Hi everyone, I am Siddharth Thorat graduate student from CU-ICAR. This is the repository to start NuScenes with sample datasets available on open source NuScenes website to visualize them.

The requirements for visualizing are:

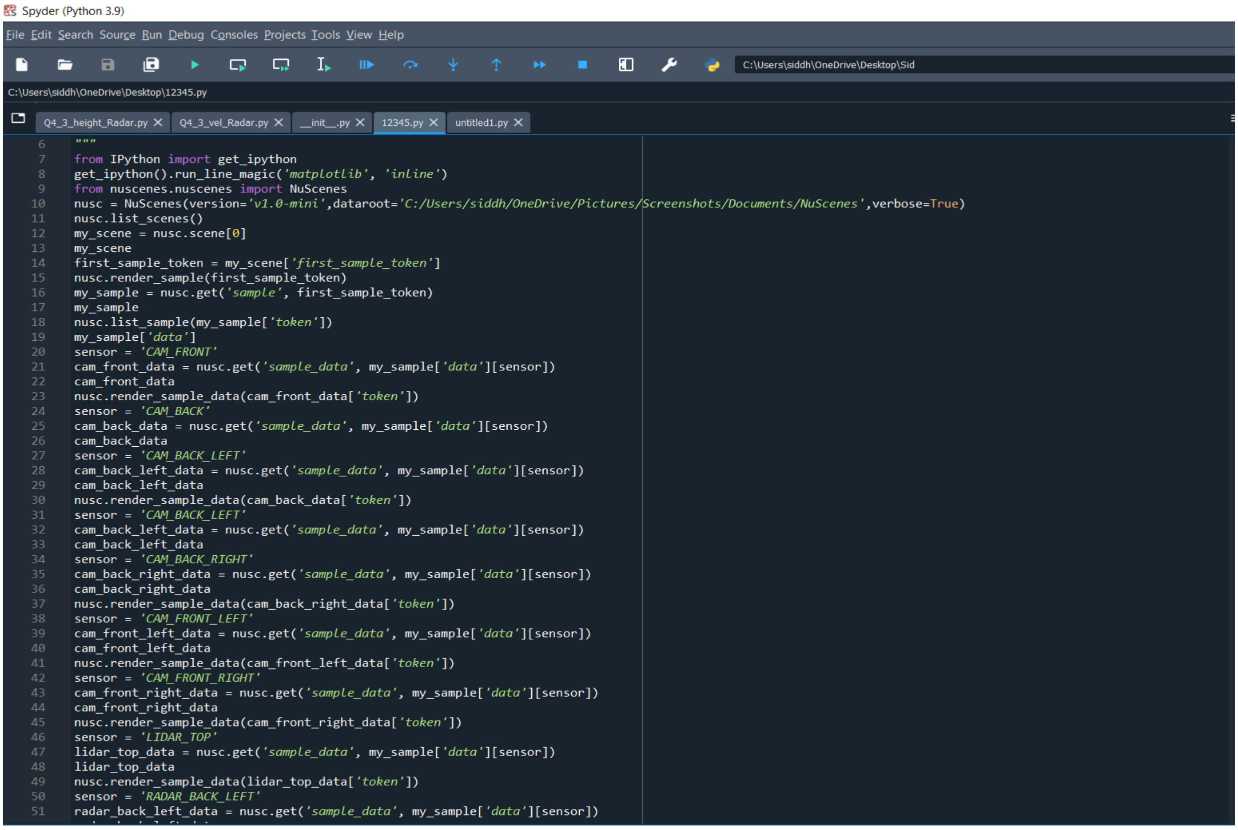
- Download NuScene mini dataset https://www.nuscenes.org/data/v1.0-mini.tgz
- Download NuScene Develop kit https://github.com/nutonomy/nuscenes-devkit
- Download Lidarseg file https://www.nuscenes.org/data/nuScenes-lidarseg-mini-v1.0.tar.bz2
- Anaconda new version
- Spyder
- pip3 ($ pip3 install nuscene-devkit)
- python3
- OpenCV

1. 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

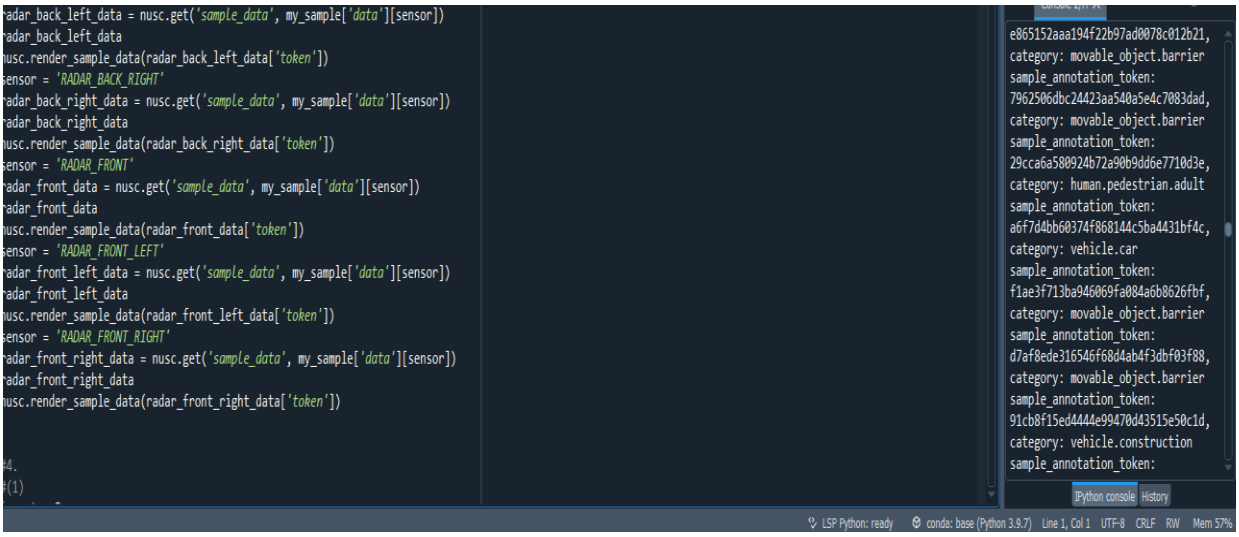
Subset of trainval, 10 scenes, used to explore the data without downloading the whole dataset.

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



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

"""
from IPython import get_ipython
get_ipython().run_line_magic('matplotlib', 'inline')
from nuscenes.nuscenes import NuScenes
nusc = NuScenes(version='v1.0-mini',dataroot='C:/Users/siddh/OneDrive/Pictures/Screenshots/Documents/NuScenes',verbose=True)
nusc.list_scenes()
my_scene = nusc.scene[0]
my_scene
first_sample_token = my_scene['first_sample_token']
nusc.render_sample(first_sample_token)
my_sample = nusc.get('sample', first_sample_token)
my_sample
nusc.list_sample(my_sample['token'])
my_sample['data']
sensor = 'CAM_FRONT'
cam_front_data = nusc.get('sample_data', my_sample['data'][sensor])
cam_front_data
nusc.render_sample_data(cam_front_data['token'])
sensor = 'CAM_BACK'
cam_back_data = nusc.get('sample_data', my_sample['data'][sensor])
cam_back_data
sensor = 'CAM_BACK_LEFT'
cam_back_left_data = nusc.get('sample_data', my_sample['data'][sensor])
cam_back_left_data
nusc.render_sample_data(cam_back_data['token'])
sensor = 'CAM_BACK_LEFT'
cam_back_left_data = nusc.get('sample_data', my_sample['data'][sensor])
cam_back_left_data
sensor = 'CAM_BACK_RIGHT'
cam_back_right_data = nusc.get('sample_data', my_sample['data'][sensor])
cam_back_right_data
nusc.render_sample_data(cam_back_right_data['token'])
sensor = 'CAM_FRONT_LEFT'
cam_front_left_data = nusc.get('sample_data', my_sample['data'][sensor])
cam_front_left_data
nusc.render_sample_data(cam_front_left_data['token'])
sensor = 'CAM_FRONT_RIGHT'
cam_front_right_data = nusc.get('sample_data', my_sample['data'][sensor])
cam_front_right_data
nusc.render_sample_data(cam_front_right_data['token'])
sensor = 'LIDAR_TOP'
lidar_top_data = nusc.get('sample_data', my_sample['data'][sensor])
lidar_top_data
nusc.render_sample_data(lidar_top_data['token'])
sensor = 'RADAR_BACK_LEFT'
radar_back_left_data = nusc.get('sample_data', my_sample['data'][sensor])
```



