Visualize range image channels (ID_S1_EX1)

Task preparations

In file loop_over_dataset.py, set the attributes for code execution in the following way:

```
data_filename = 'training_segment-
10050810020241296 53_5313_150_5333_150_with_camera_labels.tfrecord' #
show_only_frames = [0, 1]
model = "darknet"
sequence = "1"
configs_det = det.load_configs(model_name='darknet')
## Selective execution and visualization
exec_detection = []
exec_tracking = []
exec_visualization = ['show_range_image']
exec_list = make_exec_list(exec_detection, exec_tracking,
exec_visualization)
```

The resultant image is shown as

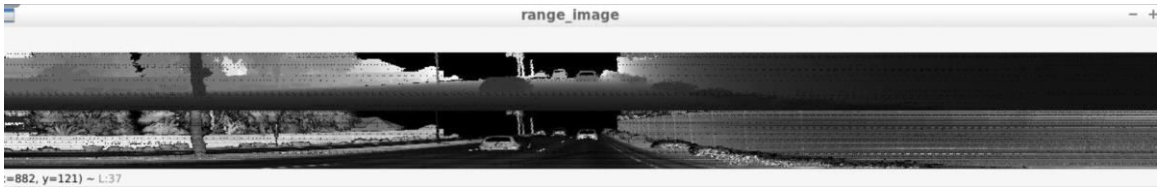

Fig 1: Range_Image

Visualize lidar point-cloud (ID_S1_EX2)

Task preparations

In file loop_over_dataset.py, set the attributes for code execution in the following way:

```
data_filename = data_filename = 'training_segment-
10963653239323173269_1924_000_1944_000_with_camera_labels.tfrecord'
show_only_frames = [0, 200]
model = "darknet"
sequence = "3"
configs_det = det.load_configs(model_name='darknet')
## Selective execution and visualization
exec_detection = []
exec_tracking = []
exec_visualization = ['show_pcl']
exec_list = make_exec_list(exec_detection, exec_tracking,
exec_visualization)
```

An example frame is shown below

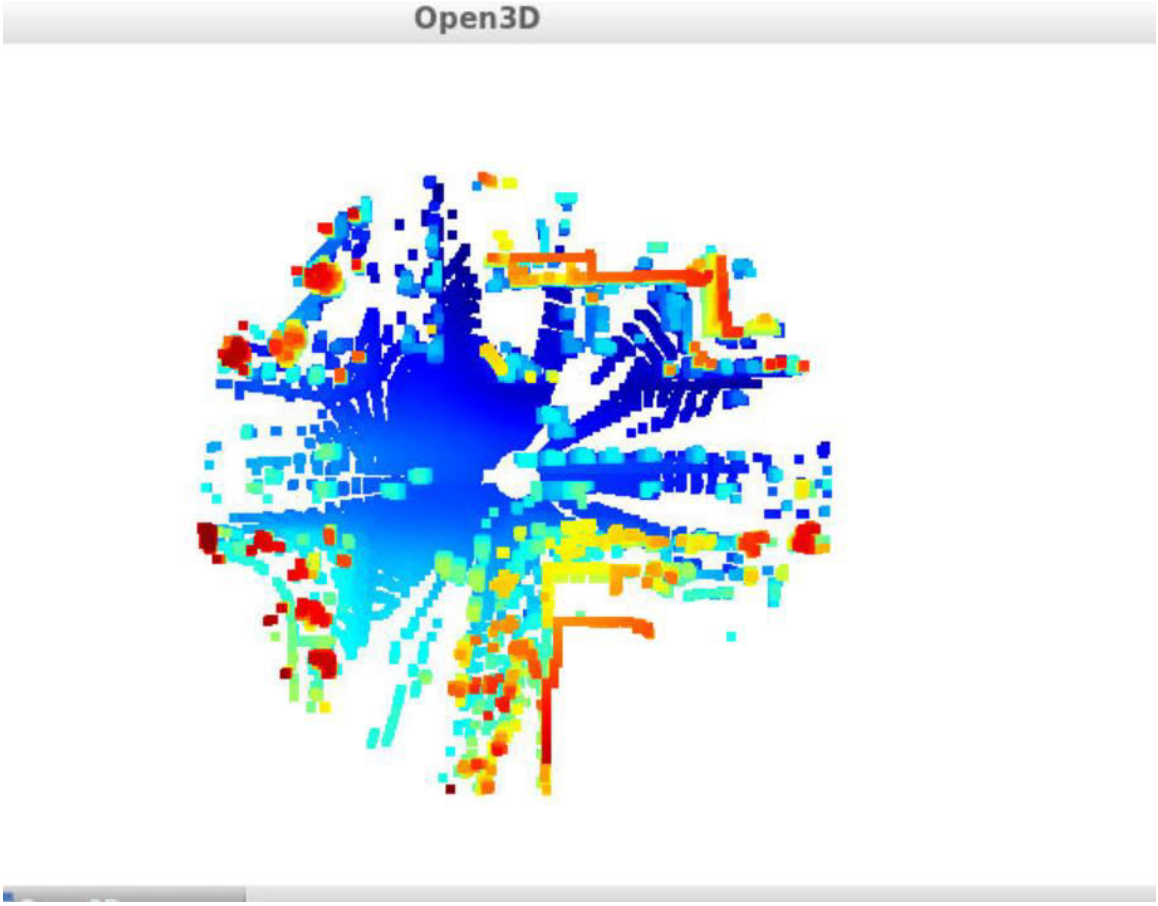

Fig 2. Visualization of LIDAR point cloud

Find 10 examples of vehicles with varying degrees of visibility in the point-cloud

