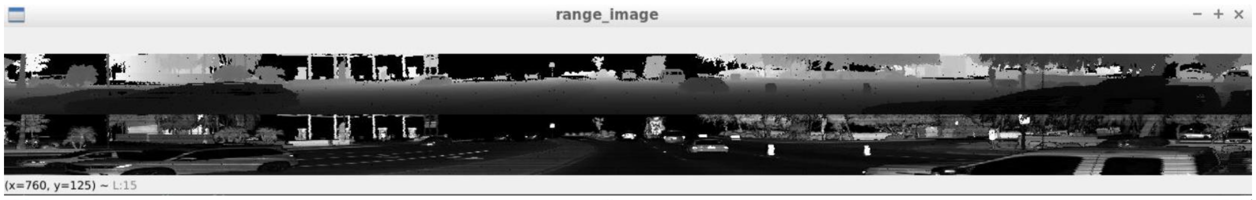
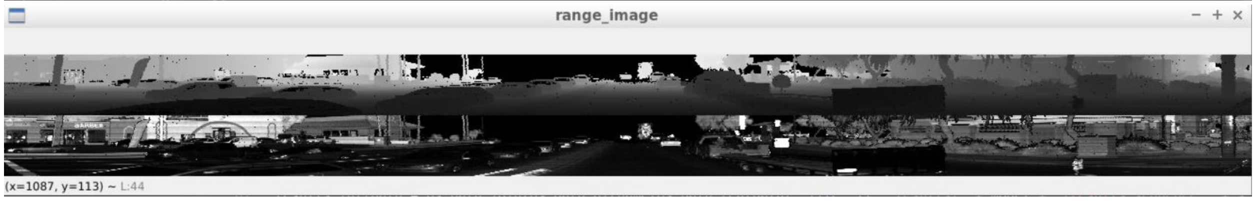
Name: Yousef Omar

Section 1 : Compute Lidar Point-Cloud from Range Image

Visualize range image channels (ID_S1_EX1)

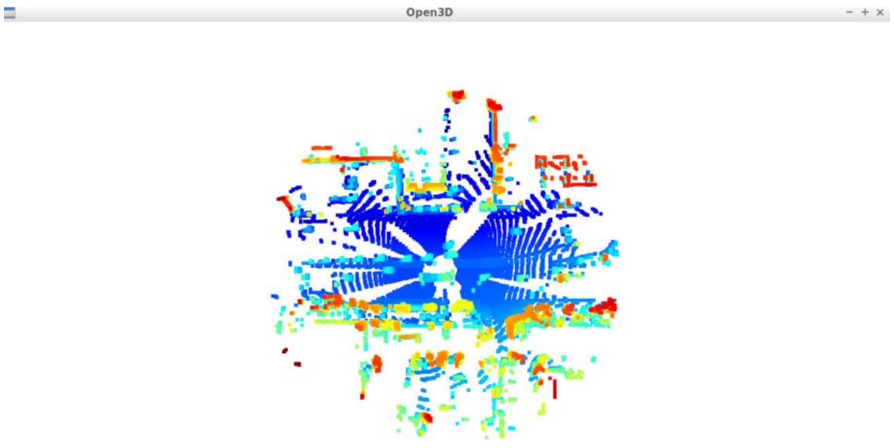
Below are a few examples from sequence #3 showing the distance information (Range) and the intensity values.

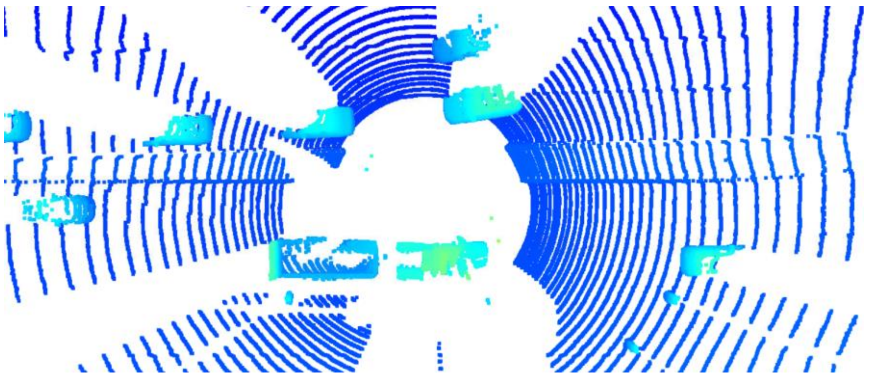


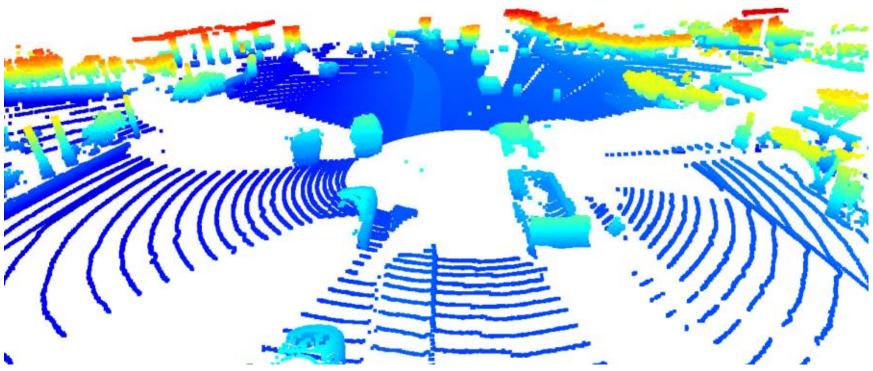


Visualize lidar point-cloud (ID_S1_EX2)

