Sensor Fusion- Mid Term Project Report

Project Title- 3D Object Detection

Section 1 : Compute Lidar Point-Cloud from Range Image

ID_S1_EX1

Introduction

The goal of the exercise is to extract and Visualize Range and Intensity channels from the Range Images in the Waymo Open dataset.

Set parameters in loop_over_dataset.py

data_filename='training_segment1005081002024129653_5313_150_5333_150_with_camera_labels.tfrecord'

show_only_frames = [0, 1]

exec_data = []

exec_detection = []

exec_tracking = []

exec_visualization = ['show_range_image']

Output Images:

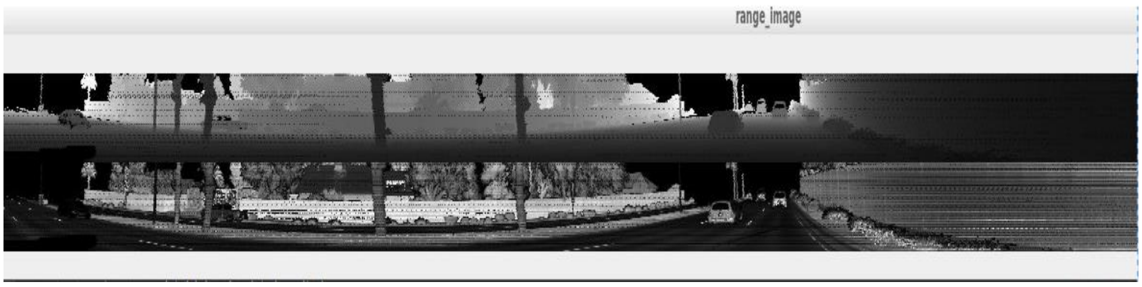


Fig1- Range Image

Visualize lidar point-cloud (ID_S1_EX2)

Introduction: The aim of this exercise is to visualize the point cloud in a 3D viewer. In the point cloud, there are objects like cars with various degrees of visibility. There is a blind spot near the center of the LiDAR. The objects near the LiDAR are denser in comparison to the ones away from it. It is easier to detect the denser objects with naked eye. The objects away from the LiDAR are sparse and many times unclear. Due to Occlusion, many objects lose significant features in the point cloud.

Set parameters in loop_over_dataset.py:

data_filename = 'training_segment-10963653239323173269_1924_000_1944_000_with_camera_labels.tfrecord'

show_only_frames = [0, 200]

exec_data = []

exec_detection = []

exec_tracking = []

exec_visualization = ['show_pcl']

Problem Statement 1

Find and Display 6 examples of vehicles with varying degrees of visibility in the point cloud

Solution:

Point cloud of the cars near the LiDAR are denser than the cars away from the LIDAR.

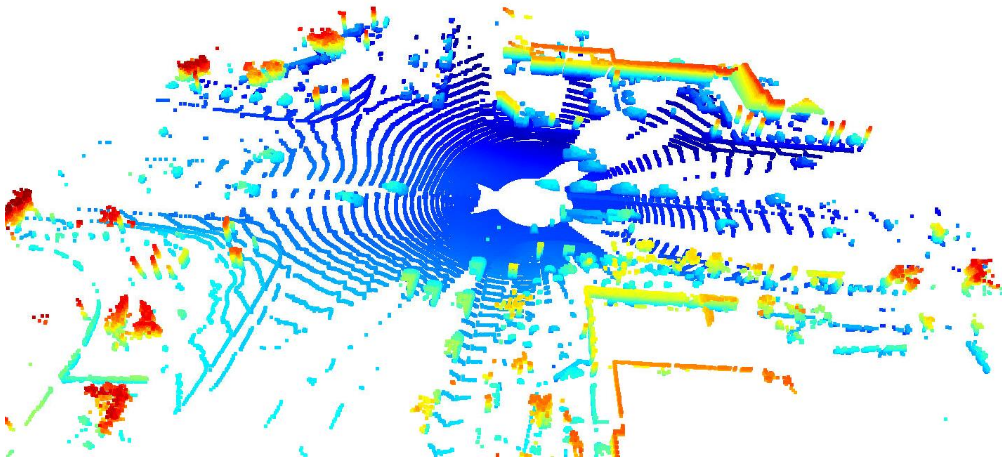


Fig2- Overall view of the point cloud with different cars.

