

## # SDCND : Sensor Fusion and Tracking

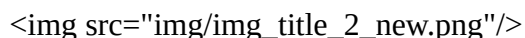
This is the project for the second course in the [Udacity Self-Driving Car Engineer Nanodegree Program](https://www.udacity.com/course/c-plus-plus-nanodegree--nd213) : Sensor Fusion and Tracking.

In this project, you'll fuse measurements from LiDAR and camera and track vehicles over time. You will be using real-world data from the Waymo Open Dataset, detect objects in 3D point clouds and apply an extended Kalman filter for sensor fusion and tracking.

The project consists of two major parts:

1. **Object detection**: In this part, a deep-learning approach is used to detect vehicles in LiDAR data based on a birds-eye view perspective of the 3D point-cloud. Also, a series of performance measures is used to evaluate the performance of the detection approach.
2. **Object tracking** : In this part, an extended Kalman filter is used to track vehicles over time, based on the lidar detections fused with camera detections. Data association and track management are implemented as well.

The following diagram contains an outline of the data flow and of the individual steps that make up the algorithm.



Also, the project code contains various tasks, which are detailed step-by-step in the code. More information on the algorithm and on the tasks can be found in the Udacity classroom.

### ## Project File Structure

```
project<br>├── dataset --> contains the Waymo Open Dataset sequences <br><br>├── misc<br>│   ├── evaluation.py --> plot functions for tracking visualization and RMSE calculation<br>│   ├── helpers.py --> misc. helper functions, e.g. for loading / saving binary files<br>│   ├── objdet_tools.py --> object detection functions without student tasks<br>│   └── params.py --> parameter file for the tracking part<br><br>├── results --> binary files with pre-computed intermediate results<br><br>├── student <br>│   ├── association.py --> data association logic for assigning measurements to tracks incl. student tasks <br>│   ├── filter.py --> extended Kalman filter implementation incl. student tasks <br>│   └── measurements.py --> sensor and measurement classes for camera and lidar incl. student tasks <br><br>├── objdet_detect.py --> model-based object detection incl. student tasks <br>├── objdet_eval.py --> performance assessment for object detection incl. student tasks <br>├── objdet_pcl.py --> point-cloud functions, e.g. for birds-eye view incl. student tasks <br>├── trackmanagement.py --> track and track management classes incl. student tasks <br><br>└── tools --> external tools<br>
```

```

├── objdet_models --> models for object detection<br>
│   └── <br>
│       ├── darknet<br>
│       │   ├── config<br>
│       │   ├── models --> darknet / yolo model class and tools<br>
│       │   ├── pretrained --> copy pre-trained model file here<br>
│       │   └── complex_yolov4_mse_loss.pth<br>
│       ├── utils --> various helper functions<br>
│       └── <br>
│           ├── resnet<br>
│           │   ├── models --> fpn_resnet model class and tools<br>
│           │   ├── pretrained --> copy pre-trained model file here <br>
│           │   └── fpn_resnet_18_epoch_300.pth <br>
│           └── utils --> various helper functions<br>
│               └── <br>
│                   └── waymo_reader --> functions for light-weight loading of Waymo sequences<br>
└── <br>
    ├── basic_loop.py<br>
    └── loop_over_dataset.py<br>

```

## ## Installation Instructions for Running Locally

### ### Cloning the Project

In order to create a local copy of the project, please click on "Code" and then "Download ZIP". Alternatively, you may of-course use GitHub Desktop or Git Bash for this purpose.

### ### Python

The project has been written using Python 3.7. Please make sure that your local installation is equal or above this version.

### ### Package Requirements

All dependencies required for the project have been listed in the file `requirements.txt`. You may either install them one-by-one using pip or you can use the following command to install them all at once:

```
`pip3 install -r requirements.txt`
```

### ### Waymo Open Dataset Reader

The Waymo Open Dataset Reader is a very convenient toolbox that allows you to access sequences from the Waymo Open Dataset without the need of installing all of the heavy-weight dependencies that come along with the official toolbox. The installation instructions can be found in `tools/waymo\_reader/README.md`.

### ### Waymo Open Dataset Files

This project makes use of three different sequences to illustrate the concepts of object detection and tracking. These are:

- Sequence 1 : `training\_segment-1005081002024129653\_5313\_150\_5333\_150\_with\_camera\_labels.tfrecord`
- Sequence 2 : `training\_segment-10072231702153043603\_5725\_000\_5745\_000\_with\_camera\_labels.tfrecord`
- Sequence 3 : `training\_segment-10963653239323173269\_1924\_000\_1944\_000\_with\_camera\_labels.tfrecord`

To download these files, you will have to register with Waymo Open Dataset first: [Open Dataset – Waymo](https://waymo.com/open/terms), if you have not already, making sure to note "Udacity" as your institution.

Once you have done so, please [click here](https://console.cloud.google.com/storage/browser/waymo\_open\_dataset\_v\_1\_2\_0\_individual\_files) to access the Google Cloud Container that holds all the sequences. Once you have been cleared for access by Waymo (which might take up to 48 hours), you can download the individual sequences.

The sequences listed above can be found in the folder "training". Please download them and put the `tfrecord`-files into the `dataset` folder of this project.

### ### Pre-Trained Models

The object detection methods used in this project use pre-trained models which have been provided by the original authors. They can be downloaded [here](https://drive.google.com/file/d/1Pqx7sShlqKSGmvshTYbNDcUEYyZwfn3A/view?usp=sharing) (darknet) and [here](https://drive.google.com/file/d/1RcEfUIF1pzDZco8PJkZ10OL-wLL2usEj/view?usp=sharing) (fpn\_resnet). Once downloaded, please copy the model files into the paths `tools/objdet\_models/darknet/pretrained` and `tools/objdet\_models/fpn\_resnet/pretrained` respectively.

### ### Using Pre-Computed Results

In the main file `loop\_over\_dataset.py`, you can choose which steps of the algorithm should be executed. If you want to call a specific function, you simply need to add the corresponding string literal to one of the following lists:

- `exec\_data` : controls the execution of steps related to sensor data.
  - `pcl\_from\_rangeimage` transforms the Waymo Open Data range image into a 3D point-cloud
  - `load\_image` returns the image of the front camera
- `exec\_detection` : controls which steps of model-based 3D object detection are performed
  - `bev\_from\_pcl` transforms the point-cloud into a fixed-size birds-eye view perspective
  - `detect\_objects` executes the actual detection and returns a set of objects (only vehicles)
  - `validate\_object\_labels` decides which ground-truth labels should be considered (e.g. based on difficulty or visibility)
  - `measure\_detection\_performance` contains methods to evaluate detection performance for a single frame

In case you do not include a specific step into the list, pre-computed binary files will be loaded instead. This enables you to run the algorithm and look at the results even without having implemented anything yet. The pre-computed results for the mid-term project need to be loaded using [this](https://drive.google.com/drive/folders/1-s46dKSrtx8rrNwnObGbly2nO3i4D7r7?usp=sharing) link. Please use the folder `darknet` first. Unzip the file within and put its content into the folder `results`.

- `exec\_tracking` : controls the execution of the object tracking algorithm
- `exec\_visualization` : controls the visualization of results

- ``show_range_image`` displays two LiDAR range image channels (range and intensity)
- ``show_labels_in_image`` projects ground-truth boxes into the front camera image
- ``show_objects_and_labels_in_bev`` projects detected objects and label boxes into the birds-eye view
- ``show_objects_in_bev_labels_in_camera`` displays a stacked view with labels inside the camera image on top and the birds-eye view with detected objects on the bottom
- ``show_tracks`` displays the tracking results
- ``show_detection_performance`` displays the performance evaluation based on all detected
- ``make_tracking_movie`` renders an output movie of the object tracking results

Even without solving any of the tasks, the project code can be executed.

The final project uses pre-computed lidar detections in order for all students to have the same input data. If you use the workspace, the data is prepared there already. Otherwise, [download the pre-computed lidar detections]([https://drive.google.com/drive/folders/1IkqFGYTF6Fh\\_d8J3UjQOSNJ2V42UDZpO?usp=sharing](https://drive.google.com/drive/folders/1IkqFGYTF6Fh_d8J3UjQOSNJ2V42UDZpO?usp=sharing)) (~1 GB), unzip them and put them in the folder ``results``.

### ## External Dependencies

Parts of this project are based on the following repositories:

- [Simple Waymo Open Dataset Reader](<https://github.com/gdlg/simple-waymo-open-dataset-reader>)
- [Super Fast and Accurate 3D Object Detection based on 3D LiDAR Point Clouds](<https://github.com/maudzung/SFA3D>)
- [Complex-YOLO: Real-time 3D Object Detection on Point Clouds](<https://github.com/maudzung/Complex-YOLOv4-Pytorch>)

### ## License

[License](LICENSE.md)

This is a template submission for the midterm course : 3D Object Detection (Midterm).

## 3D Object detection

We have used the Waymo dataset real-world data and used 3d point cloud for lidar based object detection.

- Configuring the ranges channel to 8 bit and view the range /intensity image (ID\_S1\_EX1)
- Use the Open3D library to display the lidar point cloud on a 3d viewer and identifying 6 images from point cloud.(ID\_S1\_EX2)
- Create Birds Eye View perspective (BEV) of the point cloud,assign lidar intensity values to BEV,normalize the heightmap of each BEV (ID\_S2\_EX1,ID\_S2\_EX2,ID\_S2\_EX3)
- In addition to YOLO, use the repository [<https://github.com/maudzung/SFA3D.git>]and add parameters ,instantiate fpn resnet model(ID\_S3\_EX1)
- Convert BEV coordinates into pixel coordinates and convert model output to bounding box format (ID\_S3\_EX2)
- Compute intersection over union, assign detected objects to label if IOU exceeds threshold (ID\_S4\_EX1)
- Compute false positives and false negatives, precision and recall(ID\_S4\_EX2,ID\_S4\_EX3)

The project can be run by running

python loop\_over\_dataset.py

All training/inference is done on udacity workspace.

### Step-1: Compute Lidar point cloud from Range Image

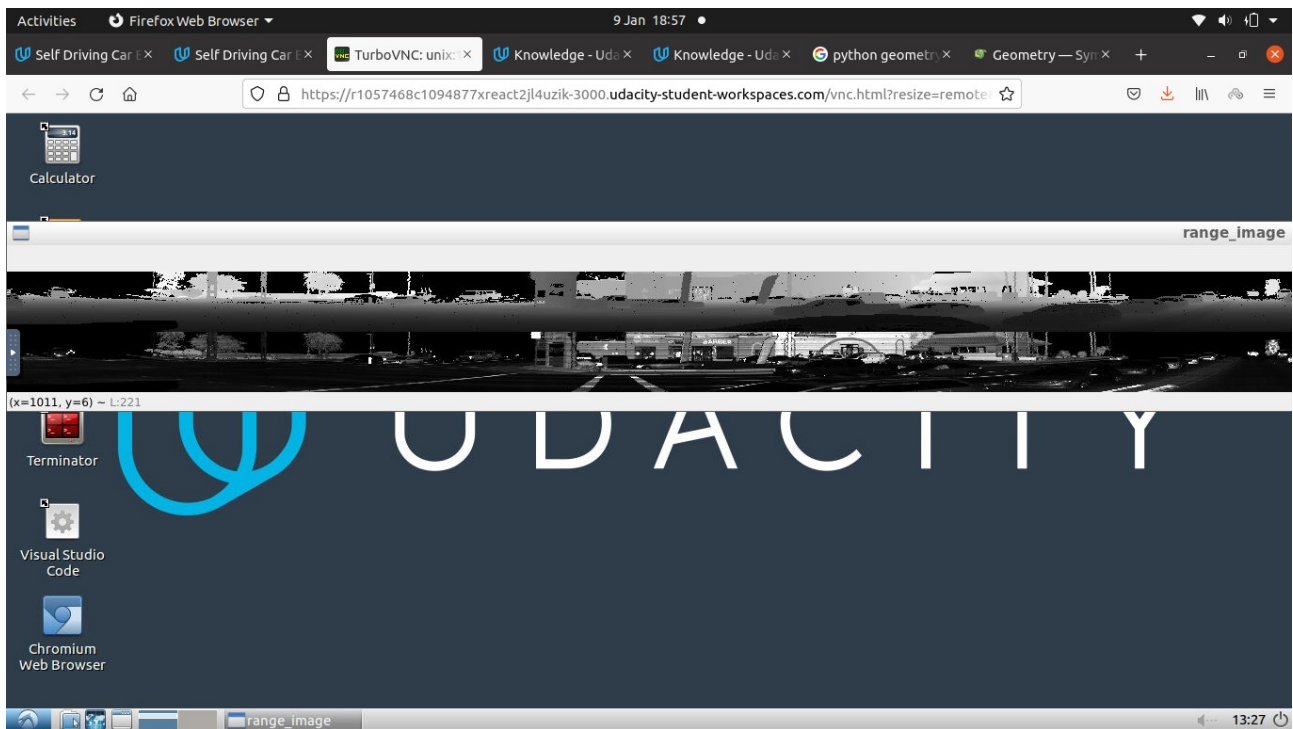
In this we are first previewing the range image and convert range and intensity channels to 8 bit format. After that, we use the openCV library to stack the range and intensity channel vertically to visualize the image.