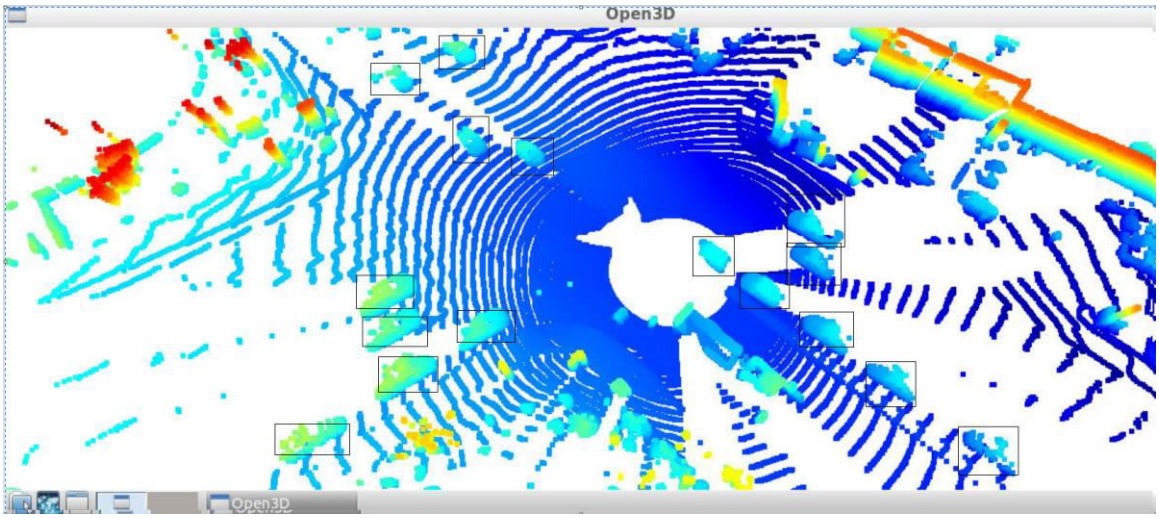

Fig 3.1

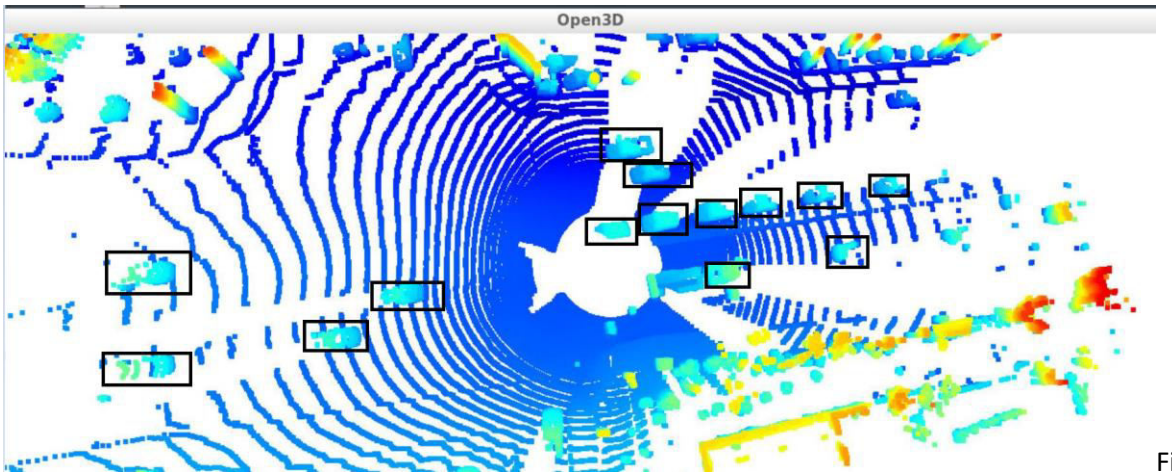

Fig 3.2

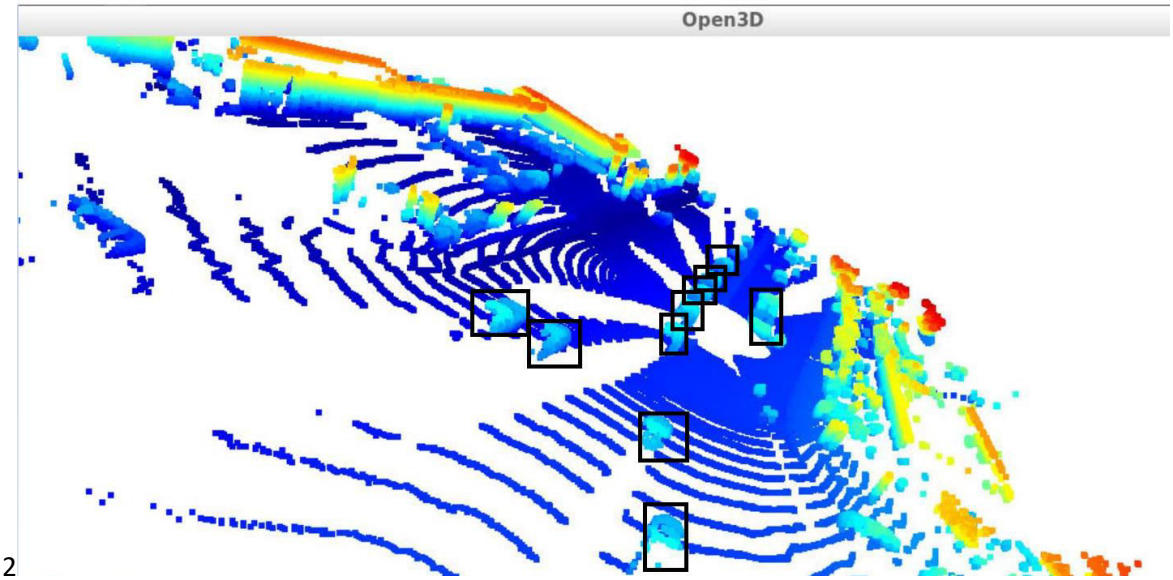

Fig 3.3

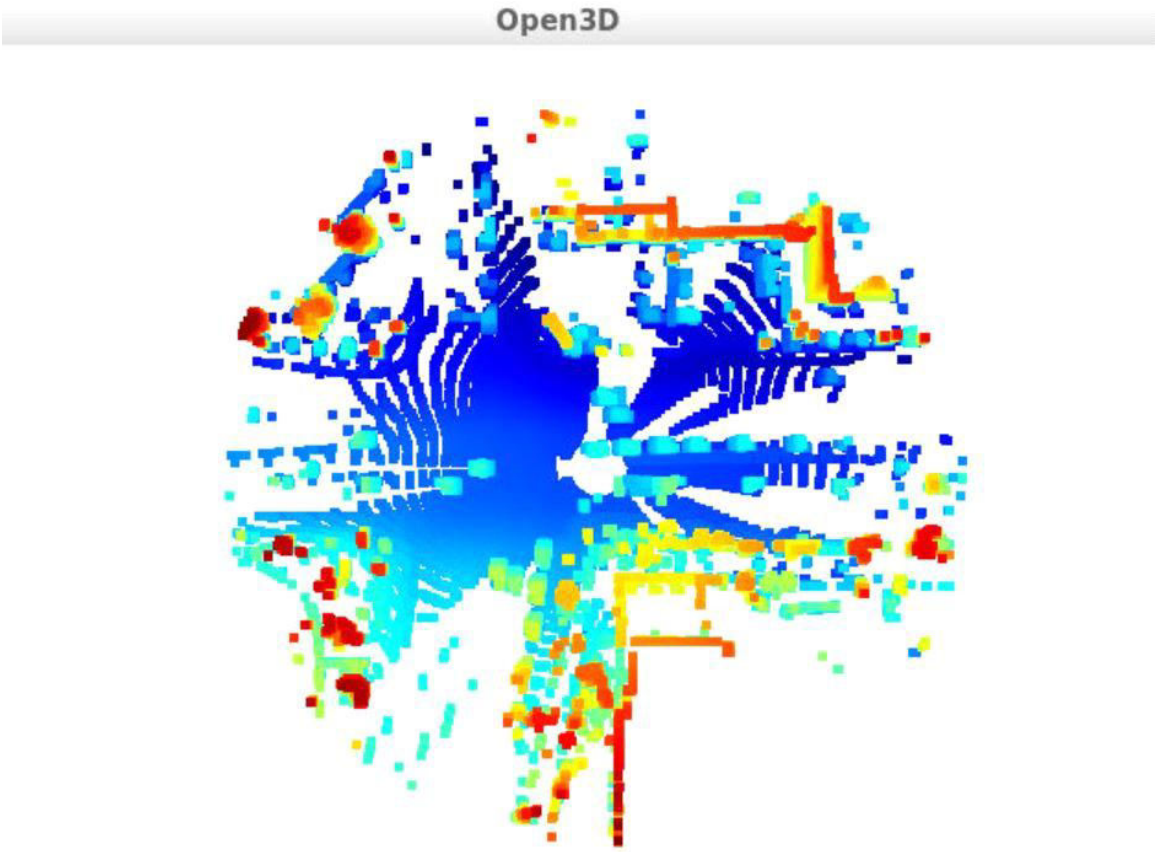

Fig 3.4

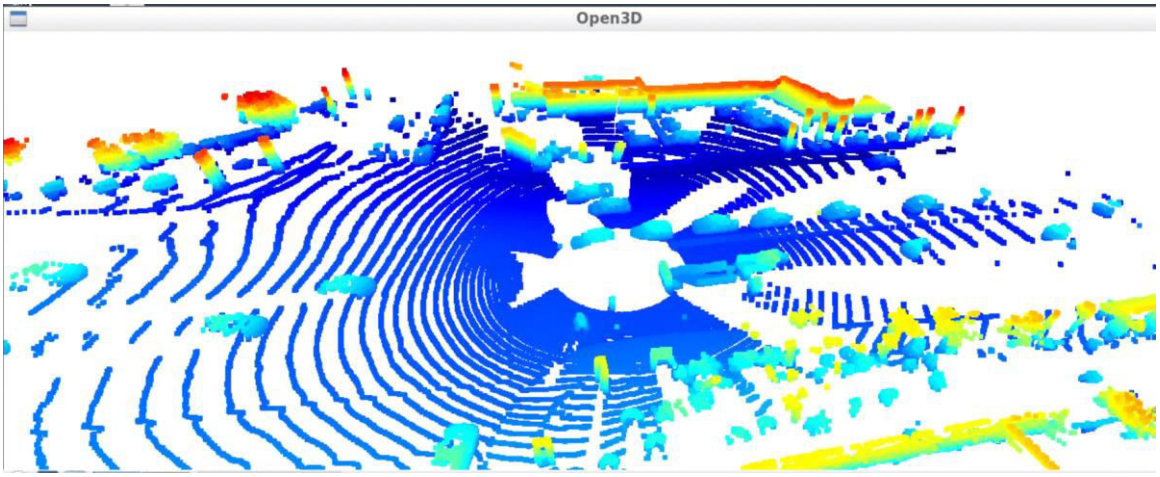

Fig 3.5

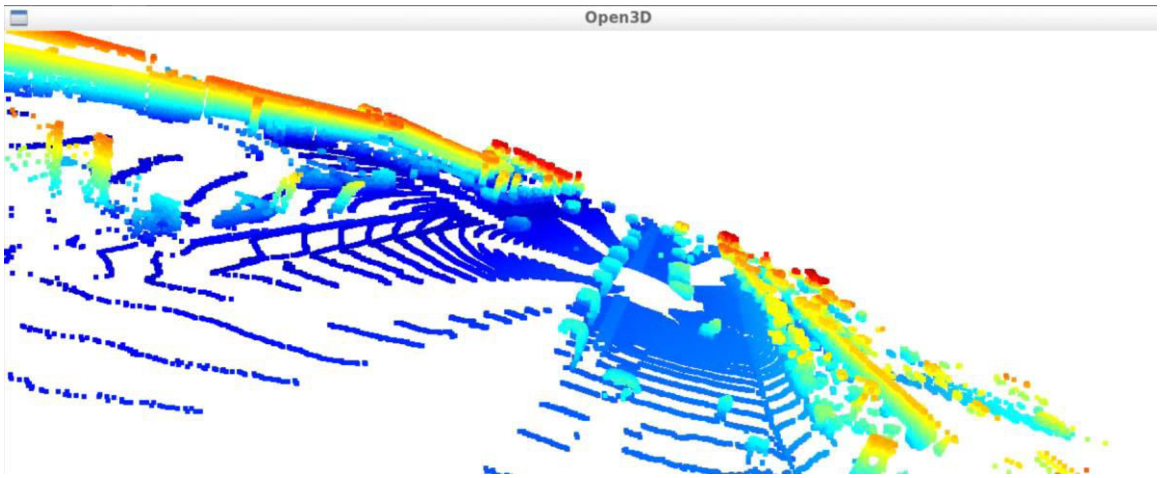

Fig 3.6

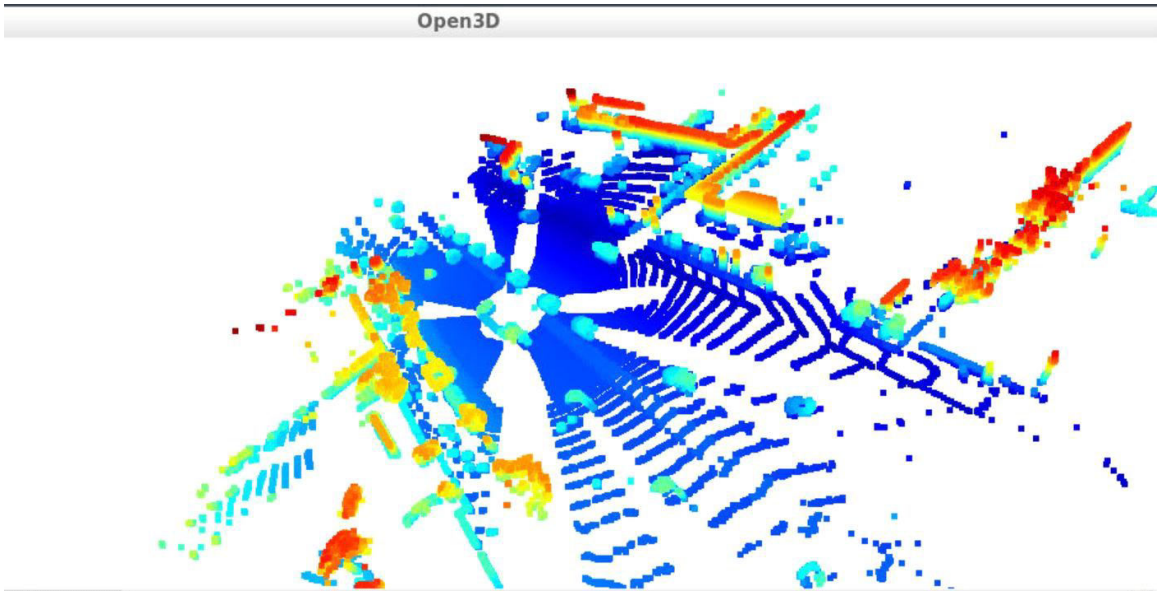

Fig 3.7