```
adar_back_left_data = nusc.get('sample_data', my_sample['data'][sensor])
adar_back_left_data
usc.render_sample_data(radar_back_left_data['token'])
ensor = 'RADAR_BACK_RIGHT'
adar_back_right_data = nusc.get('sample_data', my_sample['data'][sensor])
adar_back_right_data
usc.render_sample_data(radar_back_right_data['token'])
ensor = 'RADAR_FRONT'
adar_front_data = nusc.get('sample_data', my_sample['data'][sensor])
adar_front_data
usc.render_sample_data(radar_front_data['token'])
ensor = 'RADAR_FRONT_LEFT'
adar_front_left_data = nusc.get('sample_data', my_sample['data'][sensor])
adar_front_left_data
usc.render_sample_data(radar_front_left_data['token'])
ensor = 'RADAR_FRONT_RIGHT'
adar_front_right_data = nusc.get('sample_data', my_sample['data'][sensor])
adar_front_right_data
usc.render_sample_data(radar_front_right_data['token'])

#4.
#(1)

e865152aaa194f22b97ad0078c012b21,
category: movable_object.barrier
sample_annotation_token:
7962506dbc24423aa540a5e4c7083dad,
category: movable_object.barrier
sample_annotation_token:
29cca6a580924b72a90b9dd6e7710d3e,
category: human.pedestrian.adult
sample_annotation_token:
a6f7d4bb60374f868144c5ba4431bf4c,
category: vehicle.car
sample_annotation_token:
f1ae3f713ba946069fa084a6b8626fbf,
category: movable_object.barrier
sample_annotation_token:
d7af8ede316546f68d4ab4f3dbf03f88,
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:

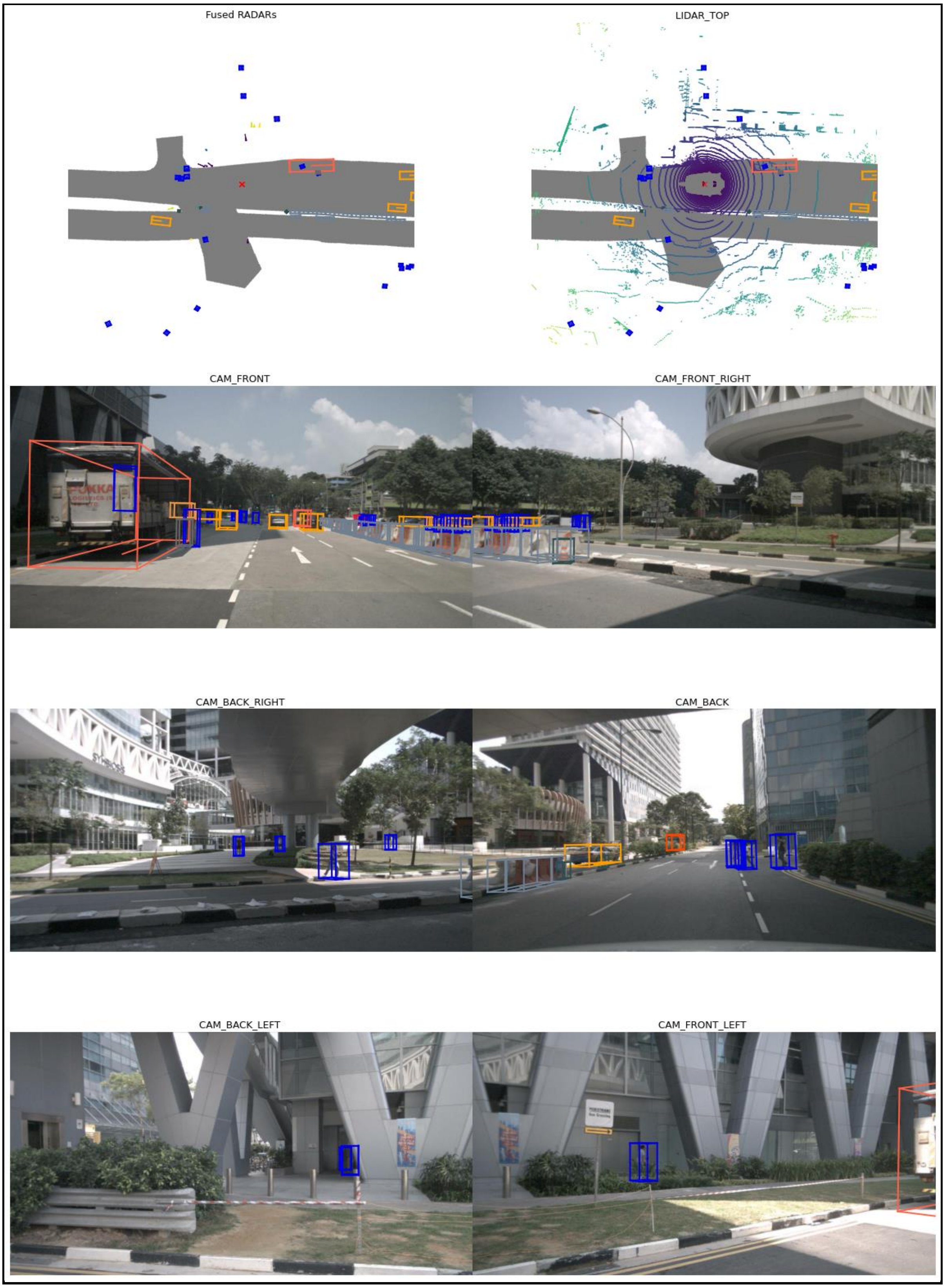



*Figure 1*

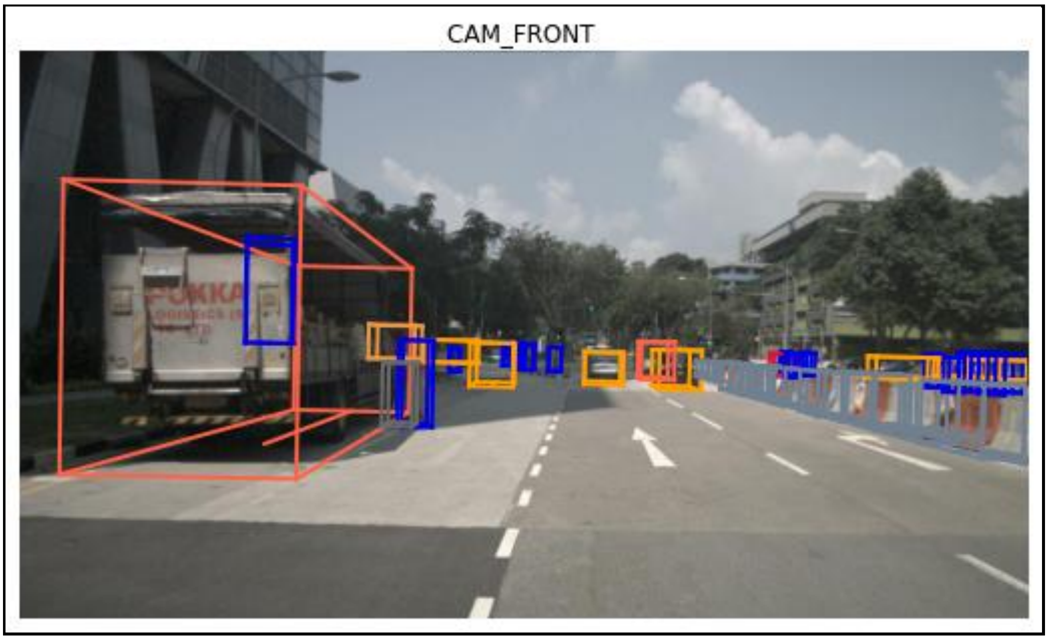



*Figure 2*

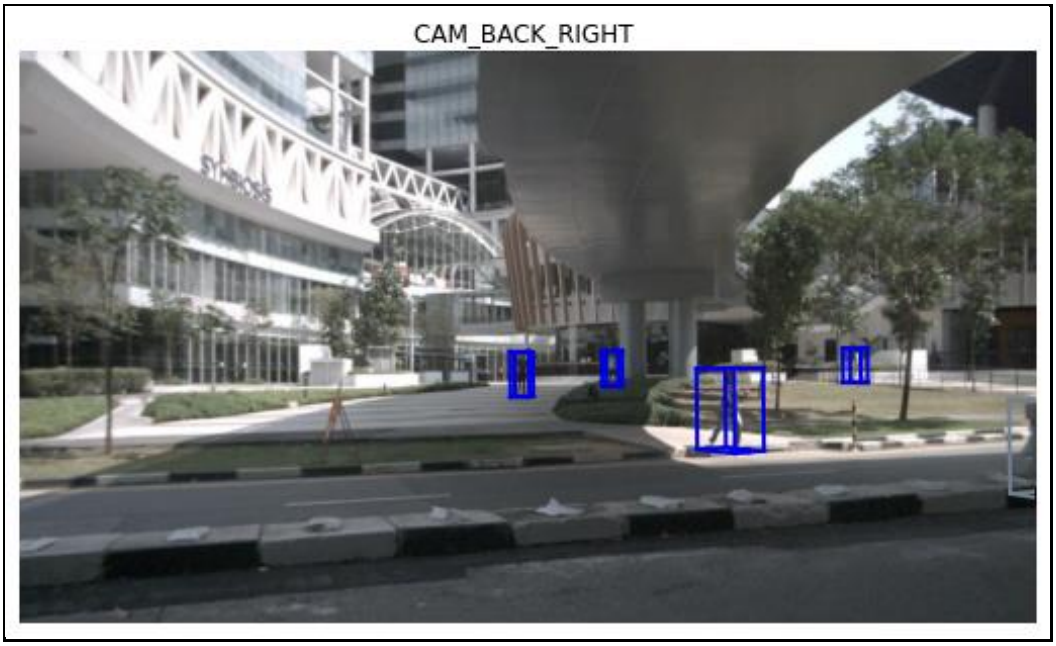



*Figure 3*

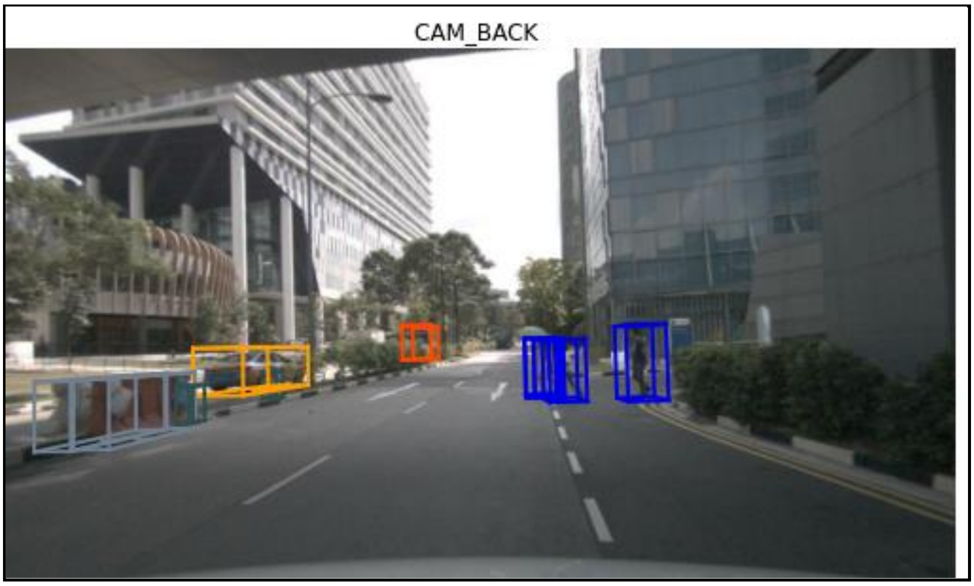



*Figure 4*

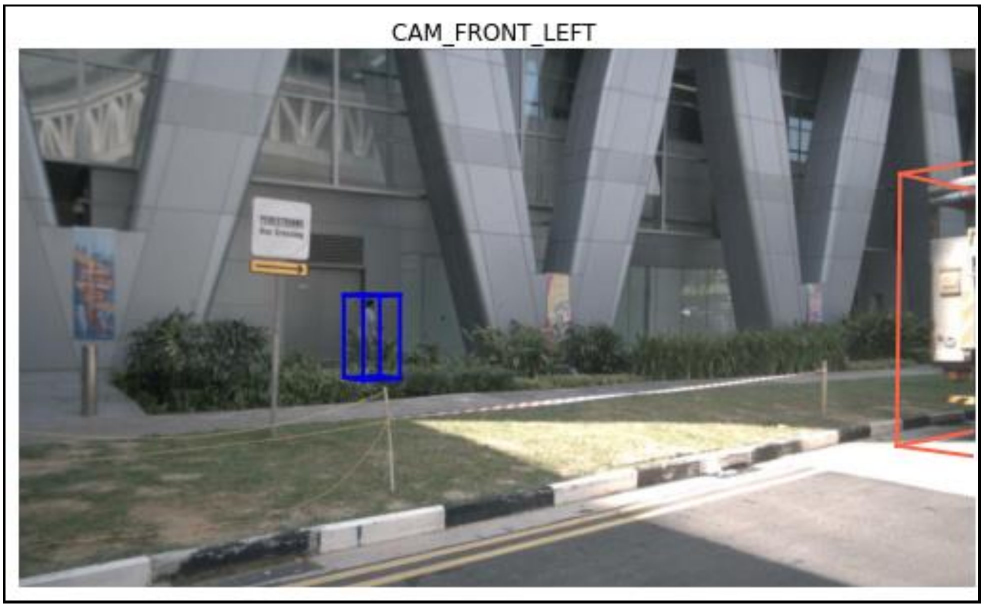



*Figure 5*

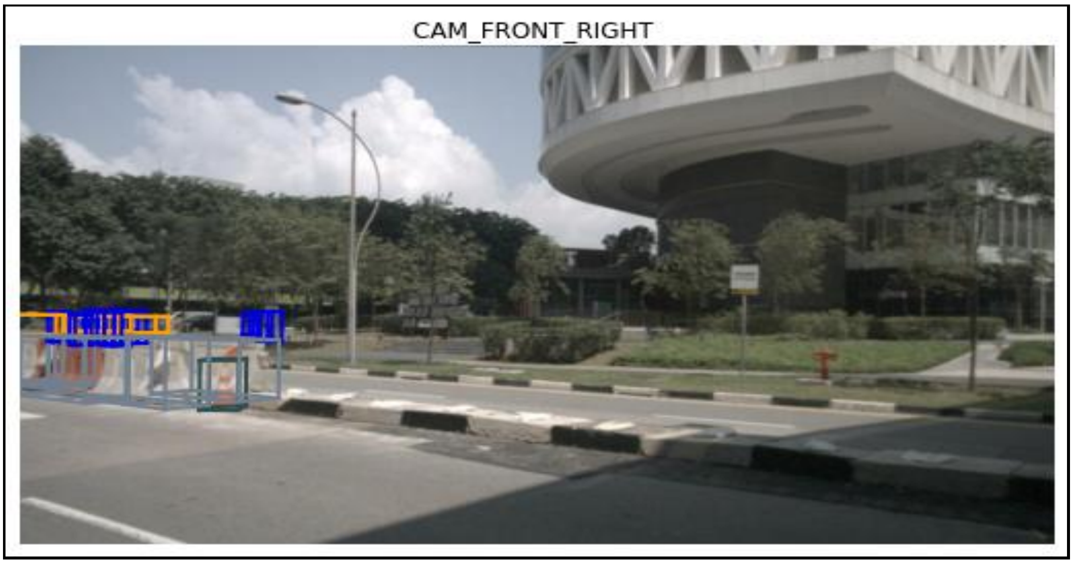



*Figure 6*

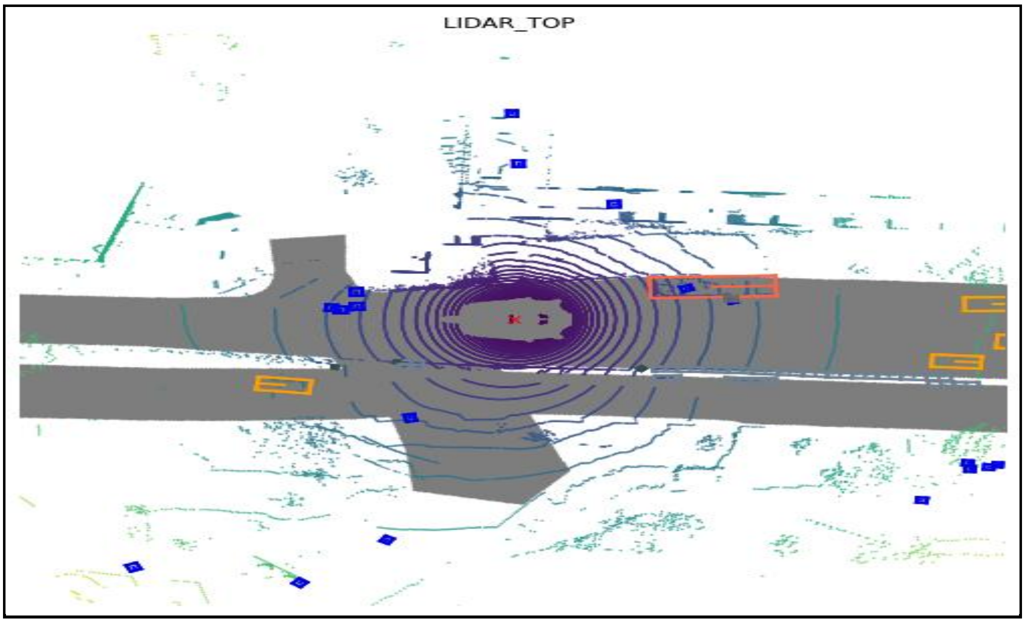



*Figure 7*

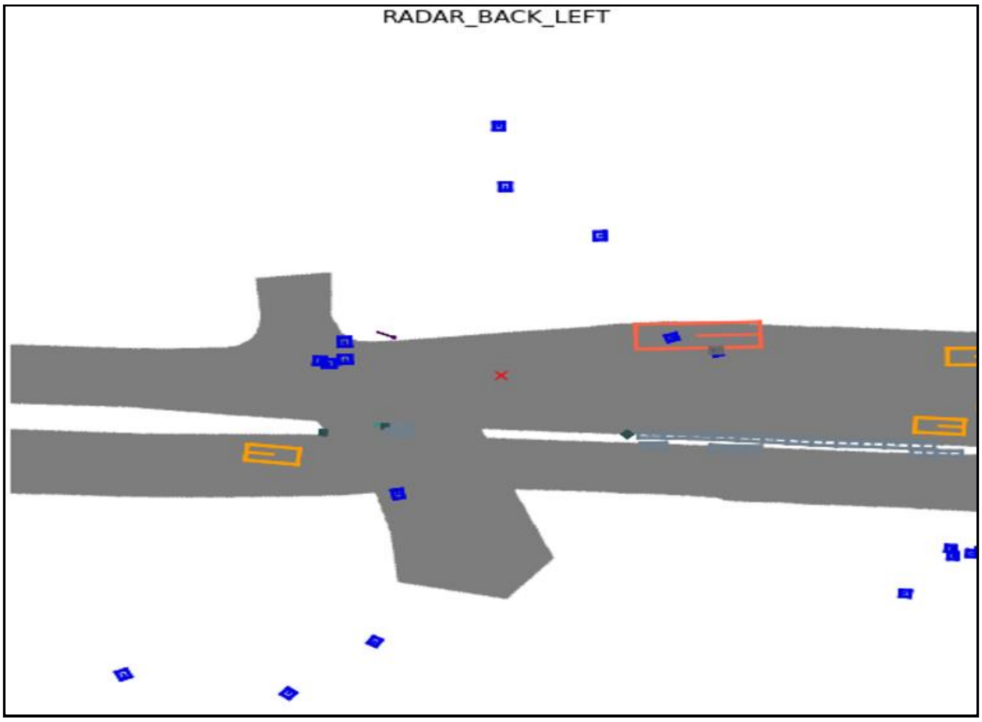



*Figure 8*

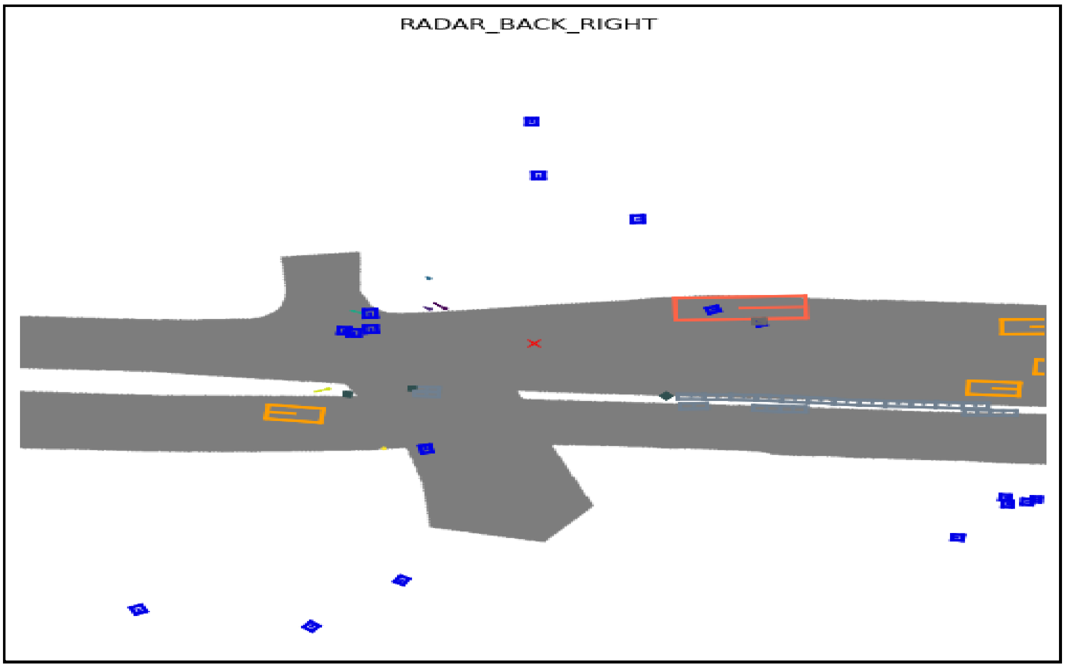



