3D Object Detection

Section 1: Compute Lidar Point-Cloud from Range Image

ID_S1_EX1: Visualize range image channels

The objective of this section is to use Waymo Open dataset (LiDAR data) to extract and visualize range and intensity channels from the range images.

Parameters

File: loop_over_dataset.py

data_filename = 'training_segment-1005081002024129653_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

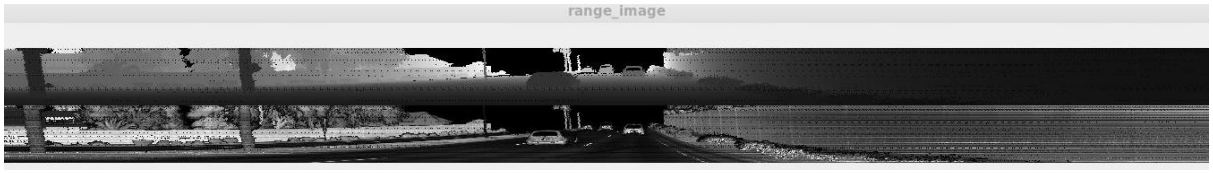


Fig.1 Range image

ID_S1_EX2: Visualize lidar point-cloud

The objective of this exercise is to display lidar point cloud in a 3D viewer to study the nature of the point cloud

Parameters

File: 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']

- *Find and display 6 examples of vehicles with varying degrees of visibility in the point-cloud*

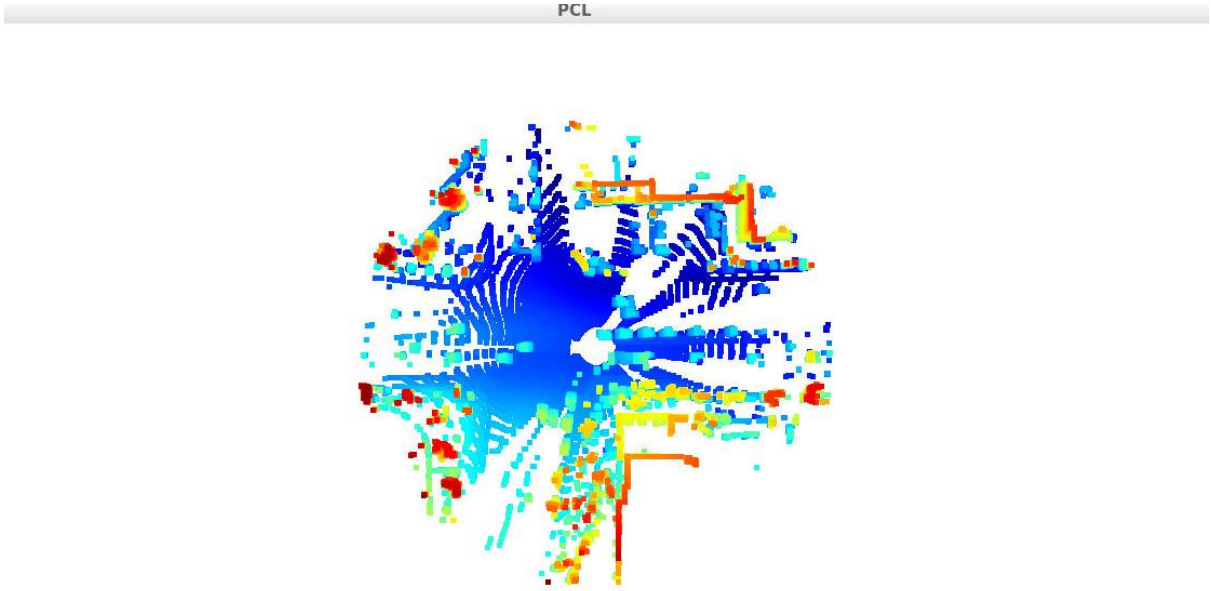


Fig 2. Top view of the point cloud. Here cars can be observed.