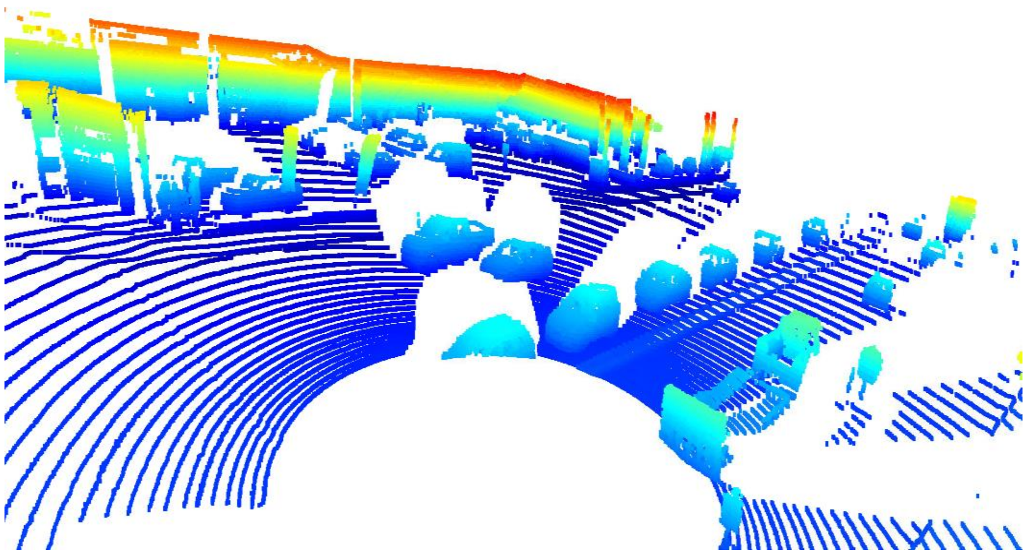


Fig3- Point Cloud depicting the cars in the area

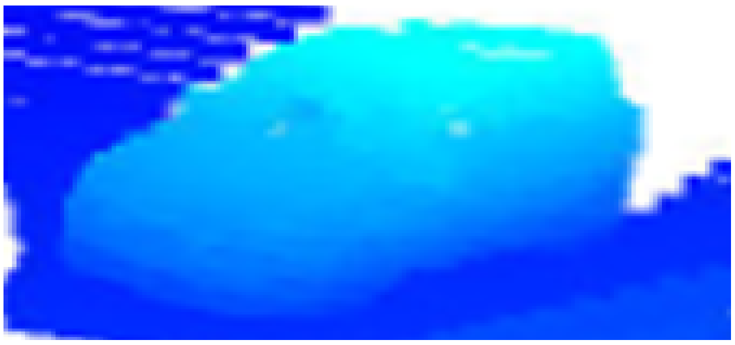


Fig4- Point cloud of Cars with clear image of Vehicle.