*Figure 9*

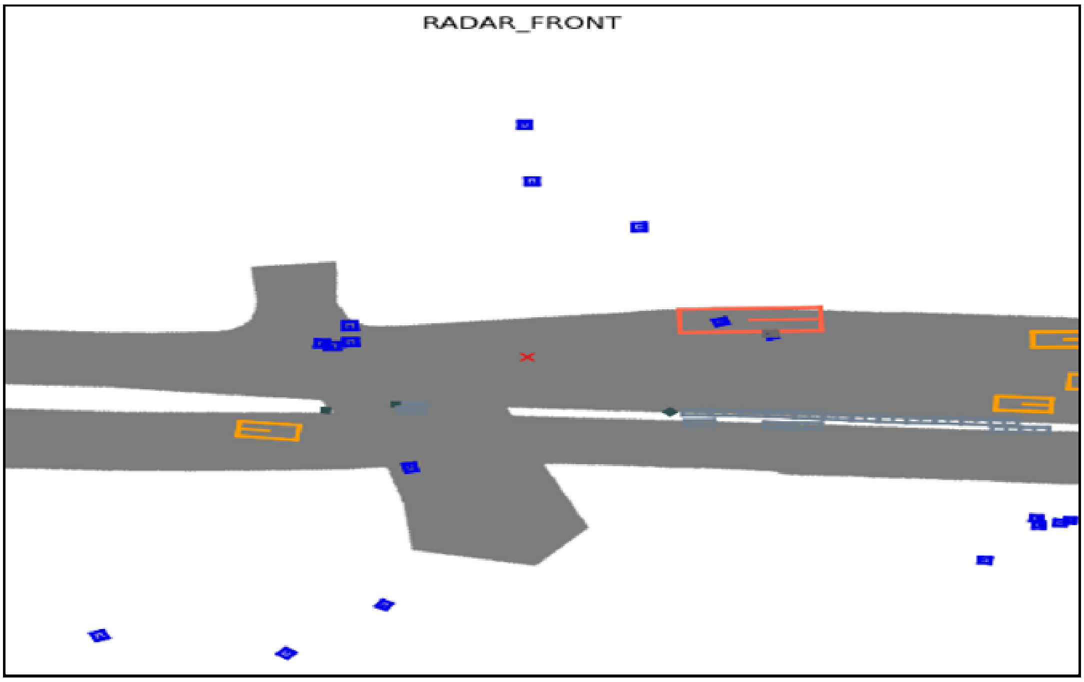



*Figure 10*

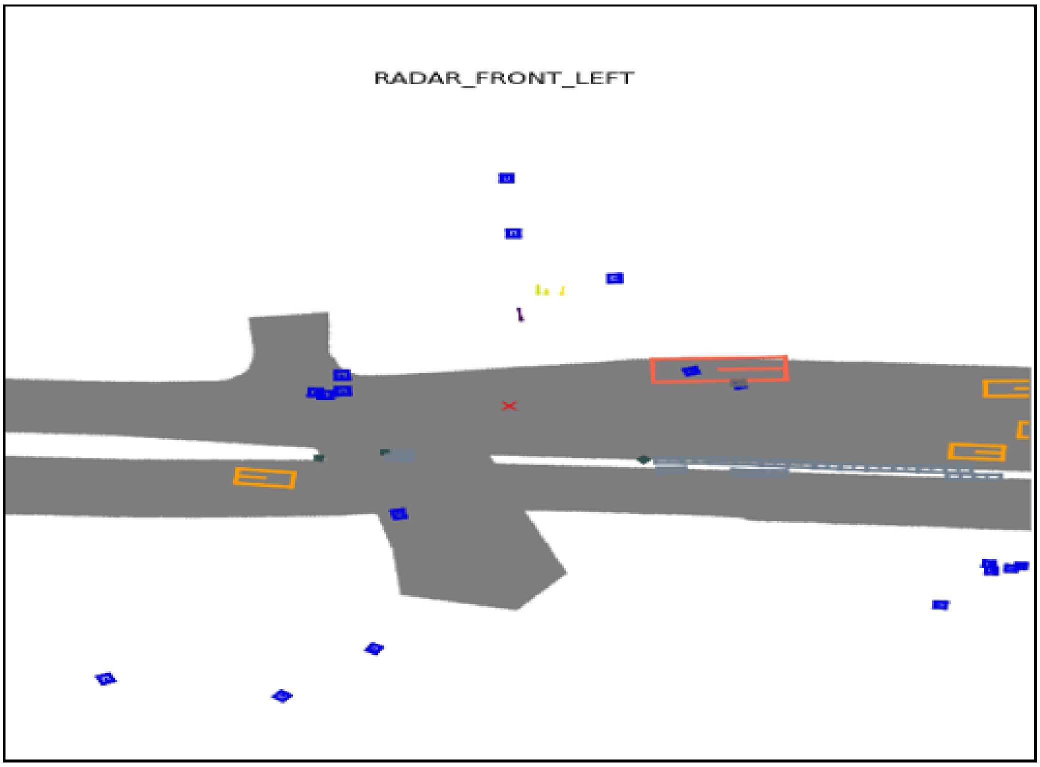



*Figure 11*

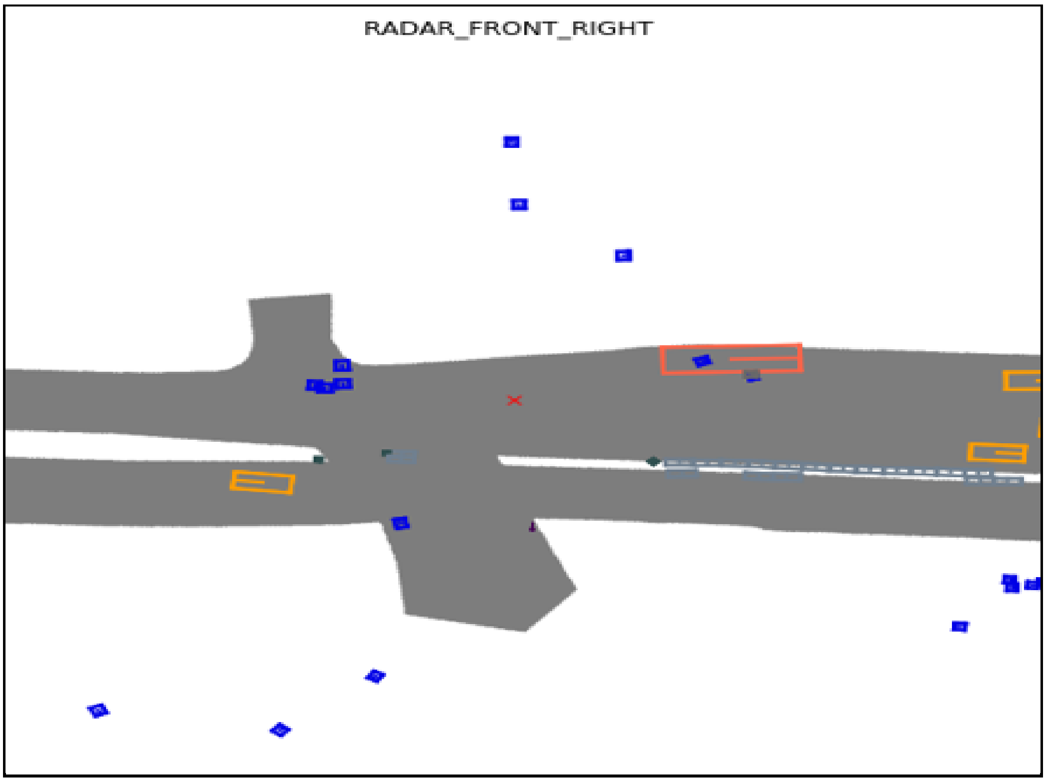



*Figure 12*

2. 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, etcl) 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:

```python
#4.
#(1)
import cv2

# Load an color image
img = cv2.imread('C:/Users/siddh/OneDrive/Pictures/Screenshots/Documents/NuScenes/image.jpg')