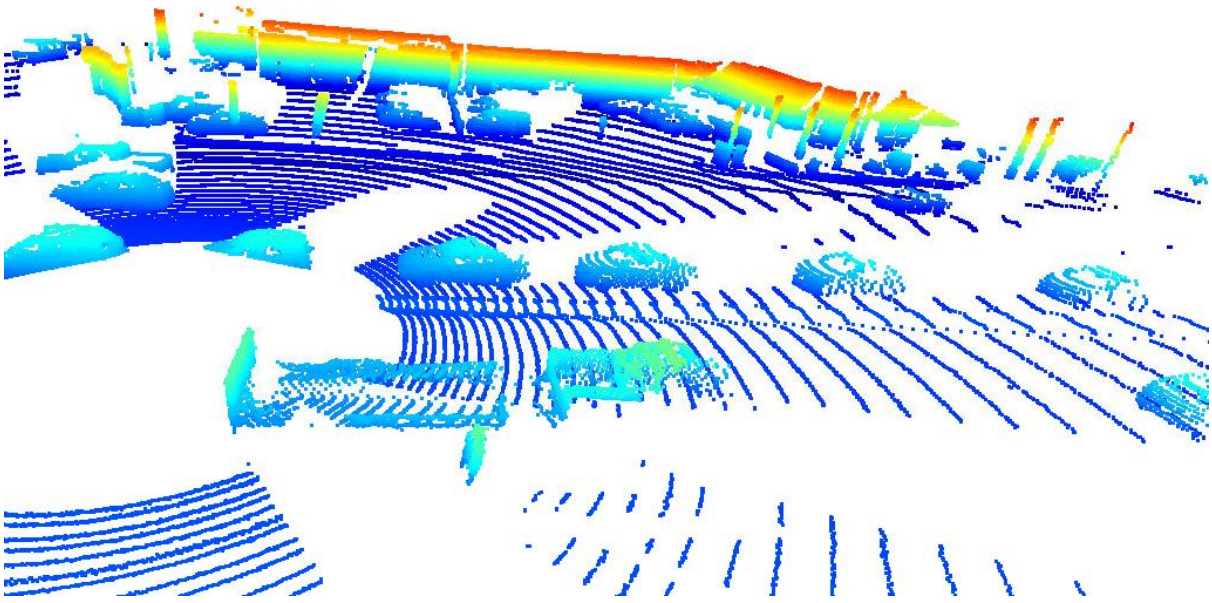


Fig 3. Point cloud showing a row of cars

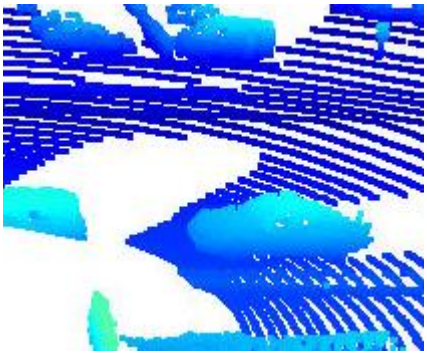


Fig. 4 Point cloud showing a relatively clear image of car

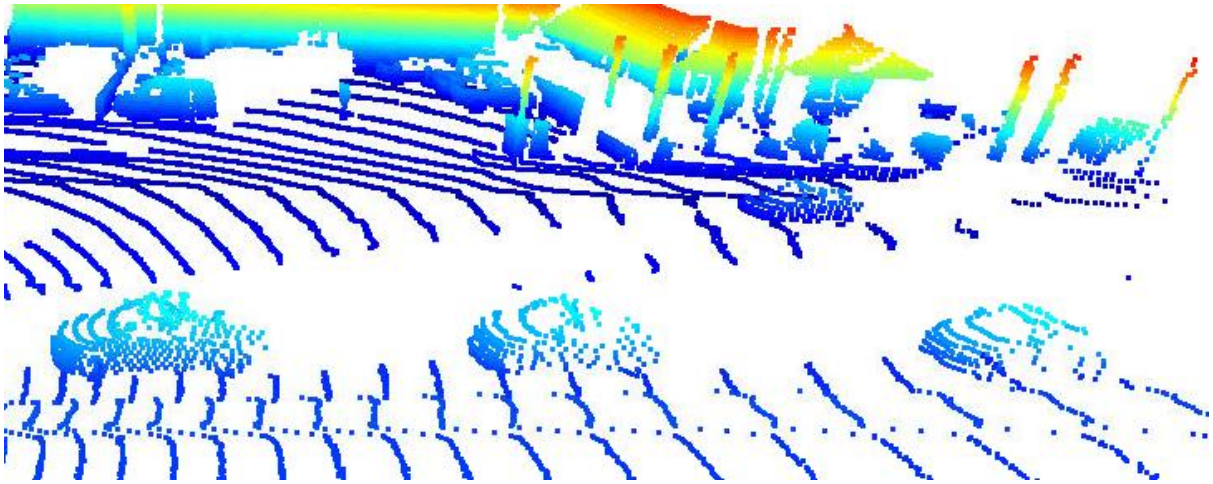


Fig 5. Point cloud with row of sparse cars

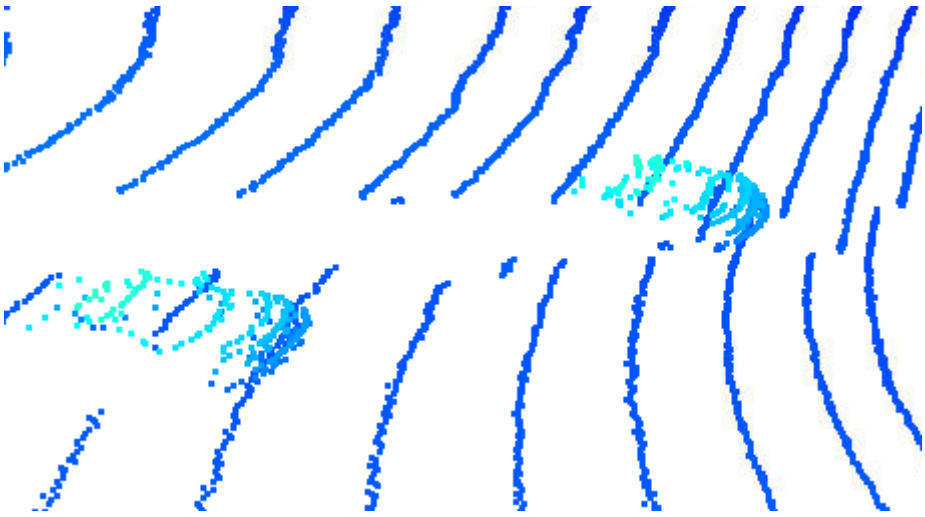
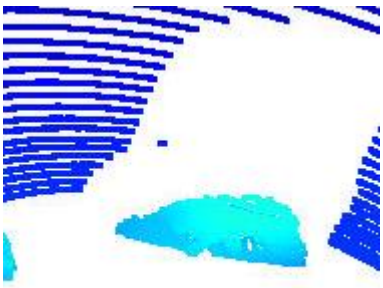


Fig 6. Less dense area where scattered point cloud of car can be observed



- Fig 7. Distorted shape of the car

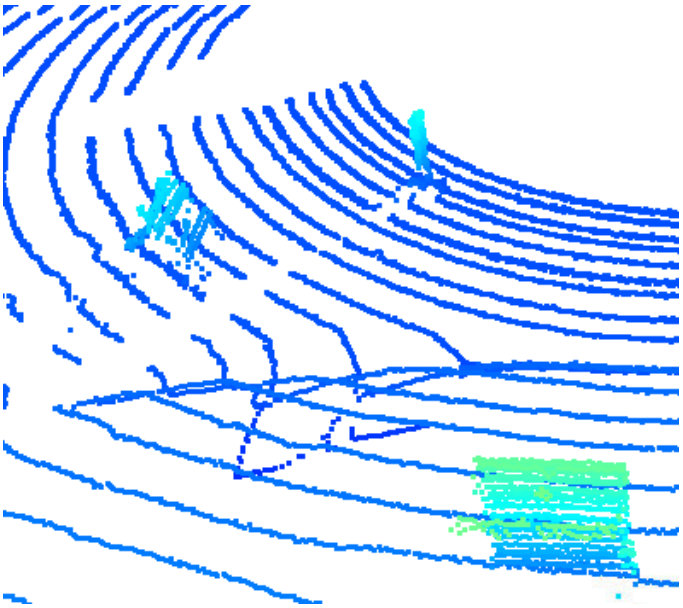


Fig 8. Distorted point cloud data. Car is visible but point cloud data is unclear

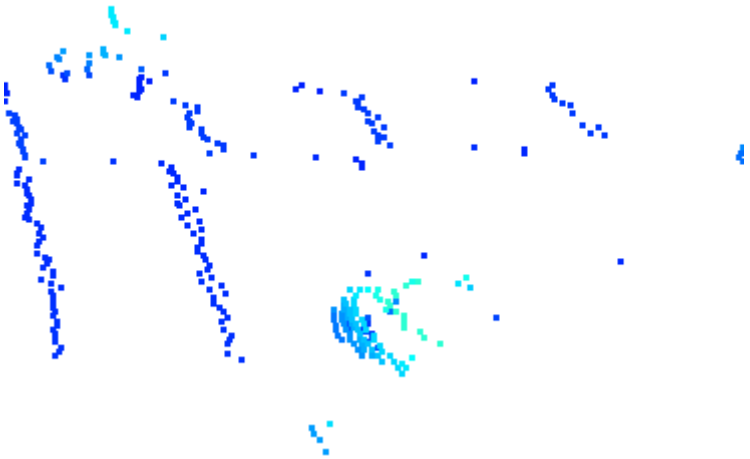


Fig. 9 Very less densed point cloud. Here also some scattered point cloud of car can be observed

- *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.*

Following are general observations

a. The cars near the centre have better visibility.
b. Front features of the car are more visible as compared to back features.
c. Tires are not clearly visible
d. Some detailed features like mirror are not visible.
e. One of the cars at front has reflection from wind screen and hence it is visible.
f. Other cars do not have reflection from the wind screen resulting in white space around the windscreen.

