# Homework 1

Student Name: Raja Manas Macherla

AuE 8930: Machine Perception and Intelligence
Instructor: Dr. Bing Li, Clemson University, Department of Automotive Engineering

* Refer to Syllabus for homework grading, submission and plagiarism policies;
* Submission to Canvas (Due: Tues. Jan. 19, 2021 11:59 pm), including:

- This document (with answers), and with your program results/visualization;
  For this homework, you may put the screenshots of the results in the submission document.
- A .zip file of source code (and data if any) with names indicating question number;

* You can find some sample codes from the course GitHub Repo;

1. Download NuScene dataset, you may need to register on that website. To save time, you can only download the mini set with its Lidarseg file. (5 points)

2. Set up the NuScene develop kit locally, you may need to install Anaconda and Jupyter notebook. (5 points)

3. Pickup a set of data, including Image, Lidar, and Radar data. Visualize them with NuScene dev-kit tutorial notebooks, reference code (which is Jupyter notebook Python). (20 points)

4. Rather than using the dev-kit, implement (total 55 points):
   (1) Visualize images (using OpenCV or others you prefer), Sample code. (5 points)
   (2) Visualize Lidar point cloud data
      a. You can refer to this sample code.
      b. Colorize points by height, intensity, and semantic label respectively. (20 points)
         i. Height is the Z value for a point.
         ii. You can get intensity from here.
         iii. You can get semantic label from the sample code.
   (3) Visualize Radar data
      a. Use another library or modify the previous sample code to make it works for the visualization of the Radar data which you picked up. (10 points)
      b. Colorize points by height and velocity respectively. (20 points)
         iv. You can find some velocity information from here.

5. Using NuScene dev-kit for the set of data which you picked up: (15 points)
   (1) Visualize Radar data projection on image (10 points).
      a. Print calibration info (between Radar and Camera sensors) by referring here.
      b. Explain the above calibration info, and the pipeline of First step ~ Fifth step.
   (2) Visualize LiDAR data projection on image (5 points).
      a. Print calibration info (between LiDAR and Camera sensors) by referring here.

Answers:

1. Downloaded the nuscenes dataset onto my laptop using wget command.

2. Set up an anaconda environment and ran the command “pip install nuscenes-devkit”

3. Visualizing camera data, radar data and lidar data using Nuscenes dataset dev-kit

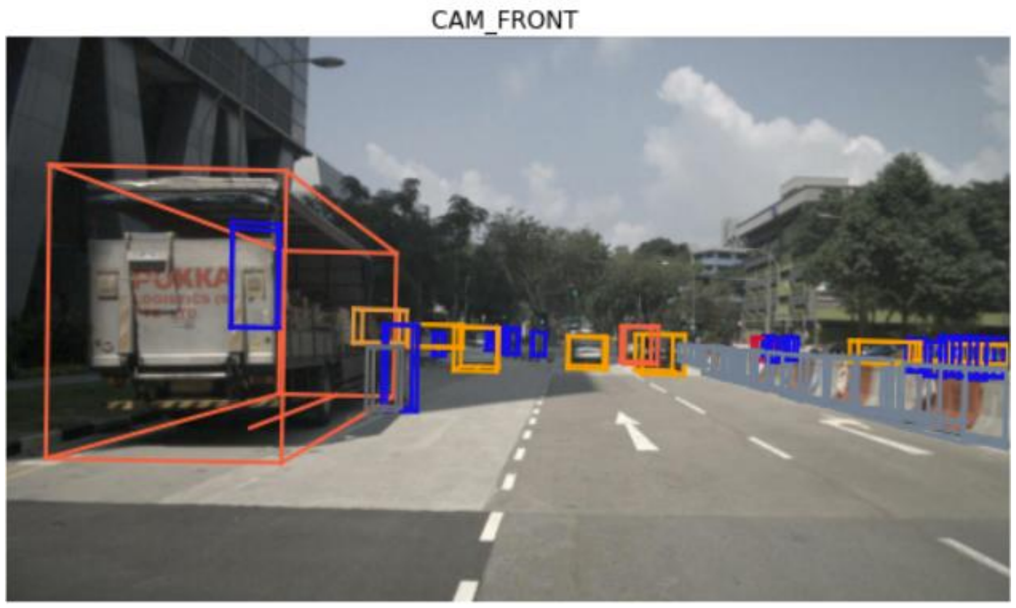


*Figure 1: Front camera image visualization using Nuscenes devkit*

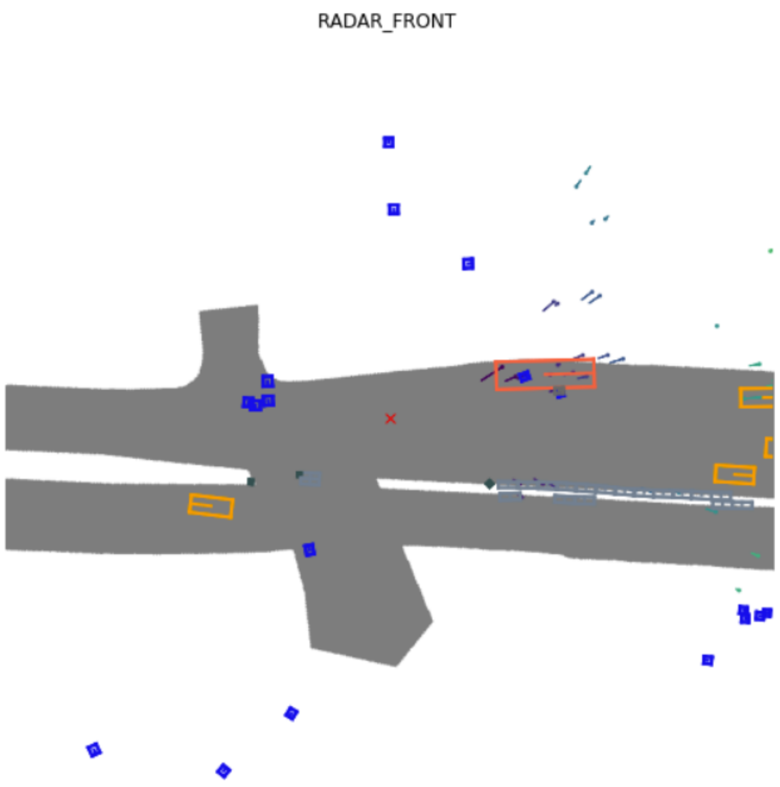


*Figure 2: Front radar data visualization using Nuscenes devkit*

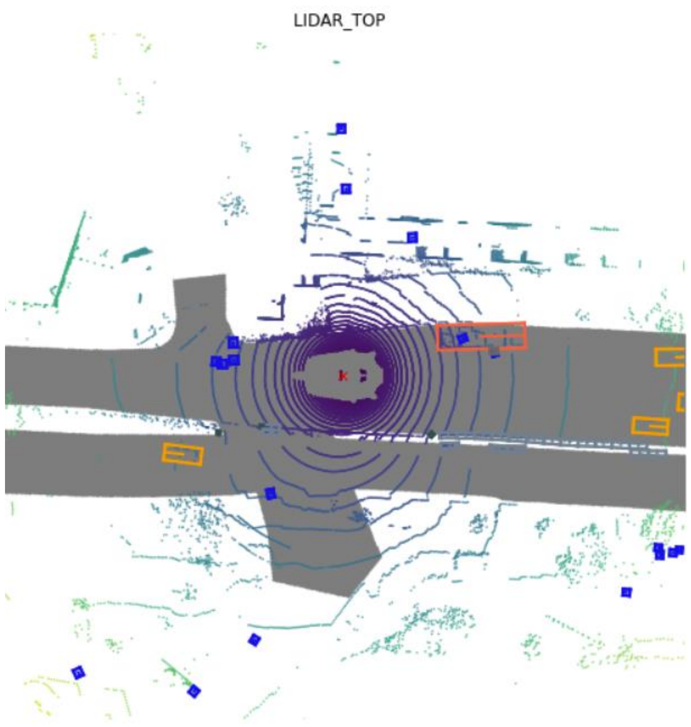


*Figure 3: Top lidar data visualization using Nuscenes devkit*

4.1. Visualizing image data using matplotlib:

```
<matplotlib.image.AxesImage at 0x14effc7a64c0>
```

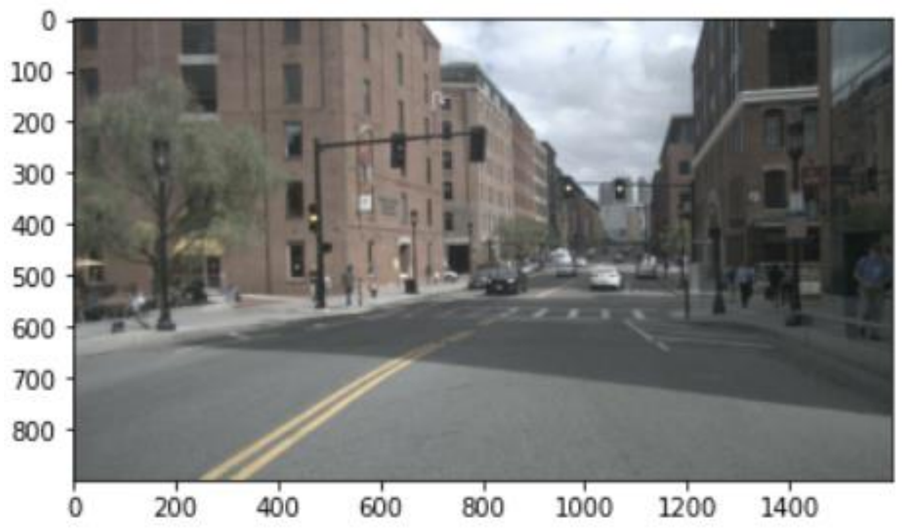


*Figure 4: Visualization of image data using Matplotlib*

4.2.a Visualizing lidar point cloud data:

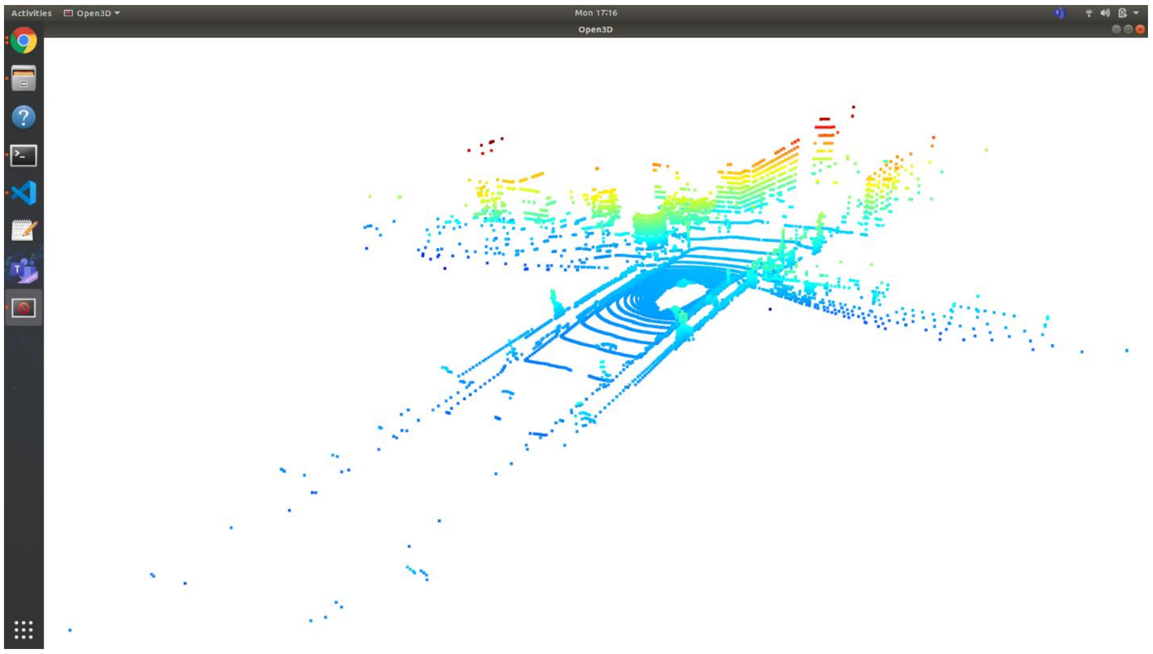


*Figure 5: Visualization of Lidar data using Open3d*

4.2.b.i) I colorized points based on just Red and Blue colors. The lowest values will have a more blue in it and the highest values will have more red in it. Below is the color based on height:

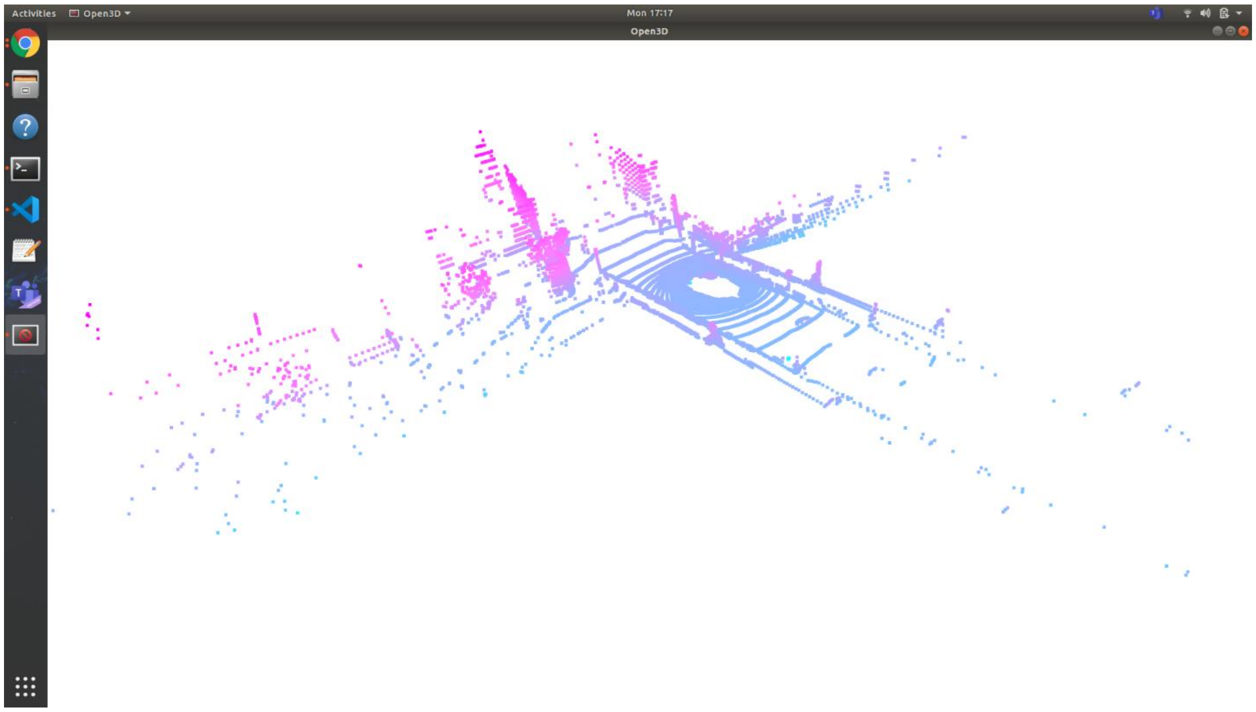


Figure 6: Coloring of Lidar data based on height

4.2.b.ii) Coloring based on intensity.

