Change in loop_over_dataset.py file

In the results folder I have created darknet and fpn_resnet folder. It helps to run the required model just by changing the model name.

```
## Prepare Waymo Open Dataset file for loading
model_name= 'darknet'  #'fpn_resnet'
data_fullpath = os.path.join(os.path.dirname(os.path.realpath(__file__)), 'dataset', data_filename) #
results_fullpath = os.path.join(os.path.dirname(os.path.realpath(__file__)), 'results', model_name)#
datafile = WaymoDataFileReader(data_fullpath)
datafile_iter = iter(datafile)  # initialize dataset iterator
```

Rather than changing it from in configs_det.

```
## Initialize object detection
configs_det = det.load_configs(model_name) # options are 'darknet', 'fpn_resnet'
model_det = det.create_model(configs_det)
```

To run the code in loop_over_dataset.py file, placed the attributes following project instructions.

Section 1: Compute Lidar Point-Cloud from Range Image

Visualize range image channels (ID_S1_EX1)

To implement ID_S1_EX1 wrote code within the function show_range_image located in the file student/objdet_pcl.py.

In step-1 extracted the lidar data and range image for the roof-mounted lidar.

In step-2 extracted the range and the intensity channel from the range image.

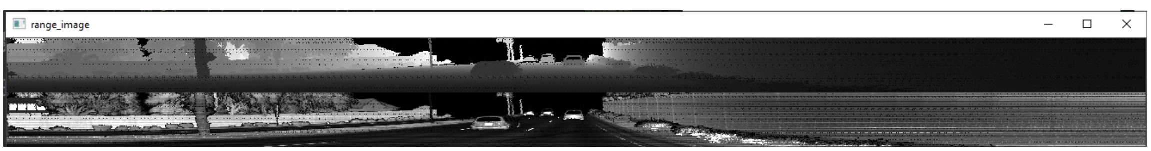
In step-4 to see the full range of values mapped the range channel onto an 8-bit scale.

In step-5 mitigated the influence of outliers using normalization with the difference between the 1- and 99-percentile. Mapped the intensity channel onto an 8-bit scale.

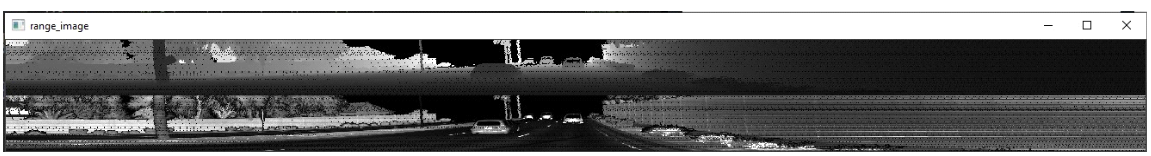
In step-6 using np.vstack stacked the range and intensity image vertically then converted it onto an 8-bit integer.

Cropped range image to +/- 90 deg. left and right of the forward-facing x-axis.