- Convert "range" channel to 8 bit
- Convert "intensity" channel to 8 bit
- Crop range image to +/- 90 degrees left and right of forward facing x axis
- Stack up range and intensity channels vertically in openCV

The changes are made in 'loop\_over\_dataset.py'

```
49  ## Set parameters and perform initializations
50
51  ## Select Waymo Open Dataset file and frame numbers
52  # data_filename = 'training_segment-1005081002024129653_5313_150_5333_150_with_camera_labels.tfrecord' # Sequence 1
53  # data_filename = 'training_segment-10072231702153043603_5725_000_5745_000_with_camera_labels.tfrecord' # Sequence 2
17- 54  data_filename = 'training_segment-10963653239323173269_1924_000_1944_000_with_camera_labels.tfrecord' # Sequence 3
55  show_only_frames = [1,10] # show only frames in interval for debugging
17- 56
57  ## Prepare Waymo Open Dataset file for loading
58  data_fullpath = os.path.join(os.path.dirname(os.path.realpath(__file__)), 'dataset', data_filename) # adjustable path
   called from another working directory
59  model = "darknet"
17- 60  sequence = "3"
4-17- 61  results_fullpath = os.path.join(os.path.dirname(os.path.realpath(__file__)), 'results/' + model + '/results_sequence_'
   model)
```

```
misc      80 np.random.seed(10) # make random values predictable
          81
results   82 ## Selective execution and visualization
student   83 exe_data=[] #option 'pcl_from_rangeimage','load_image'
tools      84 exec_detection = [] # options are 'bev_from_pcl', 'detect_objects', 'validate_object_labels', 'measure_detection_performance'; options not in
          the list will be loaded from file
.student_bashrc 85 exec_tracking = [] # options are 'perform tracking'
          86
DepthCamera_2022-01-10-17: 86 exec_visualization = ['show_range_image'] # options are 'show_range_image', 'show_bev', 'show_pcl', 'show_labels_in_image',
          'show_objects_and_labels_in_bev', 'show_objects_in_bev_labels_in_camera', 'show_tracks', 'show_detection_performance', 'make_tracking_movie'
DepthCapture_2022-01-10-17: 87 exec_list = make_exec_list(exec_detection, exec_tracking, exec_visualization)
          88
LICENSE.md 88 vis_pause_time = 0 # set pause time between frames in ms (0 = stop between frames until key is pressed)
          89
README.md  90
          91
          92
          93
          94
          95
          96
          97
          98
          99
         100
         101
         102
         103
         104
         105
         106
         107
         108
         109
         110
         111
         112
         113
         114
         115
         116
         117
         118
         119
         120
         121
         122
         123
         124
         125
         126
         127
         128
         129
         130
         131
         132
         133
         134
         135
         136
         137
         138
         139
         140
         141
         142
         143
         144
         145
         146
         147
         148
         149
         150
         151
         152
         153
         154
         155
         156
         157
         158
         159
         160
         161
         162
         163
         164
         165
         166
         167
         168
         169
         170
         171
         172
         173
         174
         175
         176
         177
         178
         179
         180
         181
         182
         183
         184
         185
         186
         187
         188
         189
         190
         191
         192
         193
         194
         195
         196
         197
         198
         199
        200
        201
        202
        203
        204
        205
        206
        207
        208
        209
        210
        211
        212
        213
        214
        215
        216
        217
        218
        219
        220
        221
        222
        223
        224
        225
        226
        227
        228
        229
        230
        231
        232
        233
        234
        235
        236
        237
        238
        239
        240
        241
        242
        243
        244
        245
        246
        247
        248
        249
        250
        251
        252
        253
        254
        255
        256
        257
        258
        259
        260
        261
        262
        263
        264
        265
        266
        267
        268
        269
        270
        271
        272
        273
        274
        275
        276
        277
        278
        279
        280
        281
        282
        283
        284
        285
        286
        287
        288
        289
        290
        291
        292
        293
        294
        295
        296
        297
        298
        299
        300
        301
        302
        303
        304
        305
        306
        307
        308
        309
        310
        311
        312
        313
        314
        315
        316
        317
        318
        319
        320
        321
        322
        323
        324
        325
        326
        327
        328
        329
        330
        331
        332
        333
        334
        335
        336
        337
        338
        339
        340
        341
        342
        343
        344
        345
        346
        347
        348
        349
        350
        351
        352
        353
        354
        355
        356
        357
        358
        359
        360
        361
        362
        363
        364
        365
        366
        367
        368
        369
        370
        371
        372
        373
        374
        375
        376
        377
        378
        379
        380
        381
        382
        383
        384
        385
        386
        387
        388
        389
        390
        391
        392
        393
        394
        395
        396
        397
        398
        399
        400
        401
        402
        403
        404
        405
        406
        407
        408
        409
        410
        411
        412
        413
        414
        415
        416
        417
        418
        419
        420
        421
        422
        423
        424
        425
        426
        427
        428
        429
        430
        431
        432
        433
        434
        435
        436
        437
        438
        439
        440
        441
        442
        443
        444
        445
        446
        447
        448
        449
        450
        451
        452
        453
        454
        455
        456
        457
        458
        459
        460
        461
        462
        463
        464
        465
        466
        467
        468
        469
        470
        471
        472
        473
        474
        475
        476
        477
        478
        479
        480
        481
        482
        483
        484
        485
        486
        487
        488
        489
        490
        491
        492
        493
        494
        495
        496
        497
        498
        499
        500
        501
        502
        503
        504
        505
        506
        507
        508
        509
        510
        511
        512
        513
        514
        515
        516
        517
        518
        519
        520
        521
        522
        523
        524
        525
        526
        527
        528
        529
        530
        531
        532
        533
        534
        535
        536
        537
        538
        539
        540
        541
        542
        543
        544
        545
        546
        547
        548
        549
        550
        551
        552
        553
        554
        555
        556
        557
        558
        559
        560
        561
        562
        563
        564
        565
        566
        567
        568
        569
        570
        571
        572
        573
        574
        575
        576
        577
        578
        579
        580
        581
        582
        583
        584
        585
        586
        587
        588
        589
        590
        591
        592
        593
        594
        595
        596
        597
        598
        599
        600
        601
        602
        603
        604
        605
        606
        607
        608
        609
        610
        611
        612
        613
        614
        615
        616
        617
        618
        619
        620
        621
        622
        623
        624
        625
        626
        627
        628
        629
        630
        631
        632
        633
        634
        635
        636
        637
        638
        639
        640
        641
        642
        643
        644
        645
        646
        647
        648
        649
        650
        651
        652
        653
        654
        655
        656
        657
        658
        659
        660
        661
        662
        663
        664
        665
        666
        667
        668
        669
        670
        671
        672
        673
        674
        675
        676
        677
        678
        679
        680
        681
        682
        683
        684
        685
        686
        687
        688
        689
        690
        691
        692
        693
        694
        695
        696
        697
        698
        699
        700
        701
        702
        703
        704
        705
        706
        707
        708
        709
        710
        711
        712
        713
        714
        715
        716
        717
        718
        719
        720
        721
        722
        723
        724
        725
        726
        727
        728
        729
        730
        731
        732
        733
        734
        735
        736
        737
        738
        739
        740
        741
        742
        743
        744
        745
        746
        747
        748
        749
        750
        751
        752
        753
        754
        755
        756
        757
        758
        759
        760
        761
        762
        763
        764
        765
        766
        767
        768
        769
        770
        771
        772
        773
        774
        775
        776
        777
        778
        779
        780
        781
        782
        783
        784
        785
        786
        787
        788
        789
        790
        791
        792
        793
        794
        795
        796
        797
        798
        799
        800
        801
        802
        803
        804
        805
        806
        807
        808
        809
        810
        811
        812
        813
        814
        815
        816
        817
        818
        819
        820
        821
        822
        823
        824
        825
        826
        827
        828
        829
        830
        831
        832
        833
        834
        835
        836
        837
        838
        839
        840
        841
        842
        843
        844
        845
        846
        847
        848
        849
        850
        851
        852
        853
        854
        855
        856
        857
        858
        859
        860
        861
        862
        863
        864
        865
        866
        867
        868
        869
        870
        871
        872
        873
        874
        875
        876
        877
        878
        879
        880
        881
        882
        883
        884
        885
        886
        887
        888
        889
        890
        891
        892
        893
        894
        895
        896
        897
        898
        899
        900
        901
        902
        903
        904
        905
        906
        907
        908
        909
        910
        911
        912
        913
        914
        915
        916
        917
        918
        919
        920
        921
        922
        923
        924
        925
        926
        927
        928
        929
        930
        931
        932
        933
        934
        935
        936
        937
        938
        939
        940
        941
        942
        943
        944
        945
        946
        947
        948
        949
        950
        951
        952
        953
        954
        955
        956
        957
        958
        959
        960
        961
        962
        963
        964
        965
        966
        967
        968
        969
        970
        971
        972
        973
        974
        975
        976
        977
        978
        979
        980
        981
        982
        983
        984
        985
        986
        987
        988
        989
        990
        991
        992
        993
        994
        995
        996
        997
        998
        999
       1000
       1001
       1002
       1003
       1004
       1005
       1006
       1007
       1008
       1009
       1010
       1011
       1012
       1013
       1014
       1015
       1016
       1017
       1018
       1019
       1020
       1021
       1022
       1023
       1024
       1025
       1026
       1027
       1028
       1029
       1030
       1031
       1032
       1033
       1034
       1035
       1036
       1037
       1038
       1039
       1040
       1041
       1042
       1043
       1044
       1045
       1046
       1047
       1048
       1049
       1050
       1051
       1052
       1053
       1054
       1055
       1056
       1057
       1058
       1059
       1060
       1061
       1062
       1063
       1064
       1065
       1066
       1067
       1068
       1069
       1070
       1071
       1072
       1073
       1074
       1075
       1076
       1077
       1078
       1079
       1080
       1081
       1082
       1083
       1084
       1085
       1086
       1087
       1088
       1089
       1090
       1091
       1092
       1093
       1094
       1095
       1096
       1097
       1098
       1099
       1100
       1101
       1102
       1103
       1104
       1105
       1106
       1107
       1108
       1109
       1110
       1111
       1112
       1113
       1114
       1115
       1116
       1117
       1118
       1119
       1120
       1121
       1122
       1123
       1124
       1125
       1126
       1127
       1128
       1129
       1130
       1131
       1132
       1133
       1134
       1135
       1136
       1137
       1138
       1139
       1140
       1141
       1142
       1143
       1144
       1145
       1146
       1147
       1148
       1149
       1150
       1151
       1152
       1153
       1154
       1155
       1156
       1157
       1158
       1159
       1160
       1161
       1162
       1163
       1164
       1165
       1166
       1167
       1168
       1169
       1170
       1171
       1172
       1173
       1174
       1175
       1176
       1177
       1178
       1179
       1180
       1181
       1182
       1183
       1184
       1185
       1186
       1187
       1188
       1189
       1190
       1191
       1192
       1193
       1194
       1195
       1196
       1197
       1198
       1199
       1200
       1201
       1202
       1203
       1204
       1205
       1206
       1207
       1208
       1209
       1210
       1211
       1212
       1213
       1214
       1215
       1216
       1217
       1218
       1219
       1220
       1221
       1222
       1223
       1224
       1225
       1226
       1227
       1228
       1229
       1230
       1231
       1232
       1233
       1234
       1235
       1236
       1237
       1238
       1239
       1240
       1241
       1242
       1243
       1244
       1245
       1246
       1247
       1248
       1249
       1250
       1251
       1252
       1253
       1254
       1255
       1256
       1257
       1258
       1259
       1260
       1261
       1262
       1263
       1264
       1265
       1266
       1267
       1268
       1269
       1270
       1271
       1272
       1273
       1274
       1275
       1276
       1277
       1278
       1279
       1280
       1281
       1282
       1283
       1284
       1285
       1286
       1287
       1288
       1289
       1290
       1291
       1292
       1293
       1294
       1295
       1296
       1297
       1298
       1299
       1300
       1301
       1302
       1303
       1304
       1305
       1306
       1307
       1308
       1309
       1310
       1311
       1312
       1313
       1314
       1315
       1316
       1317
       1318
       1319
       1320
       1321
       1322
       1323
       1324
       1325
       1326
       1327
       1328
       1329
       1330
       1331
       1332
       1333
       1334
       1335
       1336
       1337
       1338
       1339
       1340
       1341
       1342
       1343
       1344
       1345
       1346
       1347
       1348
       1349
       1350
       1351
       1352
       1353
       1354
       1355
       1356
       1357
       1358
       1359
       1360
       1361
       1362
       1363
       1364
       1365
       1366
       1367
       1368
       1369
       1370
       1371
       1372
       1373
       1374
       1375
       1376
       1377
       1378
       1379
       1380
       1381
       1382
       1383
       1384
       1385
       1386
       1387
       1388
       1389
       1390
       1391
       1392
       1393
       1394
       1395
       1396
       1397
       1398
       1399
       1400
       1401
       1402
       1403
       1404
       1405
       1406
       1407
       1408
       1409
       1410
       1411
       1412
       1413
       1414
       1415
       1416
       1417
       1418
       1419
       1420
       1421
       1422
       1423
       1424
       1425
       1426
       1427
       1428
       1429
       1430
       1431
       1432
       1433
       1434
       1435
       1436
       1437
       1438
       1439
       1440
       1441
       1442
       1443
       1444
       1445
       1446
       1447
       1448
       1449
       1450
       1451
       1452
       1453
       1454
       1455
       1456
       1457
       1458
       1459
       1460
       1461
       1462
       1463
       1464
       1465
       1466
       1467
       1468
       1469
       1470
       1471
       1472
       1473
       1474
       1475
       1476
       1477
       1478
       1479
       1480
       1481
       1482
       1483
       1484
       1485
       1486
       1487
       1488
       1489
       1490
       1491
       1492
       1493
       1494
       1495
       1496
       1497
       1498
       1499
       1500
       1501
       1502
       1503
       1504
       1505
       1506
       1507
       1508
       1509
       1510
       1511
       1512
       1513
       1514
       1515
       1516
       1517
       1518
       1519
       1520
       1521
       1522
       1523
       1524
       1525
       1526
       1527
       1528
       1529
       1530
       1531
       1532
       1533
       1534
       1535
       1536
       1537
       1538
       1539
       1540
       1541
       1542
       1543
       1544
       1545
       1546
       1547
       1548
       1549
       1550
       1551
       1552
       1553
       1554
       1555
       1556
       1557
       1558
       1559
       1560
       1561
       1562
       1563
       1564
       1565
       1566
       1567
       1568
       1569
       1570
       1571
       1572
       1573
       1574
       1575
       1576
       1577
       1578
       1579
       1580
       1581
       1582
       1583
       1584
       1585
       1586
       1587
       1588
       1589
       1590
       1591
       1592
       1593
       1594
       1595
       1596
       1597
       1598
       1599
       1600
       1601
       1602
       1603
       1604
       1605
       1606
       1607
       1608
       1609
       1610
       1611
       1612
       1613
       1614
       1615
       1616
       1617
       1618
       1619
       1620
       1621
       1622
       1623
       1624
       1625
       1626
       1627
       1628
       1629
       1630
       1631
       1632
       1633
       1634
       1635
       1636
       1637
       1638
       1639
       1640
       1641
       1642
       1643
       1644
       1645
       1646
       1647
       1648
       1649
       1650
       1651
       1652
       1653
       1654
       1655
       1656
       1657
       1658
       1659
       1660
       1661
       1662
       1663
       1664
       1665
       1666
       1667
       1668
       1669
       1670
       1671
       1672
       1673
       1674
       1675
       1676
       1677
       1678
       1679
       1680
       1681
       1682
       1683
       1684
       1685
       1686
       1687
       1688
       1689
       1690
       1691
       1692
       1693
       1694
       1695
       1696
       1697
       1698
       1699
       1700
       1701
       1702
       1703
       1704
       1705
       1706
       1707
       1708
       1709
       1710
       1711
       1712
       1713
       1714
       1715
       1716
       1717
       1718
       1719
       1720
       1721
       1722
       1723
       1724
       1725
       1726
       1727
       1728
       1729
       1730
       1731
       1732
       1733
       1734
       1735
       1736
       1737
       1738
       1739
       1740
       1741
       1742
       1743
       1744
       1745
       1746
       1747
       1748
       1749
       1750
       1751
       1752
       1753
       1754
       1755
       1756
       1757
       1758
       1759
       1760
       1761
       1762
       1763
       1764
       1765
       1766
       1767
       1768
       1769
       1770
       1771
       1772
       1773
       1774
       1775
       1776
       1777
       1778
       1779
       1780
       1781
       1782
       1783
       1784
       1785
       1786
       1787
       1788
       1789
       1790
       1791
       1792
       1793
       1794
       1795
       1796
       1797
       1798
       1799
       1800
       1801
       1802
       1803
       1804
       1805
       1806
       1807
       1808
       1809
       1810
       1811
       1812
       1813
       1814
       1815
       1816
       1817
       1818
       1819
       1820
       1821
       1822
       1823
       1824
       1825
       1826
       1827
       1828
       1829
       1830
       1831
       1832
       1833
       1834
       1835
       1836
       1837
       1838
       1839
       1840
       1841
       1842
       1843
       1844
       1845
       1846
       1847
       1848
       1849
       1850
       1851
       1852
       1853
       1854
       1855
       1856
       1857
       1858
       1859
       1860
       1861
       1862
       1863
       1864
       1865
       1866
       1867
       1868
       1869
       1870
       1871
       1872
       1873
       1874
       1875
       1876
       1877
       1878
       1879
       1880
       1881
       1882
       1883
       1884
       1885
       1886
       1887
       1888
       1889
       1890
       1891
       1892
       1893
       1894
       1895
       1896
       1897
       1898
       1899
       1900
       1901
       1902
       1903
       1904
       1905
       1906
       1907
       1908
       1909
       1910
       1911
       1912
       1913
       1914
       1915
       1916
       1917
       1918
       1919
       1920
       1921
       1922
       1923
       1924
       1925
       1926
       1927
       1928
       1929
       1930
       1931
       1932
       1933
       1934
       1935
       1936
       1937
       1938
       1939
       1940
       1941
       1942
       1943
       194
```



