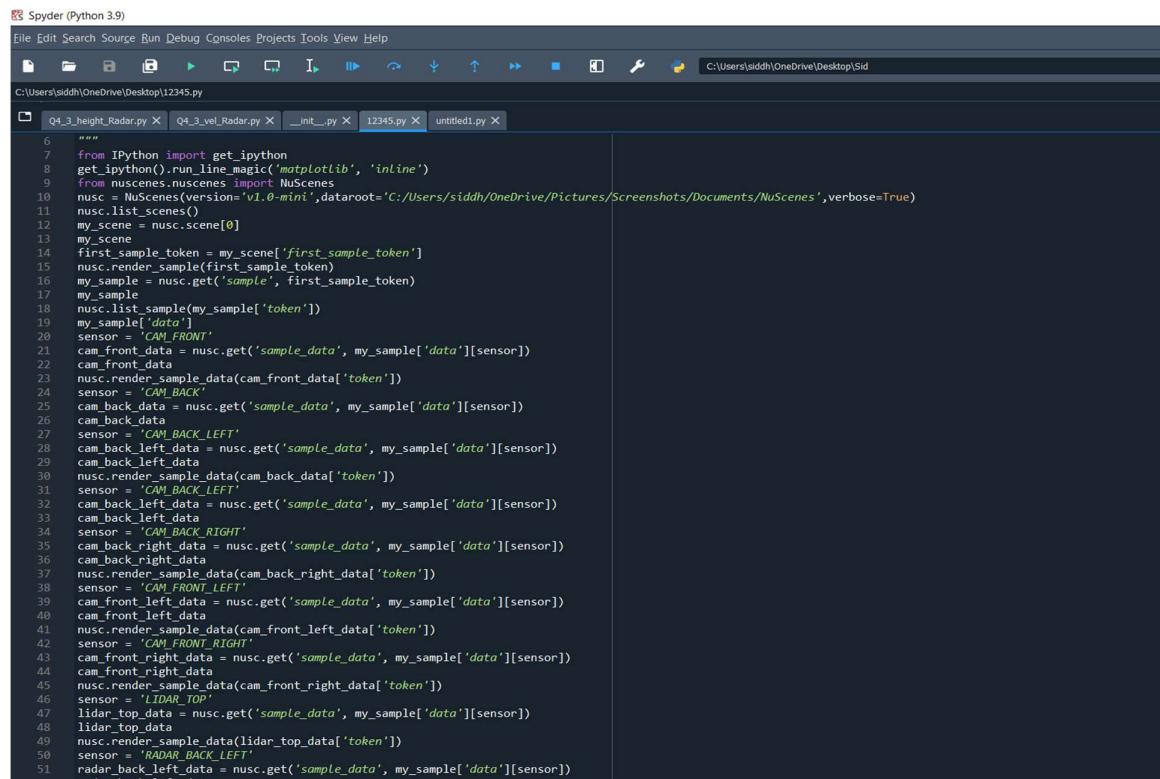


Figure 4

3. Pickup a set of data, including Image, Lidar, and Radar data. Visualize them respectively. If you use Python, you can refer to NuScene dev-kit tutorial.

- I have selected a set of data which includes Image, Lidar, and Radar data from the NuScenes dataset and visualized using Anaconda-Spyder

Program:

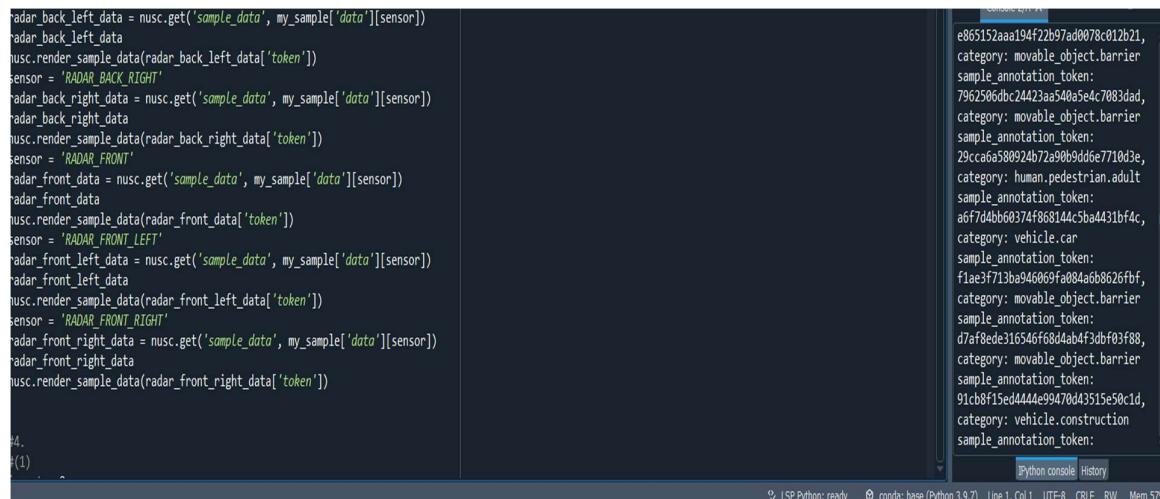


```

Spyder (Python 3.9)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\siddh\OneDrive\Desktop\12345.py
Q4_3_height_Radar.py Q4_3_vel_Radar.py __init__.py 12345.py untitled1.py

6 """
7     from IPython import get_ipython
8     get_ipython().run_line_magic('matplotlib', 'inline')
9     from nuscenes.nuscenes import NuScenes
10    nusc = NuScenes(version='v1.0-mini', dataroot='C:/Users/siddh/OneDrive/Pictures/Screenshots/Documents/NuScenes', verbose=True)
11    my_scenes = nusc.list_scenes()
12    my_scene = my_scenes[0]
13    my_scene
14    first_sample_token = my_scene['first_sample_token']
15    nusc.render_sample(first_sample_token)
16    my_sample = nusc.get('sample', first_sample_token)
17    my_sample
18    nusc.list_sample(my_sample['token'])
19    my_sample['data']
20    sensor = 'CAM_FRONT'
21    cam_front_data = nusc.get('sample_data', my_sample['data'][sensor])
22    cam_front_data
23    nusc.render_sample_data(cam_front_data['token'])
24    sensor = 'CAM_BACK'
25    cam_back_data = nusc.get('sample_data', my_sample['data'][sensor])
26    cam_back_data
27    sensor = 'CAM_BACK_LEFT'
28    cam_back_left_data = nusc.get('sample_data', my_sample['data'][sensor])
29    cam_back_left_data
30    nusc.render_sample_data(cam_back_left_data['token'])
31    sensor = 'CAM_BACK_RIGHT'
32    cam_back_right_data = nusc.get('sample_data', my_sample['data'][sensor])
33    cam_back_right_data
34    sensor = 'CAM_FRONT_LEFT'
35    cam_front_left_data = nusc.get('sample_data', my_sample['data'][sensor])
36    cam_front_left_data
37    nusc.render_sample_data(cam_front_left_data['token'])
38    sensor = 'CAM_FRONT_RIGHT'
39    cam_front_right_data = nusc.get('sample_data', my_sample['data'][sensor])
40    cam_front_right_data
41    nusc.render_sample_data(cam_front_right_data['token'])
42    sensor = 'RADAR_TOP'
43    lidar_top_data = nusc.get('sample_data', my_sample['data'][sensor])
44    lidar_top_data
45    nusc.render_sample_data(lidar_top_data['token'])
46    sensor = 'RADAR_BACK_LEFT'
47    radar_back_left_data = nusc.get('sample_data', my_sample['data'][sensor])
48
49
50
51

```



```

radar_back_left_data = nusc.get('sample_data', my_sample['data'][sensor])
radar_back_left_data
nusc.render_sample_data(radar_back_left_data['token'])
sensor = 'RADAR_BACK_RIGHT'
radar_back_right_data = nusc.get('sample_data', my_sample['data'][sensor])
radar_back_right_data
nusc.render_sample_data(radar_back_right_data['token'])
sensor = 'RADAR_FRONT'
radar_front_data = nusc.get('sample_data', my_sample['data'][sensor])
radar_front_data
nusc.render_sample_data(radar_front_data['token'])
sensor = 'RADAR_FRONT_LEFT'
radar_front_left_data = nusc.get('sample_data', my_sample['data'][sensor])
radar_front_left_data
nusc.render_sample_data(radar_front_left_data['token'])
sensor = 'RADAR_FRONT_RIGHT'
radar_front_right_data = nusc.get('sample_data', my_sample['data'][sensor])
radar_front_right_data
nusc.render_sample_data(radar_front_right_data['token'])

#(1)