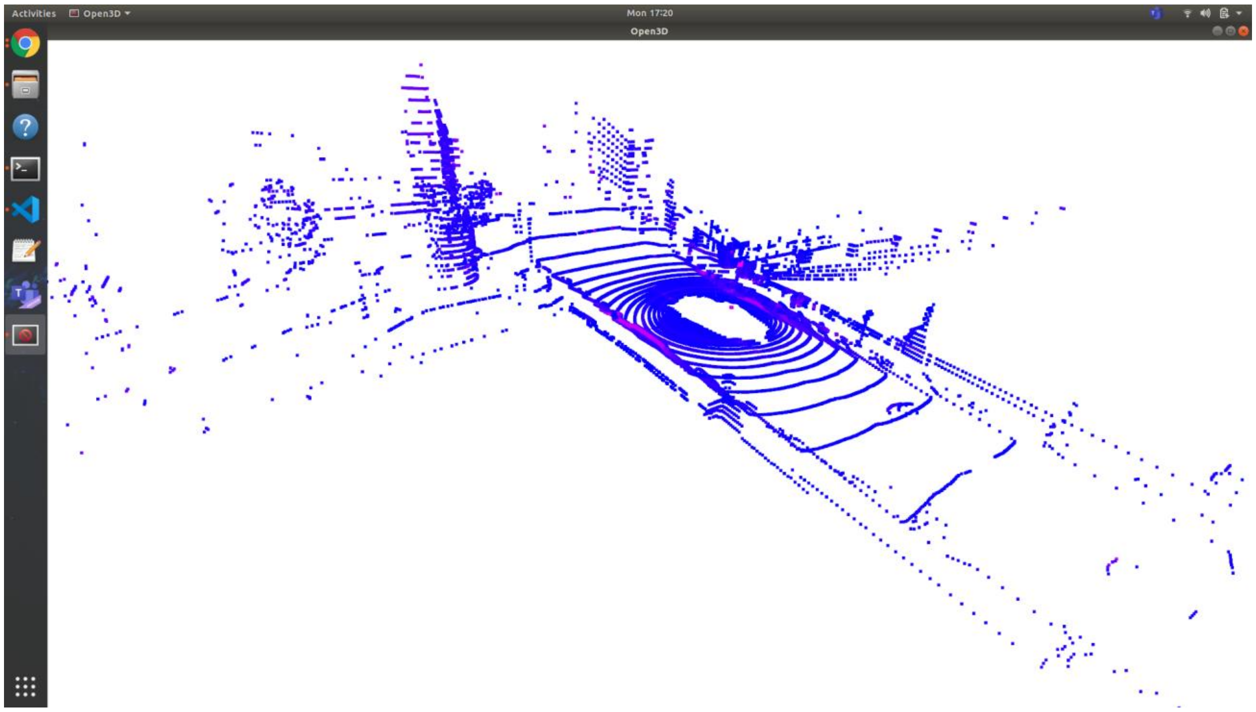


Figure 7: Coloring of Lidar data based on intensity of the points

4.2.b.iii) Coloring based on semantic label of the points.

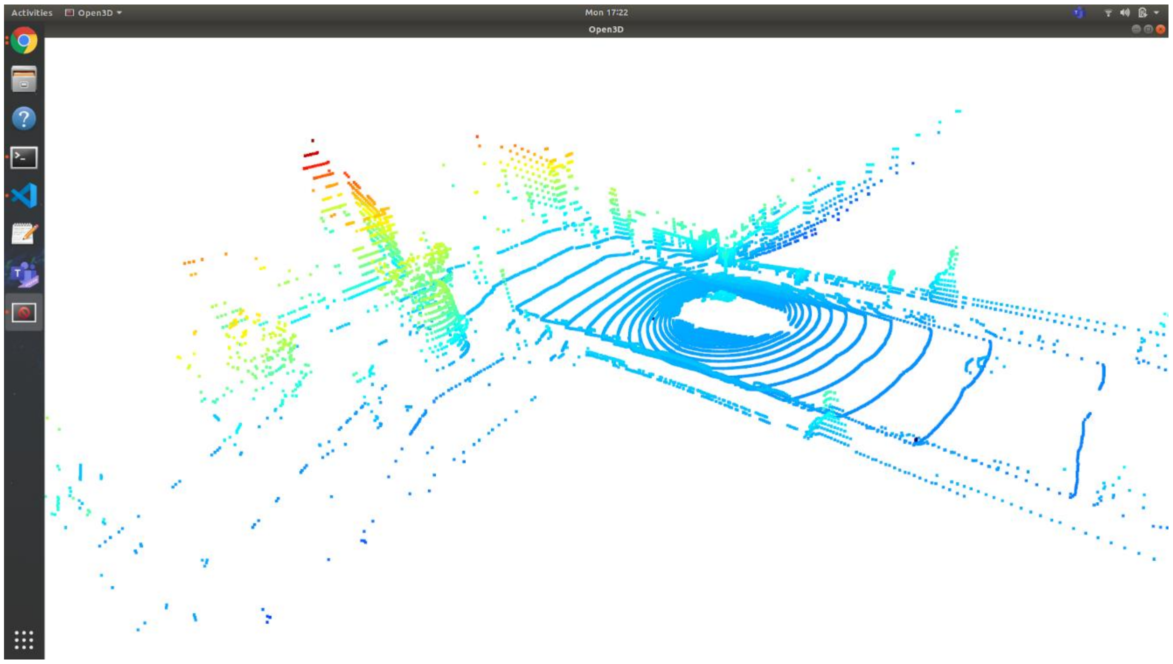


*Figure 8: Coloring based on semantic label of the points*

4.3.a) Visualizing radar pcd using Open3D.