# Show image
cv2.imshow('C:/Users/siddh/OneDrive/Pictures/Screenshots/Documents/NuScenes/image.jpg',img)
cv2.waitKey(0)
cv2.destroyAllWindows()

#(2)
import numpy as np
import open3d as o3d
import cv2, random,os
```

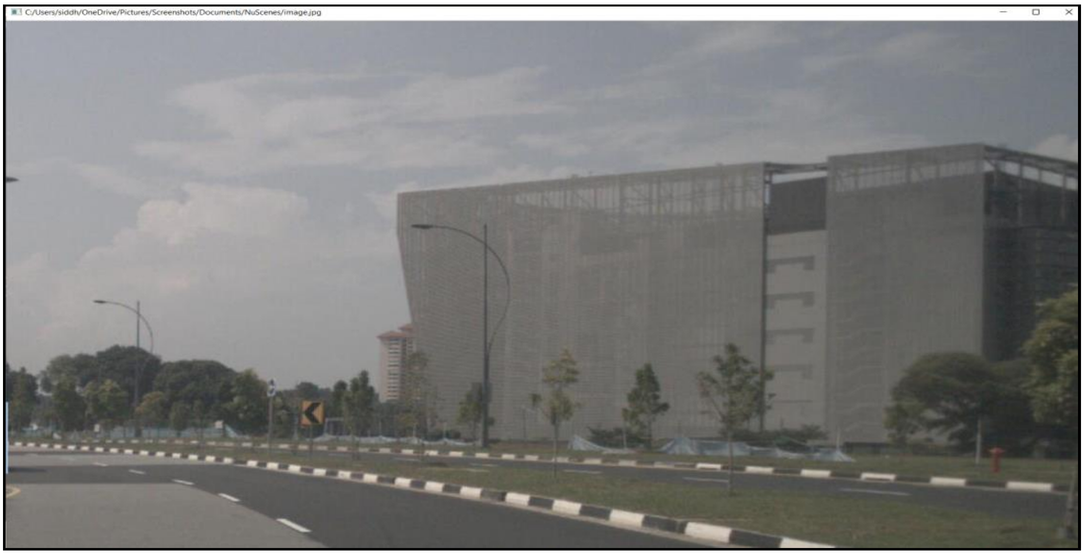



*Figure 13*

(2) Visualize Lidar point cloud data:

e. You can refer to this [sample code](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](here).
   iii. You can get semantic label from the sample above code.