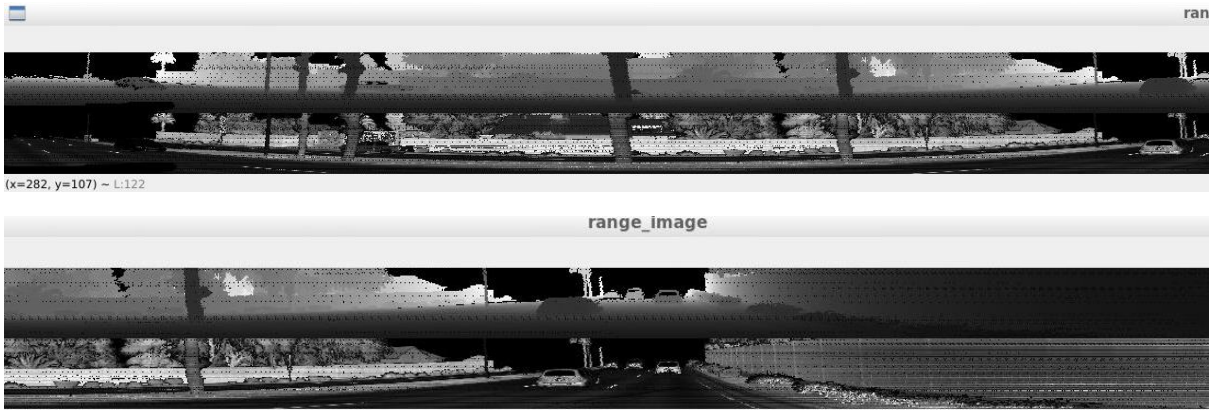


Fig. 10 Range image

Section 2: Create Birds-Eye view from LiDAR PCL

The objective of this section is to create birds eye view from LiDAR PCL and then mapping intensity and height layers of BEV map.

ID_S2_EX1: Convert sensor coordinates to BEV-map coordinates

As first step in creating birds-eye view, computing respective coordinates within the BEV coordinate space so that in subsequent tasks, the actual BEV map can be filled.

Parameters

File: loop_over_dataset.py

data_filename = 'training_segment-10050810020241296 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 = []

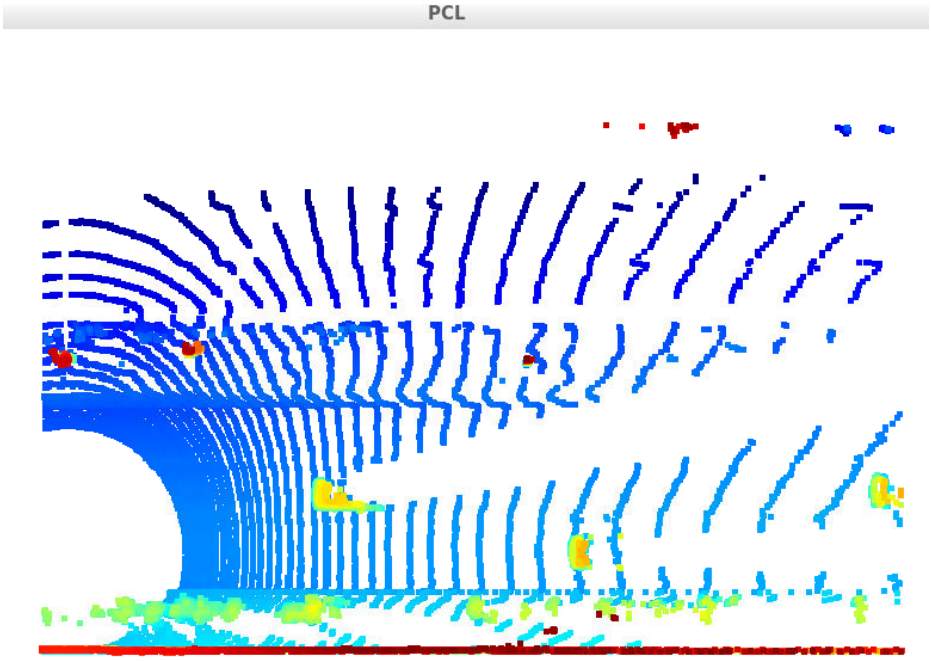


Fig. 11 BEV map

ID_S2_EX2: Compute intensity layer of the BEV map

The objective of this exercise is to fill the intensity channel of the BEV map with data from the point cloud.