```

Console (4/1) ↑

```

e865152aaa194f22b97ad0078c012b21,
category: movable_object.barrier
sample_annotation_token:
7962506dbc24423aa540a54c7083dad,
category: movable_object.barrier
sample_annotation_token:
29cca6a580924b72a90b9dd6e7710d3e,
category: human_pedestrian.adult
sample_annotation_token:
a6f7d4bb60374f868144c5ba4431bf4c,
category: vehicle.car
sample_annotation_token:
f1a3f713b946069fa0846b8862fbf,
category: movable_object.barrier
sample_annotation_token:
d7af8ede316546f6824ab4f3df03f88,
category: movable_object.barrier
sample_annotation_token:
91cb8f15ed4444e99470d43515e50c1d,
category: vehicle.construction
sample_annotation_token:

```

IPython console History

LSP Python: ready conda: base (Python 3.9.7) Line 1, Col 1 UTF-8 CRLF RW Mem 57%

Visualization Plots:

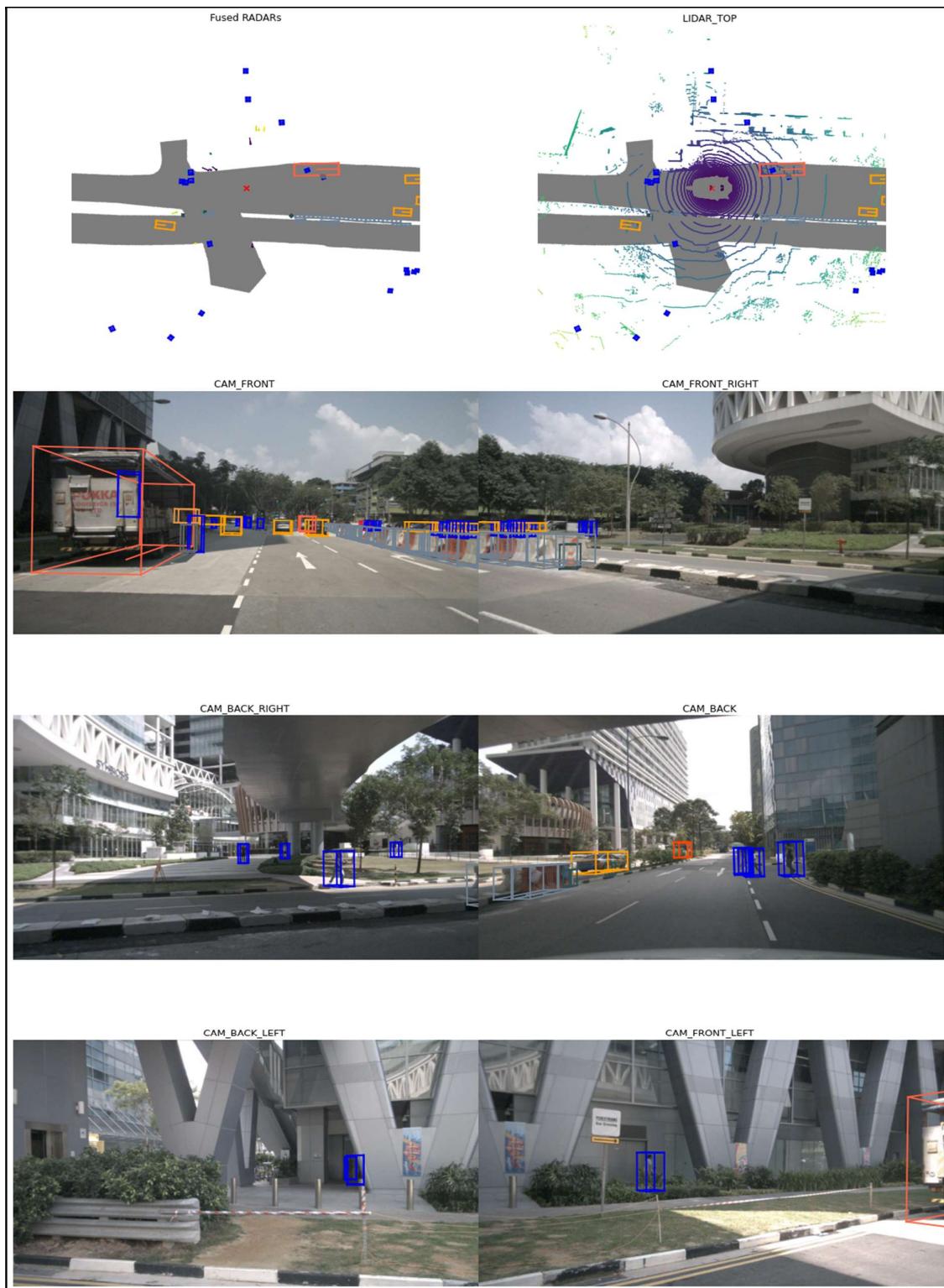


Figure 5

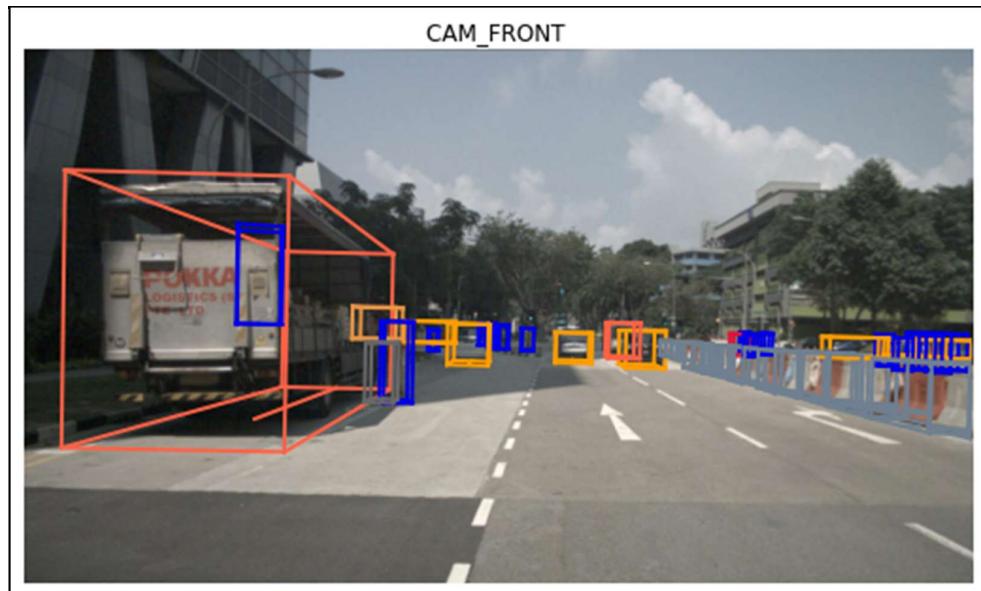


Figure 6

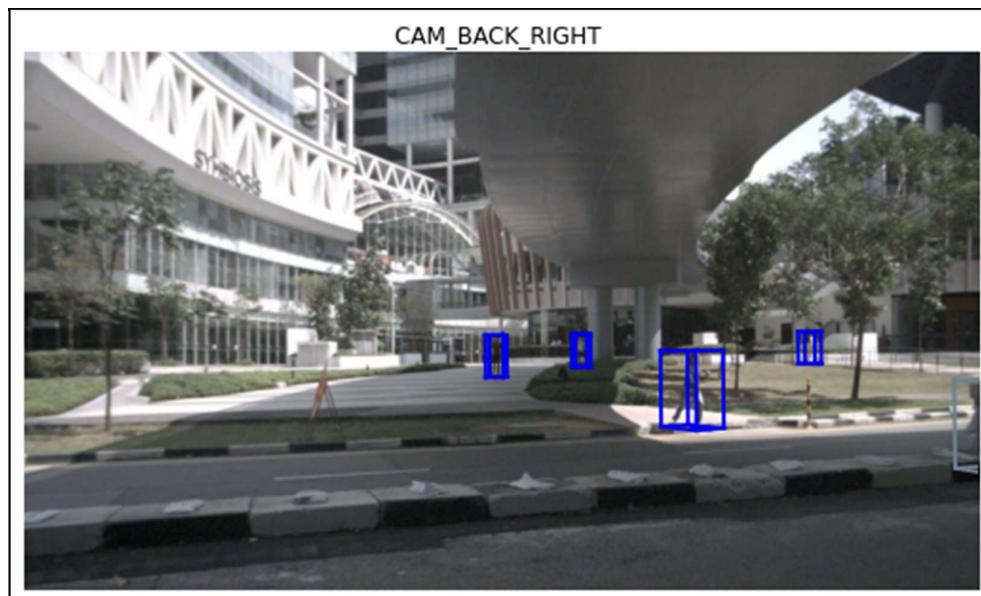


Figure 7

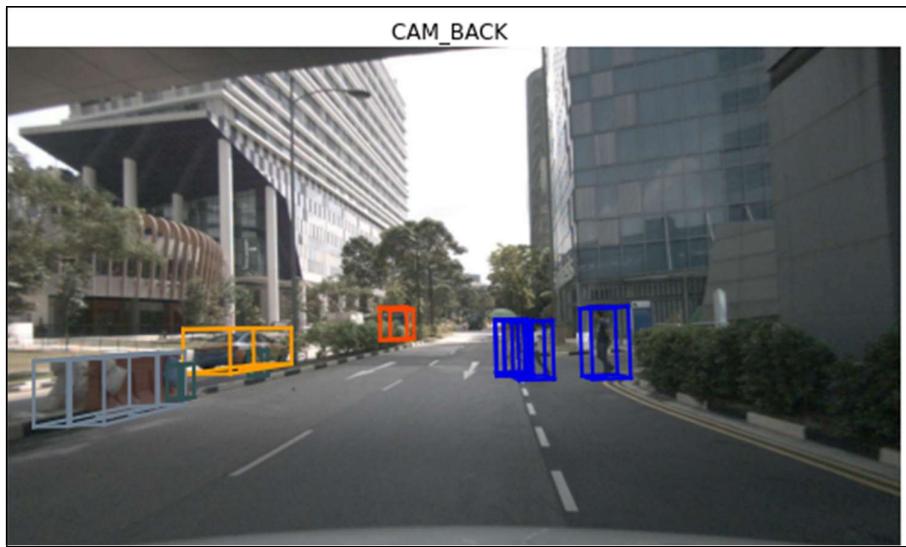


Figure 8

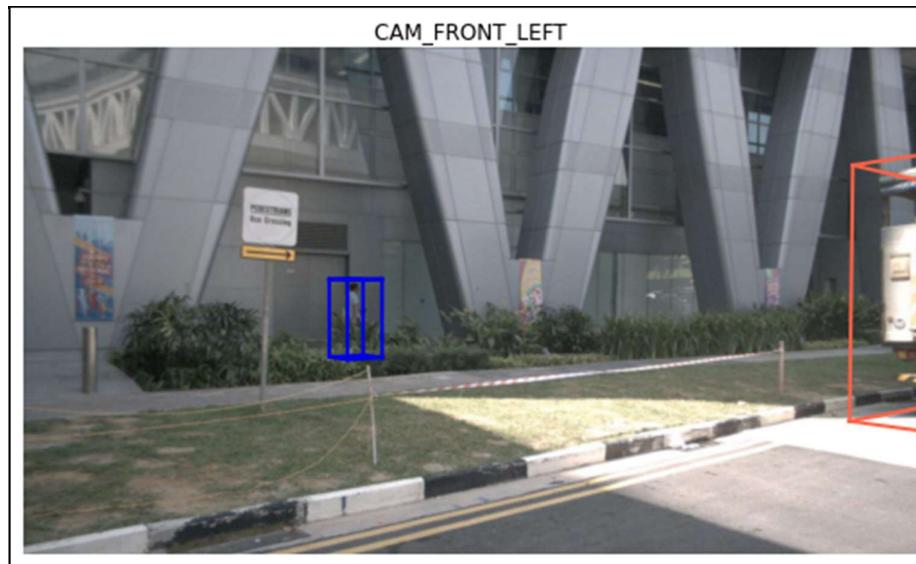


Figure 9

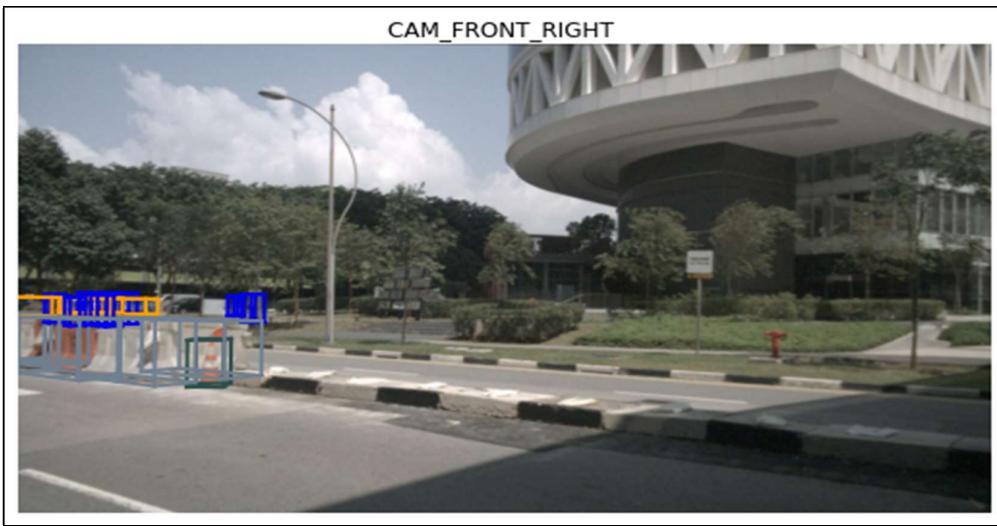


Figure 10

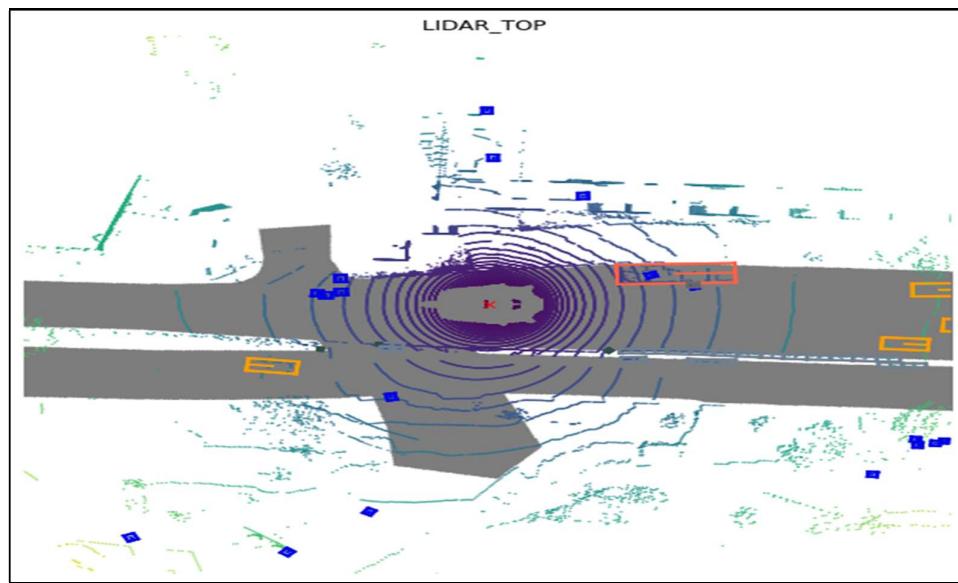


Figure 11

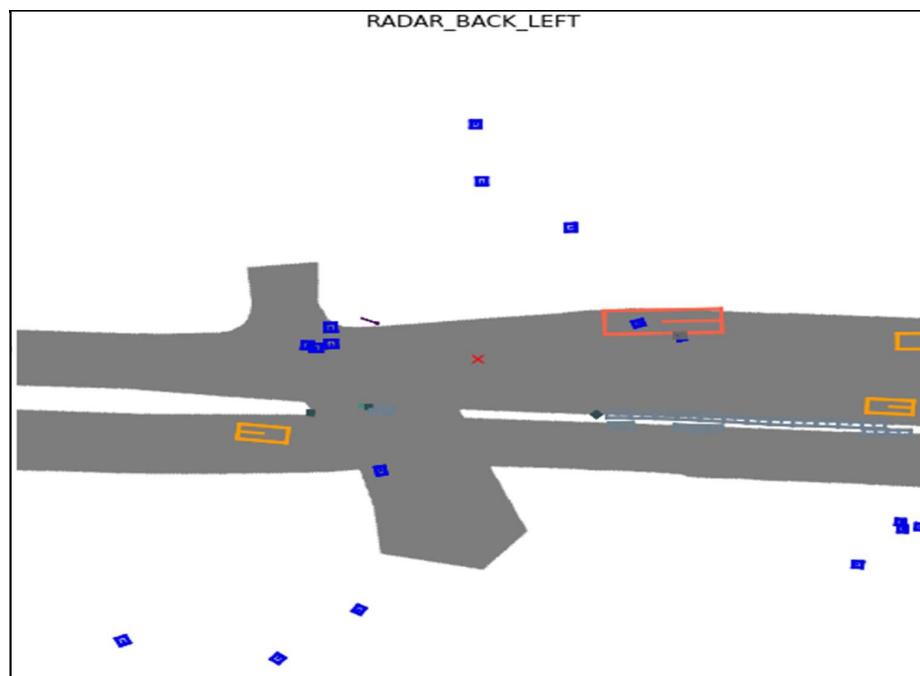


Figure 12

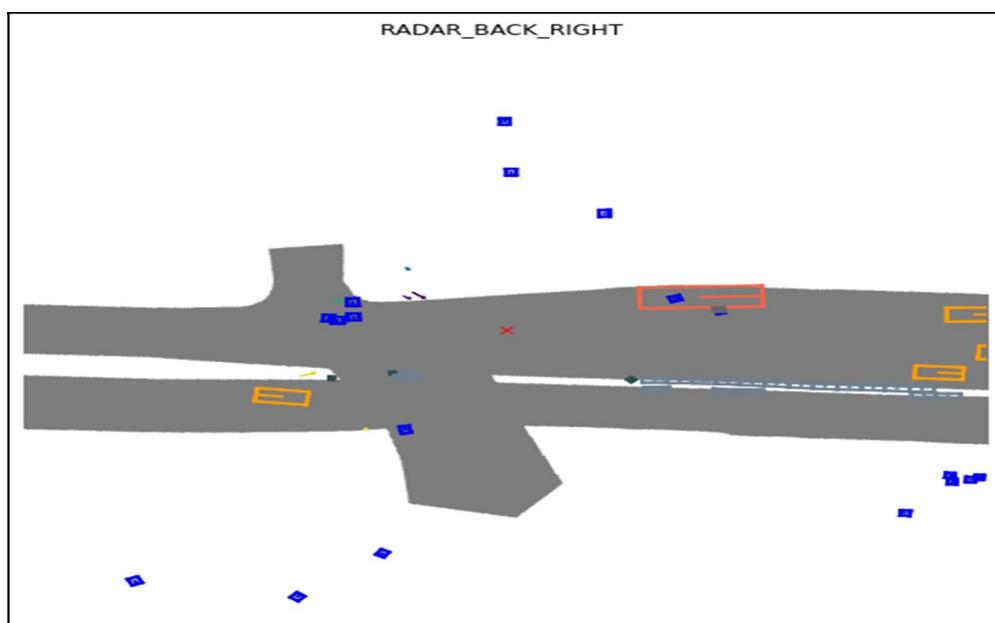


Figure 13

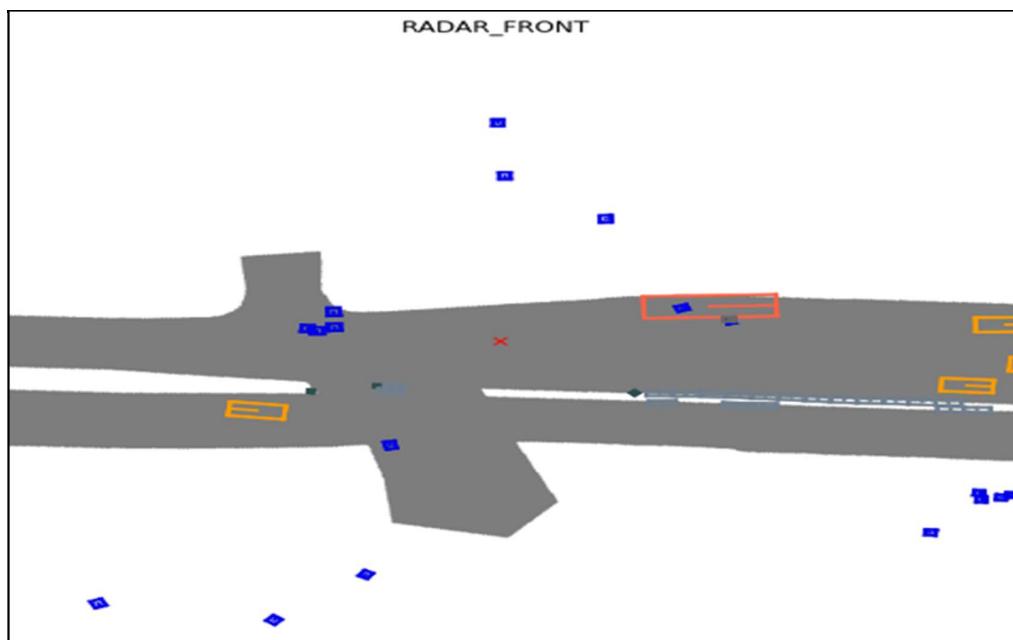


Figure 14

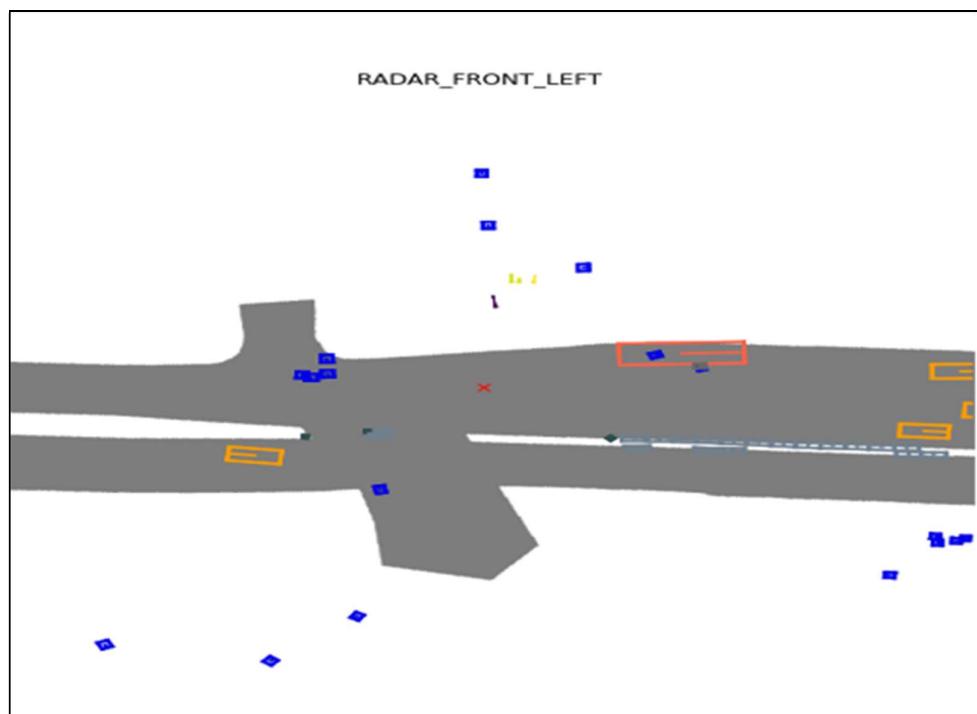


Figure 15

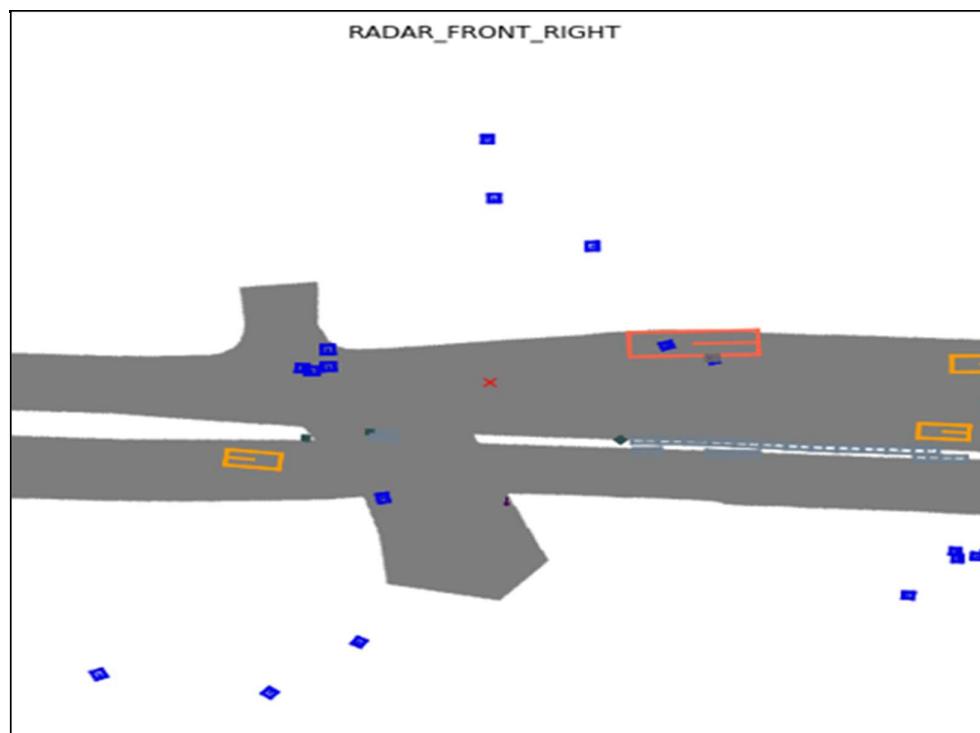


Figure 16

4. Rather than using NuScene dev-kit, implement below by yourself:
 - (1) Visualize images (you can use library OpenCV or others).
 - (2) Visualize Lidar point cloud data
 - a. You can refer to this [sample code](#).
 - b. Colorize points by height, intensity, and semantic label respectively.
 - i. Height is the Z value for a point.
 - ii. You can get intensity referring the code [here](#).
 - iii. You can get semantic label from the sample above code.
 - (3) Visualize Radar data
 - c. Use any other library (e.g., Open3D, PCL, etc) or modify the previous sample code to visualize the Radar data which you chosen.
 - d. Colorize points by below two variable aspects respectively.
 - i. For height (if it's all zero, you can colorize the points by distance).
 - ii. For velocity, you can find some velocity information from [here](#).