Output of this step for frame=50 is



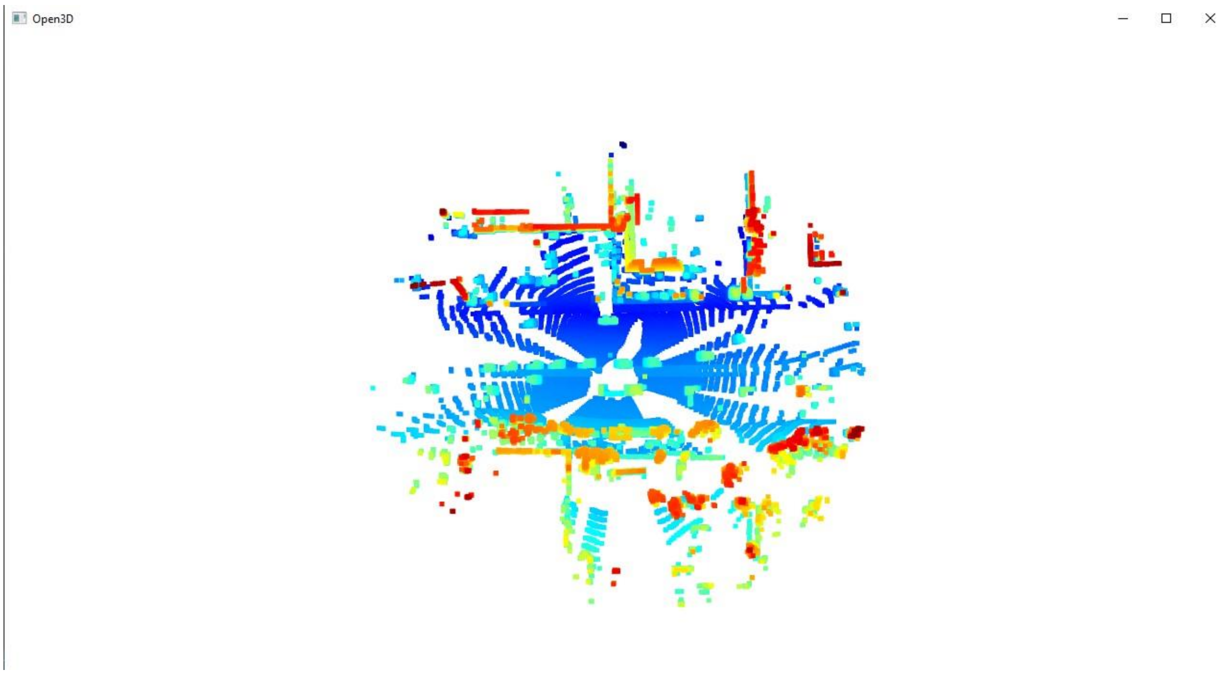
Output of this step for frame=51 is



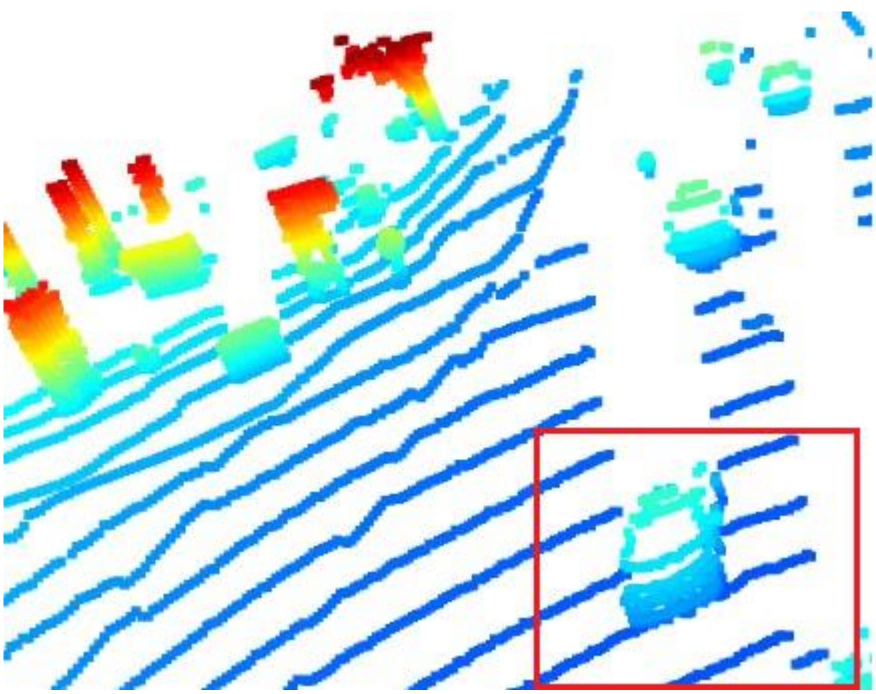
Visualize lidar point-cloud (ID_S1_EX2)

To implement ID_S1_EX2 wrote code within the function show_pcl located in the file student/objdet_pcl.py.

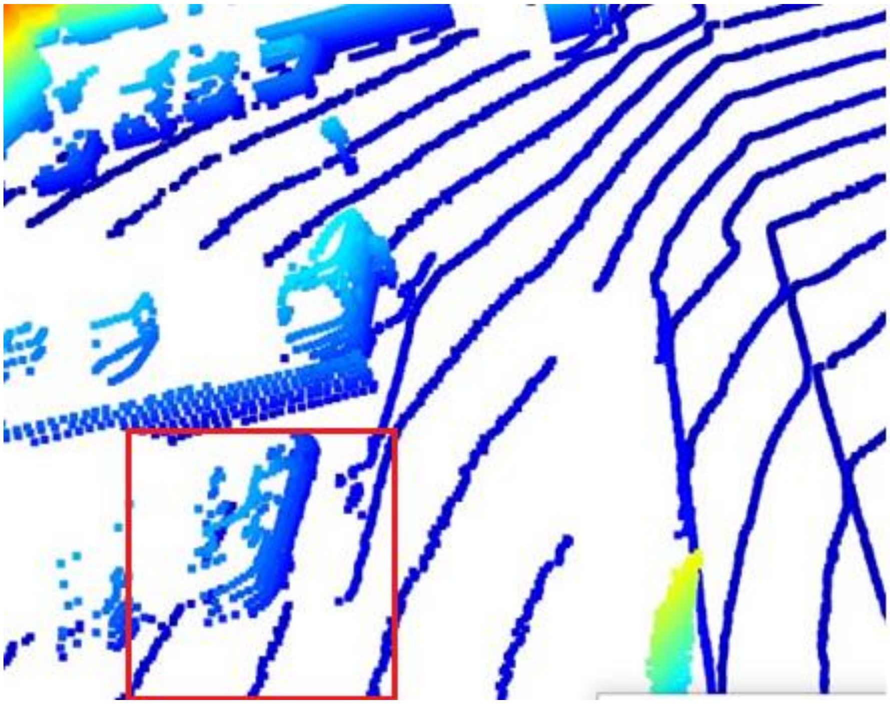
Visualized the point-cloud using the open3d module



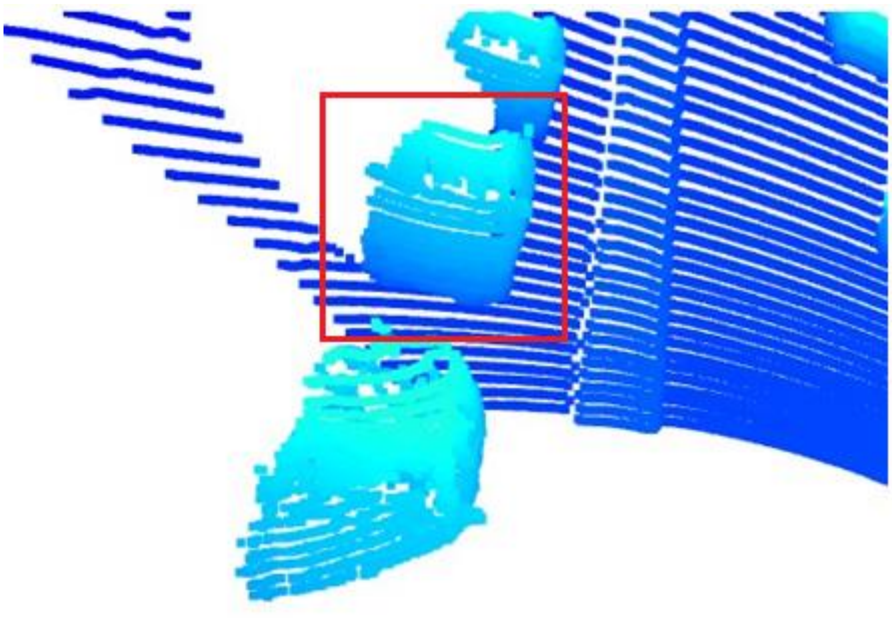
10 examples of vehicles with varying degrees of visibility in the point-cloud is given below.