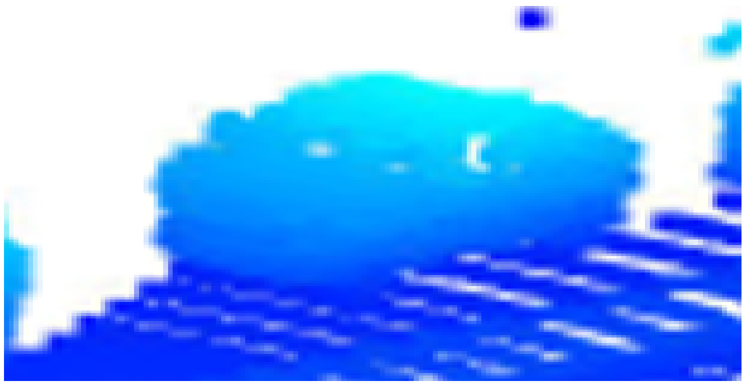


Fig5- Point cloud of Car with unclear Front Right side

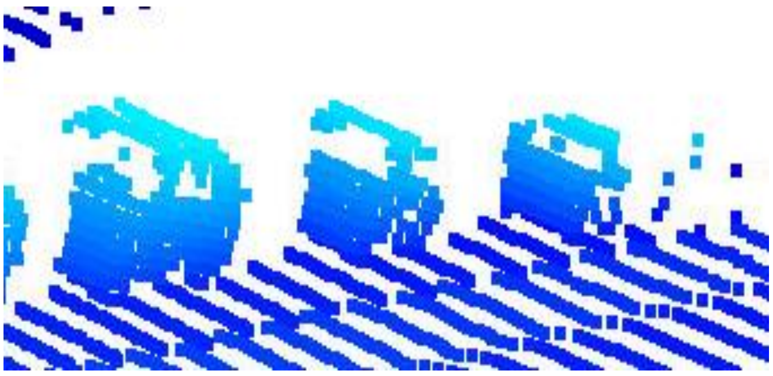


Fig6- Point Cloud with Sparse Car data

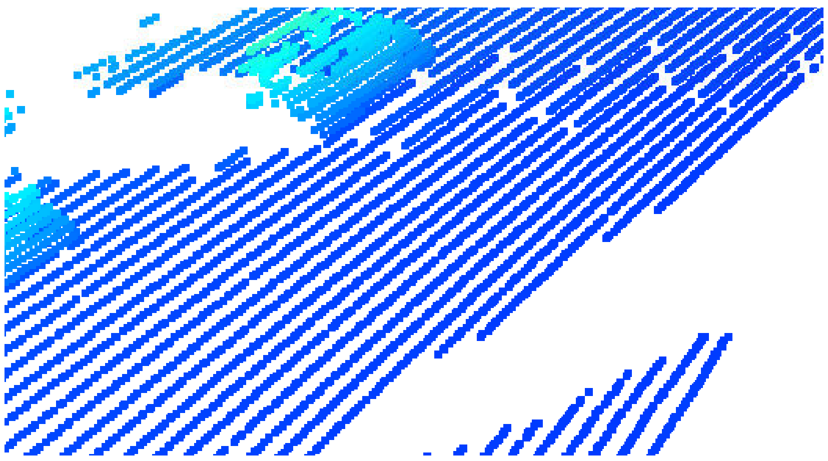


Fig7- Car is Visible but point Cloud not very clear

Fig8- Very Sparse point cloud of a Car.

Problem Statement 2

Identify vehicle features that appear as a stable feature on most vehicles (e.g. rear-bumper, tail-lights) and describe them briefly. Also, use the range image viewer from the last example to underpin your findings using the lidar intensity channel

Solution

The following are the observations

1- As we move away from the lidar, the point cloud becomes sparse, and the objects become unclear. The cars near the center have more prominent features. On one side of the Lidar point cloud, the features at the back of the car (back bumper, rear glass, back tires, back door) are more prominent. Similarly, on the other side, the features at the front of the car (Front Bonet, Windows of front side of the car (WindScreen), front bumper) are more prominent.

2- Only the first Car had complete reflection from the front window. Because of which there was a complete reflection from point cloud. The other cars did not have reflections from the windscreen because of which that area is white in the point cloud.

3- Car Bonnet and Rear bumper of the inbound and Outgoing vehicles near the Lidar were visible clearly. This was so because they were well reflected by the Lidar waves.

4- Other Vehicle features such as tires were not very clear in the point cloud. They were visible in the range image.

