Homework 2

Student Name: Jialun Yang

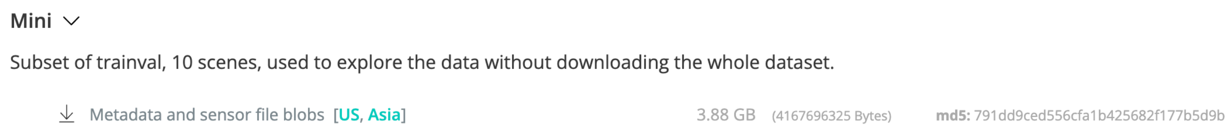
AuE 8200: Machine Perception and Intelligence
Instructor: Dr. Bing Li, Clemson University, Department of Automotive Engineering

* Refer to Syllabus for homework grading, submission and plagiarism policies;
* Submission to Canvas (Due: Feb. 13, 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 choose either Python, Matlab or any other programming language.
* You can find some sample codes from the course GitHub Repo if you use Python.

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**: (5 point)



Mini

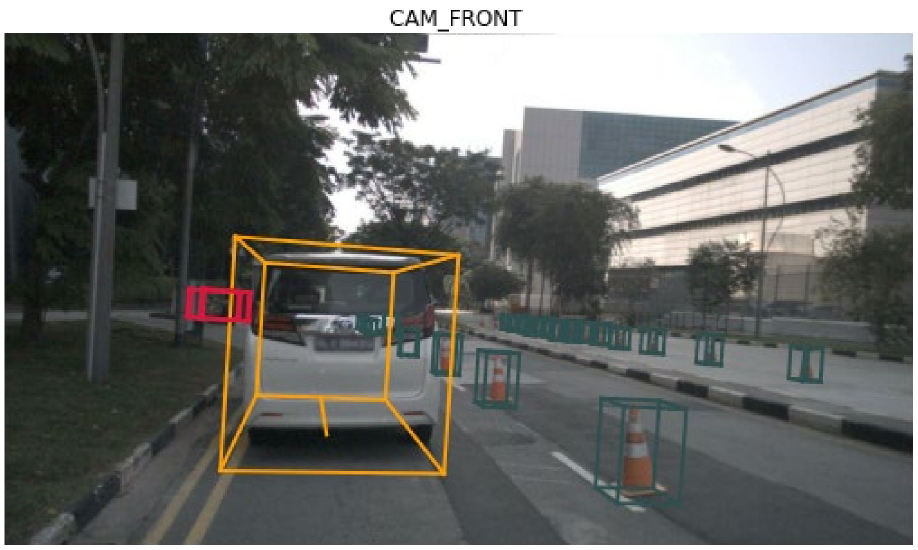
Subset of trainval, 10 scenes, used to explore the data without downloading the whole dataset.

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

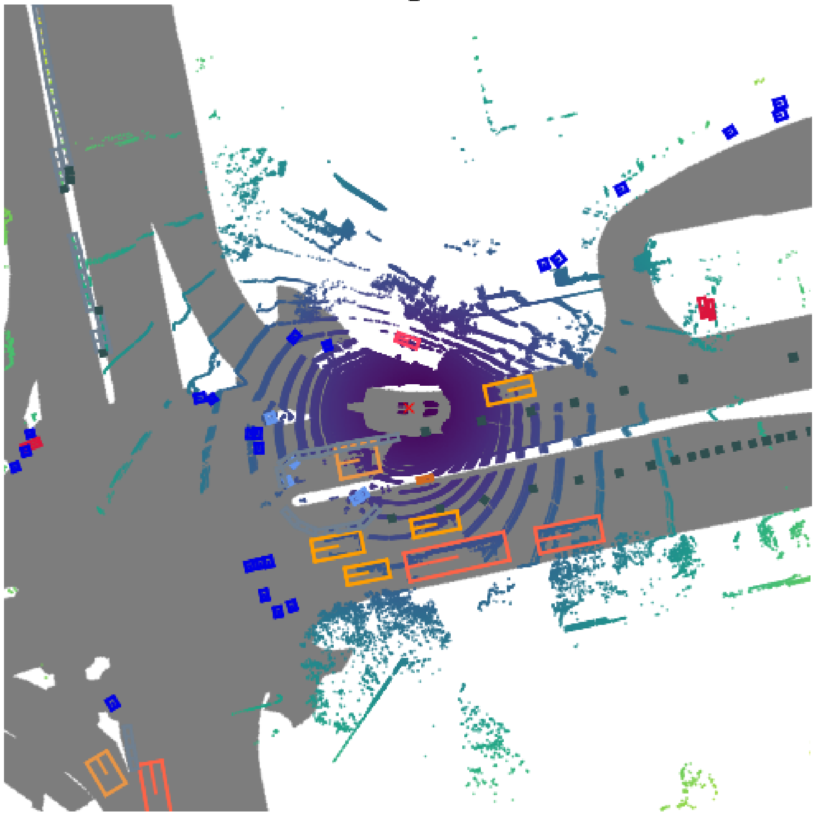
The codes are written in PyCharm and Google Colab.