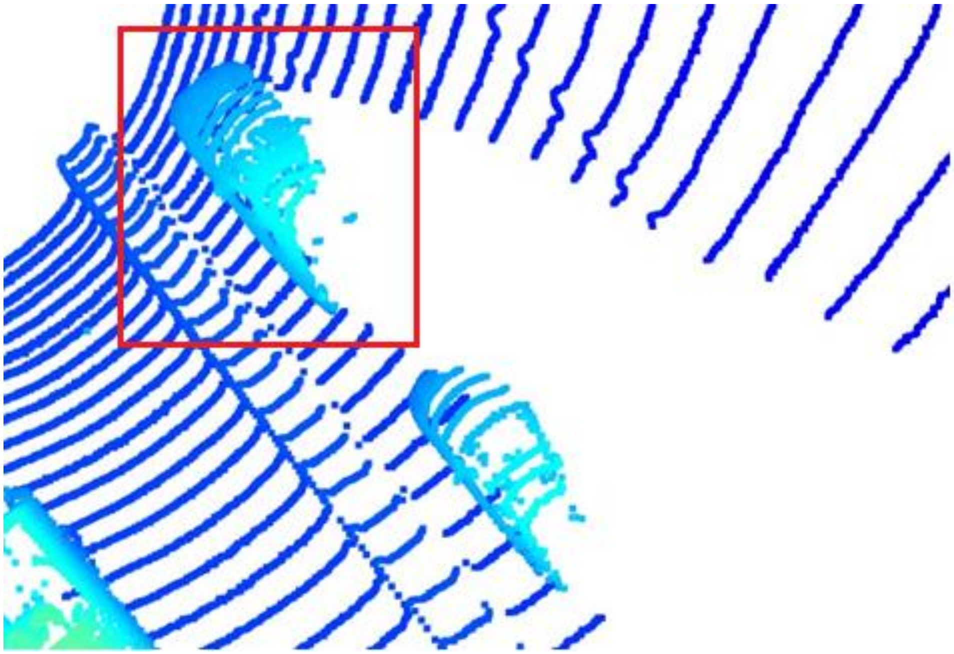
Bonnet and front bumper of vehicle are visible.



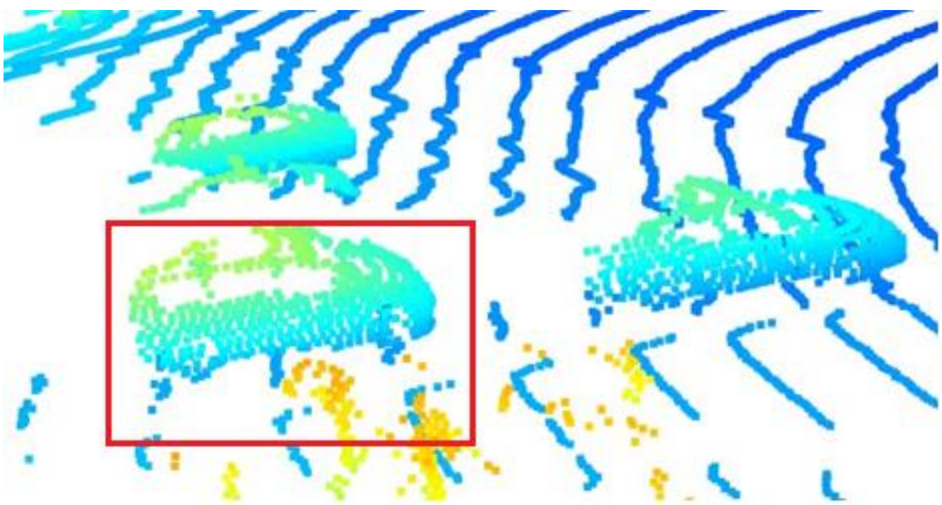
The vehicle structure, tyres and doors are visible.



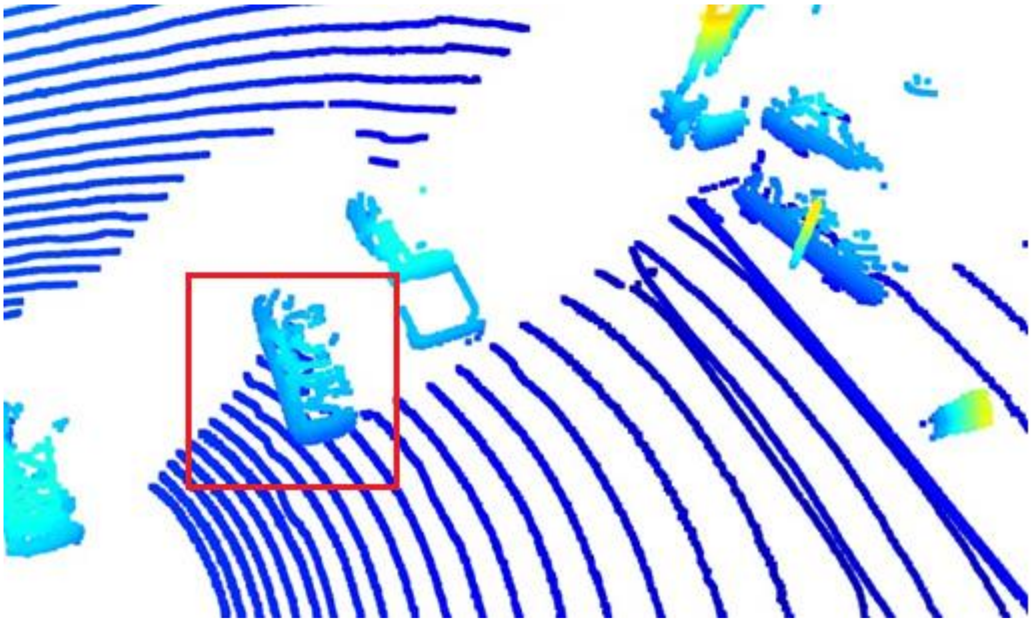
Bonnet, mirror, roof and front bumper are visible.