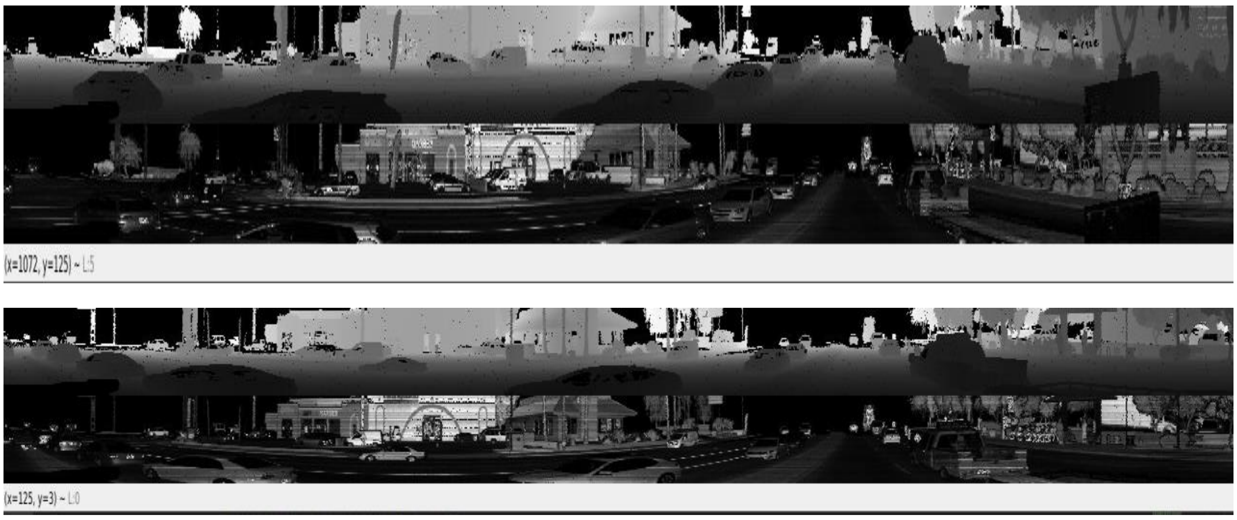


Fig9-The Range Viewer also proves the observation.

Section 2: Create Birds-Eye View from Lidar PCL

Introduction:

The aim of Section2 is to create Birds eye view from Lidar point cloud. Later, intensity layers and Height layers of the BEV map have been visualized.

Convert sensor coordinates to BEV-map coordinates (ID_S2_EX1)

Set Parameters:

data_filename=
'training_segment10050810020241296 53_5313_150_5333_150_with_camera_labels.tfrecord’

show_only_frames = [0, 1]

exec_data = ['pcl_from_rangeimage']

exec_detection = ['bev_from_pcl']

exec_tracking = []

exec_visualization = []

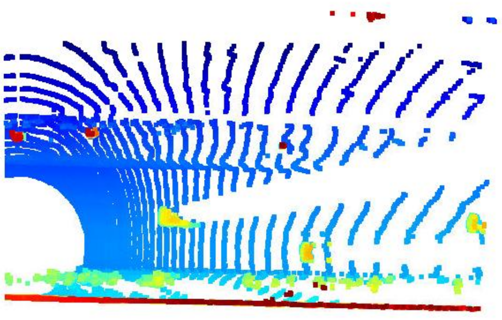


Fig10- BEV of the LiDAR point cloud

Compute intensity layer of the BEV map (ID_S2_EX2)

data_filename= 'training_segment 1005081002024129653_5313_150_5333_150_with_camera_labels.tfrecord’

show_only_frames = [0, 1]

exec_data = ['pcl_from_rangeimage']

exec_detection = ['bev_from_pcl']

exec_tracking = []

exec_visualization = []