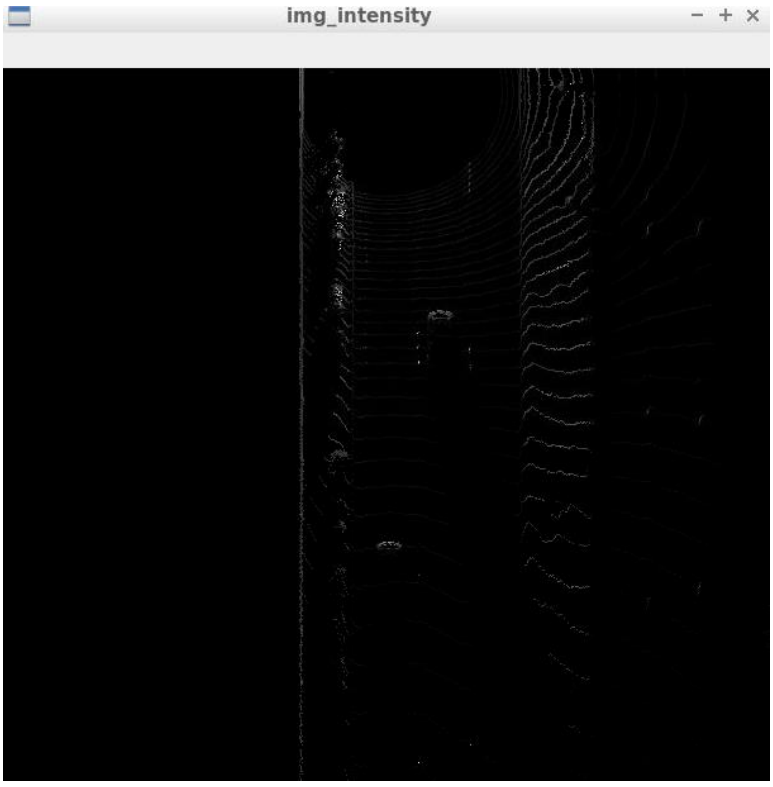


Fig. 12 Image intensity map

ID_S2_EX3: Compute height layer of the BEV map

The objective is the fill the height channel of the BEV map from the point cloud.