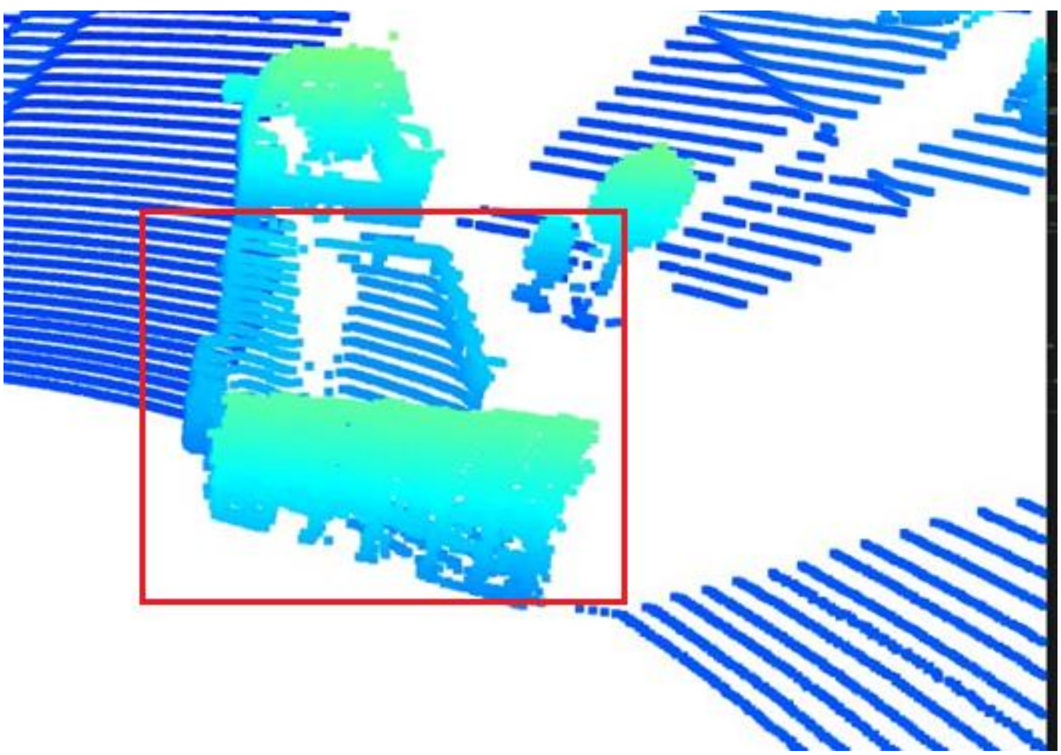
Bonnet and roof are visible.



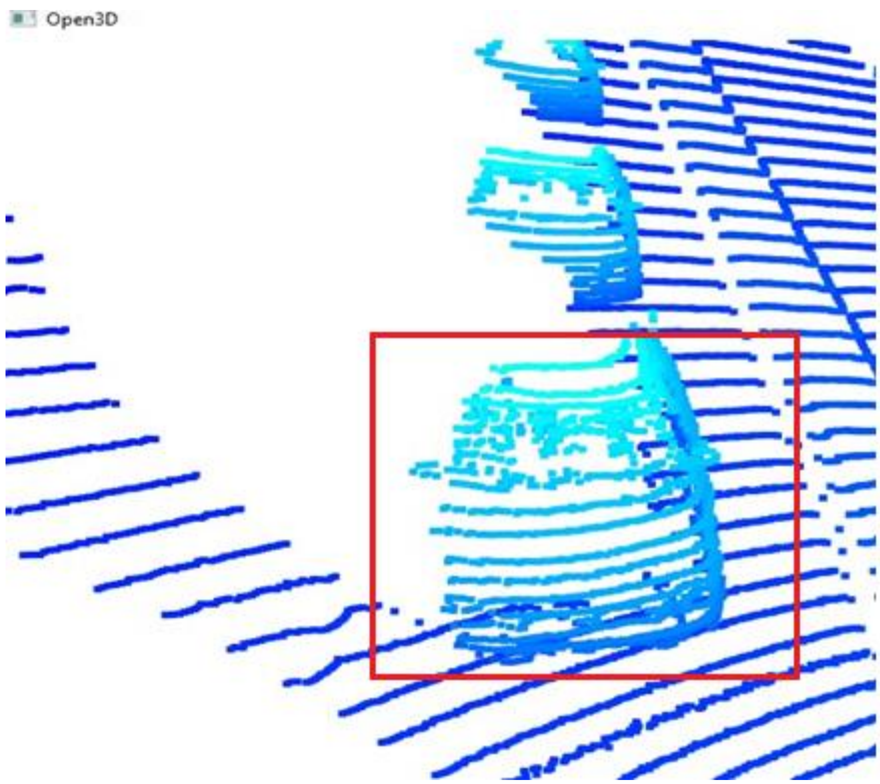
Bonnet and door are visible



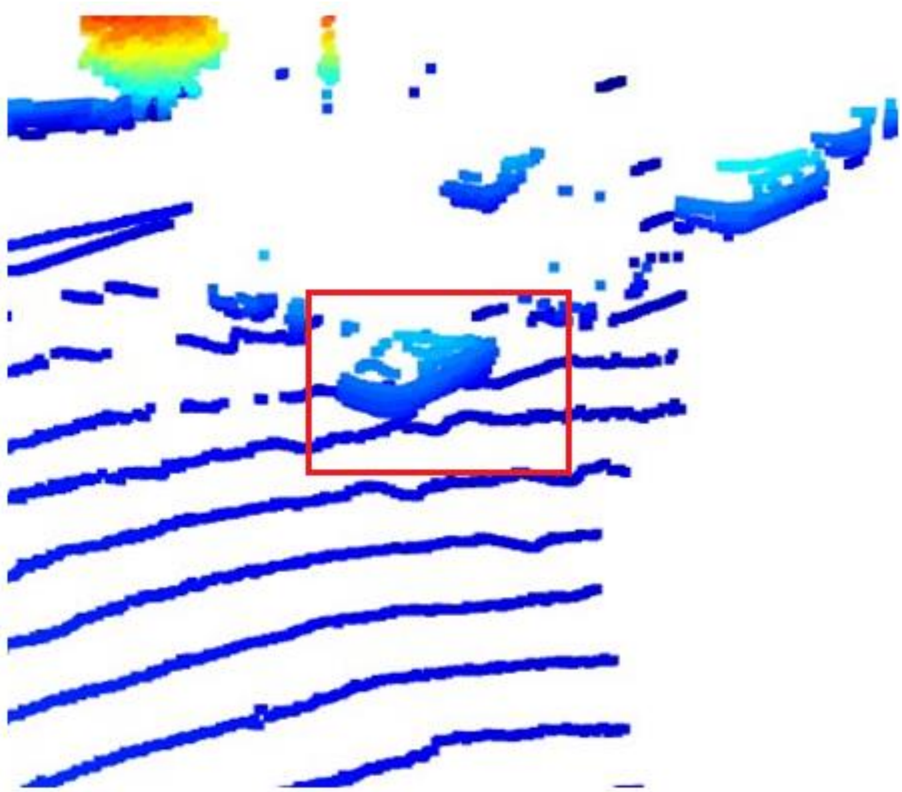
Rear bumper is visible.