Below is an example looking at different angles:

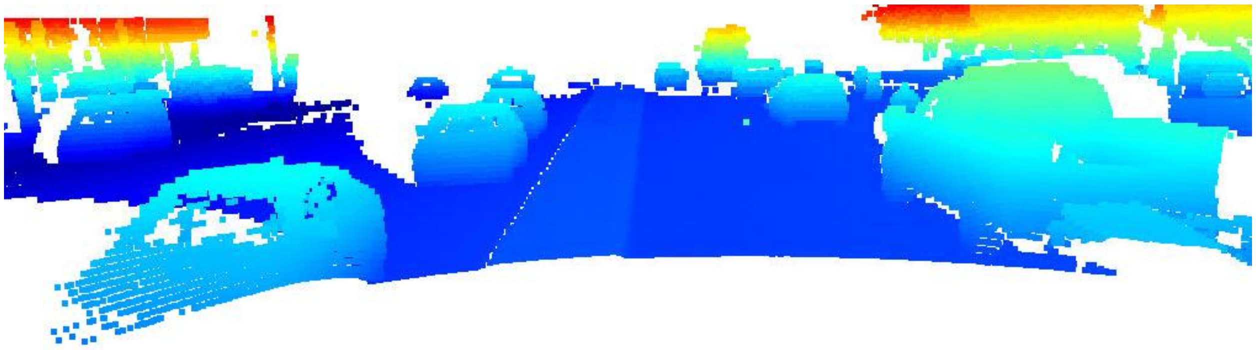




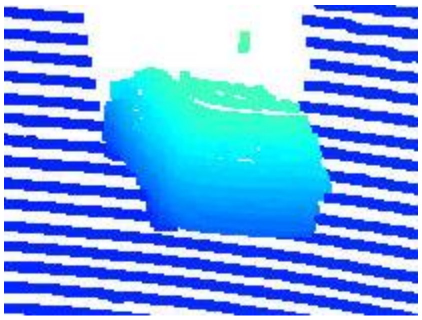


Find 10 examples of vehicles with varying degrees of visibility in the point-cloud:

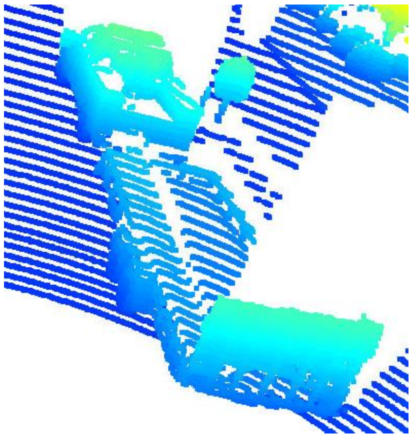
Very crowded scene of incoming and forward-facing vehicles from the vehicle's perspective:



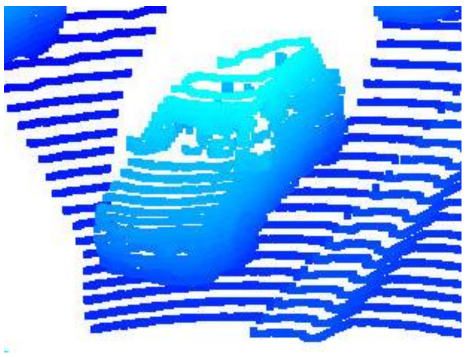
Near vehicle driving in the same direction. The point cloud is very dense capturing the rear and the left side of the vehicle very well:



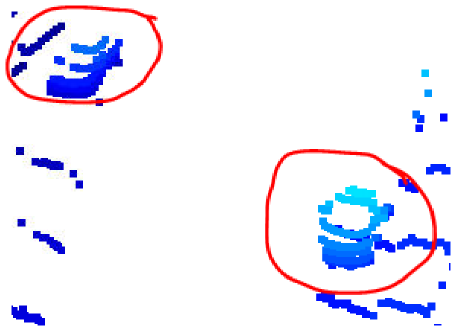
Unusual objects:



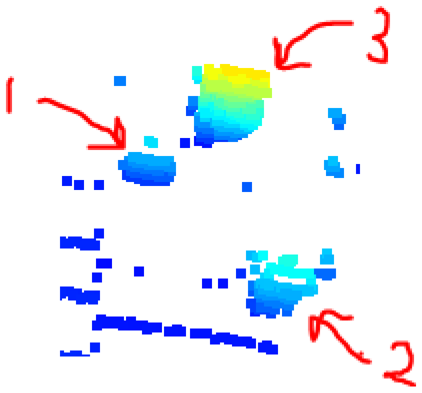
Incoming vehicle at close distance



2 incoming vehicles at far distance with very low number. Vehicle on the right is suffering from partial occlusion as well:

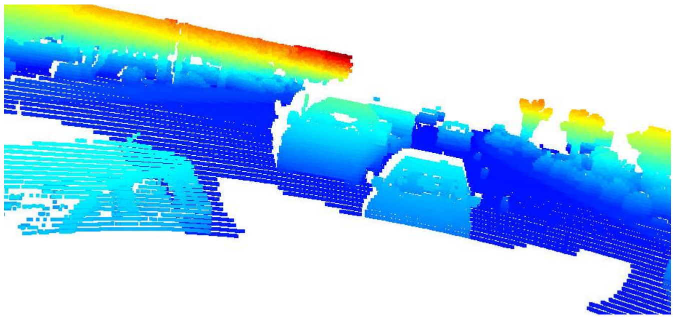


1, 2 and 3 are vehicles driving in the same direction at a very far distance. Vehicles 1 and 2 are small sedans with very limited number of lidar points. Vehicle 3 appears as a large truck.