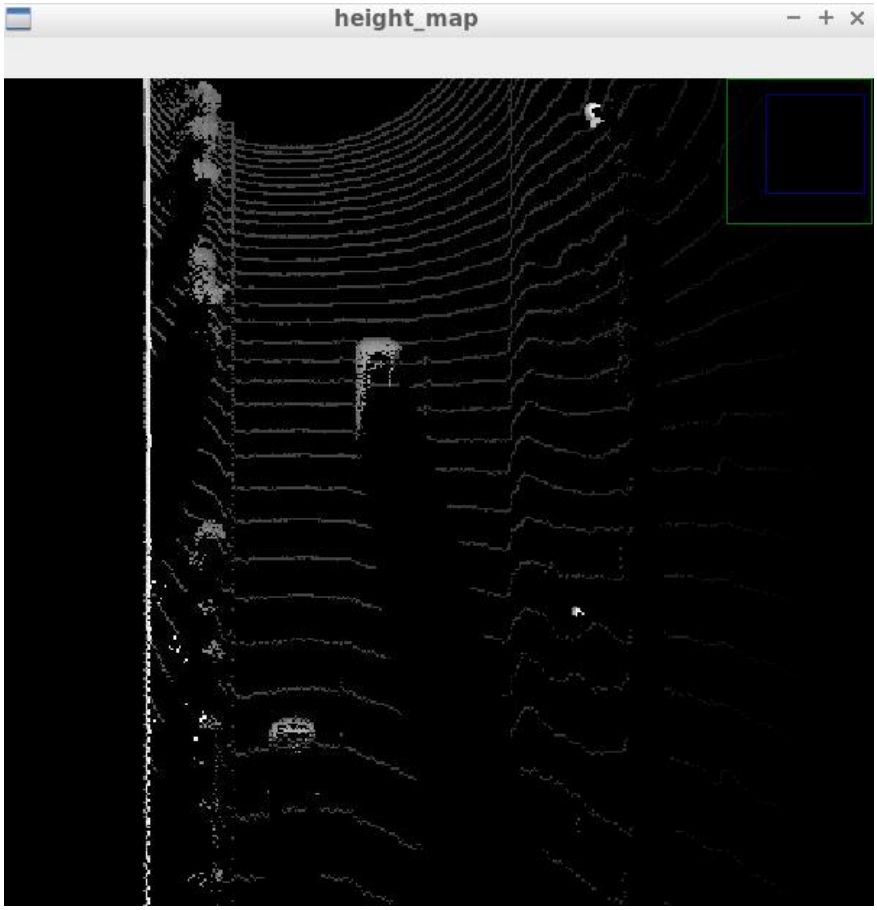


Fig. 13 Height layer map

Section 3: Model-based object detection in BEV image

The objective is to use FPN- Resnet and perform object detection. The second part of this section is to perform conversions in vehicle space.

ID_S3_EX1: Add a second model from a GitHub repo

Parameters

File: loop_over_dataset.py

data_filename = 'training_segment-1005081002024129653_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"

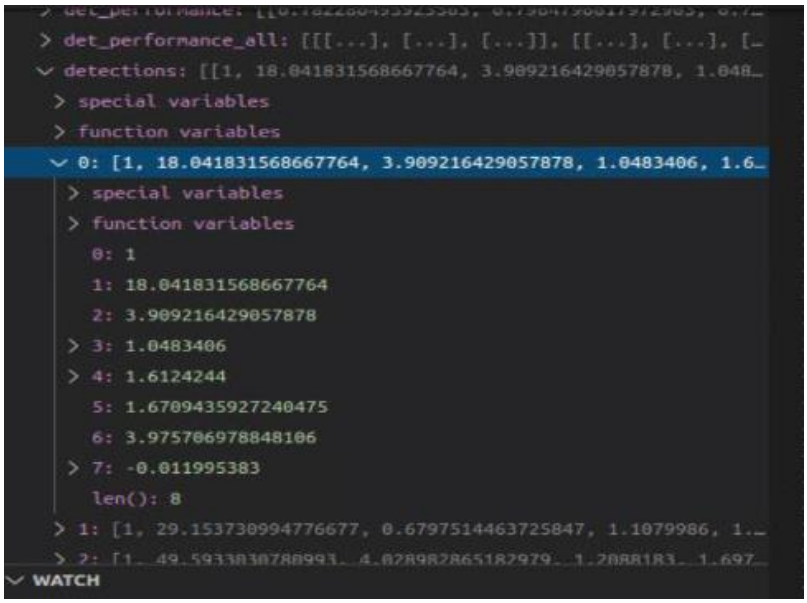


Fig. 14 Detections data after running debugging in Visual studio

ID_S3_EX2: Extract 3D bounding boxes from model responses

The objective is to convert coordinates in vehicle space before the detection and then adding 3D bounding boxes to the image.

Parameters

File: loop_over_dataset.py

data_filename = 'training_segment-1005081002024129653_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"