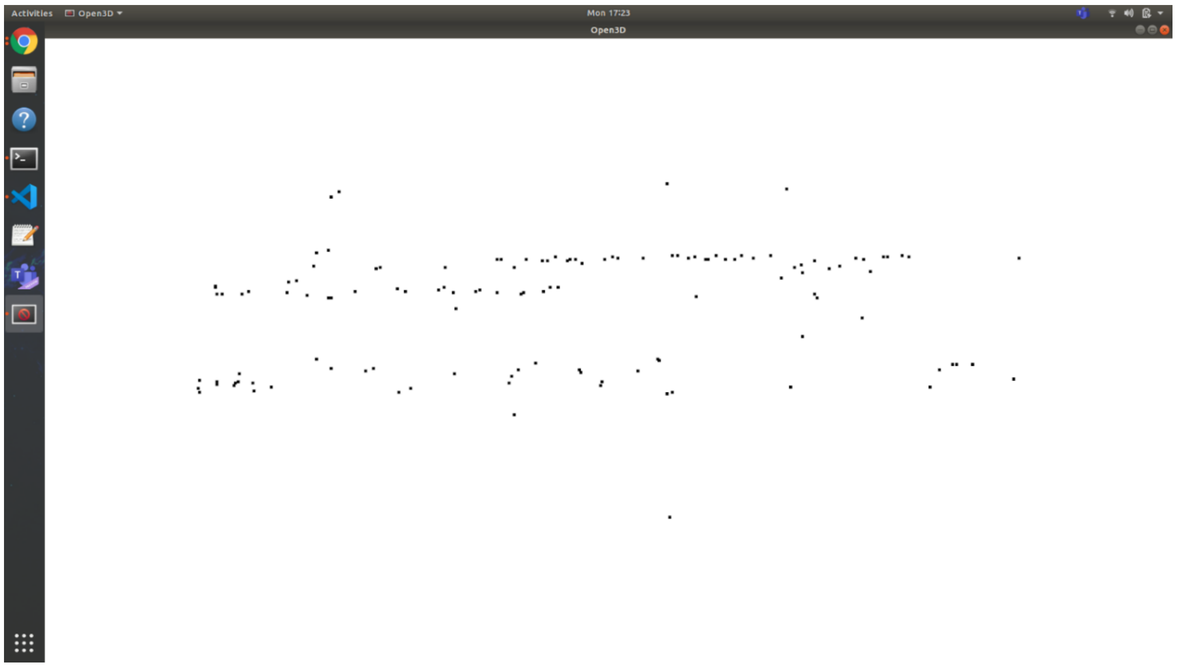


*Figure 9: Visualization of radar data using Open3d (points are black due to the lack of height information)*

4.3.b.iv) Coloring the radar pcd using height values was done but the from_file function somehow does not give height information and height values were printed, a zero vector was shown. Since all the values are zero, all the points end up being black as the picture above.

4.3.b.v) Coloring radar pcd based on vx_comp and vy_comp:

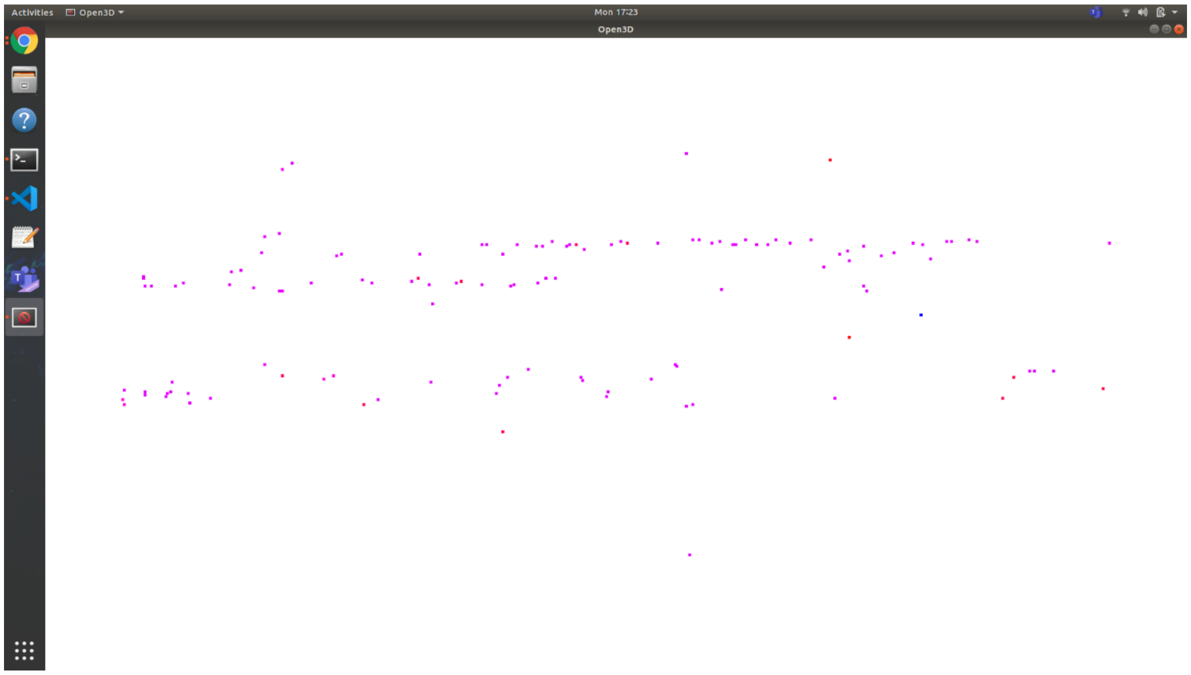


*Figure 10: Coloring of radar data based on the velocity in X direction that is compensated by the velocity of the ego vehicle.*