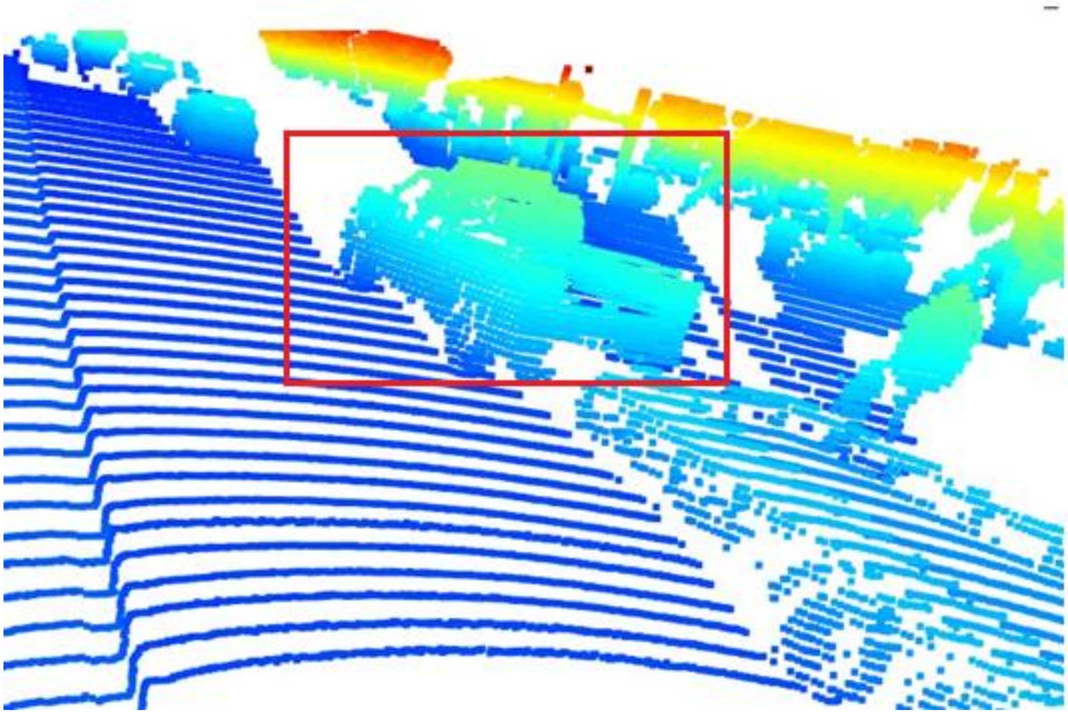
Rear bumper and tyre are visible.



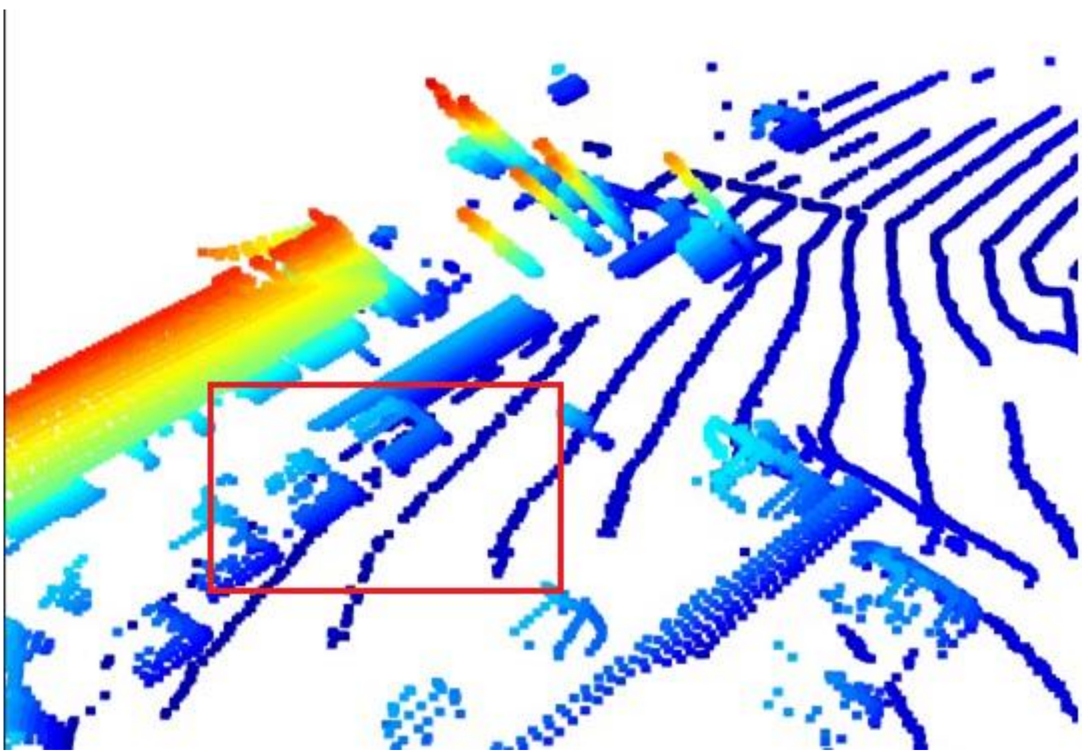
Bonnet, mirrors, windscreen, front bumper are visible.



Rear bumper and door are visible.



Wing/fender, tyres, mirror and roof are visible.



Rear bumper, boot/trunk and tyres are visible.