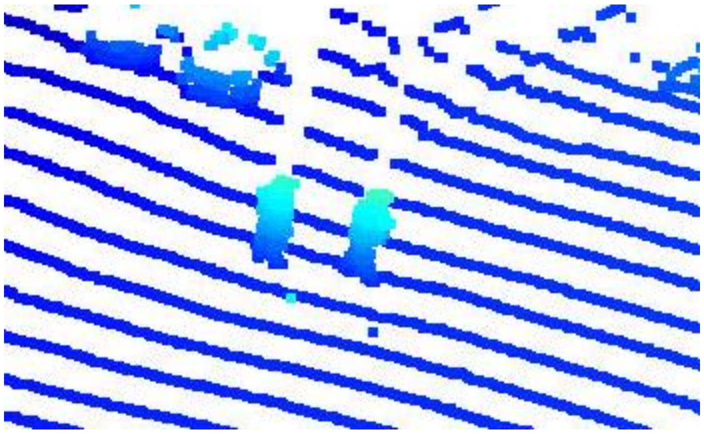


Incoming vehicle at near range, with a very large angle from the forward direction. The vehicle detects the front and the full left side of that vehicle:

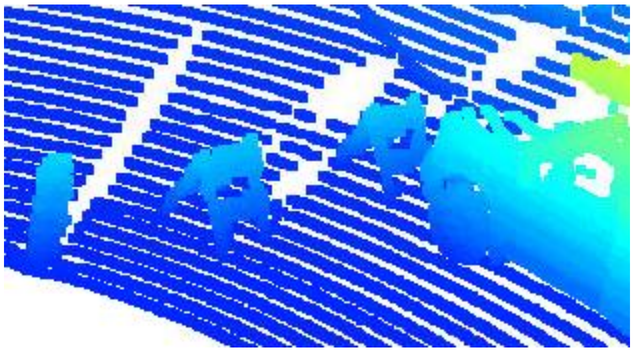
Rear view of the vehicle: If we have a rear view camera along with the current positioning of the lidar (on top of vehicle) would allow to detect incoming vehicles behind of vehicle



2 Pedestrians crossing the road:

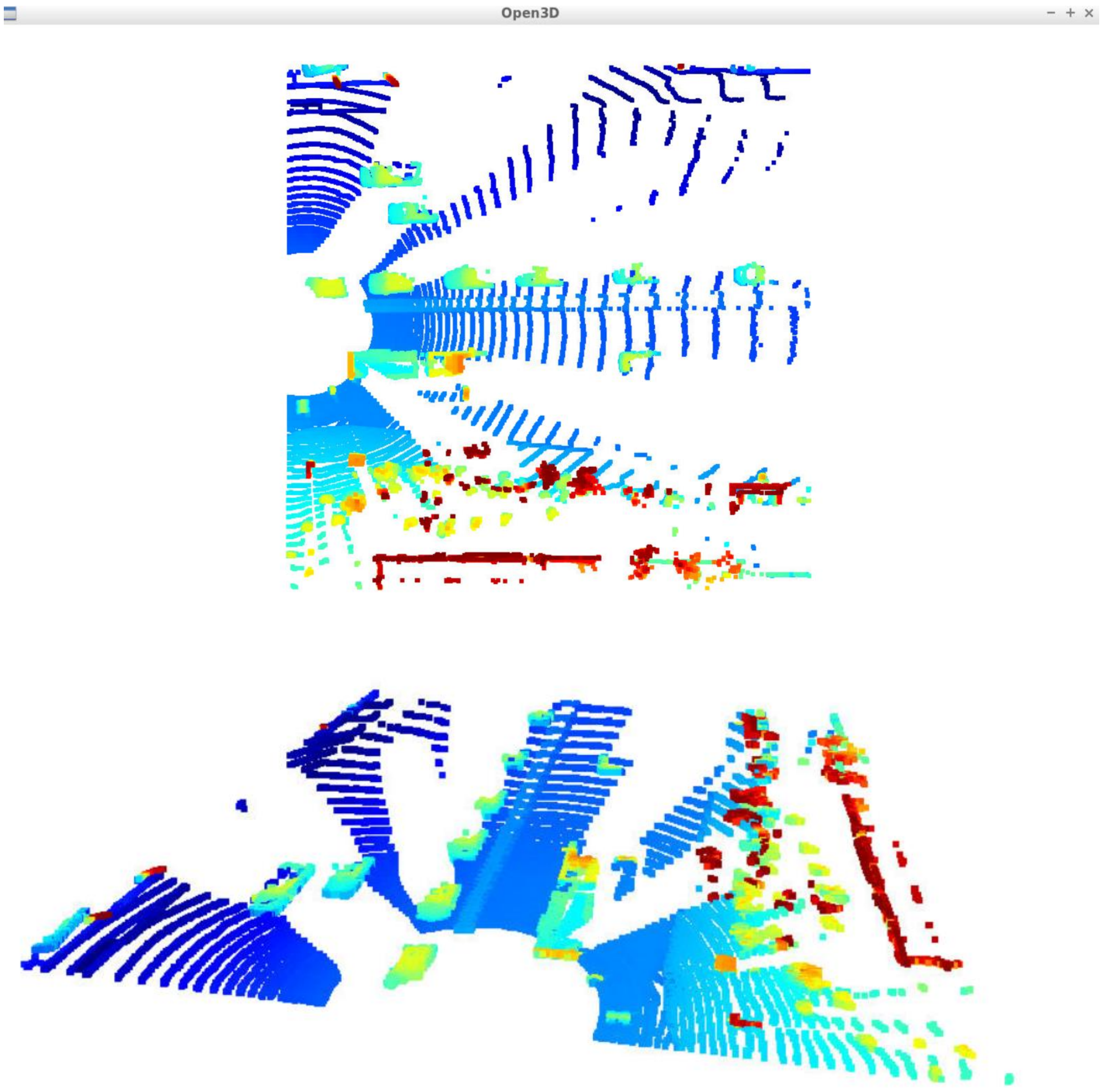


Possible Construction signs :