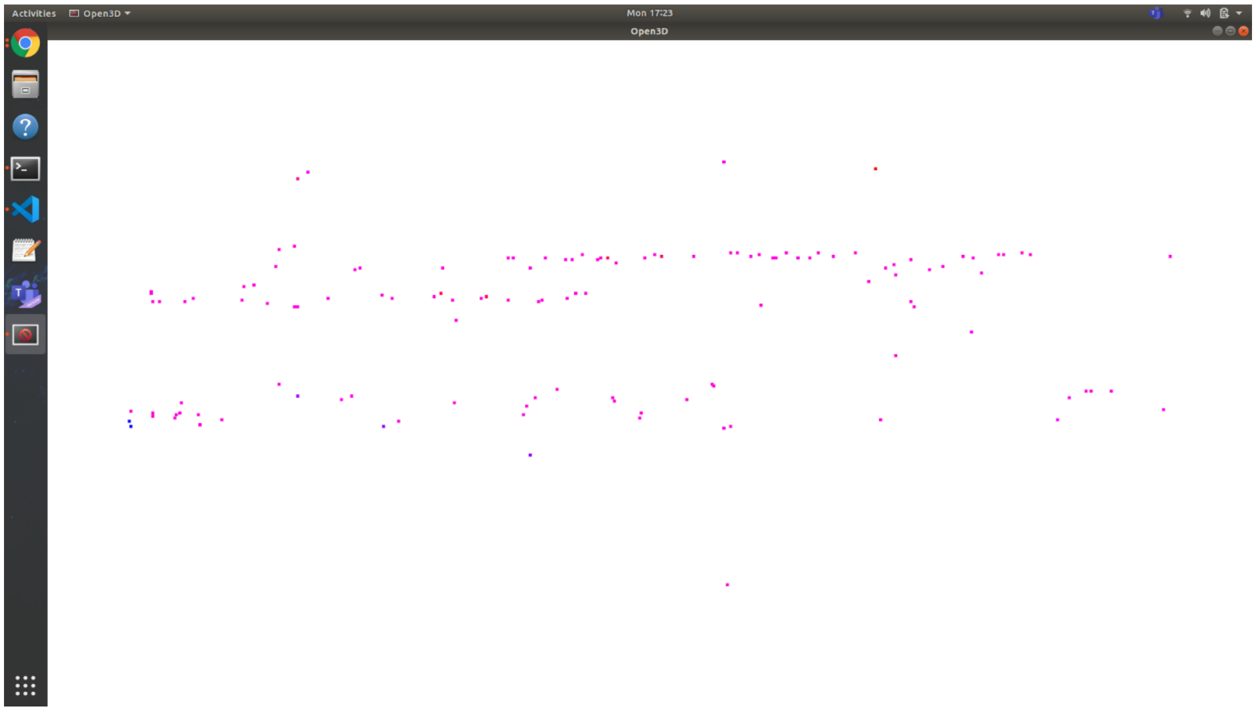


Figure 11: Coloring of radar data based on the velocity in Y direction that is compensated by the velocity of the ego vehicle.

## 5.1) Visualizing radar pcd on an image:

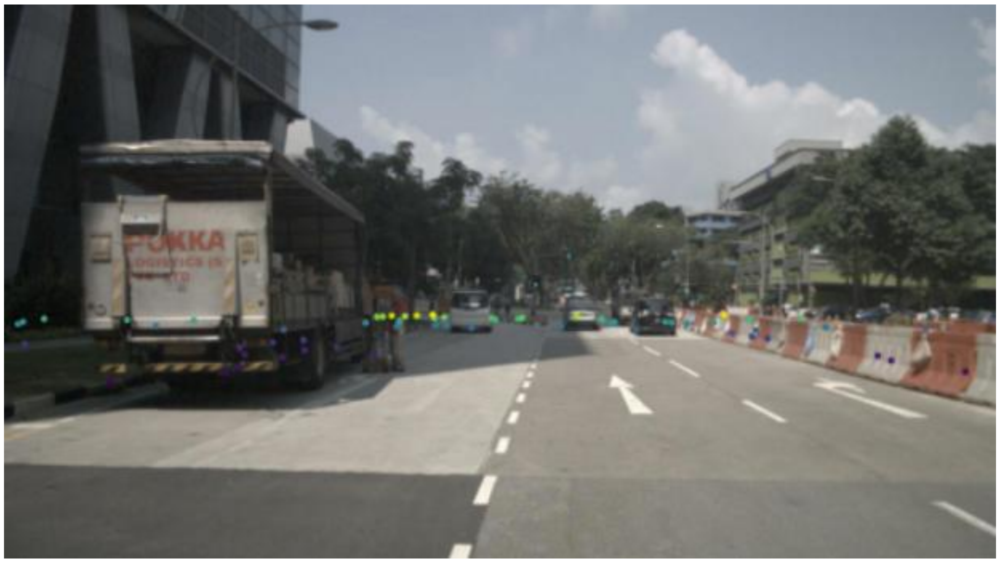
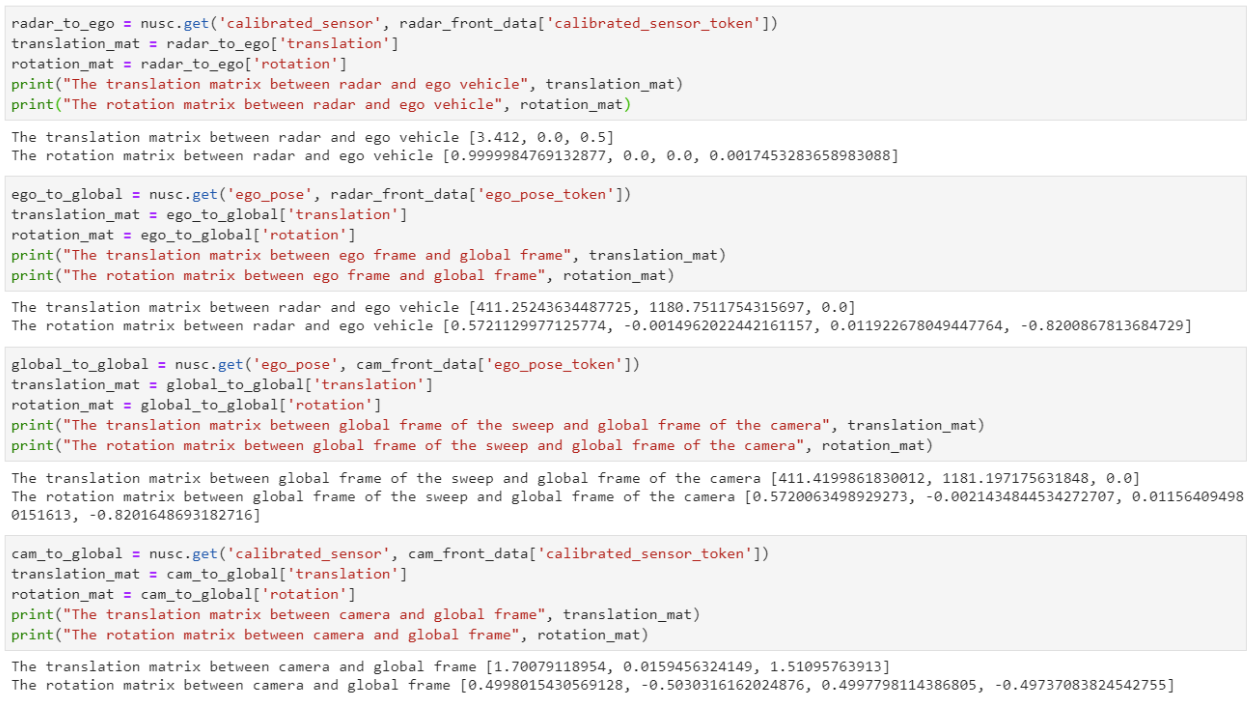