For the next part, we use the Open3D library to display the lidar point cloud on a 3D viewer and identify 10 images from point cloud

- Visualize the point cloud in Open3D
- 10 examples from point cloud with varying degrees of visibility

The changes are made in 'loop\_over\_dataset.py'

```

3D Object Detection Workspace
README.md  X  loop_over_datas...  X  objdet_pcl.py  X
74  ## Initialize tracking
75  KF = Filter() # set up Kalman filter
76  association = Association() # init data association
77  manager = Trackmanagement() # init track manager
78  lidar = None # init lidar sensor object
79  camera = None # init camera sensor object
80  np.random.seed(10) # make random values predictable
81
82  ## Selective execution and visualization
83  exe_data=[] #option 'pcl from rangeimage','load_image'
84  exec_detection = [] # options are 'bev from pcl', 'detect_objects', 'validate_object_labels', 'measure_detection'
85  # not in the list will be loaded from file
86  exec_tracking = [] # options are 'perform_tracking'
87  exec_visualization = ['show_pcl'] # options are 'show_range_image', 'show_bev', 'show_pcl', 'show_labels_in_bev',
88  # 'show_objects_and_labels_in_bev', 'show_objects_in_bev_labels_in_camera', 'show_tracks', 'show_detection_performance',
89  # 'make_tracking_movie'
90  exec_list = make_exec_list(exec_detection, exec_tracking, exec_visualization)
91  vis_pause_time = 0 # set pause time between frames in ms (0 = stop between frames until key is pressed)

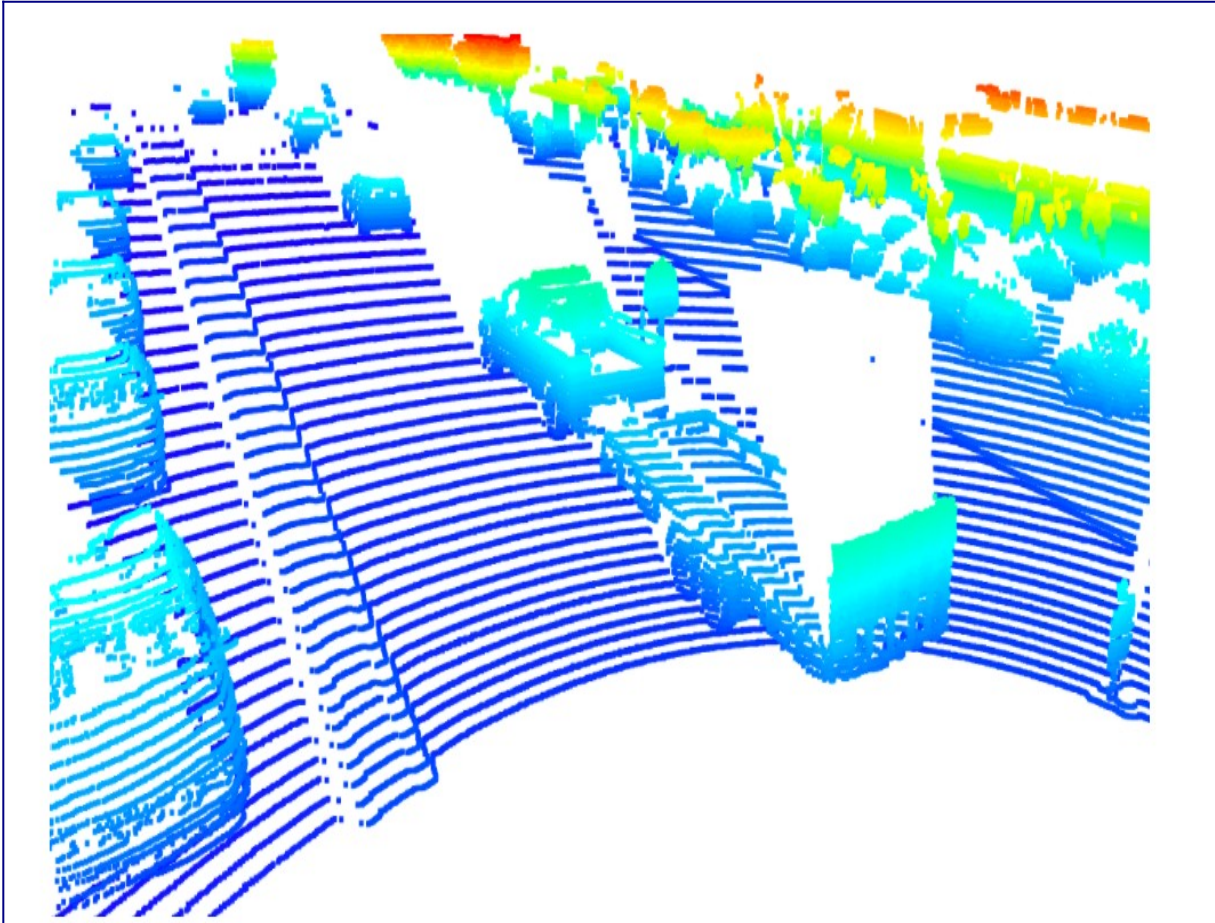
```