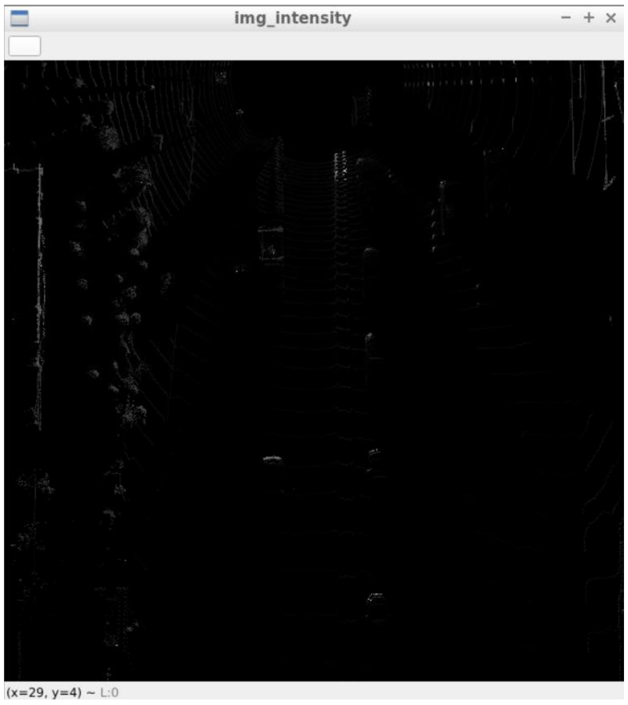
Section 2 : Create Birds-Eye View from Lidar PCL

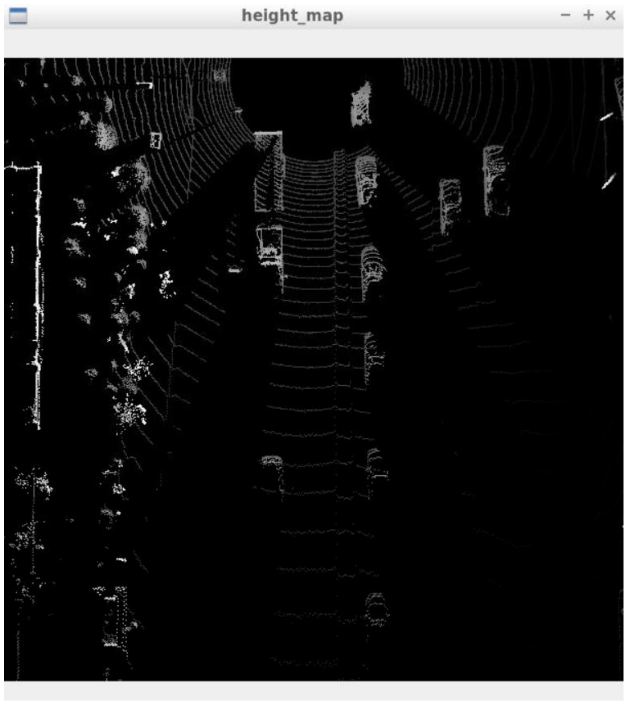
Convert sensor coordinates to BEV-map coordinates (ID_S2_EX1)



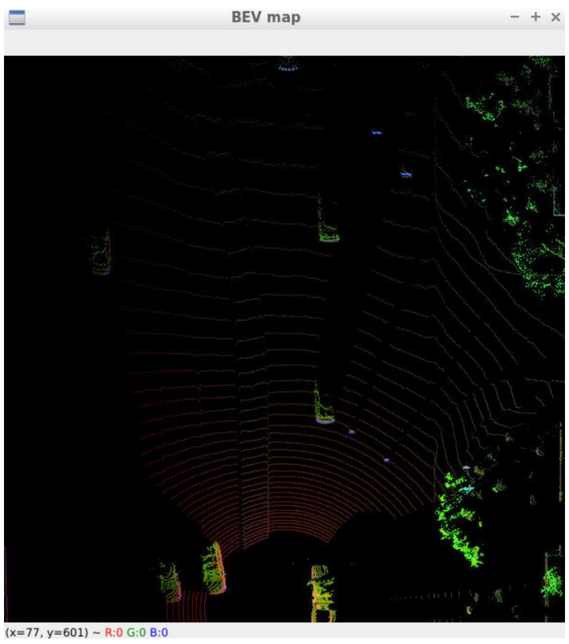
Compute layers of the BEV map (ID_S2_EX2)