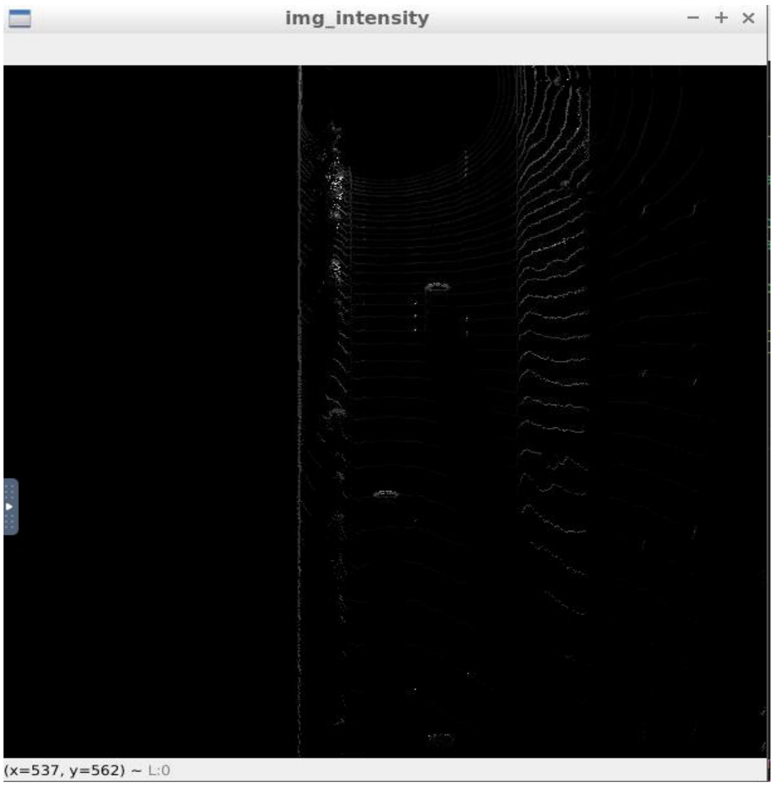


Fig11- Intensity layer of BEV Map

Compute height layer of the BEV map (ID_S2_EX3)

data_filename='training_segment-10050810020241296533_5313_150_5333_150_with_camera_labels.tfrecord'

show_only_frames = [0, 1]

exec_data = ['pcl_from_rangeimage']

exec_detection = ['bev_from_pcl']

exec_tracking = []

exec_visualization = []

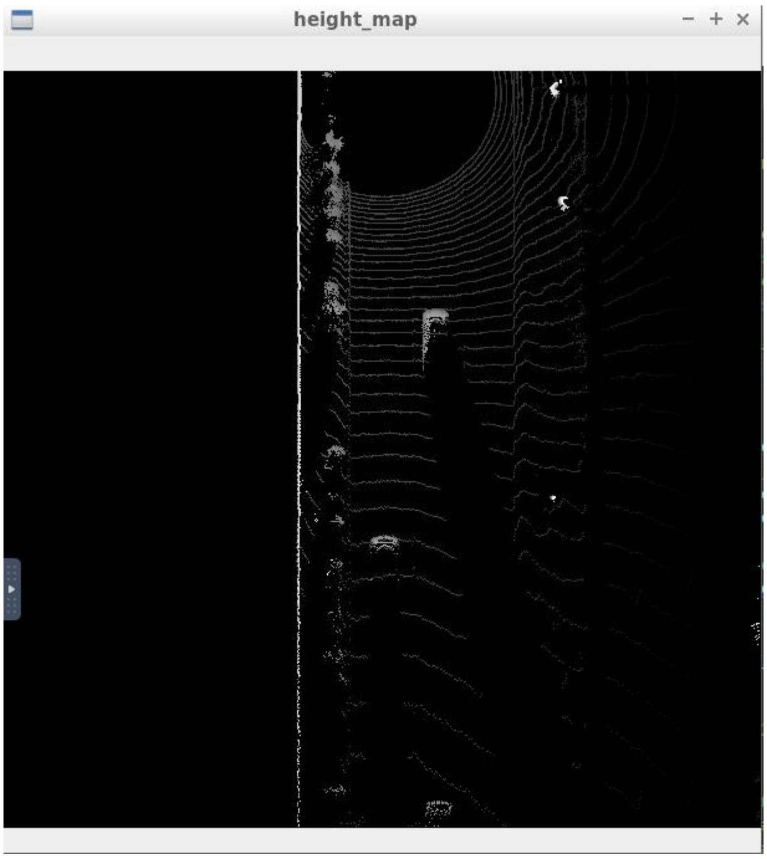


Fig12- height layer of BEV Map

Section 3 : Model-based Object Detection in BEV Image

Introduction

The aim is to add an extra model “FPN-Resnet” and perform Object detection. The model parameters were added in objdet_detect.py file. Later, conversions were made such that detections have the format [1, x, y, z, h, w, l, yaw]