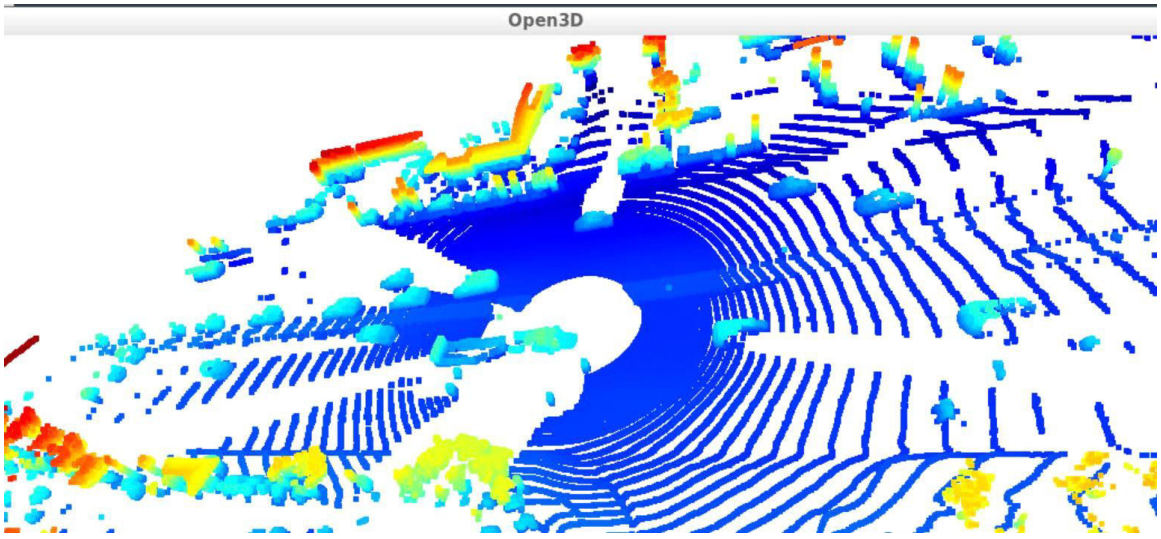

Fig 3.8

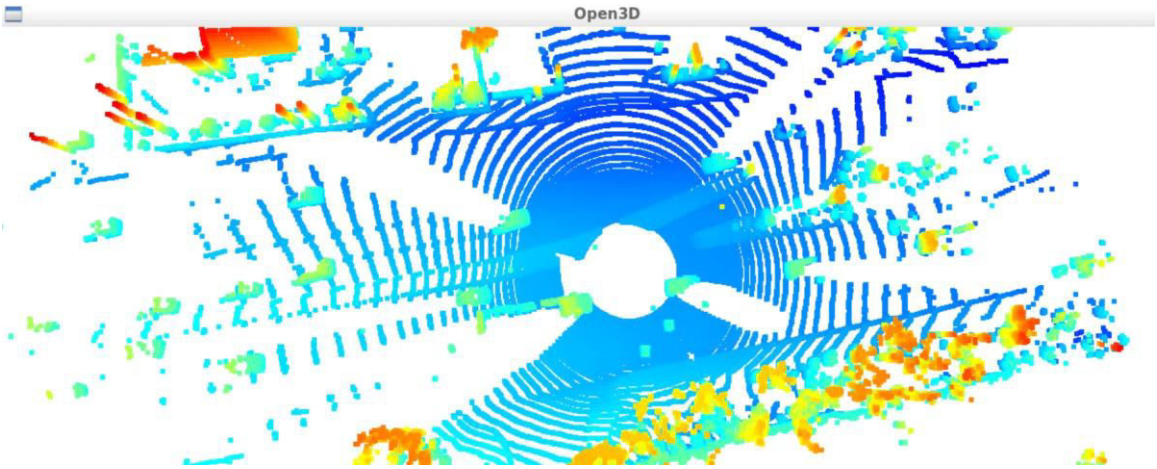

Fig 3.9

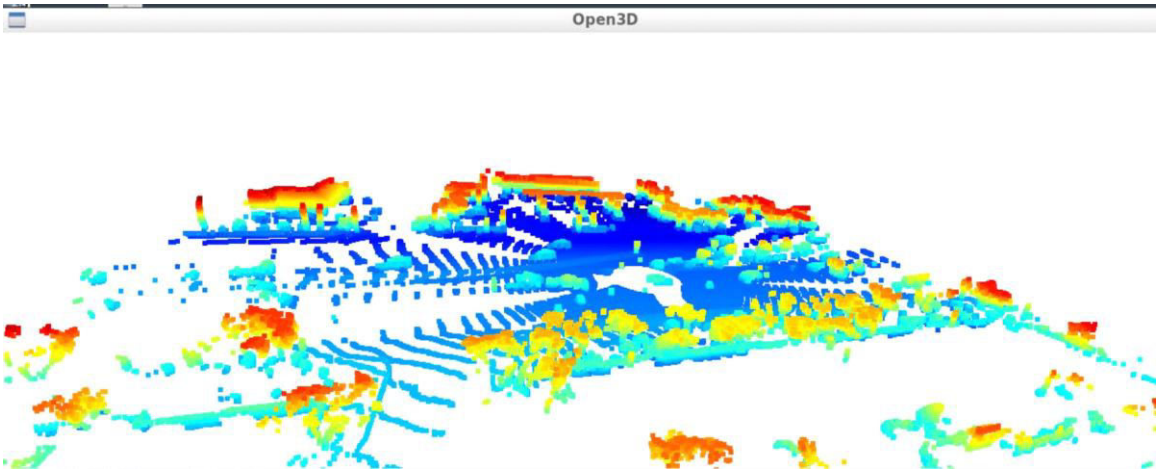

Fig 3.10

Try to identify vehicle features that appear stable in most of the inspected examples and describe them

The most prominent vehicle feature visible is the front bumper of the car.In many cases the roof as well as the sides are also visible.

Section 2 : Create Birds-Eye View from Lidar PCL

Convert sensor coordinates to BEV-map coordinates (ID_S2_EX1)

Compute intensity layer of the BEV map (ID_S2_EX2)

Compute height layer of the BEV map (ID_S2_EX3)

In file loop_over_dataset.py, set the attributes for code execution in the following way:

```
data_filename = 'training_segment-
1005081002024129653_5313_150_5333_150_with_camera_labels.tfrecord'
show_only_frames = [0, 1]
model = "darknet"
sequence = "1"
configs_det = det.load_configs(model_name='darknet')
## Selective execution and visualization
exec_detection = ['bev_from_pcl']
exec_tracking = []
exec_visualization = []
exec_list = make_exec_list(exec_detection, exec_tracking,
exec_visualization)
```