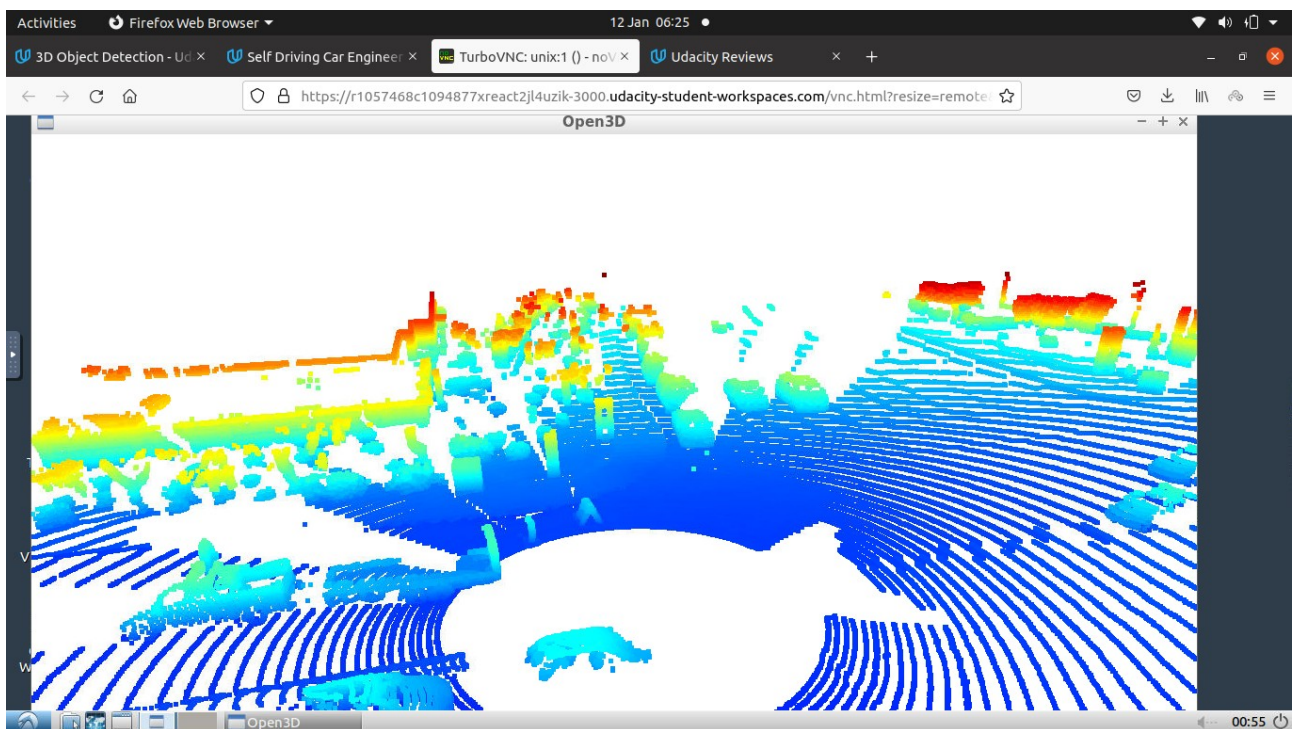
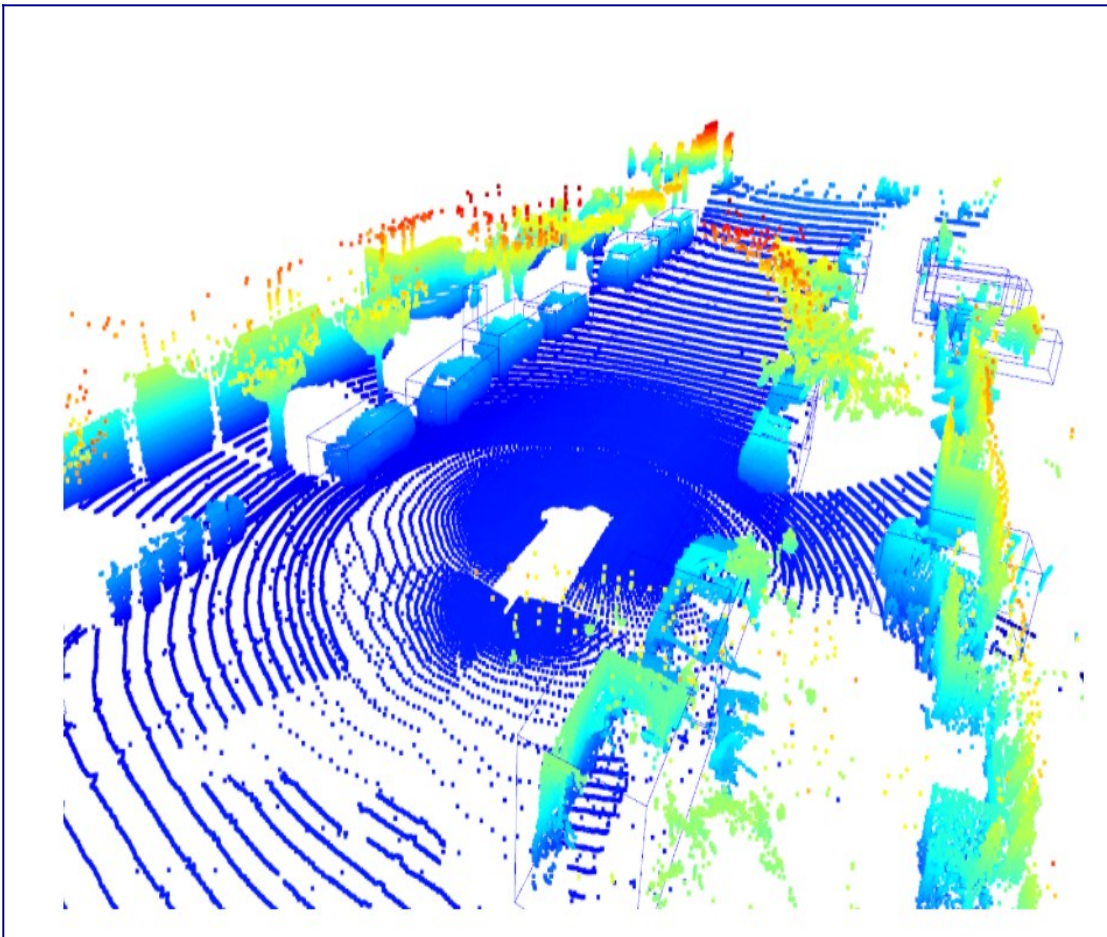
The changes are made in "objdet\_pcl.py"

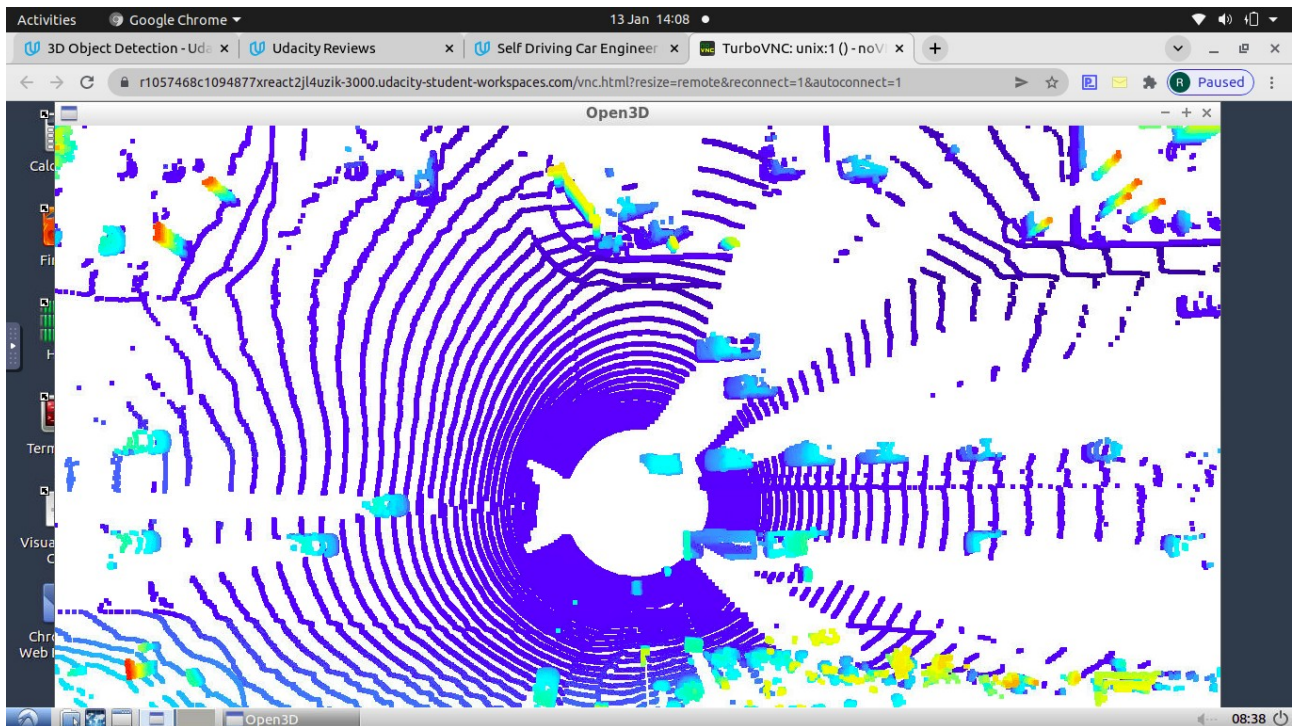
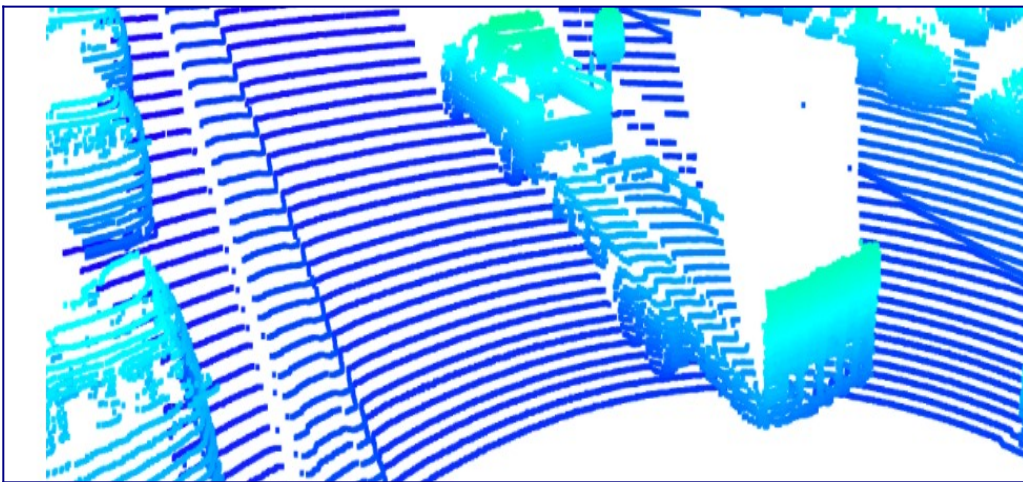
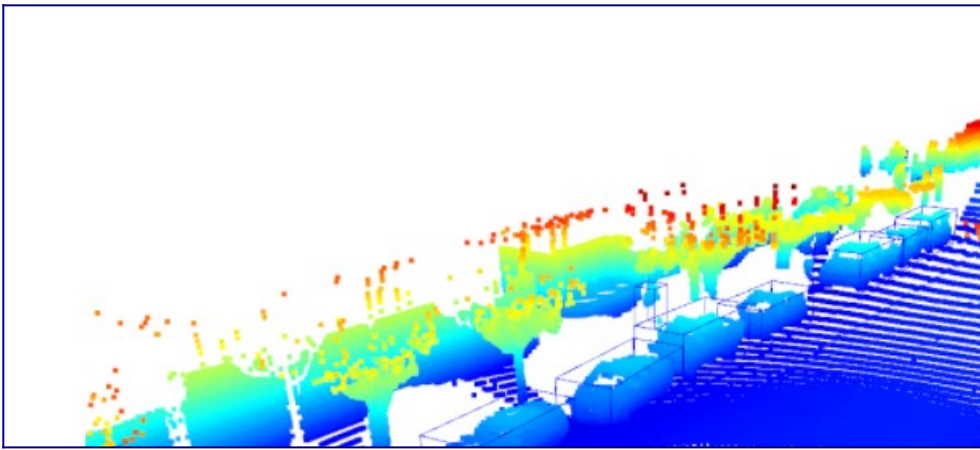
```
README.md X loop_over_dase... X objdet_eval.py X objdet_pcl.py X
--
36 # visualize lidar point-cloud
37 def show_pcl(pcl):
38
39     ##### ID_S1_EX2 START #####
40     #####
41     print("student task ID_S1_EX2")
42
43     # step 1 : initialize open3d with key callback and create window
44     viz = o3d.visualization.VisualizerWithKeyCallback()
45     viz.create_window(window_name='Open3D', width=1280, height=1080, left=50, top=50, visible=True)
46
47     global ptr
48     ptr = True
49     def Window(viz):
50         global ptr
51         print('right arrow pressed')
52         ptr = False
53         return
54     viz.register_key_callback(262, Window)
55
56     # step 2 : create instance of open3d point-cloud class
57     pcd = o3d.geometry.PointCloud()
58
59     # step 3 : set points in pcd instance by converting the point-cloud into 3d vectors (using open3d function Vector3dVector)
60     pcd.points = o3d.utility.Vector3dVector(pcl[:, :3])
61
62     # step 4 : for the first frame, add the pcd instance to visualization using add_geometry; for all other frames, use update_geometry
63     viz.add_geometry(pcd)
64
65     # step 5 : visualize point cloud and keep window open until right-arrow is pressed (key-code 262)
66     while ptr:
67         viz.poll_events()
68         viz.update_renderer()
```