Section 2: Create Birds-Eye View from Lidar PCL

Convert sensor coordinates to BEV-map coordinates (ID_S2_EX1)

To implement ID_S2_EX1 wrote code within the function bev_from_pcl located in the file student/objdet_pcl.py

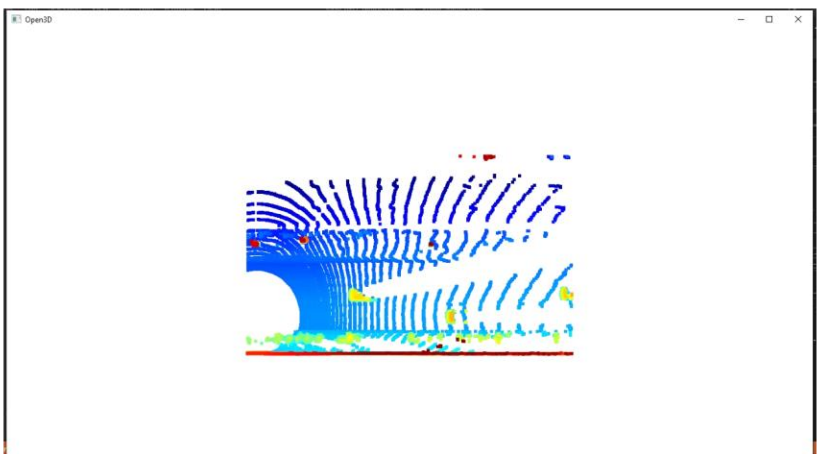
In step-1 computed discretization by dividing x-range by the bev-image height.

In step-2 created a copy of the lidar pcl and transform all matrix x-coordinates into bev-image coordinates.

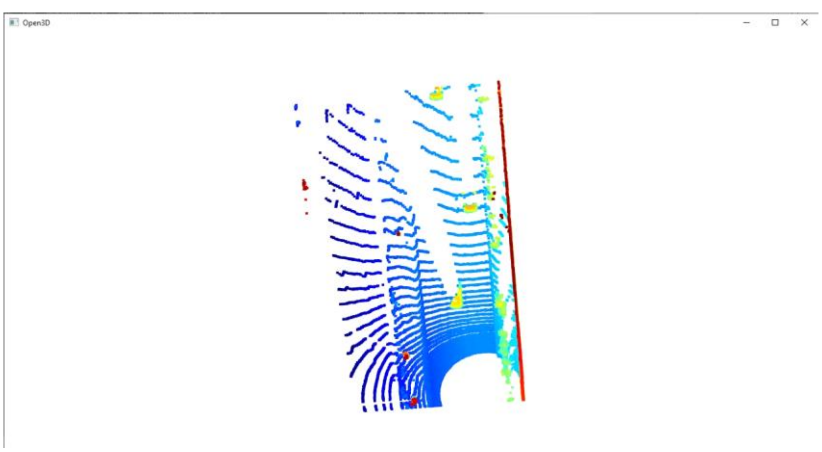
In step-3 performed the same operation as in step 2 for the y-coordinates but make sure that no negative bev-coordinates occur.

In step-4 visualized the point-cloud using the function show pcl from a previous task.

Output of this step for frame = 0 is



Rotated:



Compute intensity layer of the BEV map (ID_S2_EX2)

To implement ID_S2_EX2 wrote code within the function bev_from_pcl located in the file student/objdet_pcl.py.

In step-1 created a numpy array filled with zeros which has the same dimensions as the BEV map

In step-2 re-arranged elements in lidar_pcl_cpy by sorting first by x, then y, then -z using the numpy.lexsort.

In step-3 extracted all points with identical x and y such that only the top-most z-coordinate is kept using the numpy.unique.

In step-4 assigned the intensity value of each unique entry in lidar_top_pcl to the intensity map, intensity is scaled in such a way that vehicles are clearly visible. The influence of outliers is mitigated by normalizing intensity on the difference between the max. and min. value within the point cloud.

In step-5 to separate vehicles from the background temporarily visualized the intensity map using OpenCV.

Output of this step for frame=0 is

Compute height layer of the BEV map (ID_S2_EX3)

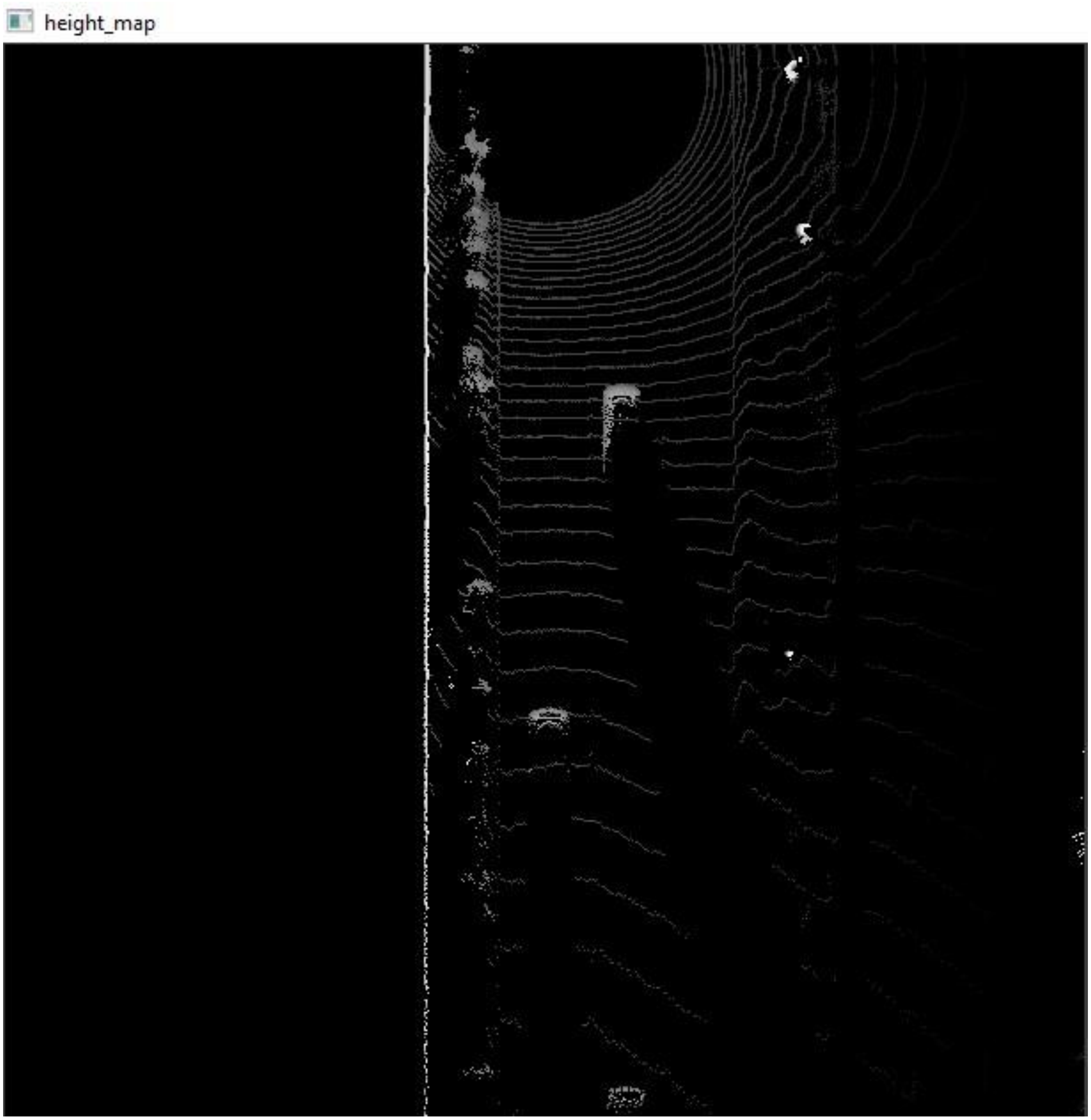
To implement ID_S2_EX3 wrote code within the function bev_from_pcl located in the file student/objdet_pcl.py.