2. 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. (5 point)

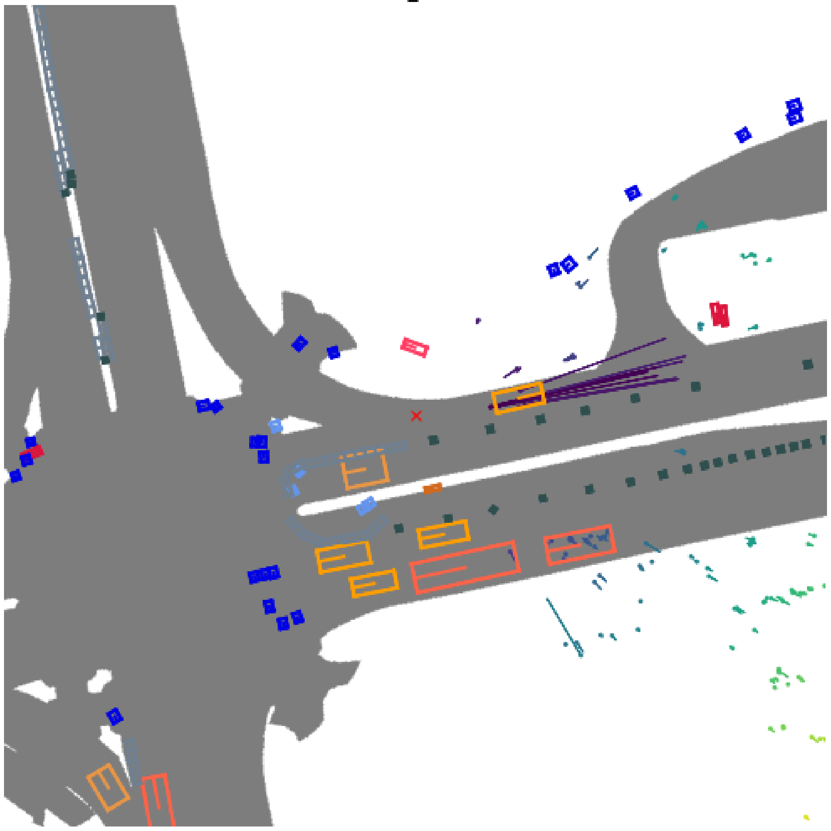
3. Pickup a set of data, including Image, Lidar, and Radar data. Visualize them respectively. If you use Python, you can refer to NuScene dev-kit tutorial reference code. (10 points)
   Comments from Siqi Zheng:
   The nuscenes library requires the matplotlib library version:
   - matplotlib 3.2.2
   - matplotlib-venn 0.11.7
   The latest version of matplotlib library seems doesn't support the nuscenes library.