The result images are

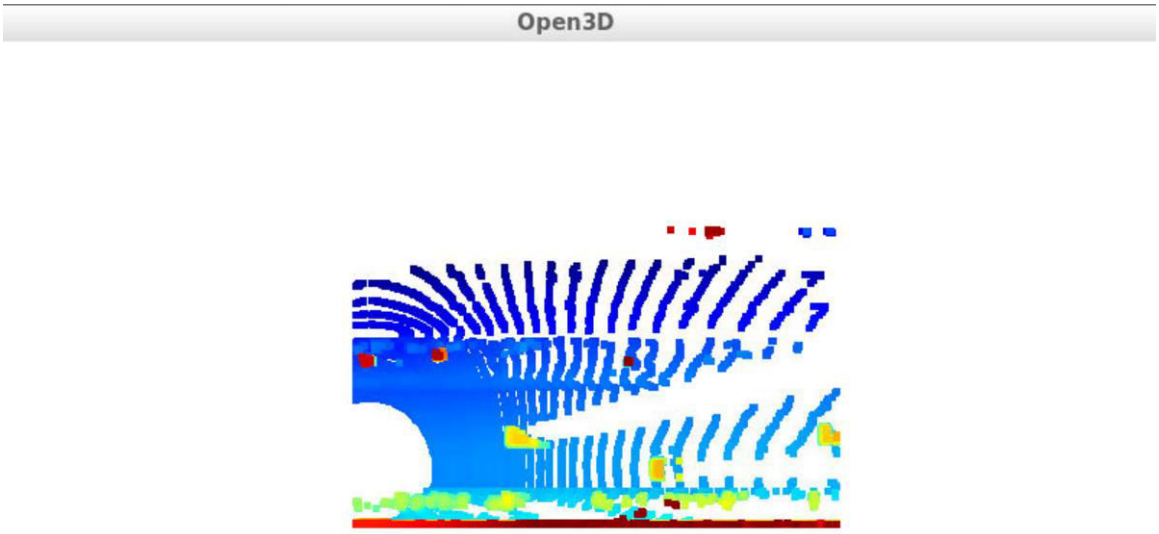

Fig 4: An example visualization to BEV map coordinates

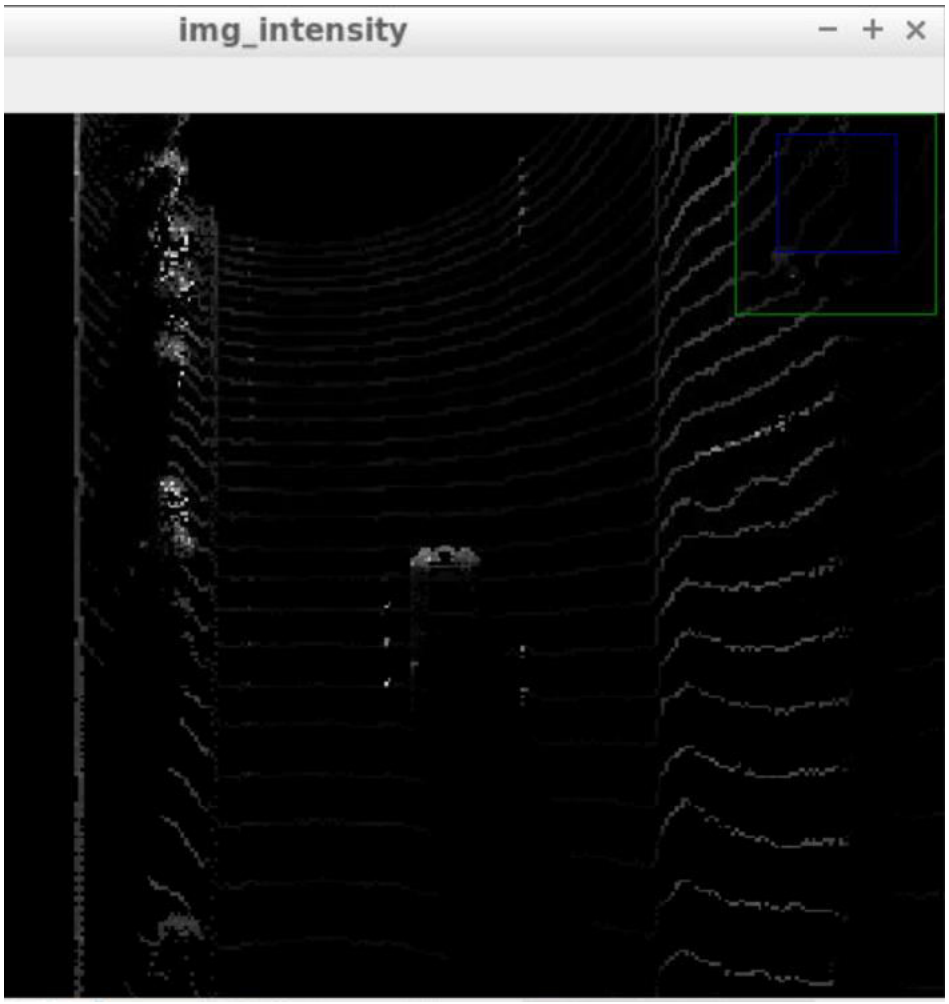

Fig 5: An example intensity layer from the BEV map