e. Program:

```python
@author: siddh
"""

import open3d as o3d
import numpy as np

seg_name="C:/Users/siddh/OneDrive/Pictures/Screenshots/Documents/NuScenes/lidarseg/v1.0-mini/0ab9ec2730894df2b48df70d0d2e84a9_lidarseg.bin"
seg=np.fromfile(seg_name, dtype=np.uint8)

color = np.zeros([len(seg), 3])
color[:, 0] = seg/32
color[:, 1] = seg/32
color[:, 2] = seg/32

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"
scan=np.fromfile(pcd_name, dtype=np.float32)
points = scan.reshape((-1, 5))[:, :4]

pcd = o3d.geometry.PointCloud()
pcd.points = o3d.utility.Vector3dVector(points[:, :3])
pcd.colors = o3d.utility.Vector3dVector(color)

o3d.visualization.draw_geometries([pcd])
```

Visualization of Lidar Image:

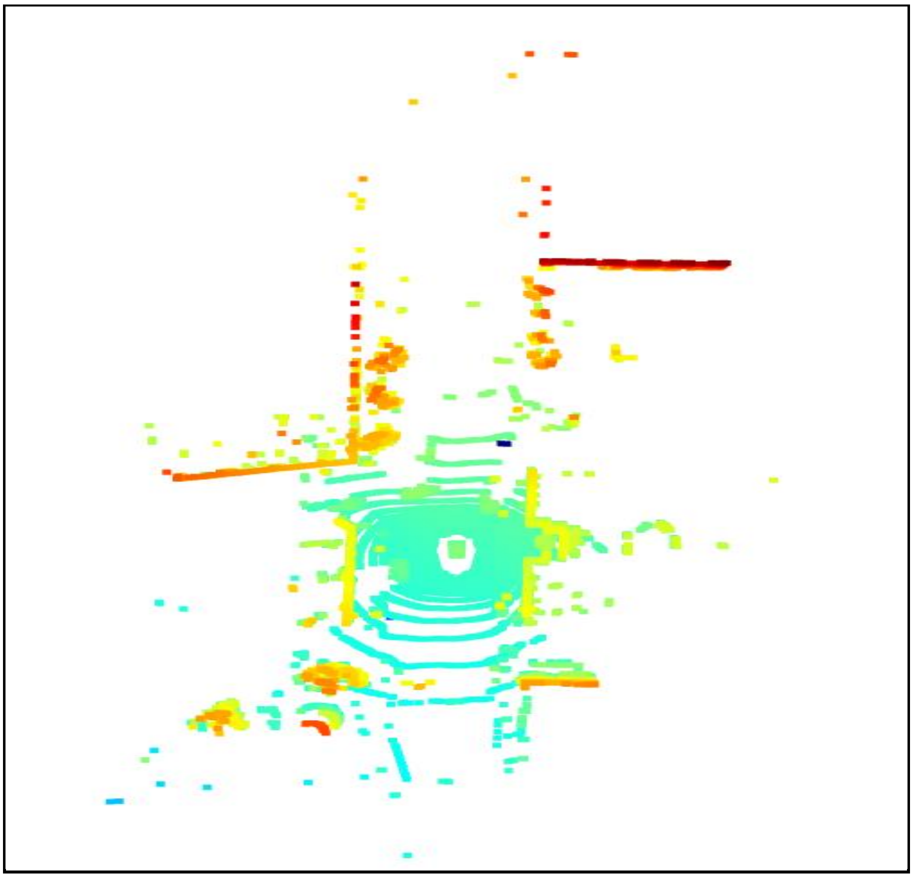


*Figure 14*

f. Program:

```python

#--------------------------------------Q.4.2

#Define PCD Path
input_path= r"C:/Users/siddh/OneDrive/Pictures/Screenshots/Documents/NuScenes/samples/LIDAR_TOP"

#Picking Random file
dataname = random.choice([
    x for x in os.listdir(input_path)
    if os.path.isfile(os.path.join(input_path, x))
])
Filename_rand= (os.path.join(input_path, dataname))
print('Displayed image is ',Filename_rand)

#Converting PCD to points
pcd_data=Filename_rand
data=np.fromfile(pcd_data, dtype=np.float32)
pointsdata = data.reshape((-1, 5))[:, :4]
print(pointsdata)

#Value of Z
Z=pointsdata[:,2]
print('The value of Z is ',Z)

#Value of Intensity
intensity=pointsdata[:,3]
print('The value of intensity is ',intensity)

#For height
color = np.zeros([len(Z), 3])
color[:, 0] = Z[1]/10
color[:, 1] = 0.5
color[:, 2] = 0.5


pcdimage = o3d.geometry.PointCloud()
pcdimage.points = o3d.utility.Vector3dVector(pointsdata[:, :3])
pcdimage.colors = o3d.utility.Vector3dVector(color)
o3d.visualization.draw_geometries([pcdimage])

#For Intensity
color = np.zeros([len(intensity), 3])
color[:, 0] = intensity[1]/10
color[:, 1] = 0.5
color[:, 2] = 0.5


pcdimage = o3d.geometry.PointCloud()
pcdimage.points = o3d.utility.Vector3dVector(pointsdata[:, :3])
pcdimage.colors = o3d.utility.Vector3dVector(color)
o3d.visualization.draw_geometries([pcdimage])

#Visualize by Segement
sem=np.fromfile(Filename_rand, dtype=np.uint8)


color = np.zeros([len(sem), 3])
color[:, 0] = sem/20
color[:, 1] = 0.5
color[:, 2] = 0.5
pcdimage = o3d.geometry.PointCloud()
pcdimage.points = o3d.utility.Vector3dVector(pointsdata[:, :3])
pcdimage.colors = o3d.utility.Vector3dVector(color)
o3d.visualization.draw_geometries([pcdimage])

```

Visualizations of h\Height, Intensity, Segmantic label:

a. Height:

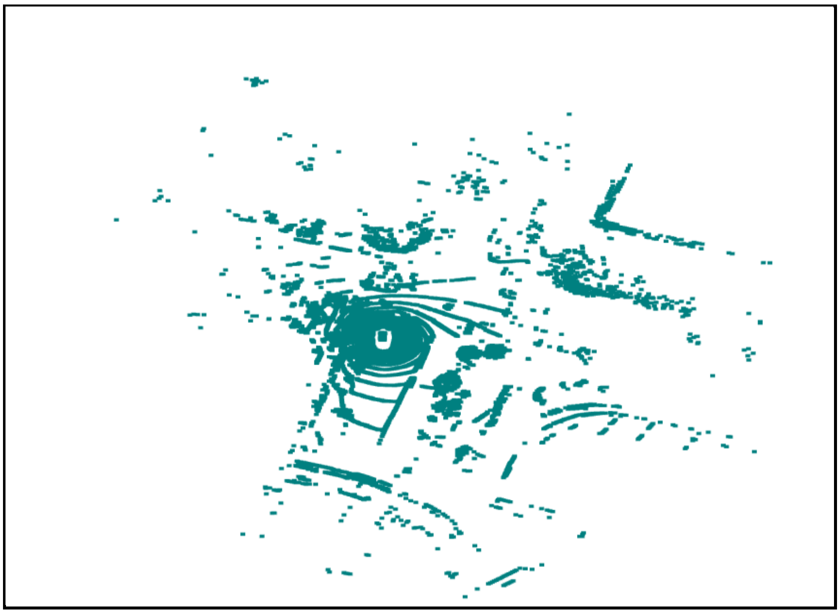


*Figure 15*

b. Intensity:

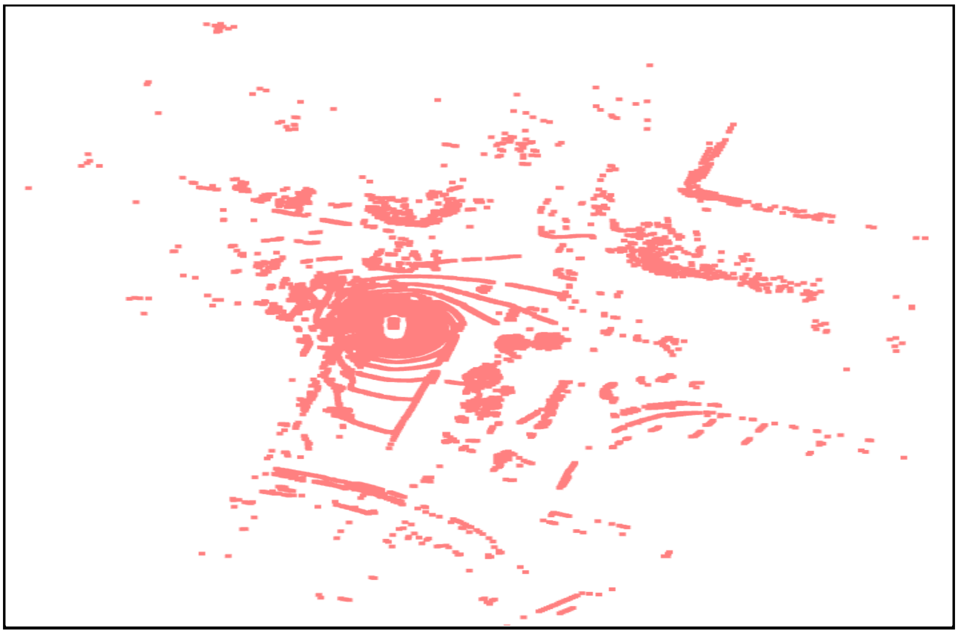


*Figure 16*

c. Segmantic label:

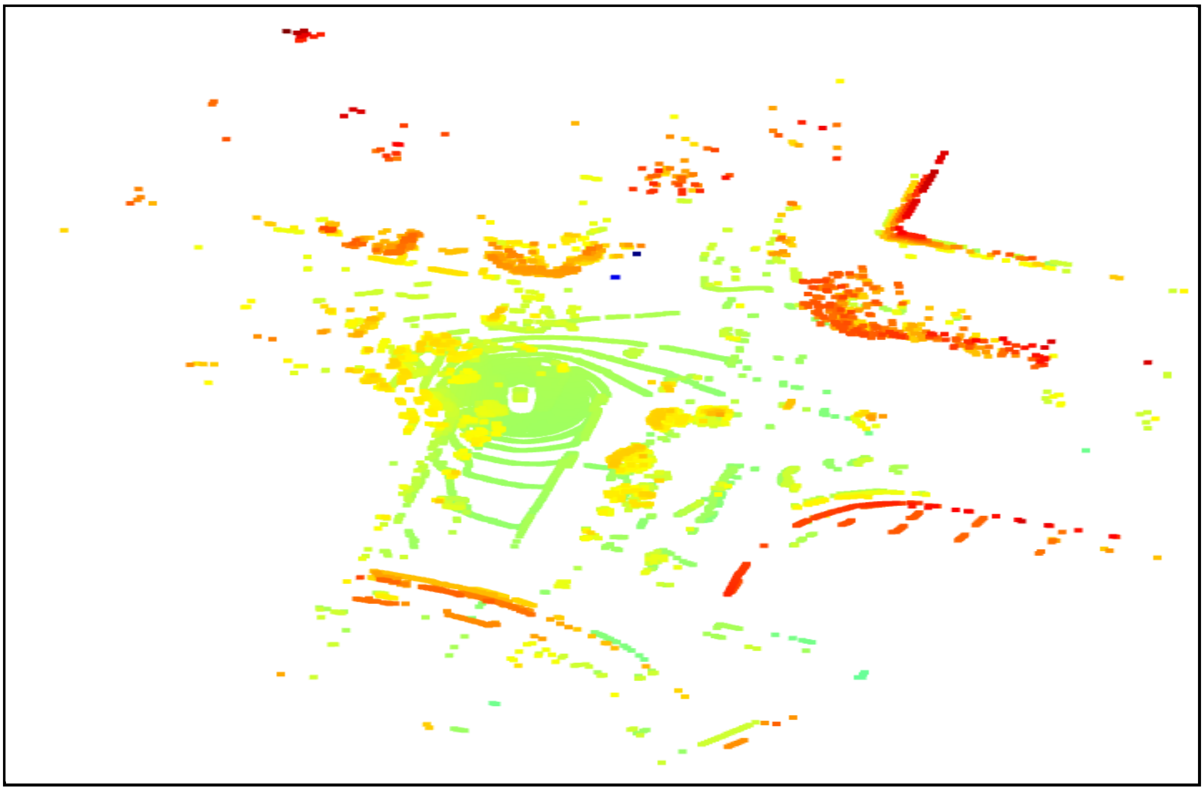


*Figure 17*

(4) Visualize Radar data

g. Use any other library (e.g., Open3D, PCL, etcl) 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](here).

g. Use any other library (e.g., Open3D, PCL, etcl) 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
```

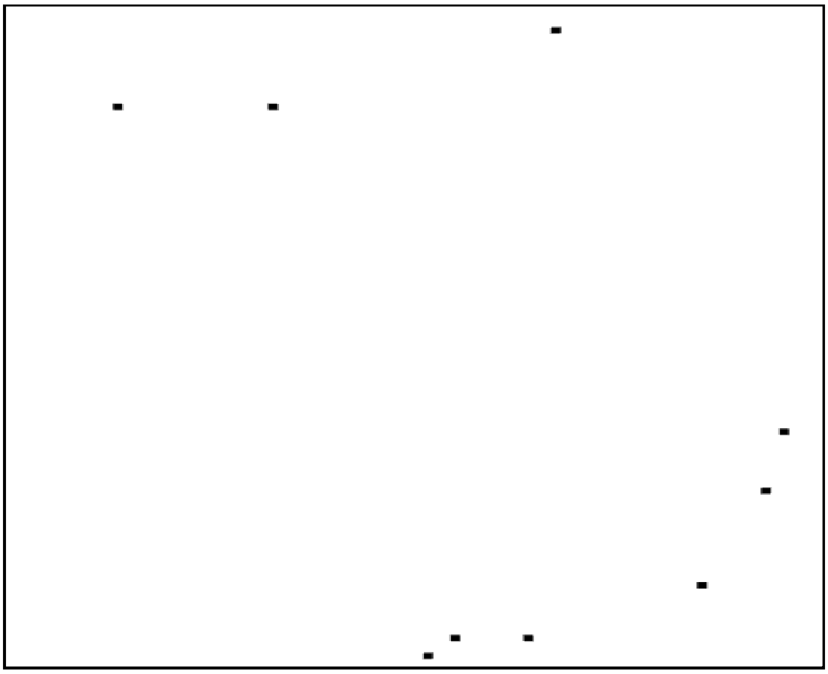


*Figure 18*

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:

```
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

nusc = NuScenes(version='v1.0-mini', dataroot='C:/Users/siddh/OneDrive/Pictures/Screenshots/Documents/NuScenes', verbose=True)
random_number=random.randint(0,100)
my_sample = nusc.sample[random_number]

pointsensor_token = my_sample['data']['RADAR_FRONT']
pointsensor = nusc.get('sample_data', pointsensor_token)
pcl_path = osp.join(nusc.dataroot, pointsensor['filename'])
#Radar plot

pc = RadarPointCloud.from_file(pcl_path)
data=np.asarray(pc.points.astype(dtype=np.float32()))

pointsdata = data.reshape((-1, 18))[:, :17]
print(pointsdata)

#Height
Z=data[2]


#Velocity
vx=data[6]
vy=data[7]