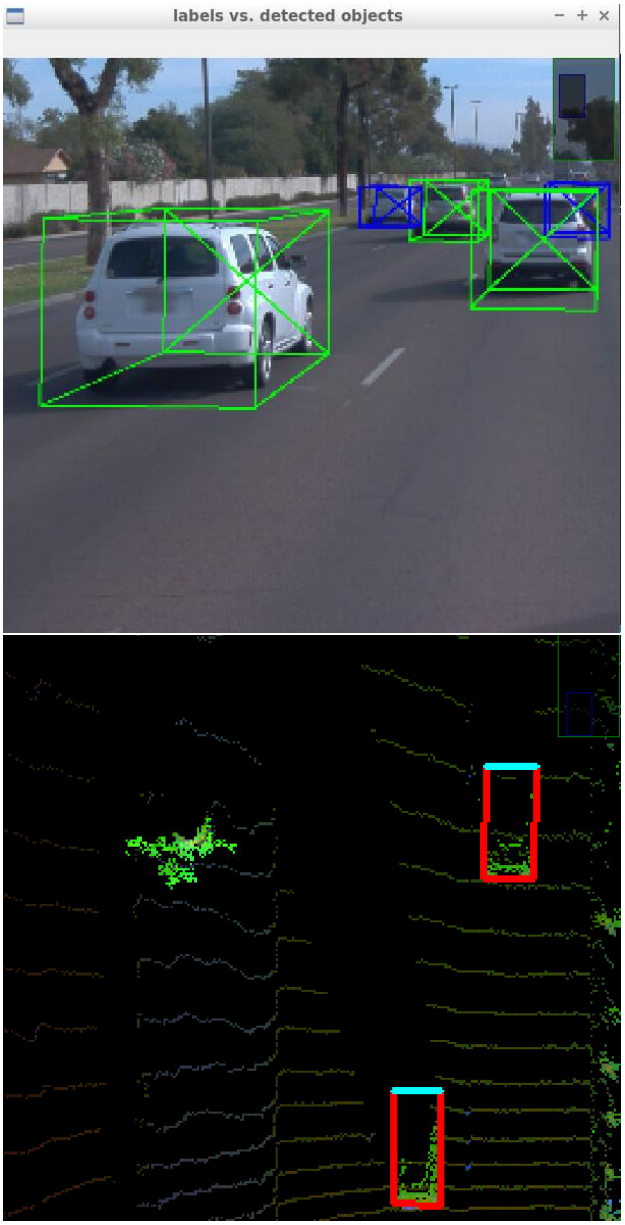


Fig. 15: 3D bounding boxes added to the image (zoomed in view)

Section 4: Performance Evaluation for Object Detection

The objective is to visualize the overall performance based on parameters like Precision, Recall, IoU and other position parameters. Steps involves computing IoU, false negative and false positives, precision and recall. The graphs are plotted as below.

ID_S4_EX1-3:

Parameters

File: loop_over_dataset.py

data_filename = 'training_segment-10050810020241296 53_5313_150_5333_150_with_camera_labels.tfrecord

show_only_frames = [50, 150]

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")