- (1) Visualize images (you can use library OpenCV or others):

Program:

```

71 #4,
72 #(1)
73 import cv2
74
75 # Load an color image
76 img = cv2.imread('C:/Users/siddh/OneDrive/Pictures/Screenshots/Documents/NuScenes/image.jpg')
77
78 # Show image
79 cv2.imshow('C:/Users/siddh/OneDrive/Pictures/Screenshots/Documents/NuScenes/image.jpg',img)
80 cv2.waitKey(0)
81 cv2.destroyAllWindows()
82
83 #(2)
84 import numpy as np
85 import open3d as o3d
86 import cv2, random,os
87
88

```

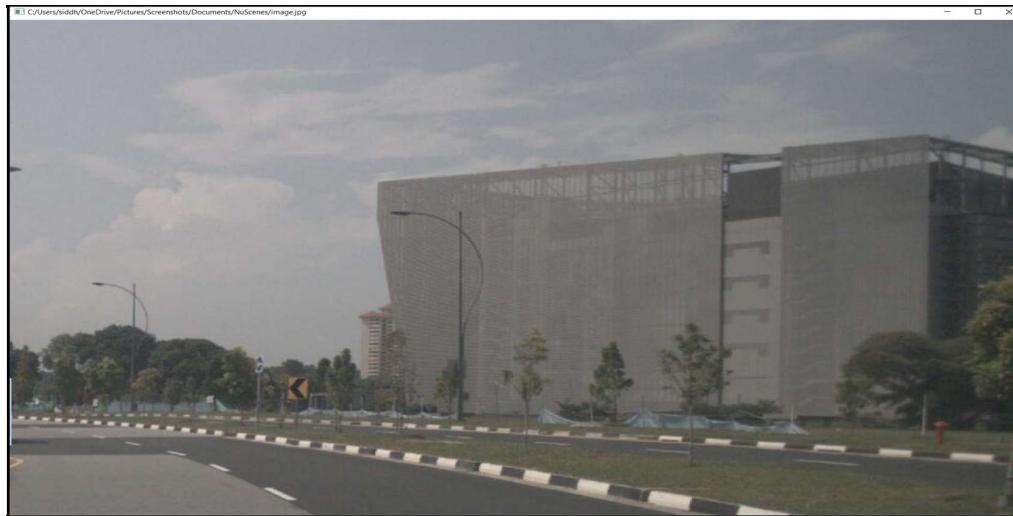


Figure 17

(2) Visualize Lidar point cloud data:

- e. You can refer to this [sample code](#).
- f. Colorize points by height, intensity, and semantic label respectively.
 - i. Height is the Z value for a point.
 - ii. You can get intensity referring the code [here](#).
 - iii. You can get semantic label from the sample above code.

e. Program:

```

4  @author: siddh
5  """
6
7
8  import open3d as o3d
9  import numpy as np
10
11 seg_name="C:/Users/siddh/OneDrive/Pictures/Screenshots/Documents/NuScenes/Lidarseg/v1.0-mini/0ab9ec2730894df2b48df70d0d2e84a9_Lidarseg.bin"
12 seg=np.fromfile(seg_name, dtype=np.uint8)
13
14 color = np.zeros([len(seg), 3])
15 color[:, 0] = seg/32
16 color[:, 1] = seg/32
17 color[:, 2] = seg/32
18
19 pcd_name="C:/Users/siddh/OneDrive/Pictures/Screenshots/Documents/NuScenes/samples/LIDAR_TOP/n008-2018-08-01-15-16-36-0400_LIDAR_TOP_1533151613398020.pcd.bin"
20 scan=np.fromfile(pcd_name, dtype=np.float32)
21 points = scan.reshape(-1, 5)[:, :4]
22
23 pcd = o3d.geometry.PointCloud()
24 pcd.points = o3d.utility.Vector3dVector(points[:, :3])
25 pcd.colors = o3d.utility.Vector3dVector(color)
26
27 o3d.visualization.draw_geometries([pcd])

```

Visualization of Lidar Image:

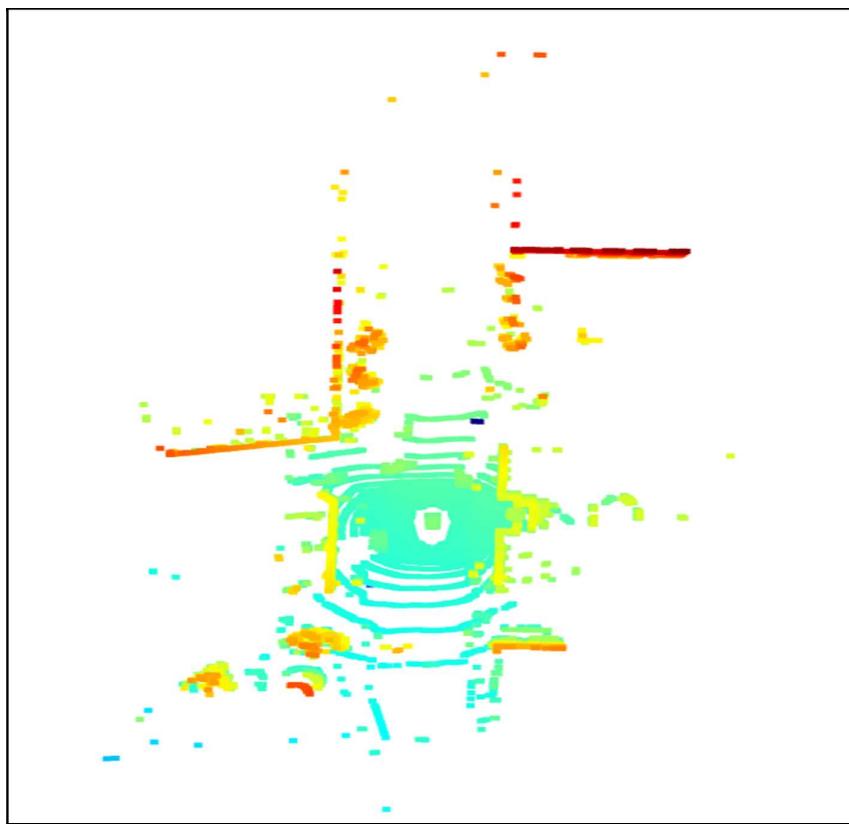


Figure 18

f. Program:

```

08     #-----Q.4.2
09
10 #Define PCD Path
11 input_path = "C:/Users/siddh/OneDrive/Pictures/Screenshots/Documents/NuScenes/samples/LIDAR_TOP"
12
13 #Picking Random file
14 dataname = random.choice([
15     x for x in os.listdir(input_path)
16     if os.path.isfile(os.path.join(input_path, x))
17 ])
18 filename_rand = (os.path.join(input_path, dataname))
19 print('Displayed image is ',filename_rand)
20
21 #Converting PCD to points
22 pcd_data = []
23 data = np.fromfile(filename_rand, dtype=np.float32)
24 pointdata = data.reshape((-1, 5))[:, :4]
25 print(pointdata)
26
27 #Value of Z
28 Z=pointdata[:,2]
29 print('The value of Z is ',Z)
30
31 #Value of Intensity
32 intensity=pointdata[:,3]
33 print('The value of Intensity is ',intensity)
34
35 #For height
36 color = np.zeros([len(Z), 3])
37 color[:, 0] = Z[:,1]/10
38 color[:, 1] = 0.5
39 color[:, 2] = 0.5
40
41
42 pcdimage = o3d.geometry.PointCloud()
43 pcdimage.points = o3d.utility.Vector3dVector(pointdata[:, :3])
44 pcdimage.colors = o3d.utility.Vector3dVector(color)
45 o3d.visualization.draw_geometries([pcdimage])
46
47 #For Intensity
48 color = np.zeros([len(intensity), 3])
49 color[:, 0] = intensity[:,1]/10
50 color[:, 1] = 0.5
51 color[:, 2] = 0.5
52
53
54 pcdimage = o3d.geometry.PointCloud()
55 pcdimage.points = o3d.utility.Vector3dVector(pointdata[:, :3])
56 pcdimage.colors = o3d.utility.Vector3dVector(color)
57 o3d.visualization.draw_geometries([pcdimage])
58
59 #Visualizations by Segment
60 see=np.fromfile(filename_rand, dtype=np.uint8)
61
62
63 color = np.zeros([len(see), 3])
64 color[:, 0] = 20
65 color[:, 1] = 0.5
66 color[:, 2] = 0.5
67
68 pcdimage = o3d.geometry.PointCloud()
69 pcdimage.points = o3d.utility.Vector3dVector(pointdata[:, :3])
70 pcdimage.colors = o3d.utility.Vector3dVector(color)
71 o3d.visualization.draw_geometries([pcdimage])
72

```

Visualizations of h\Height, Intensity, Segmantic label:

a. Height:

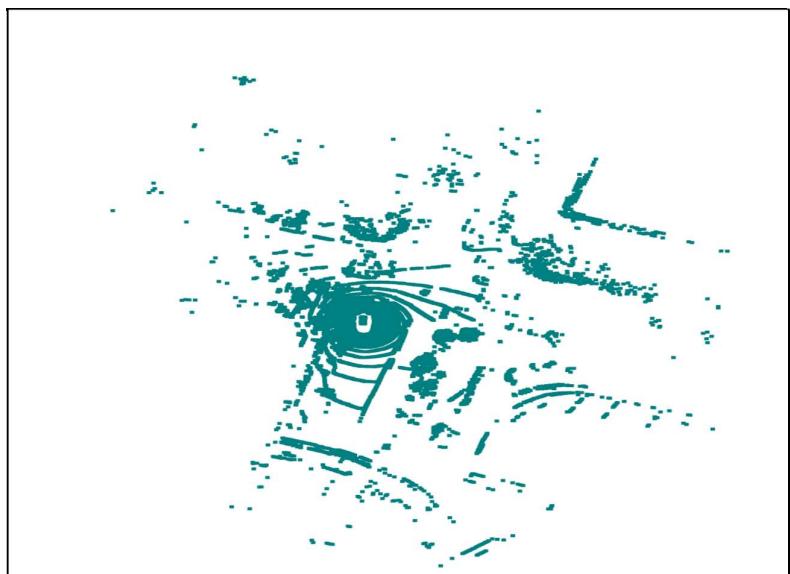


Figure 19

b. Intensity:

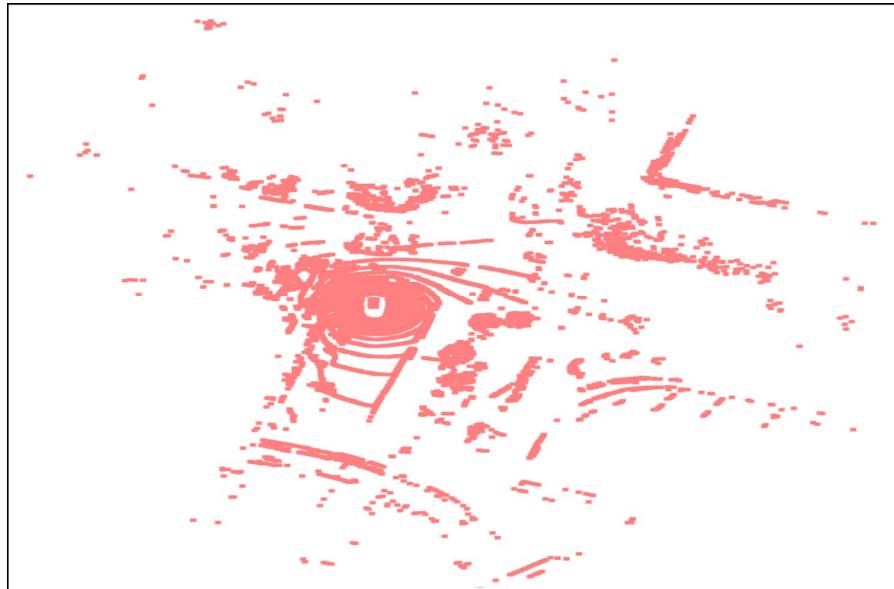


Figure 20

c. Segmantic label:

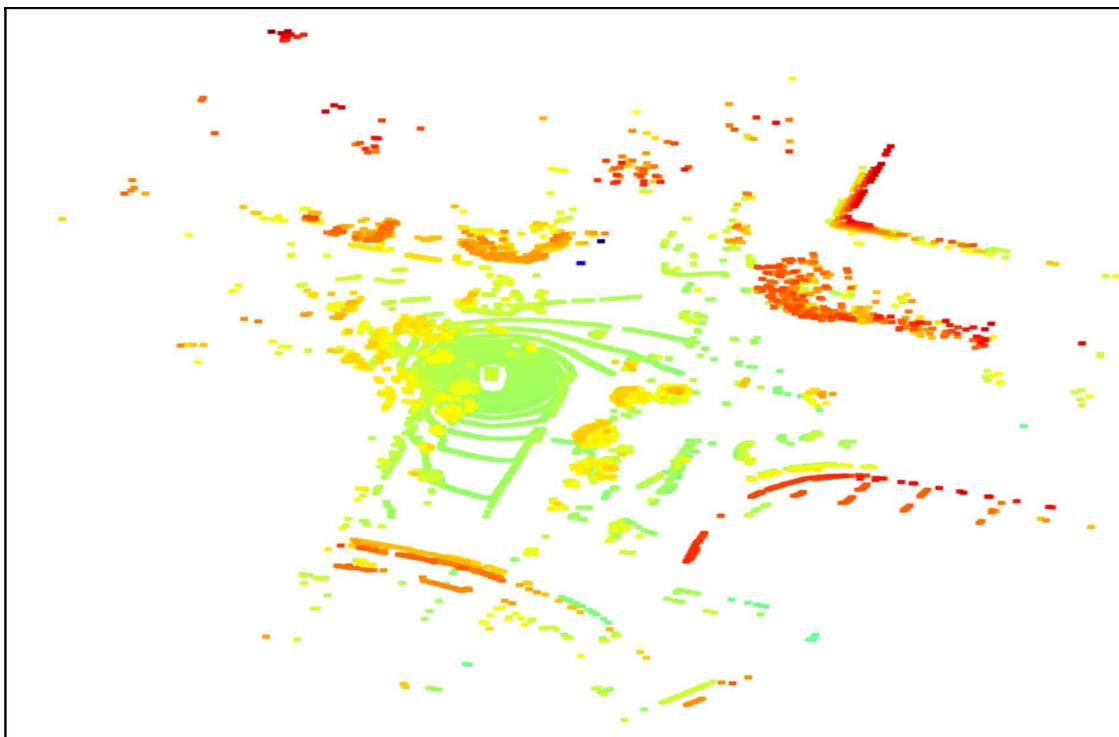


Figure 21

(4) Visualize Radar data

- g. Use any other library (e.g., Open3D, PCL, etc) or modify the previous sample code to visualize the Radar data which you chosen.
 - h. Colorize points by below two variable aspects respectively.
 - i. For height (if it's all zero, you can colorize the points by distance).
 - ii. For velocity, you can find some velocity information from [here](#).
- g. Use any other library (e.g., Open3D, PCL, etc) or modify the previous sample code to visualize the Radar data which you chosen.

Program:

```
# 4.(3d(i))
import os.path as osp
import random
import numpy as np
import open3d as o3d
from nuscenes.nuscenes import NuScenes
from nuscenes.utils.data_classes import RadarPointCloud
```

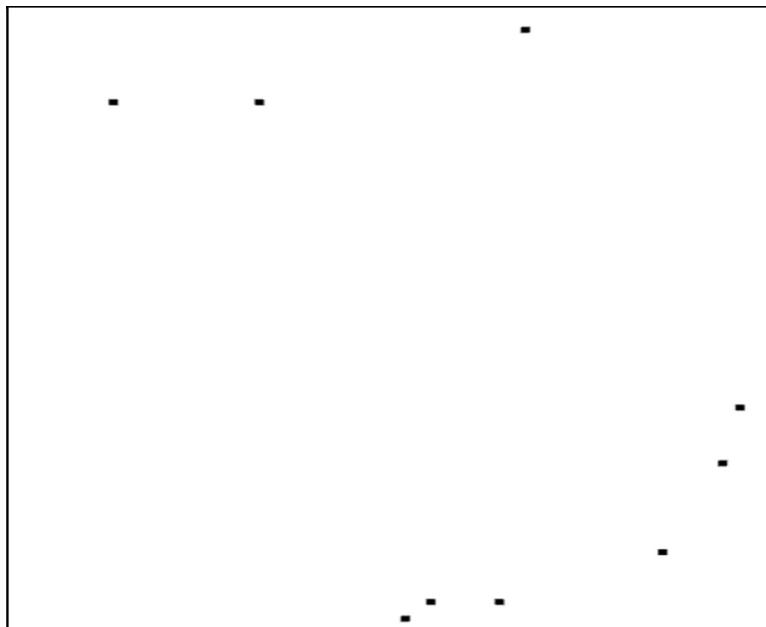


Figure 22

h. Colorize points by below two variable aspects respectively:

- iii. For height (if it's all zero, you can colorize the points by distance).
- iv. For velocity, you can find some velocity information from [here](#).