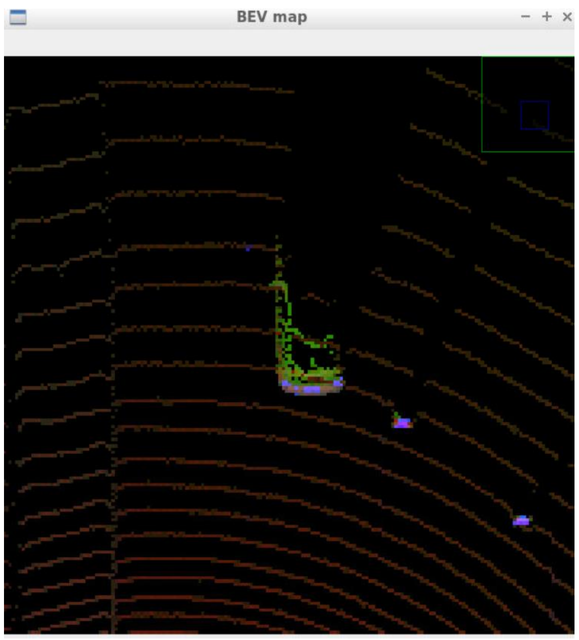
Both intensity and height layers were created for the BEV map, some images can be seen below:



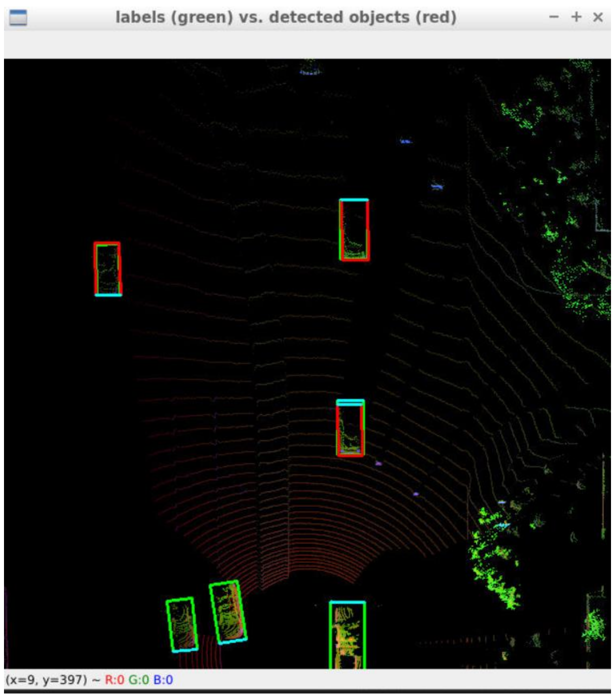


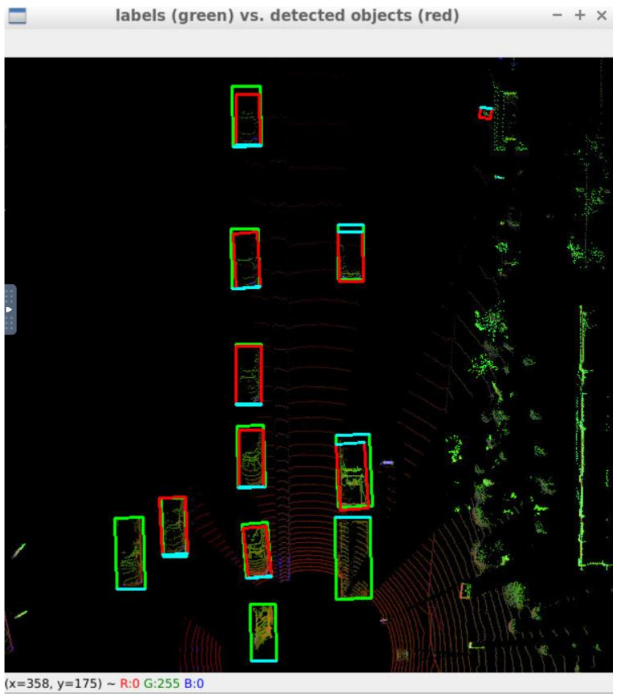
Combined result as learned from earlier exercises:





Frame 44 and frame 0 from sequence #3 showing BEV with labels and detection results from darknet:

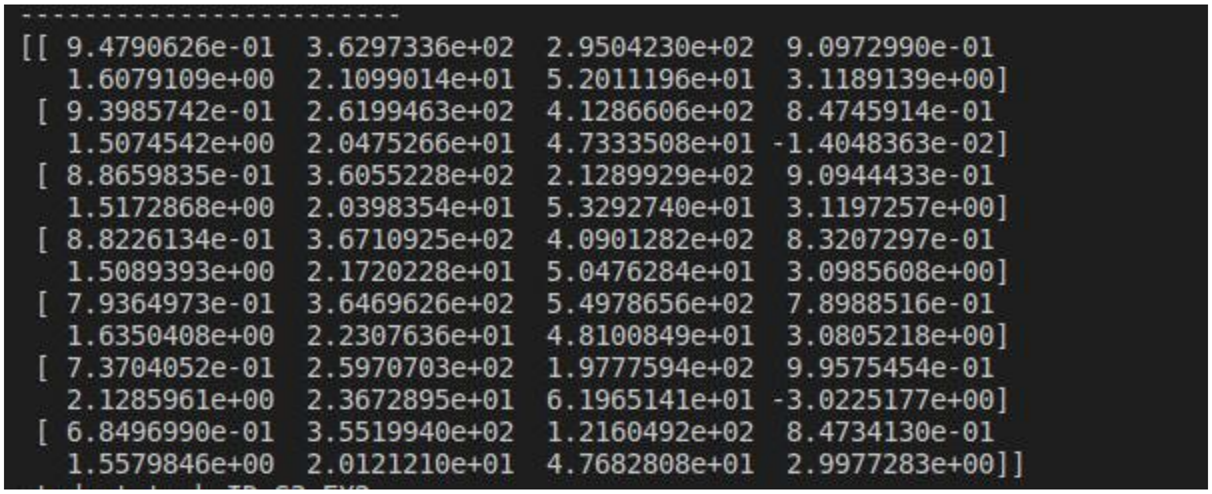

labels (green) vs. detected objects (red)
(x=9, y=397) ~ R:0 G:0 B:0


labels (green) vs. detected objects (red)
(x=358, y=175) ~ R:0 G:255 B:0

Section 3 : Model-based Object Detection in BEV Image

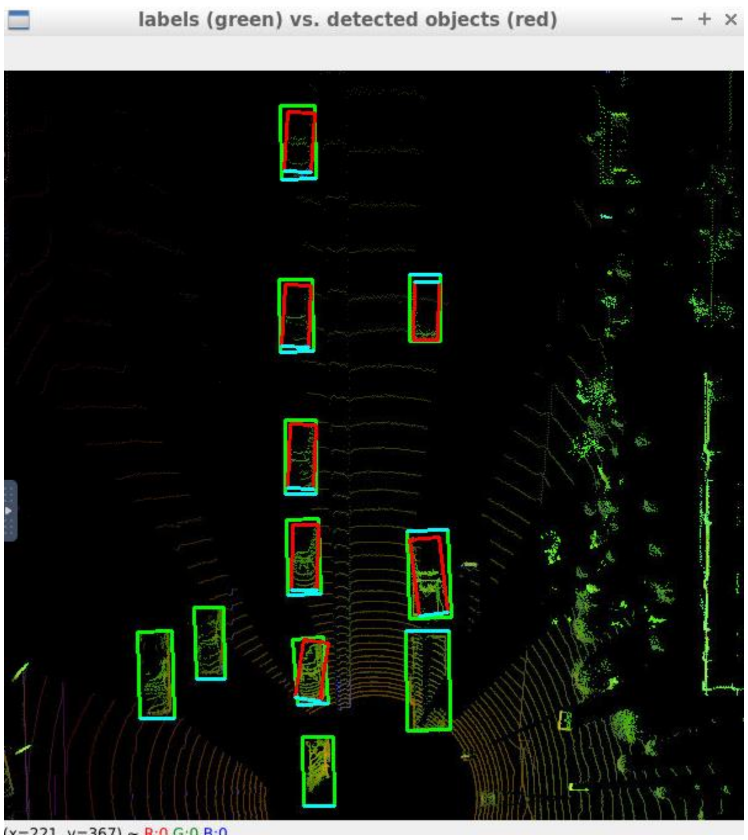
Add a second model from a GitHub repo (ID_S3_EX1)

Example of running the fpn-resnet:



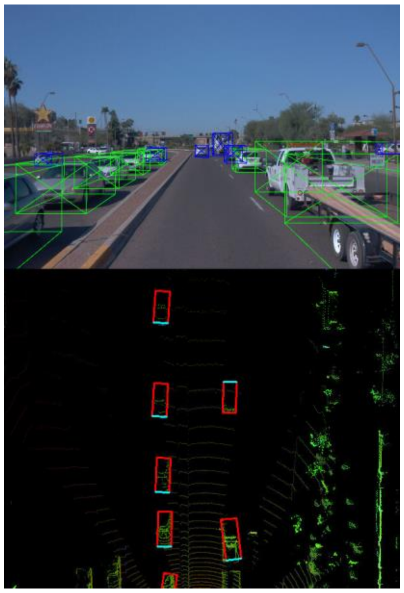
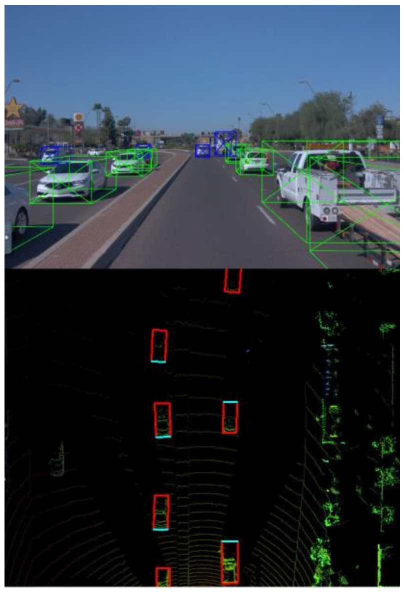
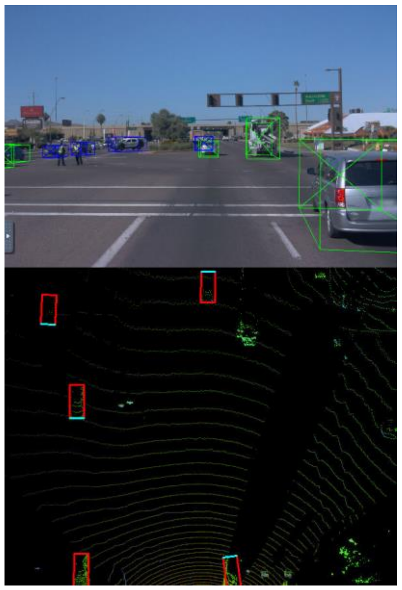
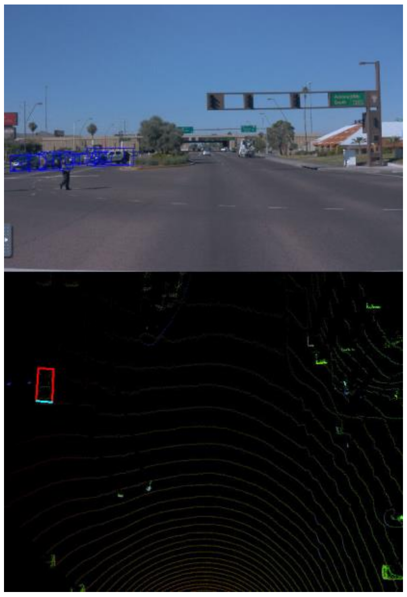
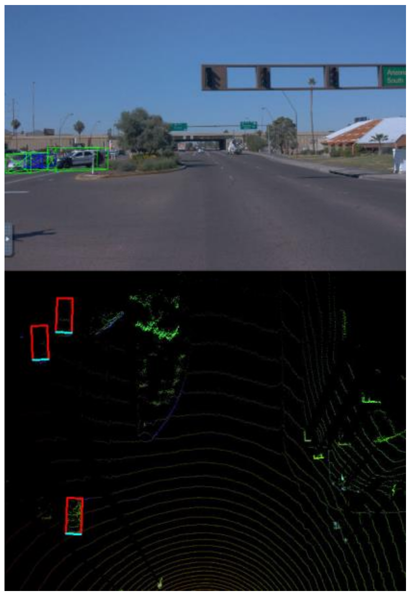
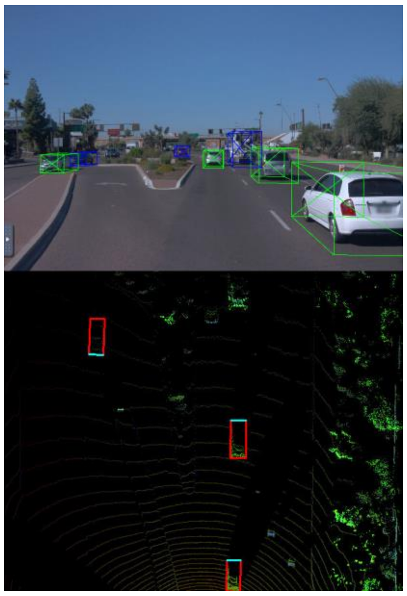
```
-------------------------
[[ 9.4790626e-01  3.6297336e+02  2.9504230e+02  9.0972990e-01
   1.6079109e+00  2.1099014e+01  5.2011196e+01  3.1189139e+00]
 [ 9.3985742e-01  2.6199463e+02  4.1286606e+02  8.4745914e-01
   1.5074542e+00  2.0475266e+01  4.7333508e+01 -1.4048363e-02]
 [ 8.8659835e-01  3.6055228e+02  2.1289929e+02  9.0944433e-01
   1.5172868e+00  2.0398354e+01  5.3292740e+01  3.1197257e+00]
 [ 8.8226134e-01  3.6710925e+02  4.0901282e+02  8.3207297e-01
   1.5089393e+00  2.1720228e+01  5.0476284e+01  3.0985608e+00]
 [ 7.9364973e-01  3.6469626e+02  5.4978656e+02  7.8988516e-01
   1.6350408e+00  2.2307636e+01  4.8100849e+01  3.0805218e+00]
 [ 7.3704052e-01  2.5970703e+02  1.9777594e+02  9.9575454e-01
   2.1285961e+00  2.3672895e+01  6.1965141e+01 -3.0225177e+00]
 [ 6.8496990e-01  3.5519940e+02  1.2160492e+02  8.4734130e-01
   1.5579846e+00  2.0121210e+01  4.7682808e+01  2.9977283e+00]]
```

As shown above, we detect 7 vehicles, and this can be illustrated from the red rectangles below:


labels (green) vs. detected objects (red)
(x=221, y=367) ~ R:0 G:0 B:0

Extract 3D bounding boxes from model response (ID_S3_EX2)

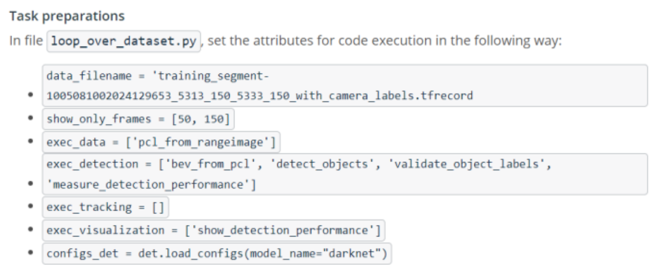
Below are several instances from Sequence #3 showing the BEV and 3D detection projected on the image:

Section 4 : Performance Evaluation for Object Detection

Compute intersection-over-union , false-negatives and false-positives, precision and recall (ID_S4_EX1,2,3)

Using the following task preperations:

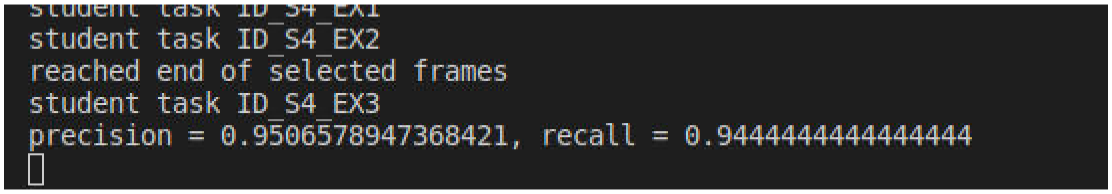


Task preparations

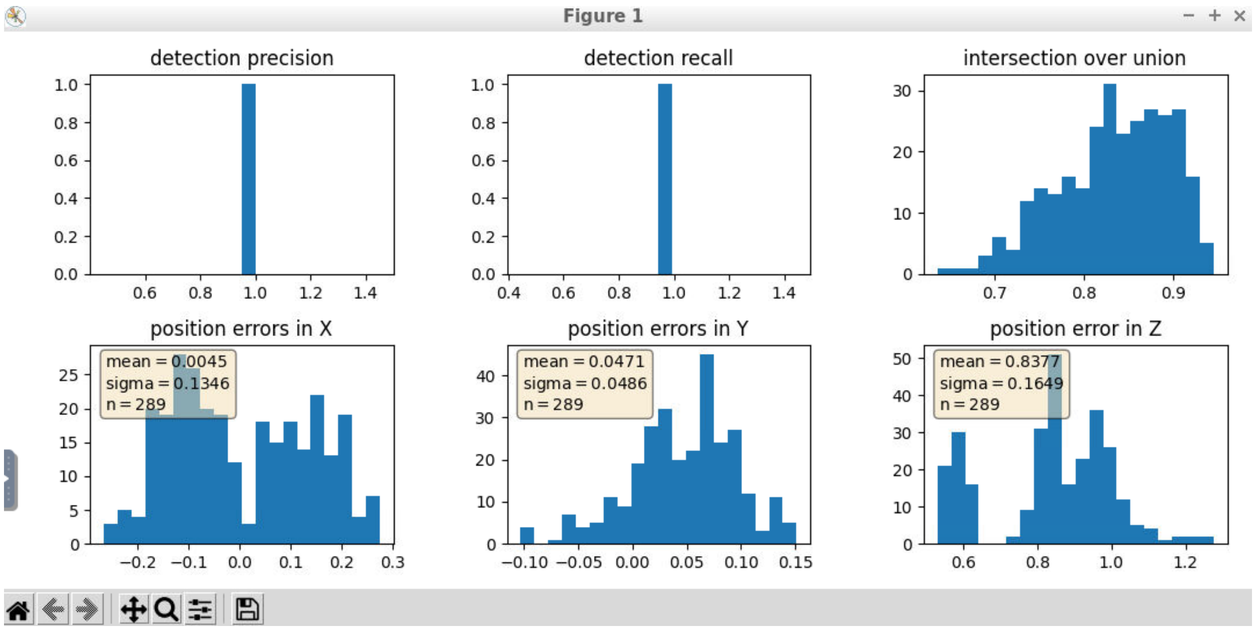
In file `loop_over_dataset.py`, set the attributes for code execution in the following way:

- `data_filename = 'training_segment-1005081002024129653_5313_150_5333_150_with_camera_labels.tfrecord`
- `show_only_frames = [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")`

I have obtained the following results:



```
student task ID_S4_EX1
student task ID_S4_EX2
reached end of selected frames
student task ID_S4_EX3
precision = 0.9506578947368421, recall = 0.9444444444444444
```



Extra test to make sure everything is done correctly:

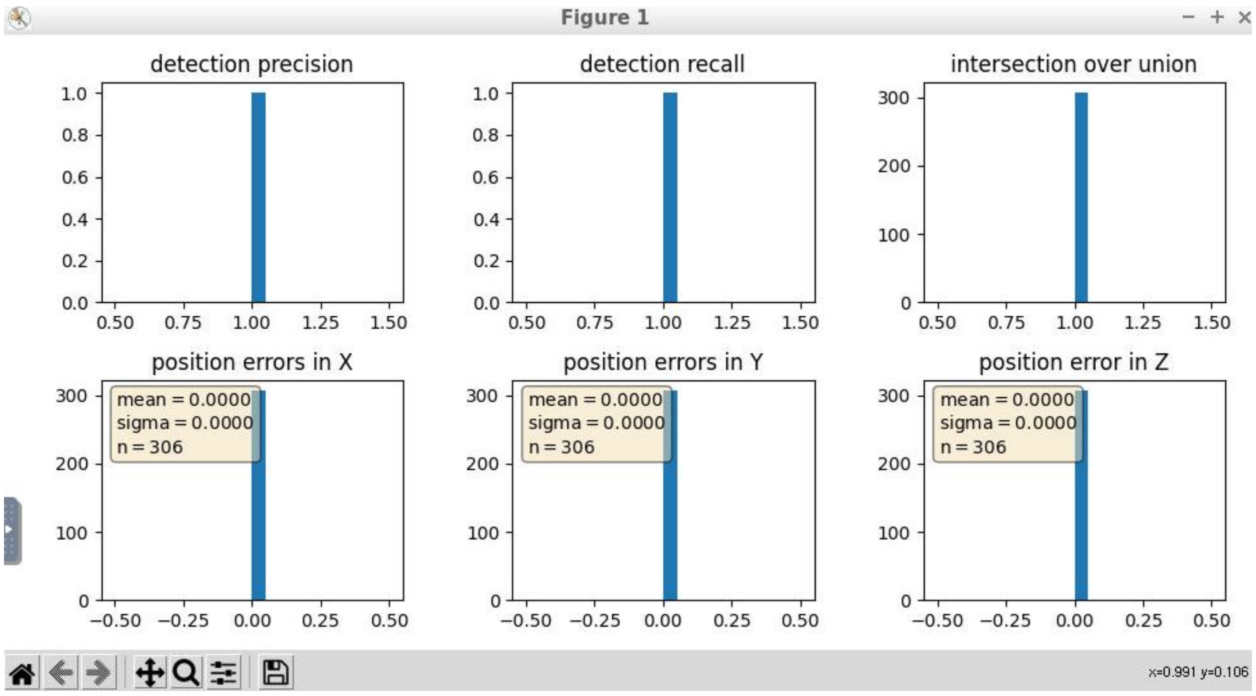


Figure 1
detection precision
detection recall
intersection over union
position errors in X
position errors in Y
position error in Z
mean = 0.0000
sigma = 0.0000
n = 306
x=0.991 y=0.106