LIDAR_TOP

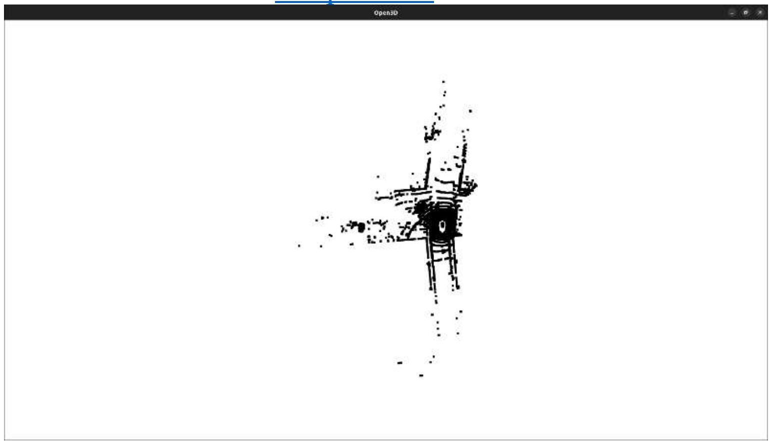


RADAR_FRONT

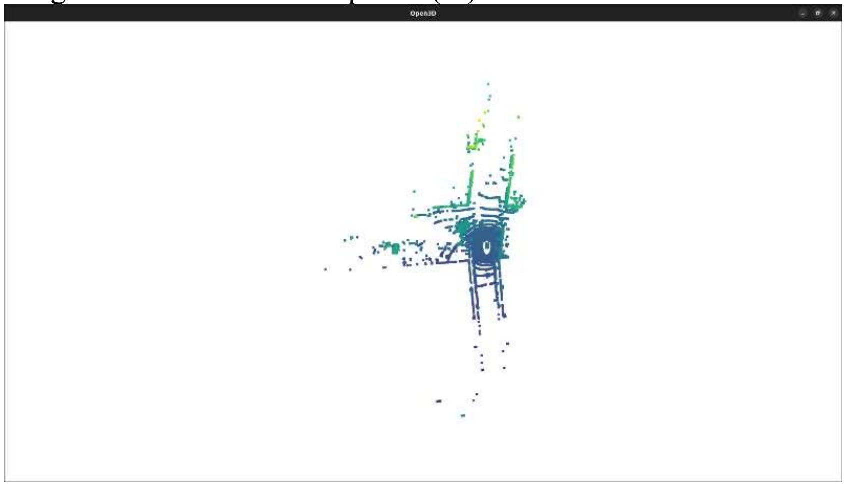


4. Rather than using NuScene dev-kit, implement below by yourself (total 35 points):
   (1) Visualize images (you can use library OpenCV or others), sample code. (5')

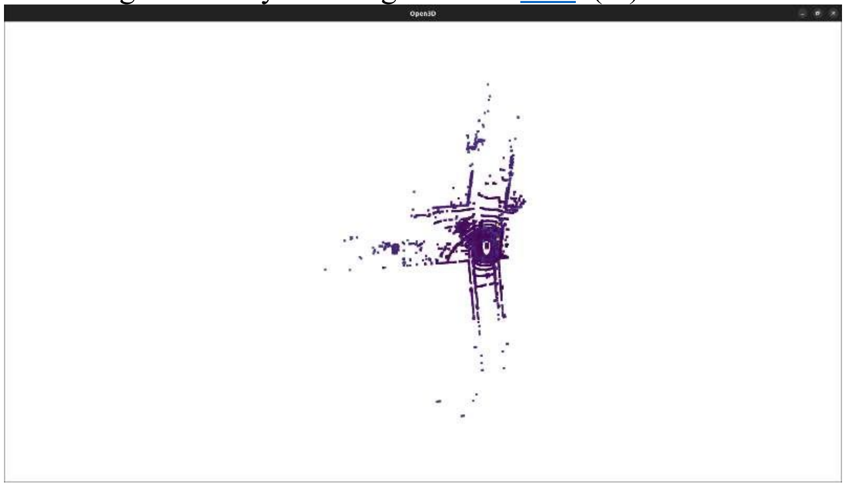
   (2) Visualize Lidar point cloud data
      a. You can refer to this sample code.