Program:

```

154 import os.path as osp
155 import random
156 import numpy as np
157 import open3d as o3d
158 from nuscenes.nuscenes import NuScenes
159 from nuscenes.utils.data_classes import RadarPointCloud
160
161 nusc = NuScenes(version='v1.0-mini', dataroot='C:/Users/siddh/OneDrive/Pictures/Screenshots/Documents/NuScenes', verbose=True)
162 random_number=random.randint(0,100)
163 my_sample = nusc.sample[random_number]
164
165 pointsensor_token = my_sample['data'][['RADAR_FRONT']]
166 pointsensor = nusc.get('sample_data', pointsensor_token)
167 pcl_path = osp.join(nusc.dataroot, pointsensor['filename'])
168 #Radar plot
169
170 pc = RadarPointCloud.from_file(pcl_path)
171 data=np.asarray(pc.points.astype(dtype=np.float32()))
172
173 pointsdata = data.reshape((-1, 18))[:, :17]
174 print(pointsdata)
175
176 #Height
177 Z=data[2]
178
179 #Velocity
180 vx=data[6]
181 vy=data[7]
182
183 #For height
184 color = np.zeros([len(Z), 3])
185 color[:, 0] = Z/10
186 color[:, 1] = 0.5
187 color[:, 2] = 0.5
188
189
190 pcdimage = o3d.geometry.PointCloud()
191 pcdimage.points = o3d.utility.Vector3dVector(pointsdata[:, :3])
192 pcdimage.colors = o3d.utility.Vector3dVector(color)
193 o3d.visualization.draw_geometries([pcdimage])

```

Visualization of height:

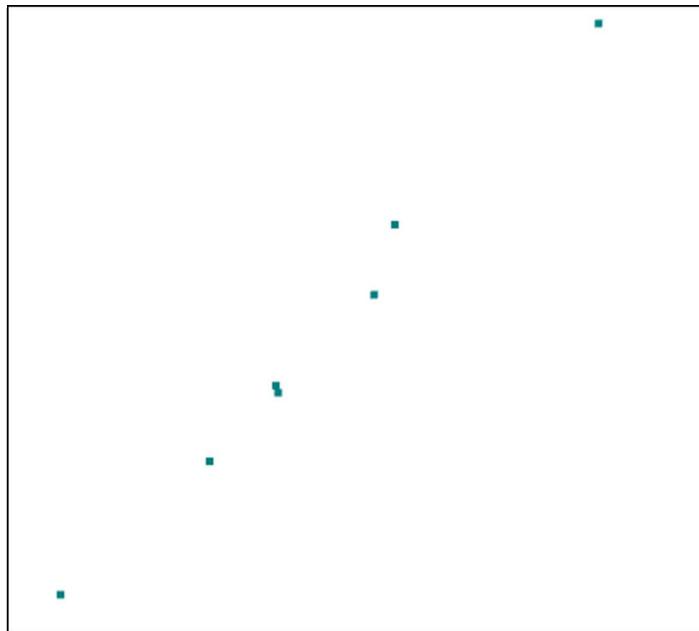


Figure 23

Program:

```

197 # 4.(3d(iii))
198 import os.path as osp
199 import random
200 import numpy as np
201 import open3d as o3d
202 from nuscenes.nuscenes import NuScenes
203 #from data_classes import RadarPointCloud
204 from nuscenes.utils.data_classes import RadarPointCloud
205
206 nusc = NuScenes(version='v1.0-mini', dataroot='C:/Users/siddh/OneDrive/Pictures/Screenshots/Documents/NuScenes', verbose=True)
207 random_number=random.randint(0,100)
208 my_sample = nusc.sample[random_number]
209
210 pointsensor_token = my_sample['data'][‘RADAR_FRONT’]
211 pointsensor = nusc.get('sample_data', pointsensor_token)
212 pcl_path = osp.join(nusc.dataroot, pointsensor[‘filename’])
213 #Radar plot
214
215 pc = RadarPointCloud.from_file(pcl_path)
216 #data=np.asarray(pc.points.astype(dtype=np.float32()))
217
218 data = pc.points.T
219
220 pointsdata = data.reshape((-1, 18))[:, :17]
221 print(pointsdata)
222
223 #Height
224 #Z=data[2]
225
226
227 #Velocity
228 vx=data[:,8]
229 vy=data[:,9]
230
231 #For height
232 color = np.zeros([len(vx), 3])
233 color[:, 0] = vx/10
234 color[:, 1] = vy/10
235 color[:, 2] = 0.5
236
237 pcdimage = o3d.geometry.PointCloud()
238 pcdimage.points = o3d.utility.Vector3dVector(pointsdata[:, :3])
239 pcdimage.colors = o3d.utility.Vector3dVector(color)
240 o3d.visualization.draw_geometries([pcdimage])
241 print('done')

```

Visualization of velocity:

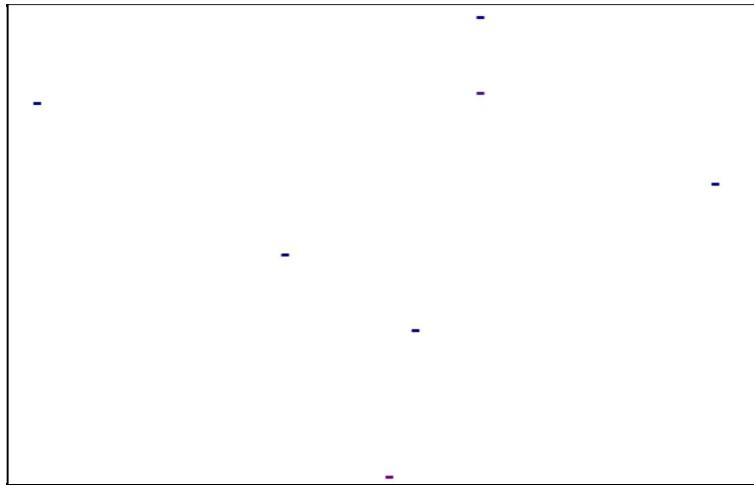


Figure 24

5. Using NuScene dev-kit for the set of data which you picked up:

(1) Visualize Radar data projection on image

- Print calibration info (between Radar and Camera sensors) by referring code [here](#).
- Explain the above calibration info, and pipeline of First–Fifth steps in the code.
- Visualize Radar data projection on image based on calibration info.

a.

Program:

```
#5(1a.)
#5(1a).

# To print calibration info (between Radar and Camera sensors) by referring code:
# Step-1: Transportation of data points from Radar Sensor to the ego vehicle frame
sensor = 'RADAR_BACK_LEFT'
t_r_s = nusc.get('sample_data', my_sample['data'][sensor])
t_r_s = nusc.get('calibrated_sensor', t_r_s['calibrated_sensor_token'])
print("RADAR_BACK_LEFT translation_values is: ",c_r_s["translation"])
print("RADAR_BACK_LEFT rotation_values is: ",c_r_s["rotation"],'\n')
# Step-2: Transformation of data points from ego vehicle frame to global frame
t_r_s = nusc.get('sample_data', my_sample['data'][sensor])
e_r_s = nusc.get('ego_pose',t_r_s['ego_pose_token'])
print("e_translation_values is: ",e_r_s["translation"])
print("e_rotation_values is: ",e_r_s["rotation"],'\n')
# Step-3: Transformation of data points from global frame to ego vehicle frame
sensor = 'CAM_BACK_LEFT'
t_c_s = nusc.get('sample_data', my_sample['data'][sensor])
t_c_s = nusc.get('calibrated_sensor', t_c_s['calibrated_sensor_token'])
print("CAM_BACK_LEFT translation_values is: ",c_c_s["translation"])
print("CAM_BACK_LEFT rotation_values is: ",c_c_s["rotation"],'\n')
# Step-4: Transformation of data points from ego vehicle frame to camera
t_s = nusc.get('sample_data', my_sample['data'][sensor])
e_c_s = nusc.get('ego_pose',t_c_s['ego_pose_token'])
print("e_translation_values is: ",e_c_s["translation"])
print("e_rotation_values is: ",e_c_s["rotation"],'\n')
# Step-5: Finally, the radar data points are present in camera.
```

```

[[ 5.59999990e+00  8.50000000e+00  0.00000000e+00  1.00000000e+00
  1.00000000e+00  5.00000000e-01 -4.75000000e+00  0.00000000e+00
  7.58259892e-02  1.15093023e-01  1.00000000e+00  3.00000000e+00
  1.90000000e+01  1.90000000e+01  0.00000000e+00  1.00000000e+00
  1.80000000e+01]
[ 8.19999981e+00 -7.50000000e+00  0.00000000e+00  1.00000000e+00
  2.00000000e+00 -4.00000000e+00 -5.00000000e+00  0.00000000e+00
  2.29304805e-02 -2.09730007e-02  1.00000000e+00  3.00000000e+00
  1.90000000e+01  1.90000000e+01  0.00000000e+00  1.00000000e+00
  1.70000000e+01]
[ 1.00000000e+01 -7.30000019e+00  0.00000000e+00  1.00000000e+00
  4.00000000e+00  6.00000000e+00 -5.00000000e+00  0.00000000e+00
  2.54133120e-02 -1.85517203e-02  1.00000000e+00  3.00000000e+00
  1.90000000e+01  1.90000000e+01  0.00000000e+00  1.00000000e+00
  1.70000000e+01]
[ 1.00000000e+01 -7.69999981e+00  0.00000000e+00  1.00000000e+00
  5.00000000e+00  4.00000000e+00 -5.00000000e+00  0.00000000e+00
  2.48855427e-02 -1.91618670e-02  1.00000000e+00  3.00000000e+00
  1.90000000e+01  1.90000000e+01  0.00000000e+00  1.00000000e+00
  1.70000000e+01]
[ 8.80000019e+00 -1.01000004e+01  0.00000000e+00  1.00000000e+00
  6.00000000e+00  3.50000000e+00 -5.00000000e+00  0.00000000e+00
  1.98937263e-02 -2.28325725e-02  1.00000000e+00  3.00000000e+00
  1.90000000e+01  1.90000000e+01  0.00000000e+00  1.00000000e+00
  1.80000000e+01]
[ 1.18000002e+01 -7.50000000e+00  0.00000000e+00  1.00000000e+00
  7.00000000e+00 -1.50000000e+00 -5.00000000e+00  0.00000000e+00
  2.65974645e-02 -1.69051699e-02  1.00000000e+00  3.00000000e+00
  1.90000000e+01  1.90000000e+01  0.00000000e+00  1.00000000e+00
  1.70000000e+01]]
done
RADAR_BACK_LEFT translation_values is: [-0.562, 0.628, 0.53]
RADAR_BACK_LEFT rotation_values is: [0.0458860416542946, 0.0, 0.0, 0.9989466808500344]

e_translation_values is: [656.4798091994182, 1603.971345141823, 0.0]
e_rotation_values is: [-0.93974100320854776, -0.009941445191752165, 0.009816534042556625, 0.34160159575337645]

CAM_BACK_LEFT translation_values is: [1.04852047718, 0.483058131052, 1.56210154484]
CAM_BACK_LEFT rotation_values is: [0.7048620297871717, -0.6907306801461466, -0.11209091960167808, 0.11617345743327073]

e_translation_values is: [656.4246963278883, 1604.0179020923947, 0.0]
e_rotation_values is: [-0.9397483329333906, -0.009977299221811859, 0.009906389280725406, 0.34157779157985546]