Point cloud images

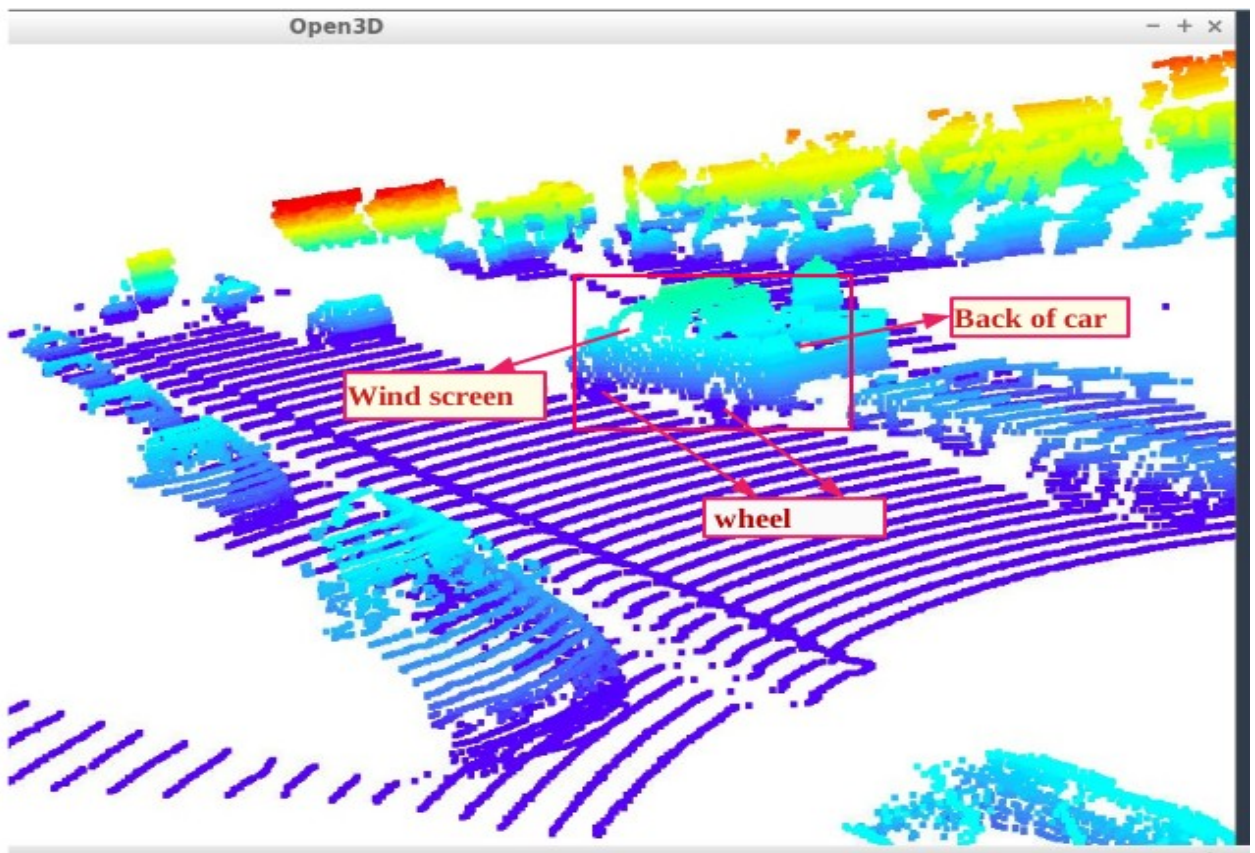
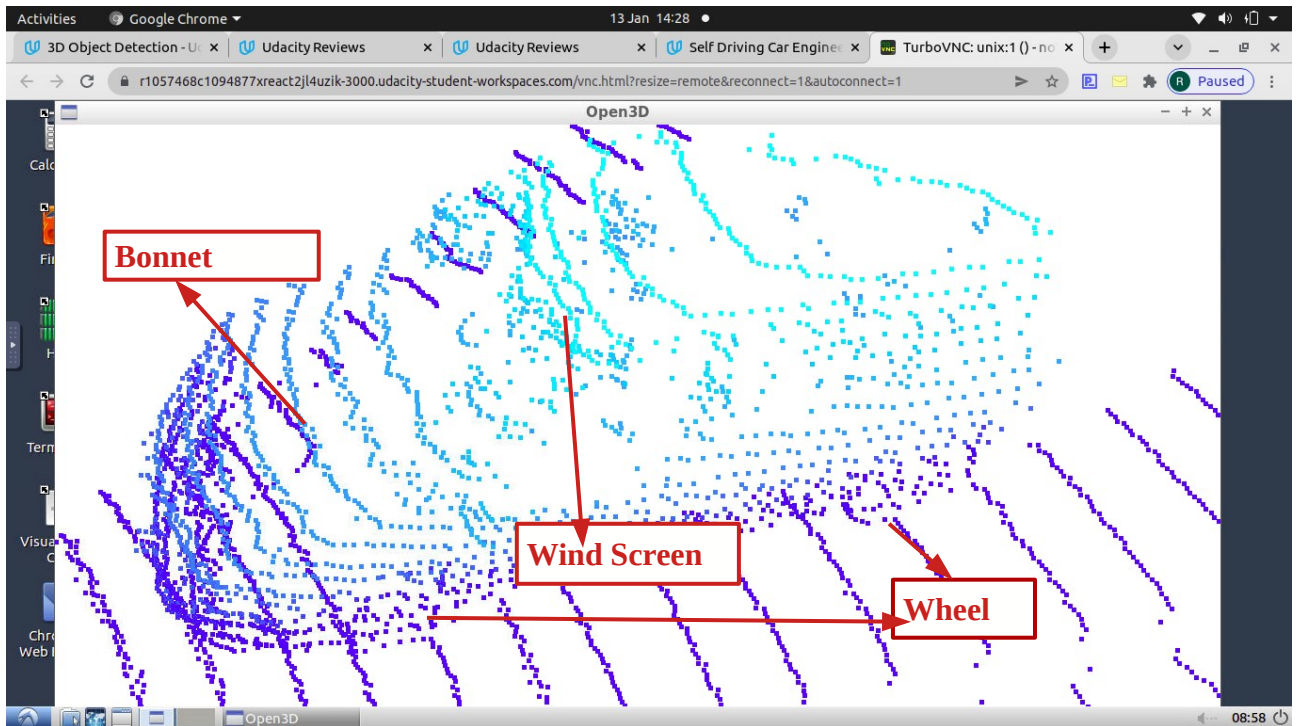