Add a second model from a GitHub repo (ID_S3_EX1)

Set parameters:

data_filename='training_segment-10050810020241296535_5313_150_5333_150_with_camera_labels.tfrecord’

show_only_frames = [50, 51]

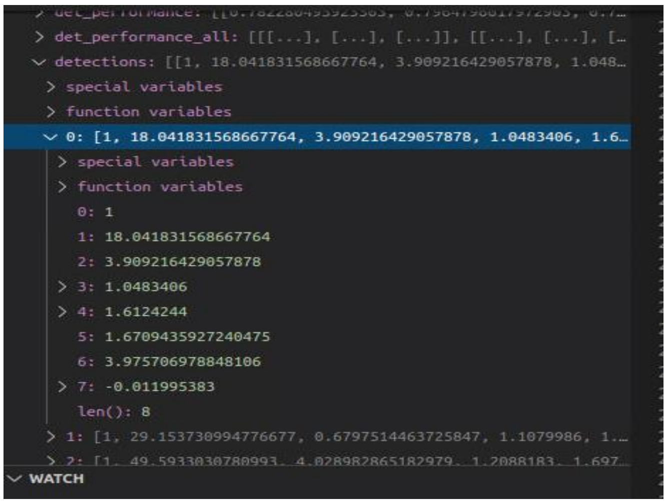
exec_data = ['pcl_from_rangeimage', 'load_image']

exec_detection = ['bev_from_pcl', 'detect_objects']

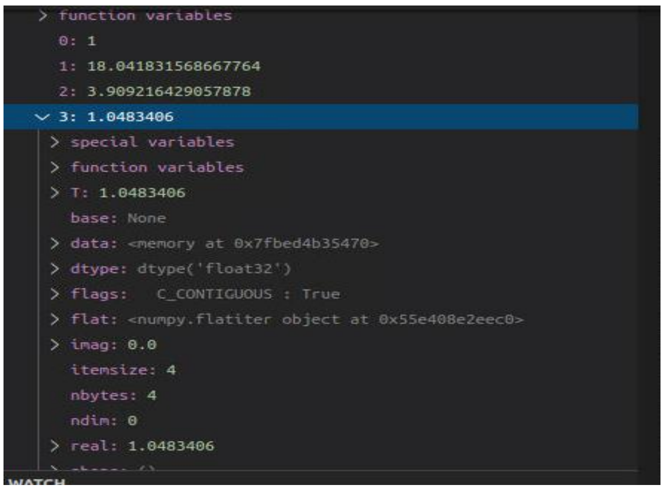
exec_tracking = []

exec_visualization = ['show_objects_in_bev_labels_in_camera']

configs_det=det.load_configs(model_name="fpn_resnet")



```
> det_performance: [[0.782280495925565, 0.7964796017972965, 0.7…
> det_performance_all: [[[...], [...], [...]], [[...], [...], […
v detections: [[1, 18.041831568667764, 3.909216429057878, 1.048…
  > special variables
  > function variables
  v 0: [1, 18.041831568667764, 3.909216429057878, 1.0483406, 1.6…
    > special variables
    > function variables
      0: 1
      1: 18.041831568667764
      2: 3.909216429057878
    > 3: 1.0483406
    > 4: 1.6124244
      5: 1.6709435927240475
      6: 3.975706978848106
    > 7: -0.011995383
      len(): 8
  > 1: [1, 29.153730994776677, 0.6797514463725847, 1.1079986, 1.…
  > 2: [1, 49.5933030780993, 4.028982865182979, 1.2088183, 1.697…
v WATCH
```



```
 > function variables
   0: 1
   1: 18.041831568667764
   2: 3.909216429057878
 v 3: 1.0483406
   > special variables
   > function variables
   > T: 1.0483406
     base: None
   > data: <memory at 0x7fbed4b35470>
   > dtype: dtype('float32')
   > flags:   C_CONTIGUOUS : True
   > flat: <numpy.flatiter object at 0x55e408e2eec0>
   > imag: 0.0
     itemsize: 4
     nbytes: 4
     ndim: 0
   > real: 1.0483406
```