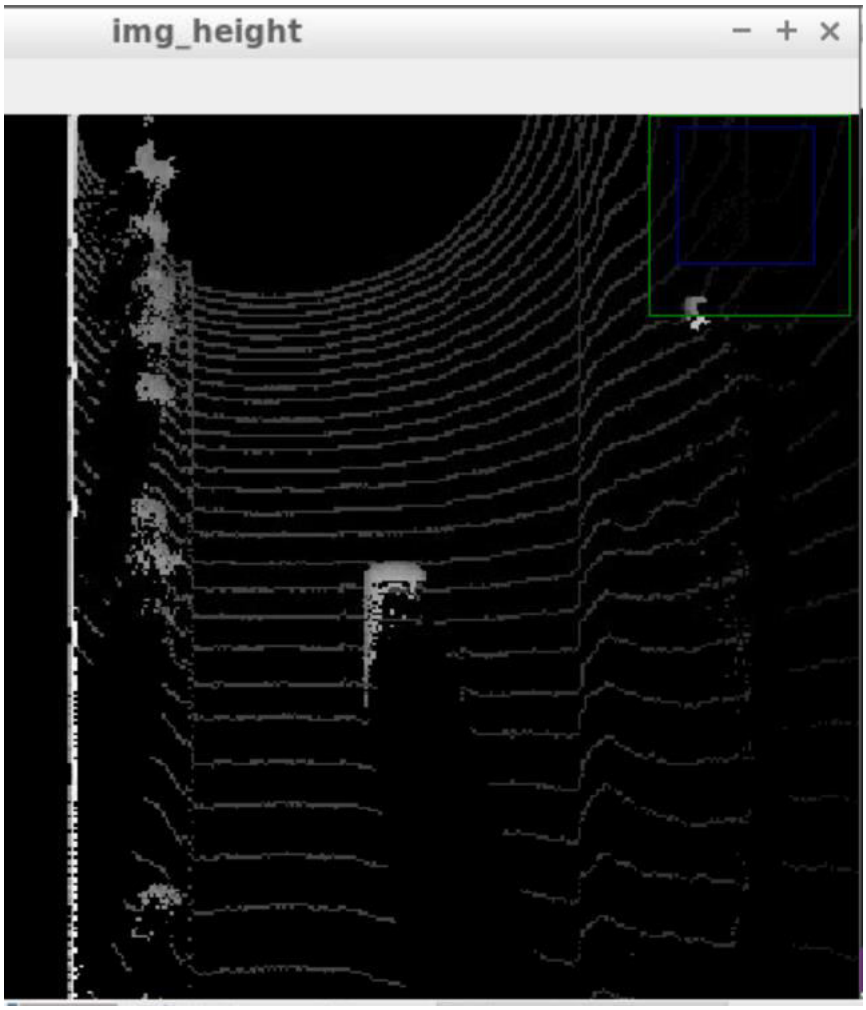

Fig 6 : An example height layer from the BEV map

Section 3 : Model-based Object Detection in BEV Image

Add a second model from a GitHub repo (ID_S3_EX1)