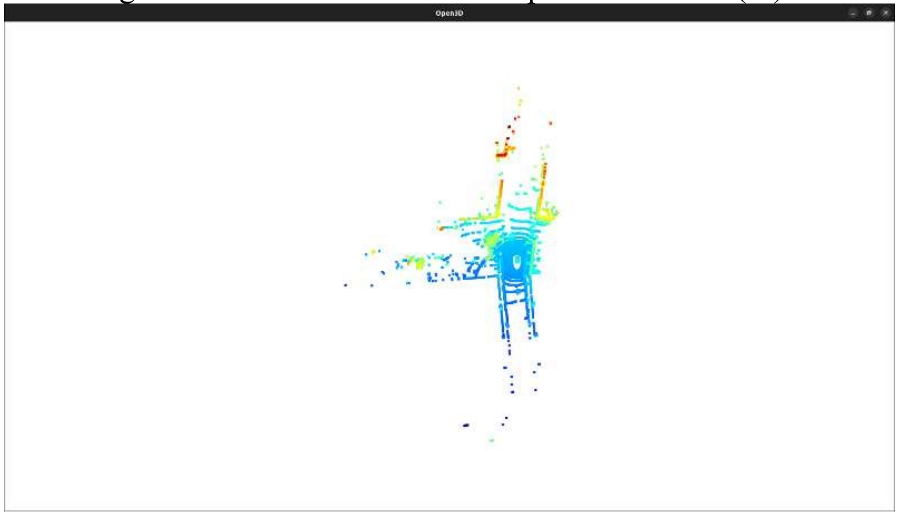
      b. Colorize points by height, intensity, and semantic label respectively.
         i. Height is the Z value for a point. (5')



         ii. You can get intensity referring the code here. (5')

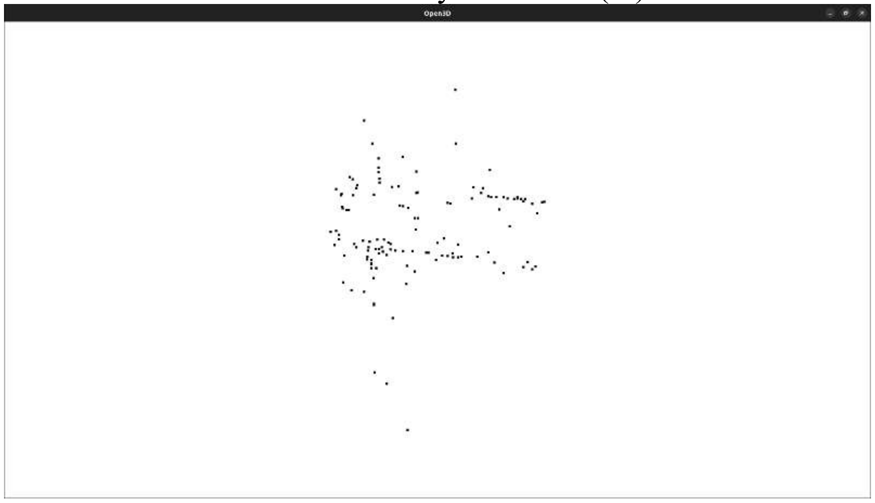


iii. You can get semantic label from the sample above code. (5')