Figure 12: Visualization of radar pcd on the camera image

## 5.1.a) Printing the four transformations between the radar and camera:



```
radar_to_ego = nusc.get('calibrated_sensor', radar_front_data['calibrated_sensor_token'])
translation_mat = radar_to_ego['translation']
rotation_mat = radar_to_ego['rotation']
print("The translation matrix between radar and ego vehicle", translation_mat)
print("The rotation matrix between radar and ego vehicle", rotation_mat)
```

```
The translation matrix between radar and ego vehicle [3.412, 0.0, 0.5]
The rotation matrix between radar and ego vehicle [0.9999984769132877, 0.0, 0.0, 0.0017453283658983088]
```

```
ego_to_global = nusc.get('ego_pose', radar_front_data['ego_pose_token'])
translation_mat = ego_to_global['translation']
rotation_mat = ego_to_global['rotation']
print("The translation matrix between ego frame and global frame", translation_mat)
print("The rotation matrix between ego frame and global frame", rotation_mat)
```

```
The translation matrix between radar and ego vehicle [411.25243634487725, 1180.7511754315697, 0.0]
The rotation matrix between radar and ego vehicle [0.5721129977125774, -0.0014962022442161157, 0.011922678049447764, -0.8200867813684729]
```

```
global_to_global = nusc.get('ego_pose', cam_front_data['ego_pose_token'])
translation_mat = global_to_global['translation']
rotation_mat = global_to_global['rotation']
print("The translation matrix between global frame of the sweep and global frame of the camera", translation_mat)
print("The rotation matrix between global frame of the sweep and global frame of the camera", rotation_mat)
```

```
The translation matrix between global frame of the sweep and global frame of the camera [411.4199861830012, 1181.197175631848, 0.0]
The rotation matrix between global frame of the sweep and global frame of the camera [0.5720063498929273, -0.0021434844534272707, 0.01156409498
0151613, -0.8201648693182716]
```

```
cam_to_global = nusc.get('calibrated_sensor', cam_front_data['calibrated_sensor_token'])
translation_mat = cam_to_global['translation']
rotation_mat = cam_to_global['rotation']
print("The translation matrix between camera and global frame", translation_mat)
print("The rotation matrix between camera and global frame", rotation_mat)
```

```
The translation matrix between camera and global frame [1.70079118954, 0.0159456324149, 1.51095763913]
The rotation matrix between camera and global frame [0.4998015430569128, -0.5030316162024876, 0.4997798114386805, -0.49737083824542755]
```

*Figure 13: Calibration information between radar data and camera*

5.1.b) There are five steps involved in registering a transform between radar pcd and the camera:
1. When the radar points are generated, they are generated with respect to the frame of the radar sensor. The first step is to transform all the points to the ego vehicle frame of reference. This is done using the translation in X, Y, Z axes and rotation matrices in the X,Y,Z axes between the radar frame and the ego vehicle frame. It is printed above in the code snippet.
2. Once the points are transformed to the ego vehicle frame, they are transformed into the global map frame of reference using the translation and rotation matrices between the ego vehicle and the global map.
3. The timestamps of the camera image and the radar data might not be the same and we need to ensure that we transform the radar data to the correct position of the vehicle when the image was taken. Therefore, we do a global-to-global transformation between camera timestamp and radar timestamp.
4. Then a transform is made between the global frame of the radar data to the camera frame using the transpose of the translation and the rotation matrices.
5. All the depths of the points are taken for the visualization on the camera image.

5.2) Visualizing Lidar pcd on the camera image.