Fig13- Detections data

Extract 3D bounding boxes from model response (ID_S3_EX2)

Set Parameters:

data_filename= "training_segment-10050810020241296 53_5313_150_5333_150_with_camera_labels.tfrecord"

show_only_frames = [50, 51]

exec_data = ['pcl_from_rangeimage', 'load_image']

exec_detection = ['bev_from_pcl', 'detect_objects']

exec_tracking = []

exec_visualization = ['show_objects_in_bev_labels_in_camera']

configs_det = det.load_configs(model_name="fpn_resnet")

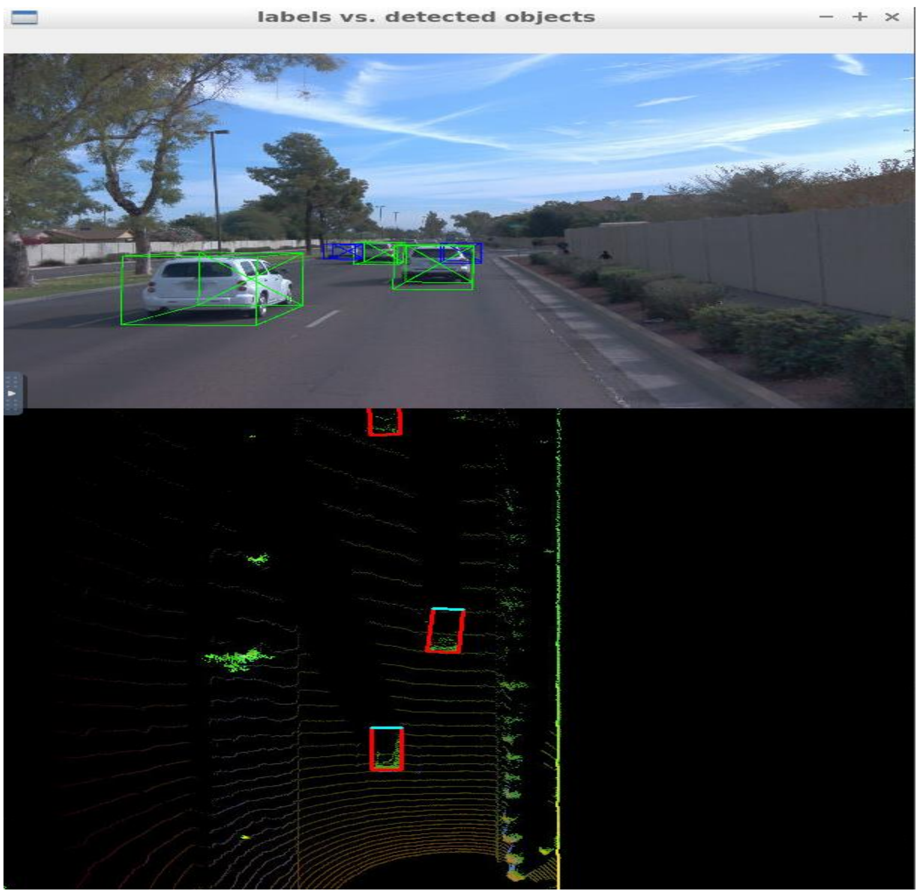


Fig14- 3D bounding boxes added to the Images

Compute intersection-over-union between labels and detections (ID_S4_EX1- EX 3)

Introduction

The aim is to compute and plot the graphs for various parameters such as IOU, Precision and Recall. Before calculating the parameters, the number of true positives, false positives and false negatives were calculated for each frame. Other metrics such as F1 score can also be calculated. These graphs have been drawn two times. Before and after changing the parameter **configs_det.use_labels_as_objects.**