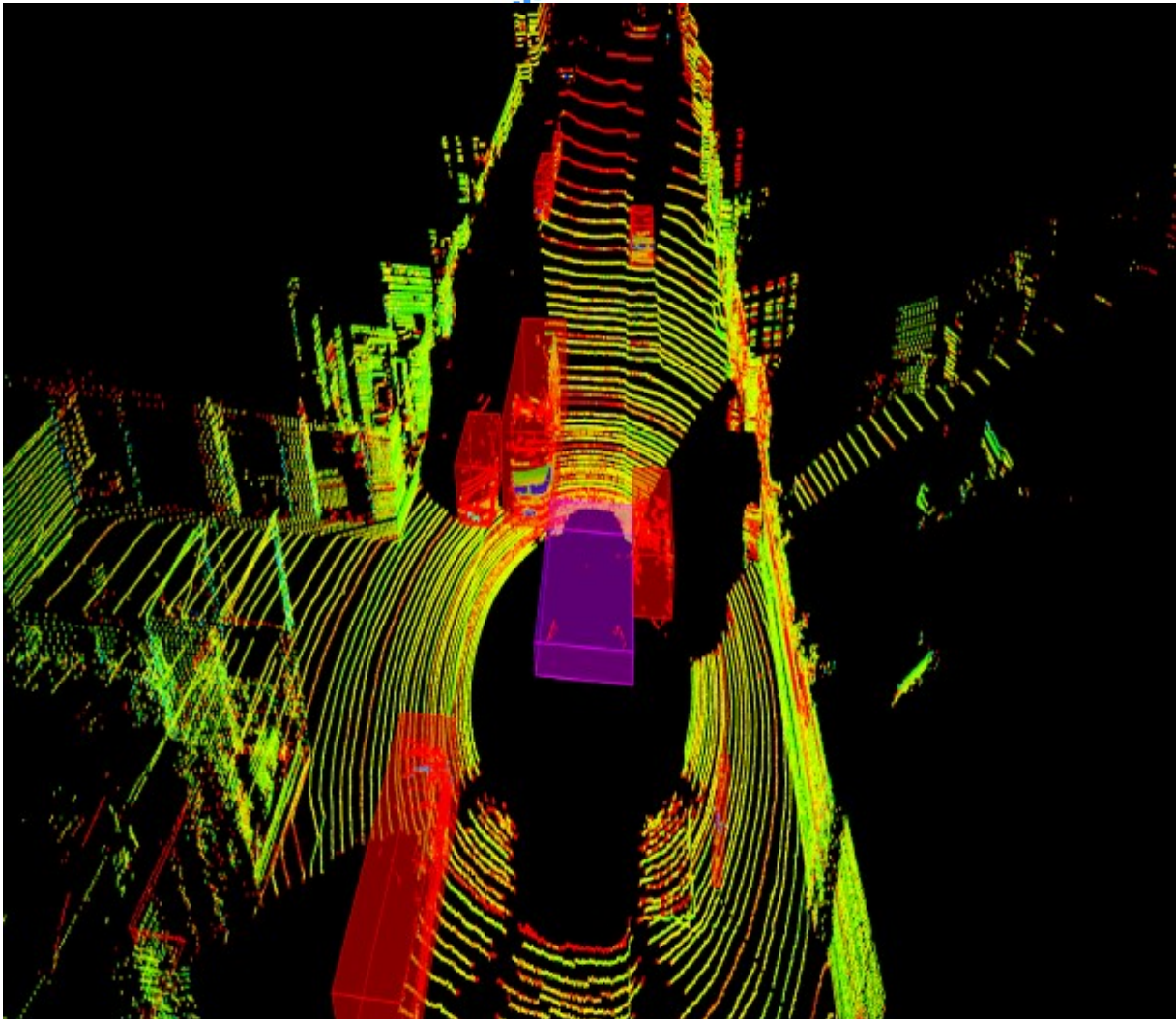
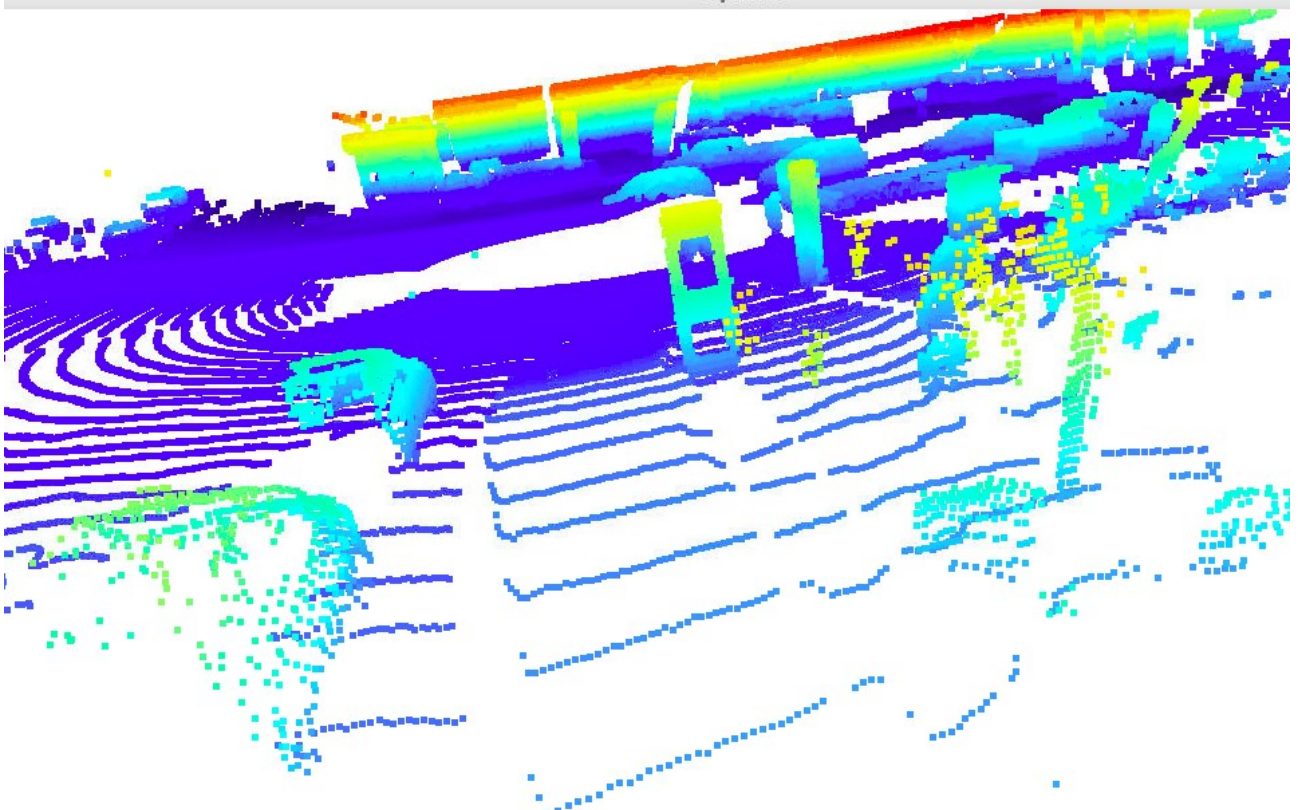














Stable features include the tail lights, the rear bumper majorly. In some cases the additional features include the headover lights, car front lights, rear window shields. These are identified through the intensity channels . The chassis of the car is the most prominent identifiable feature from the lidar perspective. The images are analysed with different settings and the rear lights are the major stable components, also the bounding boxes are correctly assigned to the cars (used from Step-3).

## Step-2: Creaate BEV from Lidar PCL

In this case, we are:

- Converting the coordinates to pixel values
- Assigning lidar intensity values to the birds eye view BEV mapping
- Using sorted and pruned point cloud lidar from the previous task
- Normalizing the height map in the BEV
- Compute and map the intensity values

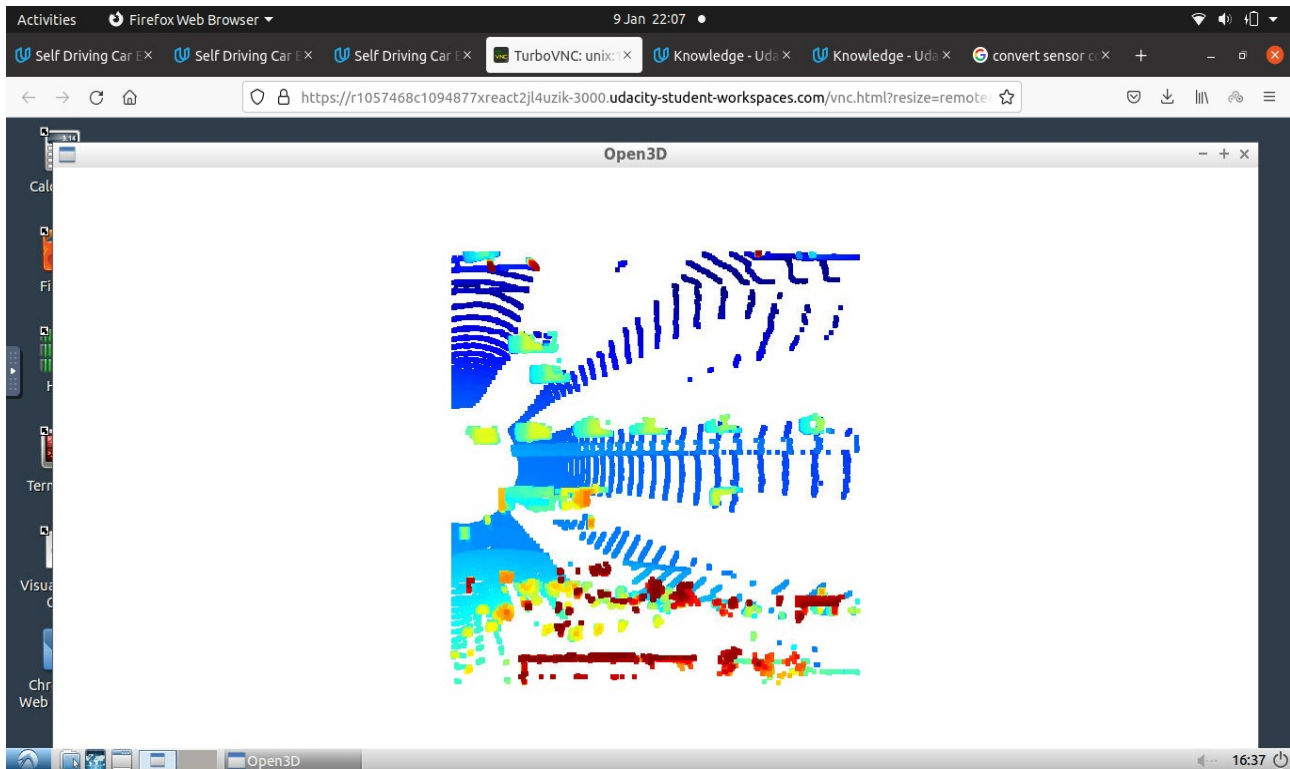
The changes are in the 'loop\_over\_dataset.py'

```
82 ## Selective execution and visualization
83 exe_data=['pcl_from_rangeimage'] #option 'pcl_from_rangeimage','load_image'
84 exec_detection = ['bev_from_pcl'] # options are 'bev_from_pcl', 'detect_objects', 'validate_object_labels', 'mea
loaded from file
85 exec_tracking = [] # options are 'perform_tracking'
86 exec_visualization = [] # options are 'show_range_image', 'show_bev', 'show_pcl', 'show_labels_in_image', 'show
'show_objects_in_bev_labels_in_camera', 'show_tracks', 'show_detection_performance', 'make_tracking_movie'
87 exec_list = make_exec_list(exec_detection, exec_tracking, exec_visualization)
88 vis_pause_time = 0 # set pause time between frames in ms (0 = stop between frames until key is pressed)
89
90
```

The changes are also in the "objdet\_pcl.py"

```
3D Object Detection Workspace
README.md  X  loop_over_dase...  X  objdet_pcl.py  X
121
122 # create birds-eye view of lidar data
123 def bev_from_pcl(lidar_pcl, configs):
124
125     # remove lidar points outside detection area and with too low reflectivity
126     mask = np.where((lidar_pcl[:, 0] >= configs.lim_x[0]) & (lidar_pcl[:, 0] <= configs.lim_x[1]) &
127                     (lidar_pcl[:, 1] >= configs.lim_y[0]) & (lidar_pcl[:, 1] <= configs.lim_y[1]) &
128                     (lidar_pcl[:, 2] >= configs.lim_z[0]) & (lidar_pcl[:, 2] <= configs.lim_z[1]))
129     lidar_pcl = lidar_pcl[mask]
130
131     # shift level of ground plane to avoid flipping from 0 to 255 for neighboring pixels
132     lidar_pcl[:, 2] = lidar_pcl[:, 2] - configs.lim_z[0]
133
134     # convert sensor coordinates to bev-map coordinates (center is bottom-middle)
135     ##### ID_S2_EX1 START #####
136     #####
137     print("Student task ID_S2_EX1")
138
139
140     ## step 1 : compute bev-map discretization by dividing x-range by the bev-image height (see configs)
141     bev_discret = (configs.lim_x[1] - configs.lim_x[0]) / configs.bev_height
142
143
144     ## step 2 : create a copy of the lidar pcl and transform all metrix x-coordinates into bev-image coordinates
145     copy_lidar_pcl = np.copy(lidar_pcl)
146     copy_lidar_pcl[:, 0] = np.int(np.floor(copy_lidar_pcl[:, 0] / bev_discret))
147
148     # step 3 : perform the same operation as in step 2 for the y-coordinates but make sure that no negative bev-coordinates occur
149     copy_lidar_pcl[:, 1] = np.int(np.floor(copy_lidar_pcl[:, 1] / bev_discret) + (configs.bev_width + 1) / 2)
150     copy_lidar_pcl[:, 1] = np.abs(copy_lidar_pcl[:, 1])
151
152     # step 4 : visualize point-cloud using the function show_pcl from a previous task
```