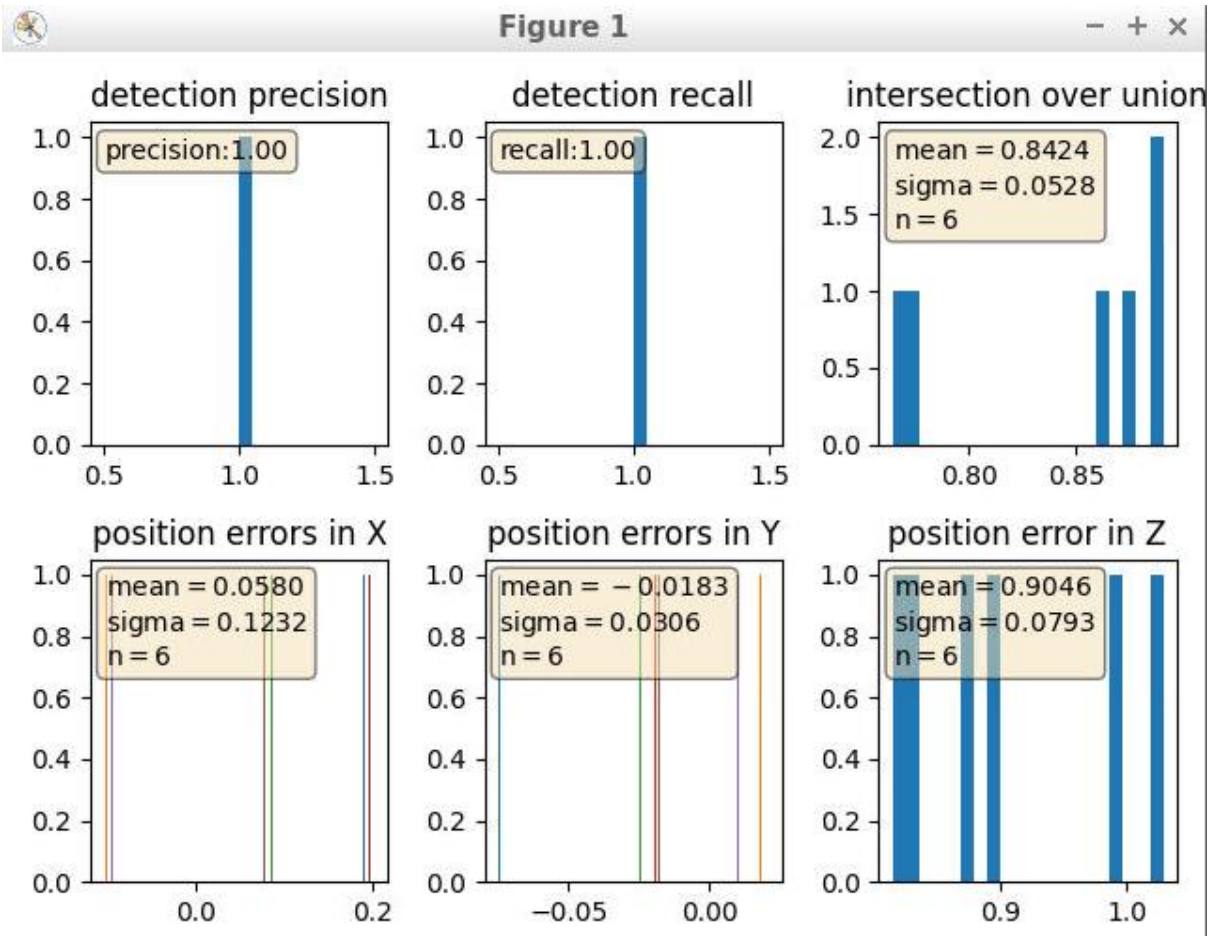


Fig. 16: Graphs setting configs_det.use_labels_as_objects as False

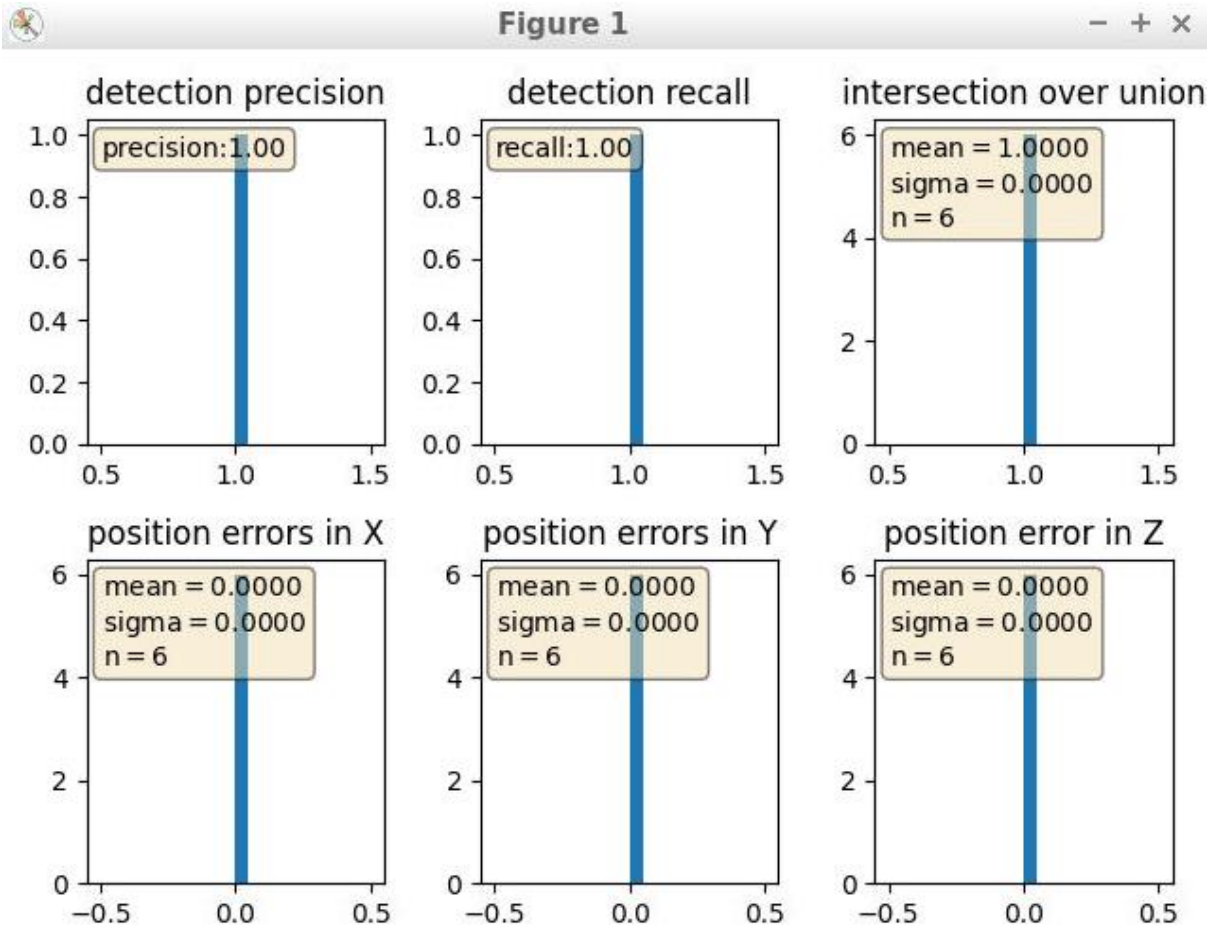


Fig. 17 Graphs setting configs_det.use_labels_as_objects as True