```

1b. Explain the above calibration info, and pipeline of First~Fifth steps in the code

- Firstly, the data points are transferred from the Radar sensor to ego vehicle frame by using (translation and rotation) calibration data.
- Secondly, the data points are transferred from ego vehicle frame to global frame by using ego pose in calibration data (translation and rotation).
- Thirdly, the data points are now transformed from global frame to ego vehicle frame by using calibration data (translation and rotating).
- Next, the data points are transformed from ego vehicle frame to the camera frame.
- Finally, the radar data points are present in the Camera.

1c. Visualize Radar data projection on image based on calibration info.

```

1 import random
2 from nuscenes.nuscenes import NuScenes
3 from Nusc_file import NuScenesExplorer
4
5
6 nusc = NuScenes(version='v1.0-mini', dataroot='C:/Users/siddh/OneDrive/Pictures/Screenshots/Documents/NuScenes', verbose=True)
7 random_number=random.randint(0,100)
8 my_sample = nusc.sample[random_number]
9
10
11 #Radar plot
12 nusc.render_pointcloud_in_image(my_sample['token'], pointsensor_channel='RADAR_FRONT')
13 NuScenesExplorer(nusc).map_pointcloud_to_image(my_sample['data'][['RADAR_FRONT']],my_sample['data'][['CAM_FRONT']])

```



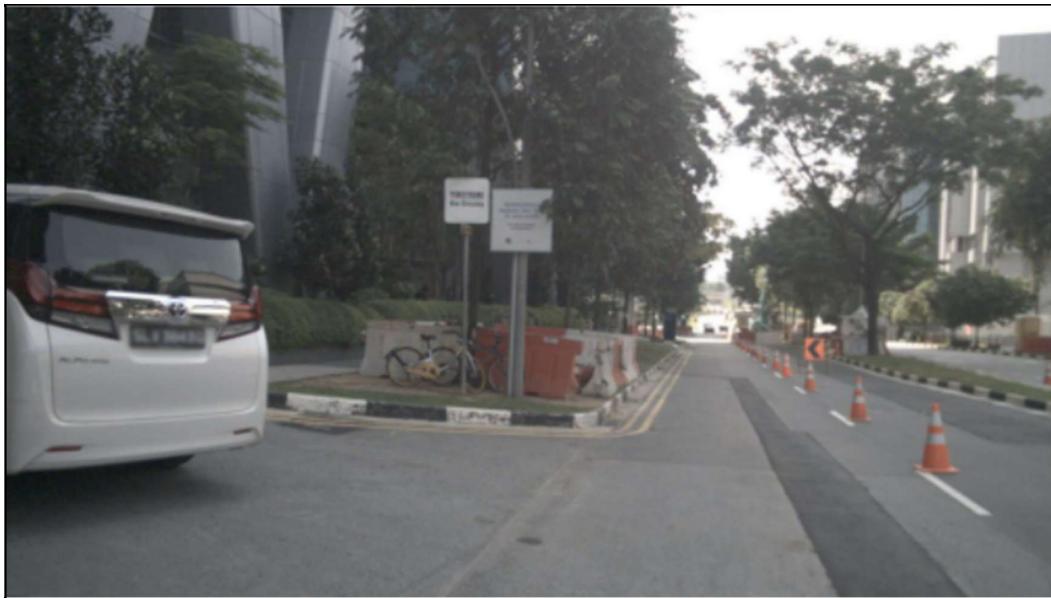


Figure 25

- If we observe keenly, we can see that they are blue and red colour-based distance values where Red is the maximum and Blue is the minimum.
- (2) Visualize LiDAR data projection on image
- c. Print and explain the calibration info (between LiDAR and Camera sensors) by referring [here](#).
 - e. Visualize LiDAR data projection on image based on calibration info.

2c. Print and explain the calibration info (between LiDAR and Camera sensors):

Program:

```
## To print calibration info (between Radar and Camera sensors) by referring code:
# Step-1: Transportation of data points from Radar Sensor to the ego vehicle frame
sensor = 'LIDAR_TOP'
t_l_s = nusc.get('sample_data', my_sample['data'][sensor])
t_l_s = nusc.get('calibrated_sensor', t_l_s["calibrated_sensor_token"])
print("LIDAR_TOP translation_values is: ",c_l_s["translation"])
print("LIDAR_TOP rotation_values is: ",c_l_s["rotation"], "\n")
# Step-2: Transformation of data points from ego vehicle frame to global frame
t_l_s = nusc.get('sample_data', my_sample['data'][sensor])
e_l_s = nusc.get('ego_pose',t_l_s["ego_pose_token"])
print("e_translation_values is: ",e_l_s["translation"])
print("e_rotation_values is: ",e_l_s["rotation"], "\n")
# Step-3: Transformation of data points from global frame to ego vehicle frame
sensor = 'CAM_BACK_LEFT'
t_c_s = nusc.get('sample_data', my_sample['data'][sensor])
t_c_s = nusc.get('calibrated_sensor', t_c_s["calibrated_sensor_token"])
print("CAM_BACK_LEFT translation_values is: ",c_c_s["translation"])
print("CAM_BACK_LEFT rotation_values is: ",c_c_s["rotation"], "\n")
# Step-4: Transformation of data points from ego vehicle frame to camera
t_s = nusc.get('sample_data', my_sample['data'][sensor])
e_c_s = nusc.get('ego_pose',t_c_s["ego_pose_token"])
print("e_translation_values is: ",e_c_s["translation"])
print("e_rotation_values is: ",e_c_s["rotation"], "\n")
# Step-5: Finally, the radar data points is present in camera.
```

c. Explanation of above calibration info:

- Firstly, the data points are transferred from the Radar sensor to ego vehicle frame by using (translation and rotation) calibration data.
- Secondly, the data points are transferred from ego vehicle frame to global frame by using ego pose in calibration data (translation and rotation).
- Thirdly, the data points are now transformed from global frame to ego vehicle frame by using calibration data (translation and rotating).
- Next, the data points are transformed from ego vehicle frame to the camera frame.
- Finally, the radar data points are present in the Camera.

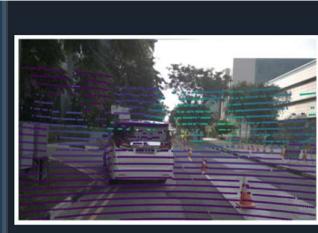
e. Visualize LiDAR data projection on image based on calibration info.

Program:

```

1 import random
2 from nuscenes.nuscenes import NuScenes
3 from Nusc_file import NuScenesExplorer
4
5
6 nusc = NuScenes(version='v1.0-mini', dataroot='C:/Users/siddh/OneDrive/Pictures/Screenshots/Documents/NuScenes', verbose=True)
7 random_number=random.randint(0,100)
8 my_sample = nusc.sample[random_number]
9
10 #lidar plot
11 nusc.render_pointcloud_in_image(my_sample['token'], pointsensor_channel='LIDAR_TOP', render_intensity=False)
12 NuScenesExplorer(nusc).map_pointcloud_to_image(my_sample['data']['LIDAR_TOP'],my_sample['data']['CAM_FRONT'])
13

```



Visualization of LiDAR data projection on image based on calibration info:



Figure 26