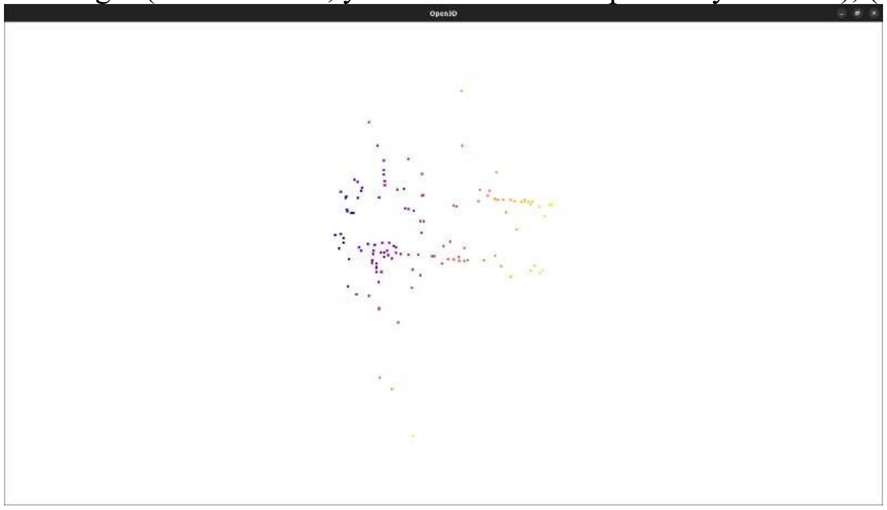
(3) Visualize Radar data

c. Use any other library (e.g, Open3D, PCL, etcl) or modify the previous sample code to visualize the Radar data which you chosen. (5')

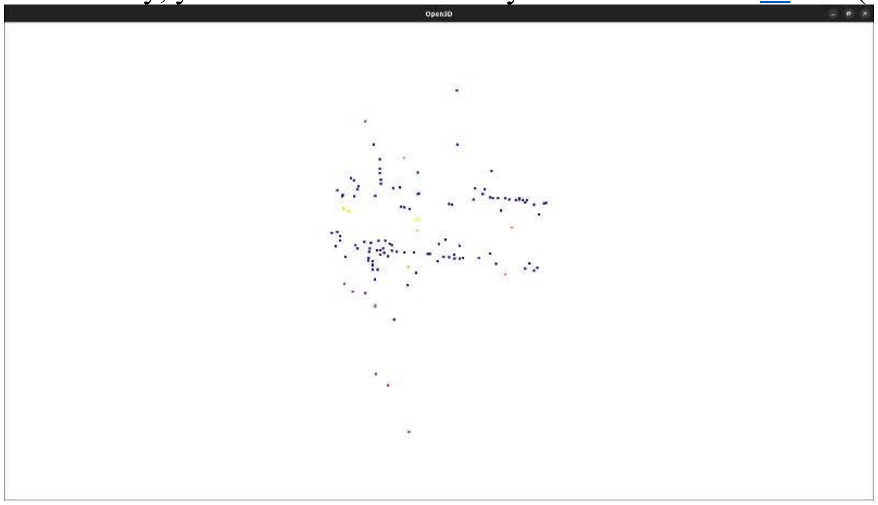


d. Colorize points by below two variable aspects respectively.