#For height
color = np.zeros([len(Z), 3])
color[:, 0] = Z/10
color[:, 1] = 0.5
color[:, 2] = 0.5


pcdimage = o3d.geometry.PointCloud()
pcdimage.points = o3d.utility.Vector3dVector(pointsdata[:, :3])
pcdimage.colors = o3d.utility.Vector3dVector(color)
o3d.visualization.draw_geometries([pcdimage])
```

Visualization of height:

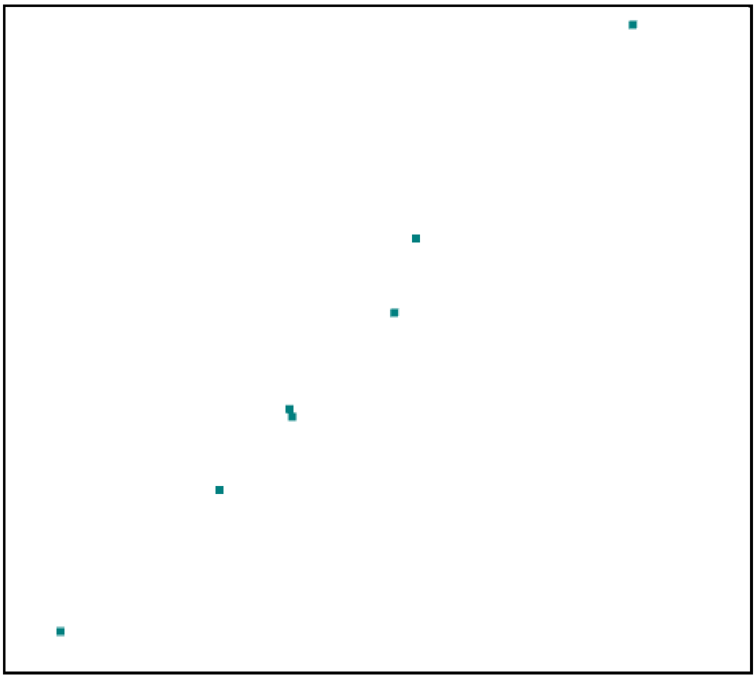


*Figure 19*

Program:

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

nusc = NuScenes(version='v1.0-mini', dataroot='C:/Users/siddh/OneDrive/Pictures/Screenshots/Documents/NuScenes', verbose=True)
random_number=random.randint(0,100)
my_sample = nusc.sample[random_number]

pointsensor_token = my_sample['data']['RADAR_FRONT']
pointsensor = nusc.get('sample_data', pointsensor_token)
pcl_path = osp.join(nusc.dataroot, pointsensor['filename'])
#Radar plot

pc = RadarPointCloud.from_file(pcl_path)
#data=np.asarray(pc.points.astype(dtype=np.float32()))

data = pc.points.T

pointsdata = data.reshape((-1, 18))[:, :17]
print(pointsdata)

#Height
#Z=data[2]


#Velocity
vx=data[:,8]
vy=data[:,9]

#For height
color = np.zeros([len(vx), 3])
color[:, 0] = vx/10
color[:, 1] = vy/10
color[:, 2] = 0.5

pcdimage = o3d.geometry.PointCloud()
pcdimage.points = o3d.utility.Vector3dVector(pointsdata[:, :3])
pcdimage.colors = o3d.utility.Vector3dVector(color)
o3d.visualization.draw_geometries([pcdimage])
print('done')
```

Visualization of velocity:

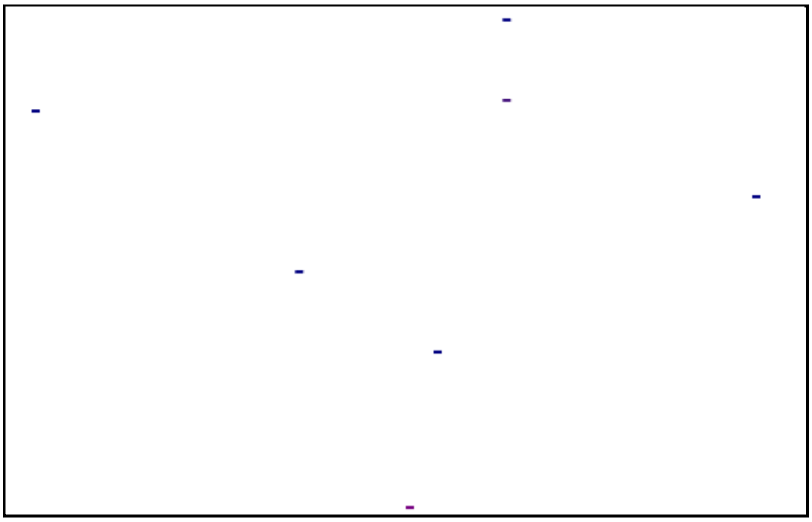


*Figure 20*

3. Using NuScene dev-kit for the set of data which you picked up:
   (1) Visualize Radar data projection on image
      a. Print calibration info (between Radar and Camera sensors) by referring code [here](here).
      b. Explain the above calibration info, and pipeline of First~Fifth steps in the code.
      c. 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
ensor = 'RADAR_BACK_LEFT'
_r_s = nusc.get('sample_data', my_sample['data'][sensor])
_r_s = nusc.get('calibrated_sensor', t_r_s["calibrated_sensor_token"])
rint("RADAR_BACK_LEFT translation_values is: ",c_r_s["translation"])
rint("RADAR_BACK_LEFT rotation_values is: ",c_r_s["rotation"],"\n")
 Step-2: Transfomation of data points from ego vehicle frame to global frame
_r_s = nusc.get('sample_data', my_sample['data'][sensor])
_r_s = nusc.get('ego_pose',t_r_s["ego_pose_token"])
rint("e_translation_values is: ",e_r_s["translation"])
rint("e_rotation_values is: ",e_r_s["rotation"],"\n")
 Step-3: Transformation of data points from global frame to ego vehicle frame
ensor = 'CAM_BACK_LEFT'
_c_s = nusc.get('sample_data', my_sample['data'][sensor])
_c_s = nusc.get('calibrated_sensor', t_c_s["calibrated_sensor_token"])
rint("CAM_BACK_LEFT translation_values is: ",c_c_s["translation"])
rint("CAM_BACK_LEFT rotation_values is: ",c_c_s["rotation"],"\n")
 Step-4: Transformation of data points from ego vehicle frame to camera
_s = nusc.get('sample_data', my_sample['data'][sensor])
_c_s = nusc.get('ego_pose',t_c_s["ego_pose_token"])
rint("e_translation_values is: ",e_c_s["translation"])
rint("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:  [-0.9397410032054776, -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.

```python
import random
from nuscenes.nuscenes import NuScenes
from Nusc_file import NuScenesExplorer


nusc = NuScenes(version='v1.0-mini', dataroot='C:/Users/siddh/OneDrive/Pictures/Screenshots/Documents/NuScenes', verbose=True)
random_number=random.randint(0,100)
my_sample = nusc.sample[random_number]


#Radar plot
nusc.render_pointcloud_in_image(my_sample['token'], pointsensor_channel='RADAR_FRONT')
NuScenesExplorer(nusc).map_pointcloud_to_image(my_sample['data']['RADAR_FRONT'],my_sample['data']['CAM_FRONT'])
```

*Figure 21*

- ➢ 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])
c_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: Transfomation ofdata points frpom 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])
c_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 ofdata 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.

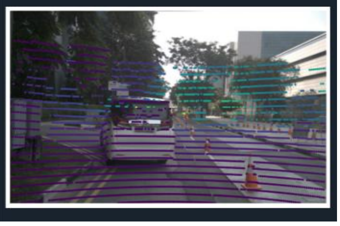
e. Visualize LiDAR data projection on image based on calibration info.

Program:

```
import random
from nuscenes.nuscenes import NuScenes
from Nusc_file import NuScenesExplorer


nusc = NuScenes(version='v1.0-mini', dataroot='C:/Users/siddh/OneDrive/Pictures/Screenshots/Documents/NuScenes', verbose=True)
random_number=random.randint(0,100)
my_sample = nusc.sample[random_number]

#lidar plot
nusc.render_pointcloud_in_image(my_sample['token'], pointsensor_channel='LIDAR_TOP', render_intensity=False)
NuScenesExplorer(nusc).map_pointcloud_to_image(my_sample['data']['LIDAR_TOP'],my_sample['data']['CAM_FRONT'])
```



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

*Figure 22*