In step-1 created a numpy array filled with zeros which has the same dimensions as the BEV map.

In step-2 assigned the height value of each unique entry in lidar_pcl_top to the height map.

In step-3 to separate vehicles from the background temporarily visualized the intensity map using OpenCV.

Output of this step for frame=0 is



Section 3: Model-based Object Detection in BEV Image

Added a second model from a GitHub repo (ID_S3_EX1)

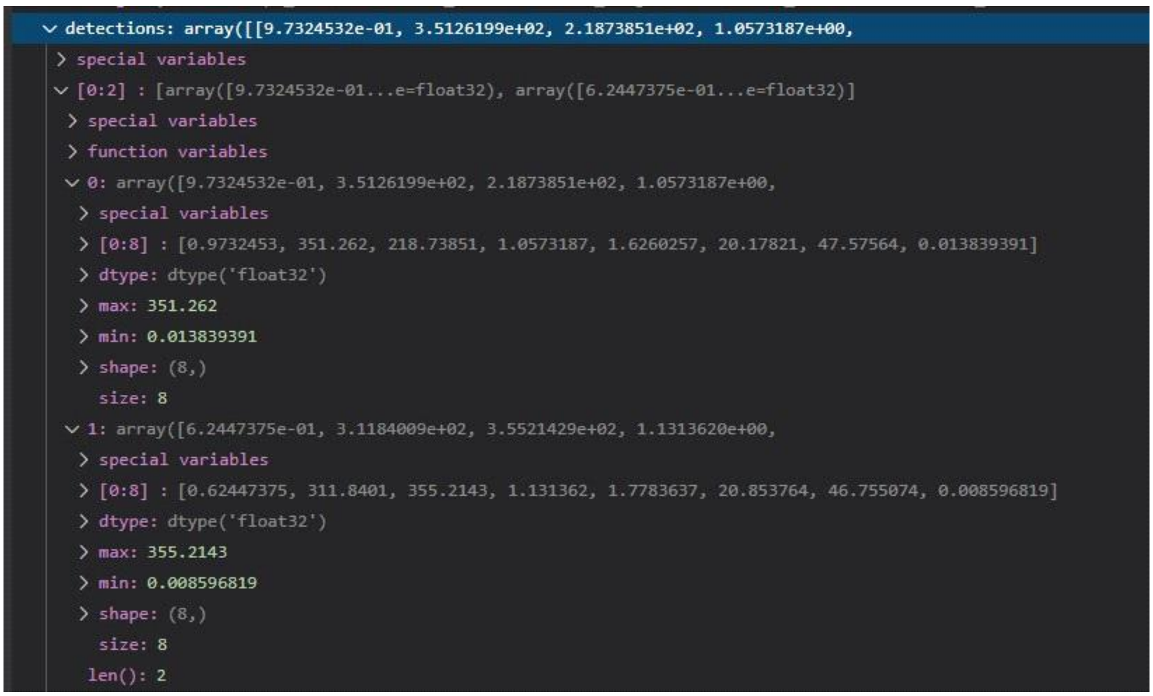
To implement ID_S3_EX1 wrote code within the function detect_objects, load_configs_model and create_model located in the file student/objdet_detect.py.

From SFA3D->test.py extract the relevant parameters from SFA3D->test.py->parse_test_configs() and added them to the the configs structure in load_configs_model.

Instantiate the model for fpn_resnet in create_model.

decoded output and performed post-processing for object detections.

Output of this step for frame = 50



Extract 3D bounding boxes from model response (ID_S3_EX2)

To implement ID_S3_EX2 wrote code within detect_objects located in the file student/objdet_detect.py.

In step-1 checked whether there are any detections.

In step-2 looped over all detections.

In step-3 performed the conversion using the limits for x, y and z set in the configs structure.

In step-4 appended the current object to the 'objects' array.

Output of this step for frame = 50 is

Output of this step for frame = 51

labels vs. detected objects

Section 4: Performance Evaluation for Object Detection

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

To implement ID_S4_EX1 wrote code within detect_objects located in the file student/objdet_eval.py.

In step-1 extracted the four corners of the current label bounding-box.

In step-2 loop over all detected objects.

In step-3 extracted the four corners of the current detection.

In step-4 computed the center distance between label and detection bounding-box in x, y, and z.

In step-5 computed the intersection over union (IOU) between label and detection bounding-box.

In step-6 checked the IOU value it exceeds min_iou threshold, store [iou,dist_x, dist_y, dist_z] in matches_lab_det and increases the TP count.