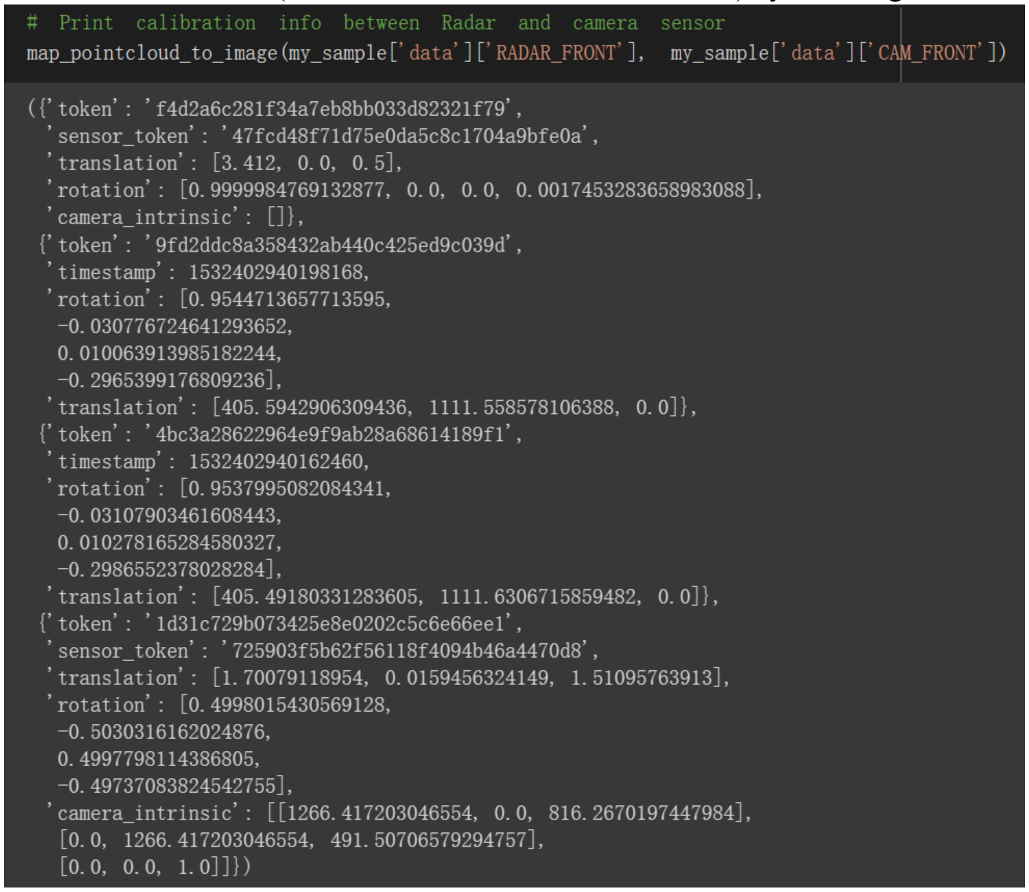
i. For height (if it's all zero, you can colorize the points by distance), (5')



ii. For velocity, you can find some velocity information from [he](#) `re. (5')



5. Using NuScene dev-kit for the set of data which you picked up: (45 points)
   (1) Visualize Radar data projection on image
      a. Print calibration info (between Radar and Camera sensors) by referring code [here](#). (5')



```
# Print calibration info between Radar and camera sensor
map_pointcloud_to_image(my_sample['data']['RADAR_FRONT'], my_sample['data']['CAM_FRONT'])
```

```
({'token': 'f4d2a6c281f34a7eb8bb033d82321f79',
  'sensor_token': '47fcd48f71d75e0da5c8c1704a9bfe0a',
  'translation': [3.412, 0.0, 0.5],
  'rotation': [0.9999984769132877, 0.0, 0.0, 0.0017453283658983088],
  'camera_intrinsic': []},
 {'token': '9fd2ddc8a358432ab440c425ed9c039d',
  'timestamp': 1532402940198168,
  'rotation': [0.9544713657713595,
   -0.030776724641293652,
   0.010063913985182244,
   -0.2965399176809236],
  'translation': [405.5942906309436, 1111.558578106388, 0.0]},
 {'token': '4bc3a28622964e9f9ab28a68614189f1',
  'timestamp': 1532402940162460,
  'rotation': [0.9537995082084341,
   -0.03107903461608443,
   0.010278165284580327,
   -0.2986552378028284],
  'translation': [405.49180331283605, 1111.6306715859482, 0.0]},
 {'token': '1d31c729b073425e8e0202c5c6e66ee1',
  'sensor_token': '725903f5b62f56118f4094b46a4470d8',
  'translation': [1.70079118954, 0.0159456324149, 1.51095763913],
  'rotation': [0.4998015430569128,
   -0.5030316162024876,
   0.4997798114386805,
   -0.49737083824542755],
  'camera_intrinsic': [[1266.417203046554, 0.0, 816.2670197447984],
   [0.0, 1266.417203046554, 491.50706579294757],
   [0.0, 0.0, 1.0]]})
```

      b. Explain the above calibration info, and pipeline of First~Fifth steps in the code. (10')
      As we can see in the above screenshot, the mainly calibration info is the translation matrix and rotation matrix in the function map_pointcloud_to_image. The rotation

matrix is used to convert a point in space from one coordinate system while maintaining the orientation of the point in space. The translation matrix depict how the position of a point in space changes as it transformed from one coordinate to another. Combining the two kinds of matrix together, we can transform and combine the points in two coordinates to complete the calibration.