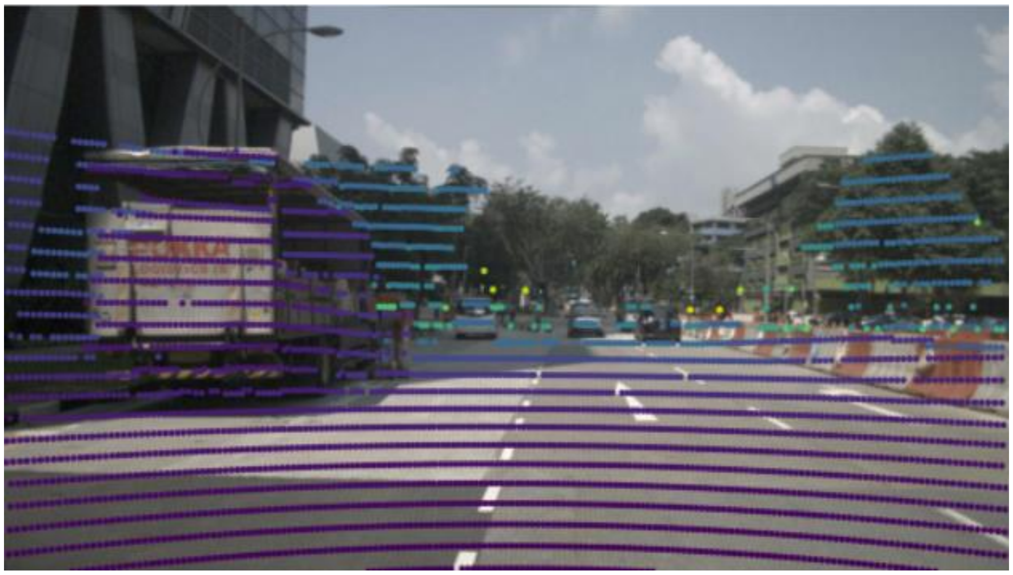
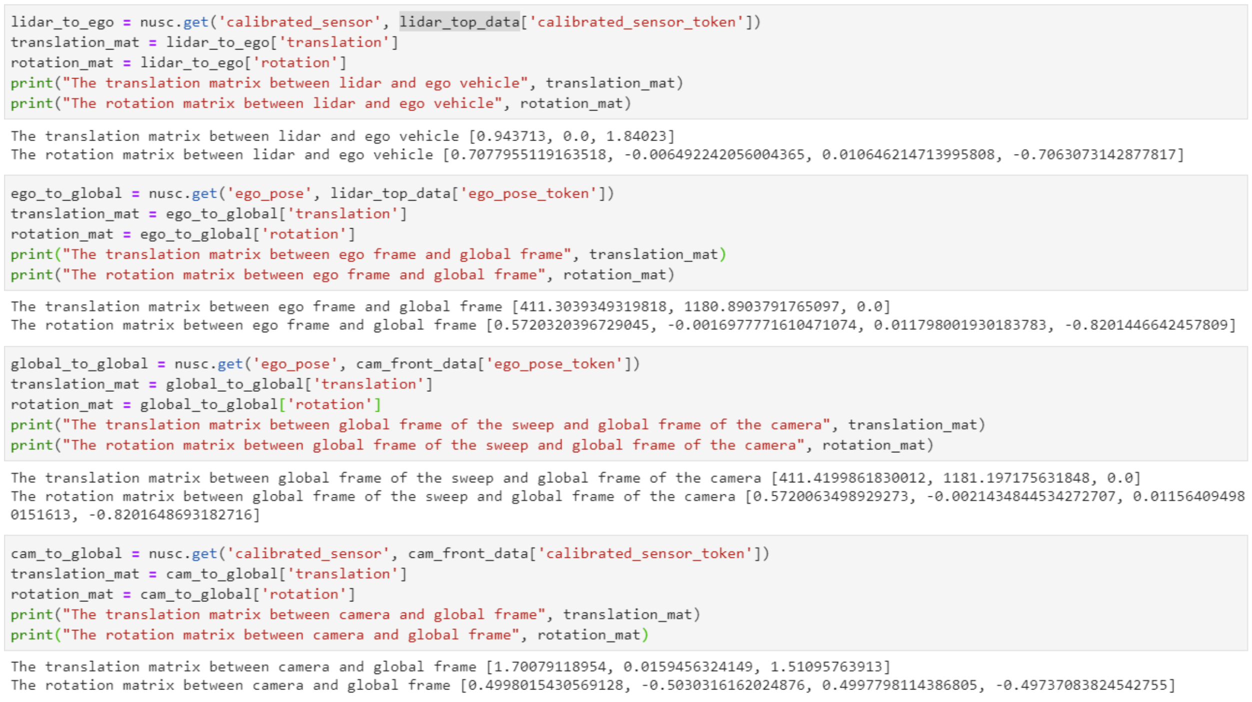


*Figure 14: Visualization of lidar pcd on the camera image*

Printing the calibration information between Lidar data and the camera image:



```
lidar_to_ego = nusc.get('calibrated_sensor', lidar_top_data['calibrated_sensor_token'])
translation_mat = lidar_to_ego['translation']
rotation_mat = lidar_to_ego['rotation']
print("The translation matrix between lidar and ego vehicle", translation_mat)
print("The rotation matrix between lidar and ego vehicle", rotation_mat)
```

```
The translation matrix between lidar and ego vehicle [0.943713, 0.0, 1.84023]
The rotation matrix between lidar and ego vehicle [0.7077955119163518, -0.006492242056004365, 0.010646214713995808, -0.7063073142877817]
```

```
ego_to_global = nusc.get('ego_pose', lidar_top_data['ego_pose_token'])
translation_mat = ego_to_global['translation']
rotation_mat = ego_to_global['rotation']
print("The translation matrix between ego frame and global frame", translation_mat)
print("The rotation matrix between ego frame and global frame", rotation_mat)
```

```
The translation matrix between ego frame and global frame [411.3039349319818, 1180.8903791765097, 0.0]
The rotation matrix between ego frame and global frame [0.5720320396729045, -0.0016977771610471074, 0.011798001930183783, -0.8201446642457809]
```

```
global_to_global = nusc.get('ego_pose', cam_front_data['ego_pose_token'])
translation_mat = global_to_global['translation']
rotation_mat = global_to_global['rotation']
print("The translation matrix between global frame of the sweep and global frame of the camera", translation_mat)
print("The rotation matrix between global frame of the sweep and global frame of the camera", rotation_mat)
```

```
The translation matrix between global frame of the sweep and global frame of the camera [411.4199861830012, 1181.197175631848, 0.0]
The rotation matrix between global frame of the sweep and global frame of the camera [0.5720063498929273, -0.0021434844534272707, 0.01156409498
0151613, -0.8201648693182716]
```

```
cam_to_global = nusc.get('calibrated_sensor', cam_front_data['calibrated_sensor_token'])
translation_mat = cam_to_global['translation']
rotation_mat = cam_to_global['rotation']
print("The translation matrix between camera and global frame", translation_mat)
print("The rotation matrix between camera and global frame", rotation_mat)
```

```
The translation matrix between camera and global frame [1.70079118954, 0.0159456324149, 1.51095763913]
The rotation matrix between camera and global frame [0.4998015430569128, -0.5030316162024876, 0.4997798114386805, -0.49737083824542755]
```

*Figure 15: Calibration information between lidar pcd and camera image*