Output of this step for frame = 50

```
ious: [0.8234347776989478, 0.8883710045949664, 0.8752899290800235]
  > special variables
  > function variables
    0: 0.8234347776989478
    1: 0.8883710045949664
    2: 0.8752899290800235
    len(): 3
```

```
center_devs: [[tensor(0.1402), tensor(-0.0197), 1.0292643213596193], [tensor(-0.0835), tensor(0.0698), 0.8291298942401681], [tensor(0.08…
  > special variables
  > function variables
  > 0: [tensor(0.1402), tensor(-0.0197), 1.0292643213596193]
  > 1: [tensor(-0.0835), tensor(0.0698), 0.8291298942401681]
  > 2: [tensor(0.0837), tensor(0.0251), 0.8929607095304846]
    len(): 3
```

Compute false-negatives and false-positives (ID_S4_EX2)

To implement ID_S4_EX2 wrote code within detect_objects located in the file student/objdet_eval.py.

computed the total number of positives, number of false negatives, number of false positives presented in the scene.

Output of this step for frame=50

Compute precision and recall (ID_S4_EX3)

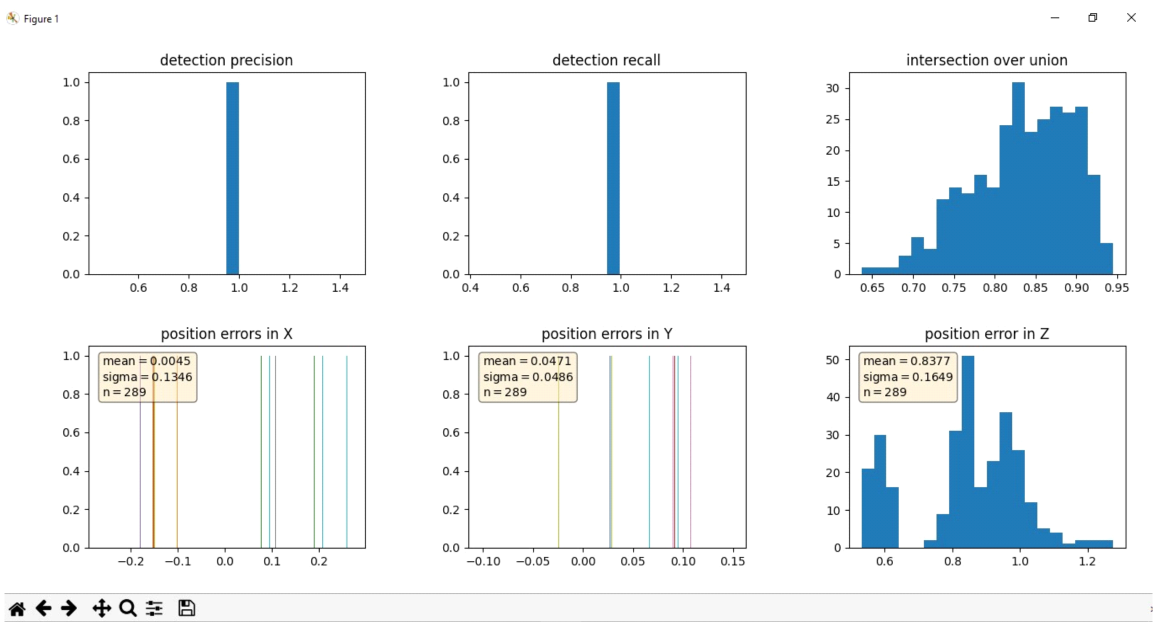
To implement ID_S4_EX3 wrote code within detect_objects located in the file student/objdet_eval.py.

In step-1 extracted the total number of positives, true positives, false negatives and false positives.

In step-2,3 computed precision and recall.

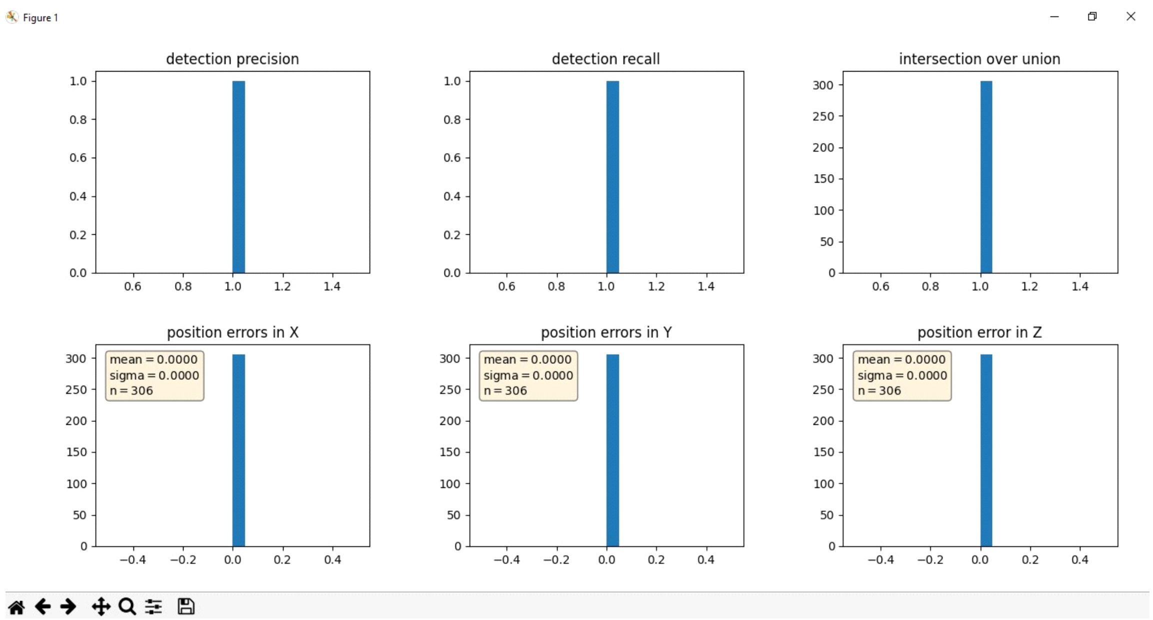
Output of this step for the frame = 50 to 150

```
student task ID_S4_EX3
precision = 0.9506578947368421, recall = 0.9444444444444444
```



configs_det.use_labels_as_objects set to True.

```
student task ID_S4_EX2
reached end of selected frames
student task ID_S4_EX3
precision = 1.0, recall = 1.0
```



I faced difficulty to implement EX-2 of step-1 and Ex-1,2 of step 2.