The pipeline of First-Fifth steps:

① Form the relationship between calibrated sensor and ego vehicle, then transform the pointcloud to ego frame.

② Form the relationship between ego vehicle and global frame, then transform the points to global frame.

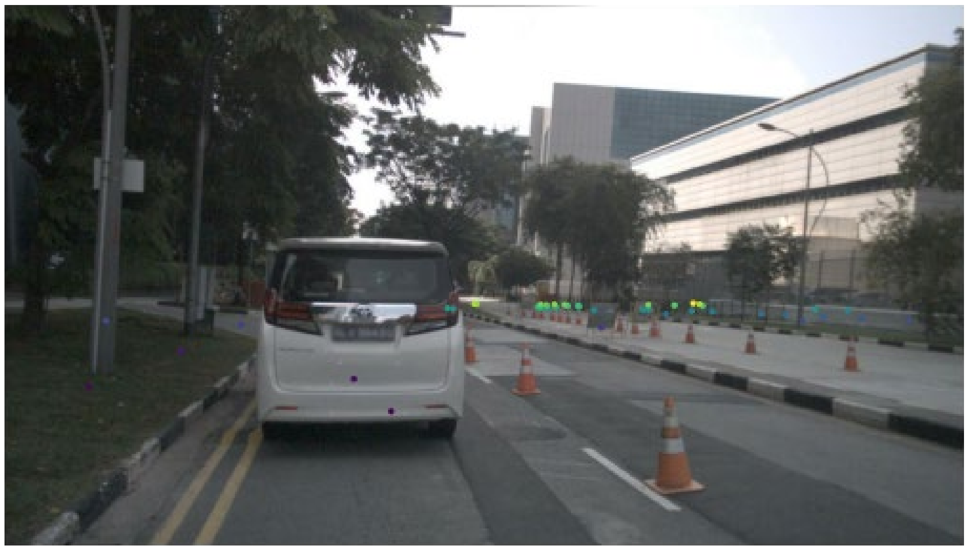
③ Form the relationship between global frame and the ego vehicle frame, then transform the points coordinate into the camera coordinate.

④ Form the relationship between ego vehicle frame and calibrating sensor, and finally we got the two data in one coordinate.

⑤ Got the depths of the pointcloud in one image by combining the results in step 1~4.