Extract 3D bounding boxes from model response (ID_S3_EX2)

Task preparations

In file loop_over_dataset.py, set the attributes for code execution in the following way:

```
data_filename = 'training_segment-10059810020241296 53_5313_150_5333_150_with_camera_labels.tfrecord'
show_only_frames = [50, 51]
model = "resnet"
sequence = "1"
configs_det = det.load_configs(model_name='fpn_resnet')
## Selective execution and visualization
exec_detection = ['bev_from_pcl','detect_objects']
exec_tracking = []
exec_visualization = ['show_objects_in_bev_labels_in_camera']
exec_list = make_exec_list(exec_detection, exec_tracking, exec_visualization)
```

In the file objdet_tools.py in 'misc' the following changes were made

```
# show combined view
out_img = cv2.resize( out_img,( 500, 500))
cv2.imshow('labels vs. detected objects', out_img)
```

The result image is

Fig 7 : 3D bounding boxes added to the images

Section 4 : Performance Evaluation for Object Detection

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

Compute precision and recall (ID_S4_EX3)

```
data_filename = 'training_segment-10059810020241296 53_5313_150_5333_150_with_camera_labels.tfrecord'
show_only_frames = [50, 150]
model = "darknet"
sequence = "1"
configs_det = det.load_configs(model_name='darknet')
## Selective execution and visualization
exec_detection = ['bev_from_pcl', 'detect_objects', 'validate_object_labels', 'measure_detection_performance']
exec_tracking = []
exec_visualization = ['show_detection_performance']
exec_list = make_exec_list(exec_detection, exec_tracking, exec_visualization)
```

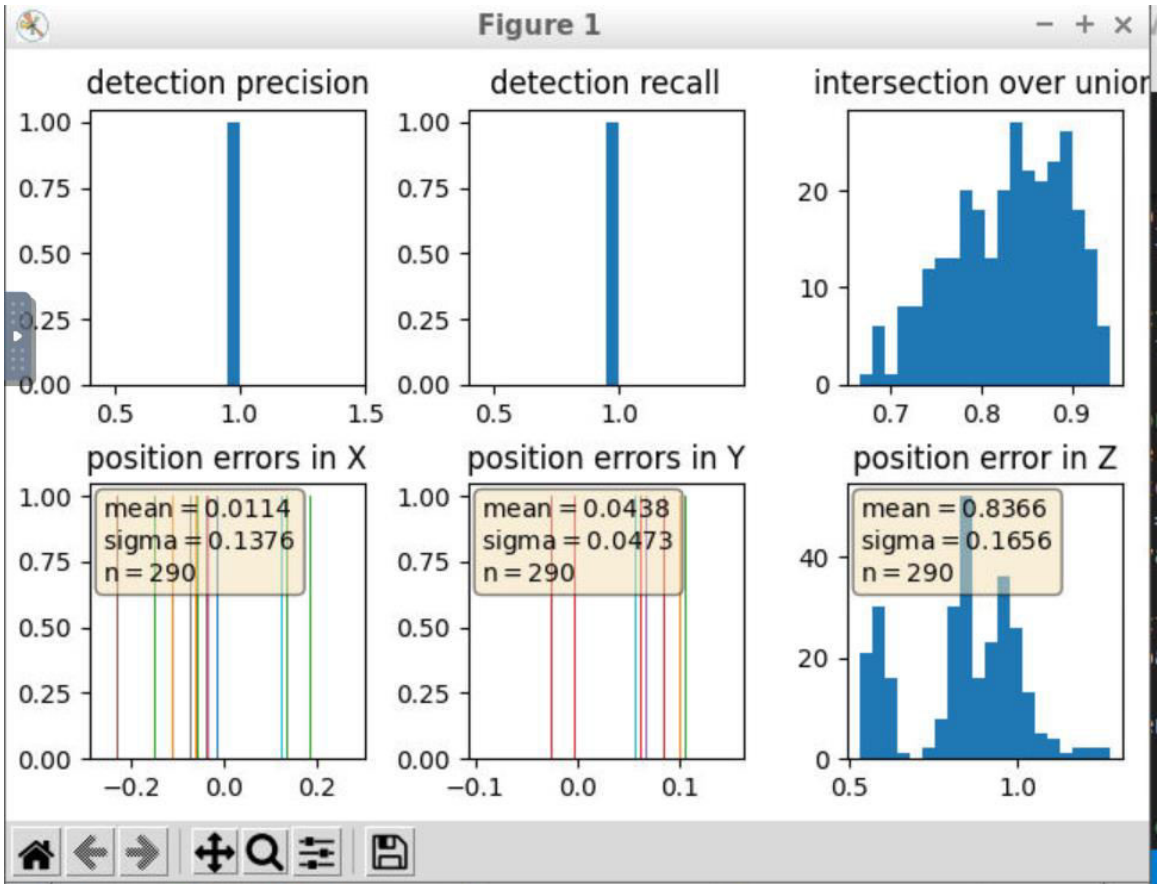

Fig 8. Graphing performance metrics