Set Parameters

data_filename = “training_segment-10050810020241296 53_5313_150_5333_150_with_camera_labels.tfrecord”

show_only_frames = [50, 51]

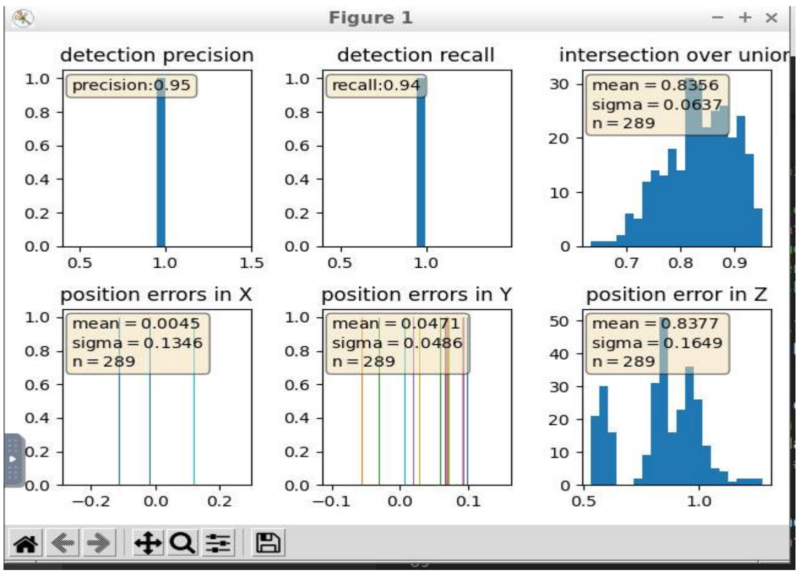
exec_data = ['pcl_from_rangeimage']

exec_detection = ['bev_from_pcl', 'detect_objects', 'validate_object_labels', 'measure_detection_performance']

exec_tracking = []

exec_visualization = ['show_detection_performance']

configs_det = det.load_configs(model_name="darknet") graphs after setting the value of Parameter **configs_det.use_labels_as_objects** as FALSE



graphs after setting the value of Parameter **configs_det.use_labels_as_objects** as TRUE

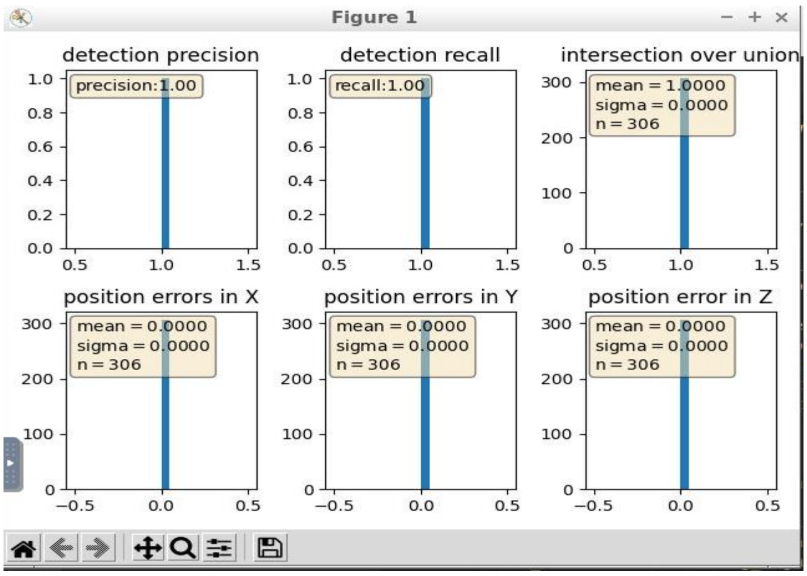


Figure 1
detection precision
precision:1.00
detection recall
recall:1.00
intersection over union
mean = 1.0000
sigma = 0.0000
n = 306
position errors in X
mean = 0.0000
sigma = 0.0000
n = 306
position errors in Y
mean = 0.0000
sigma = 0.0000
n = 306
position error in Z
mean = 0.0000
sigma = 0.0000
n = 306