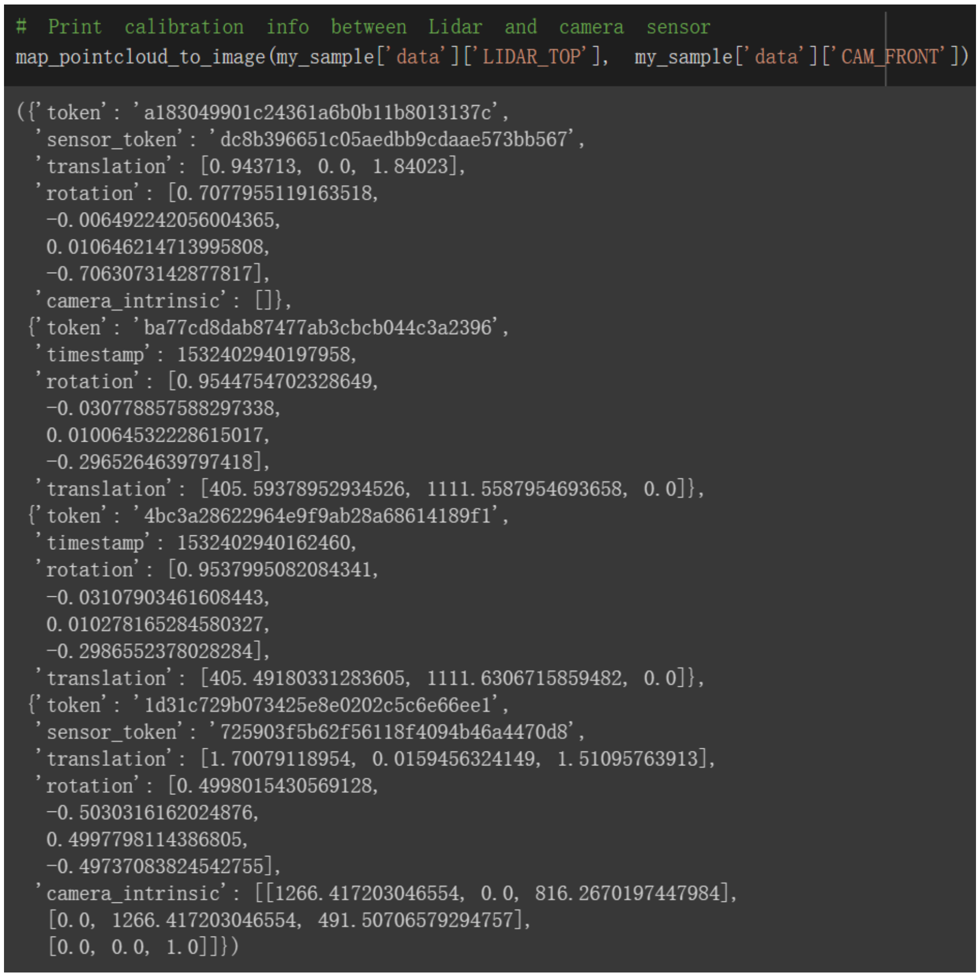
Step 1~5 go through the transformation action of the points to put them in one coordinate so that we can do further calibration.

c. Visualize Radar data projection on image based on calibration info. (10')



(2) Visualize LiDAR data projection on image

d. Print and explain the calibration info (between LiDAR and Camera sensors) by referring here. (5')



```
# Print calibration info between Lidar and camera sensor
map_pointcloud_to_image(my_sample['data']['LIDAR_TOP'], my_sample['data']['CAM_FRONT'])
```

```
({'token': 'a183049901c24361a6b0b11b8013137c',
  'sensor_token': 'dc8b396651c05aedbb9cdaae573bb567',
  'translation': [0.943713, 0.0, 1.84023],
  'rotation': [0.7077955119163518,
   -0.006492242056004365,
   0.010646214713995808,
   -0.7063073142877817],
  'camera_intrinsic': []},
 {'token': 'ba77cd8dab87477ab3cbcb044c3a2396',
  'timestamp': 1532402940197958,
  'rotation': [0.9544754702328649,
   -0.030778857588297338,
   0.010064532228615017,
   -0.2965264639797418],
  'translation': [405.59378952934526, 1111.5587954693658, 0.0]},
 {'token': '4bc3a28622964e9f9ab28a68614189f1',
  'timestamp': 1532402940162460,
  'rotation': [0.9537995082084341,
   -0.03107903461608443,
   0.010278165284580327,
   -0.2986552378028284],
  'translation': [405.49180331283605, 1111.6306715859482, 0.0]},
 {'token': '1d31c729b073425e8e0202c5c6e66ee1',
  'sensor_token': '725903f5b62f56118f4094b46a4470d8',
  'translation': [1.70079118954, 0.0159456324149, 1.51095763913],
  'rotation': [0.4998015430569128,
   -0.5030316162024876,
   0.4997798114386805,
   -0.49737083824542755],
  'camera_intrinsic': [[1266.417203046554, 0.0, 816.2670197447984],
   [0.0, 1266.417203046554, 491.50706579294757],
   [0.0, 0.0, 1.0]]})
```

e. Visualize LiDAR data projection on image based on calibration info. (15')