A sample preview of the BEV:



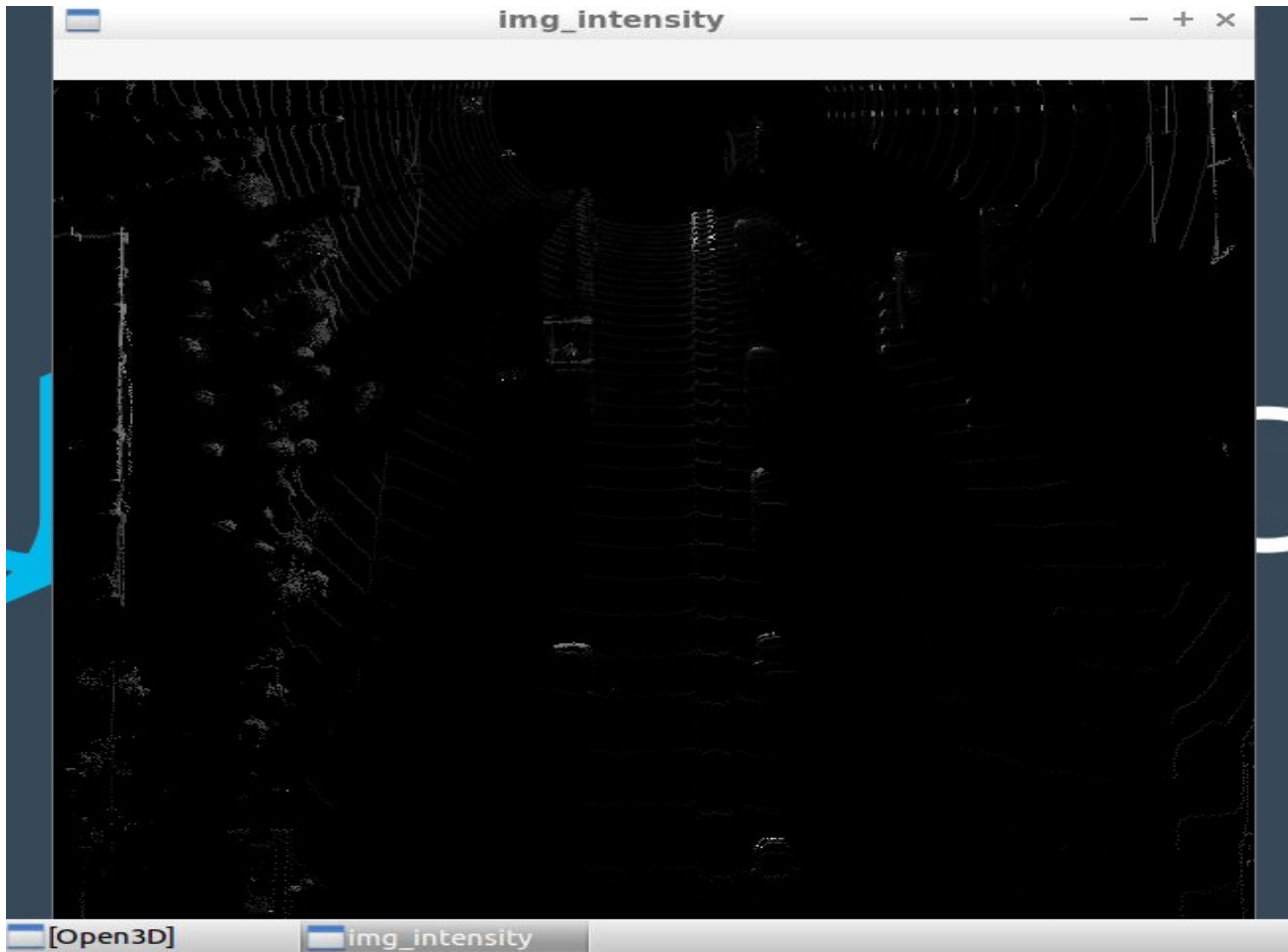
A preview of the intensity layer:

```
3D Object Detection Workspace
README.md  X  loop_over_dats...  X  objdet_pcl.py  X
158 # Compute intensity layer of the BEV map
159 ##### ID_S2_EX2 START #####
160 #####
161 print("student task ID_S2_EX2")
162 # refer from github
163 ## step 1 : create a numpy array filled with zeros which has the same dimensions as the BEV map
164 map_intensity = np.zeros((configs.bev_height, configs.bev_width))
165
166 # step 2 : re-arrange elements in lidar_pcl_cpy by sorting first by x, then y, then -z (use numpy.lexsort)
167 copy_lidar_pcl[copy_lidar_pcl[:,3]>1.0,3] = 1.0
168 intensity_pointer = np.lexsort((-copy_lidar_pcl[:, 2], copy_lidar_pcl[:, 1], copy_lidar_pcl[:, 0]))
169 lidar_pcl_top = copy_lidar_pcl[intensity_pointer]
170
171 ## step 3 : extract all points with identical x and y such that only the top-most z-coordinate is kept (use numpy.unique)
172 ## also, store the number of points per x,y-cell in a variable named "counts" for use in the next task
173 lidar_pcl_int, idxs, count = np.unique(copy_lidar_pcl[:, 0:2], axis=0, return_index=True, return_counts=True)
174 lidar_pcl_top = copy_lidar_pcl[idxs]
175
176 ## step 4 : assign the intensity value of each unique entry in lidar_top_pcl to the intensity map
177 ## make sure that the intensity is scaled in such a way that objects of interest (e.g. vehicles) are clearly visible
178 ## also, make sure that the influence of outliers is mitigated by normalizing intensity on the difference between the max. a
179 cloud
180 map_intensity[np.int_(lidar_pcl_top[:, 0]),
181              np.int_(lidar_pcl_top[:, 1])] = lidar_pcl_top[:, 3] / (np.amax(lidar_pcl_top[:, 3]) - np.amin(lidar_pcl_top[:, 3]))
182
183 ## step 5 : temporarily visualize the intensity map using OpenCV to make sure that vehicles separate well from the background
184 img_intensity = map_intensity * 256
185 img_intensity = img_intensity.astype(np.uint8)
186 cv2.imshow('img_intensity', img_intensity)
187 cv2.waitKey(0)
188 cv2.destroyAllWindows()
```

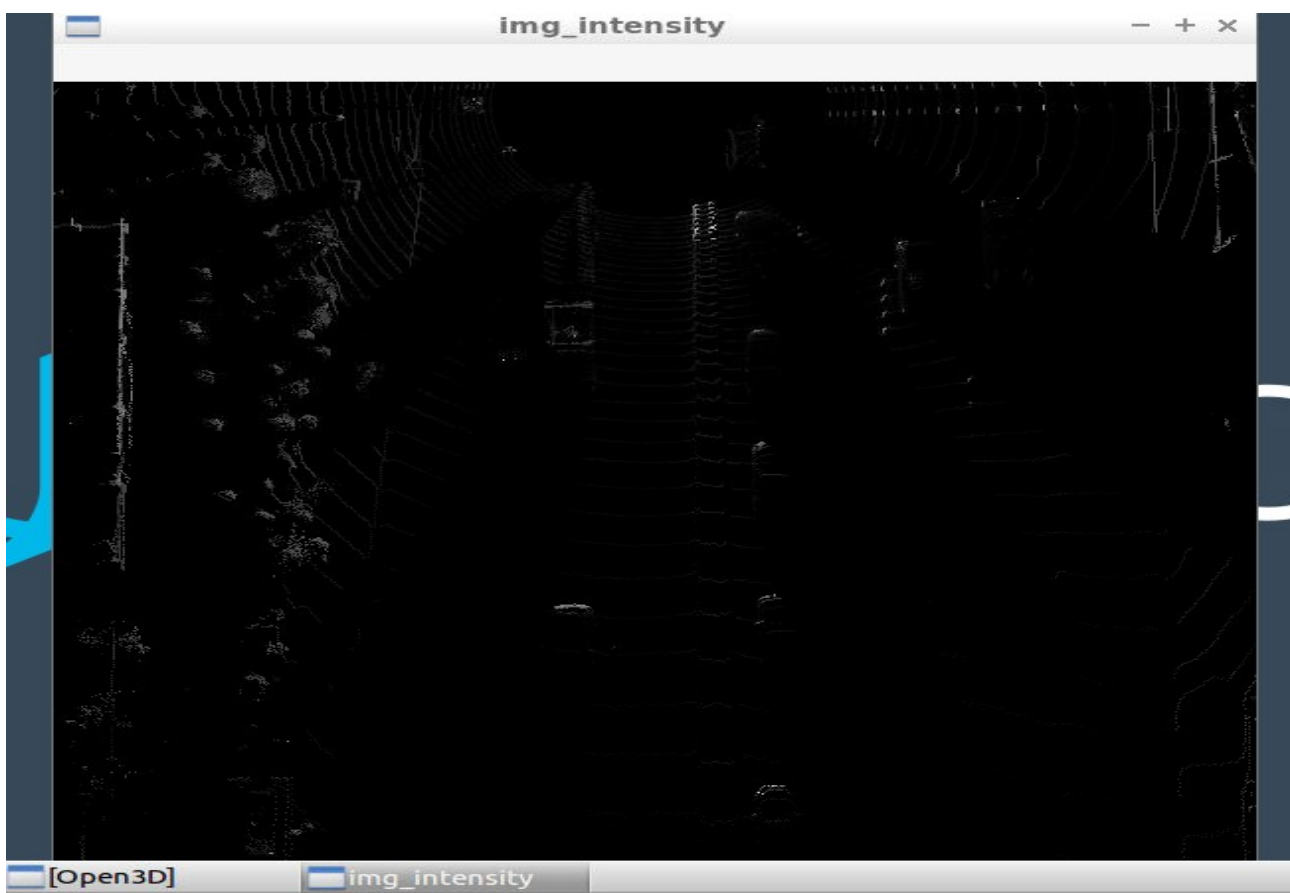
The 'lidar\_pcl\_top' is used in this case, shown in the Figure:

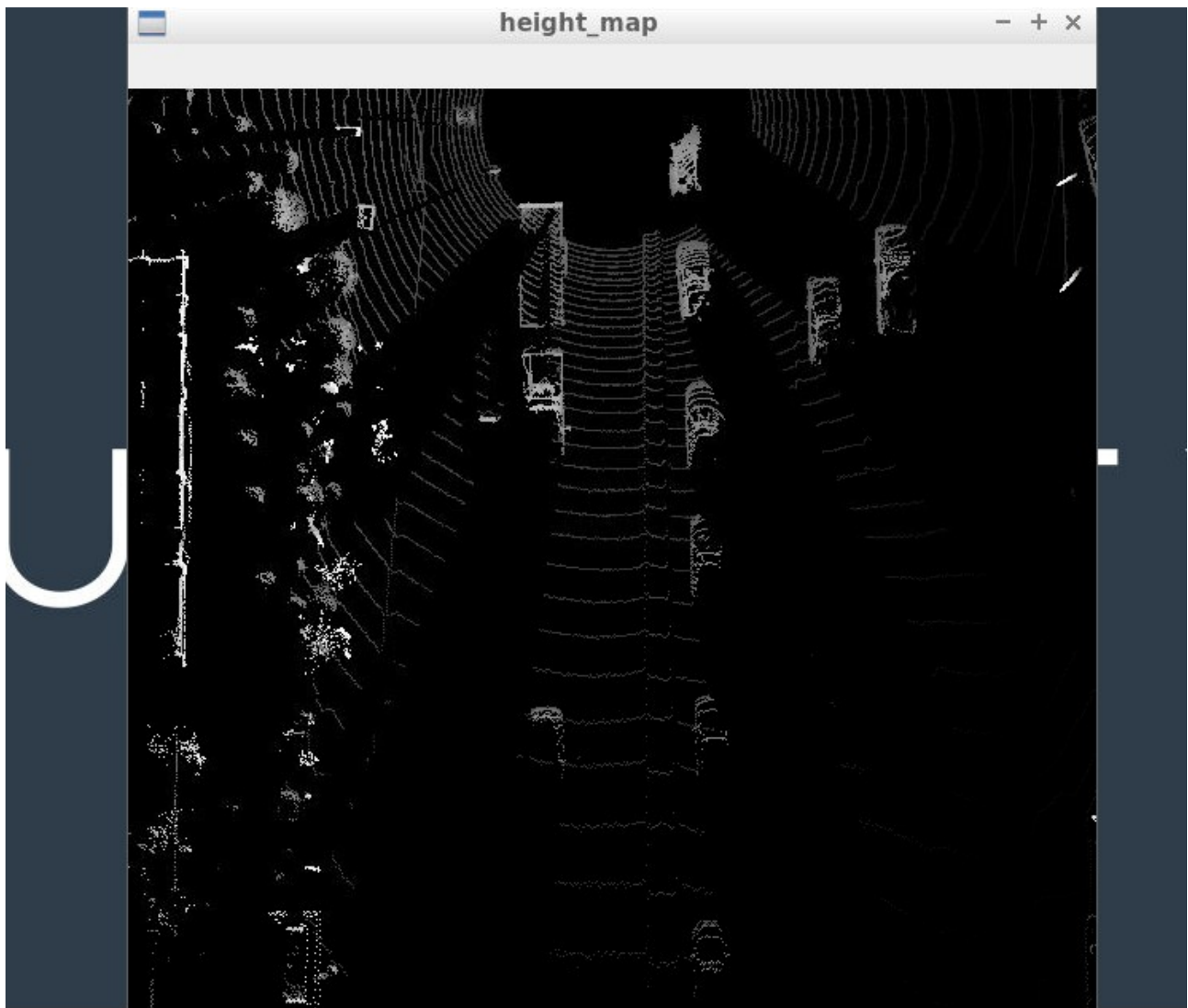


The corresponding intensity channel:



The corresponding normalized height channel:





### Step-3: Model Based Object Detection in BEV Image

Here, particularly the test.py file and extracting the relevant configurations from 'parse\_test\_configs()' and added them in the 'load\_configs\_model' config structure by looking into this repo [<https://github.com/maudzung/SFA3D.git>].

- Instantiating the fpn resnet model from the cloned repository configs
- Extracting 3d bounding boxes from the responses
- Transforming the pixel to vehicle coordinates
- Model output tuned to the bounding box format [class-id, x, y, z, h, w, l, yaw]

The changes are in "loop\_over\_dataset.py"

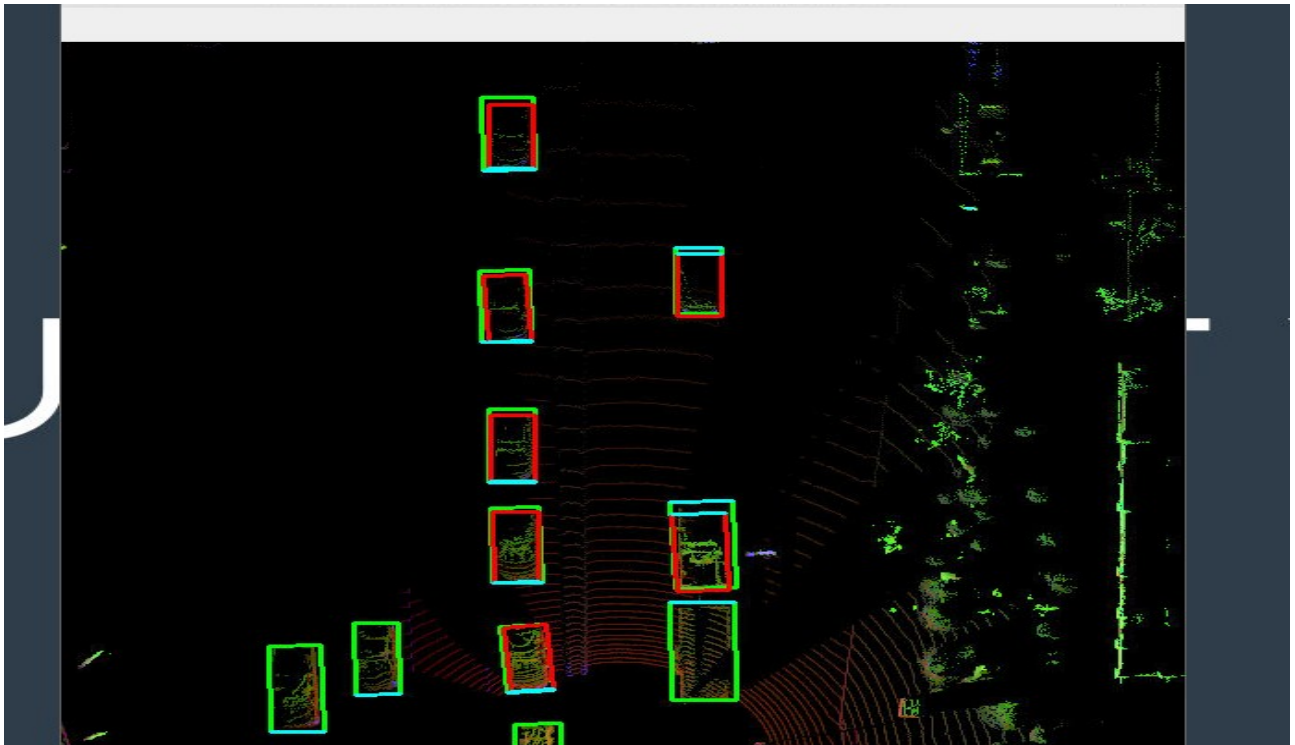
```
81
82 ## Selective execution and visualization
83 exe_data=['pcl_from_rangeimage','load_image'] #option 'pcl_from_rangeimage','load_image'
84 exec_detection = ['bev_from_pcl','detect_objects'] # options are 'bev_from_pcl', 'detect_objects', 'validate_object_labels', 'measure_det
the list will be loaded from file
85 exec_tracking = [] # options are 'perform_tracking'
86 exec_visualization = ['show_objects_in_bev_labels_in_camera'] # options are 'show_range_image', 'show_bev', 'show_pcl', 'show_labels_in_
'show_objects_and_labels_in_bev', 'show_objects_in_bev_labels_in_camera', 'show_tracks', 'show_detection_performance', 'make_tracking_mov
87 exec_list = make_exec_list(exec_detection, exec_tracking, exec_visualization)
88 vis_pause_time = 0 # set pause time between frames in ms (0 = stop between frames until key is pressed)
89
```

The changes for the detection are inside the "objdet\_detect.py" file:

```
README.md X loop_over_datase... X objdet_detect.py X
181 # detect trained objects in birds-eye view
182 def detect_objects(input_bev_maps, model, configs):
183
184     # deactivate autograd engine during test to reduce memory usage and speed up computations
185     with torch.no_grad():
186
187         # perform inference
188         outputs = model(input_bev_maps)
189
190         # decode model output into target object format
191         if 'darknet' in configs.arch:
192
193             # perform post-processing
194             output_post = post_processing_v2(outputs, conf_thresh=configs.conf_thresh, nms_thresh=configs.nms_thresh)
195             detections = []
196             for sample_i in range(len(output_post)):
197                 if output_post[sample_i] is None:
198                     continue
199                 detection = output_post[sample_i]
200                 for obj in detection:
201                     x, y, w, l, im, re, _, _ = obj
202                     yaw = np.arctan2(im, re)
203                     detections.append([1, x, y, 0.0, 1.50, w, l, yaw])
204
205         elif 'fpn_resnet' in configs.arch:
206             # decode output and perform post-processing
207
208             ##### ID_S3_EX1-5 START #####
209             #####
210             print("student task ID_S3_EX1-5")
211             outputs['hm_cen'] = _sigmoid(outputs['hm_cen'])
212             outputs['cen_offset'] = _sigmoid(outputs['cen_offset'])
213             # detections size (batch_size, K, 10)
214             detections = decode(outputs['hm_cen'], outputs['cen_offset'], outputs['direction'], outputs['z_coor'],
215                                 outputs['dim'], K=40) #K=configs.k
216             detections = detections.cpu().numpy().astype(np.float32)
217             # print(detections)
218             detections = post_processing(detections, configs)
219             detections = detections[0][1]
```

As the model input is a three-channel BEV map, the detected objects will be returned with coordinates and properties in the BEV coordinate space. Thus, before the detections can move along in the processing pipeline, they need to be converted into metric coordinates in vehicle space.

A sample preview of the bounding box images:



## Step-4: Performance detection for 3D Object Detection

In this step, the performance is computed by getting the IOU between labels and detections to get the false positive and false negative values. The task is to compute the geometric overlap between the bounding boxes of labels and the detected objects:

- Assigning a detected object to a label if IOU exceeds threshold



- Computing the degree of geometric overlap
- For multiple matches objects/detections pair with maximum IOU are kept
- Computing the false negative and false positive values
- Computing precision and recall over the false positive and false negative values

The changes in the code are:

```

81
82 ## Selective execution and visualization
83 exe_data=['pcl_from_rangeframe'] #option 'pcl_from_rangeframe','load_image'
84 exec_detection = ['bev_from_pcl','detect_objects','validate_object_labels', 'measure_detection_performance'] # option
85 'detect_objects', 'validate_object_labels', 'measure_detection_performance'; options not in the list will be loaded f
86 exec_tracking = [] # options are 'perform_tracking'
87 exec_visualization = ['show_detection_performance'] # options are 'show_range_image', 'show_bev', 'show_pcl', 'show_l
88 'show_objects_and_labels_in_bev', 'show_objects_in_bev_labels_in_camera', 'show_tracks', 'show_detection_performance'
89 exec_list = make_exec_list(exec_detection, exec_tracking, exec_visualization)
90 vis_pause_time = 0 # set pause time between frames in ms (0 = stop between frames until key is pressed)
91
92

```

The changes for "objdet\_eval.py" where the precision and recall are calculated as functions of false positives and negatives:

```

3D Object Detection workspace
README.md X loop_over_data... X objdet_eval.py X
87 center_devs.append(best_match[1:])
88
89
90 ##### ID_S4_EX2 START #####
91 #####
92 print("student task ID_S4_EX2")
93
94 # compute positives and negatives for precision/recall
95
96 ## step 1 : compute the total number of positives present in the scene
97 all_positives = labels_valid.sum()
98
99 ## step 2 : compute the number of false negatives
100 true_positives= len(ious)
101 false_negatives = all_positives-true_positives
102
103 ## step 3 : compute the number of false positives
104 false_positives = len(detections)-all_positives
105
106 #####
107 ##### ID_S4_EX2 END #####
108
109 pos_negs = [all_positives, true_positives, false_negatives, false_positives]
110 det_performance = [ious, center_devs, pos_negs]
111
112 return det_performance
113

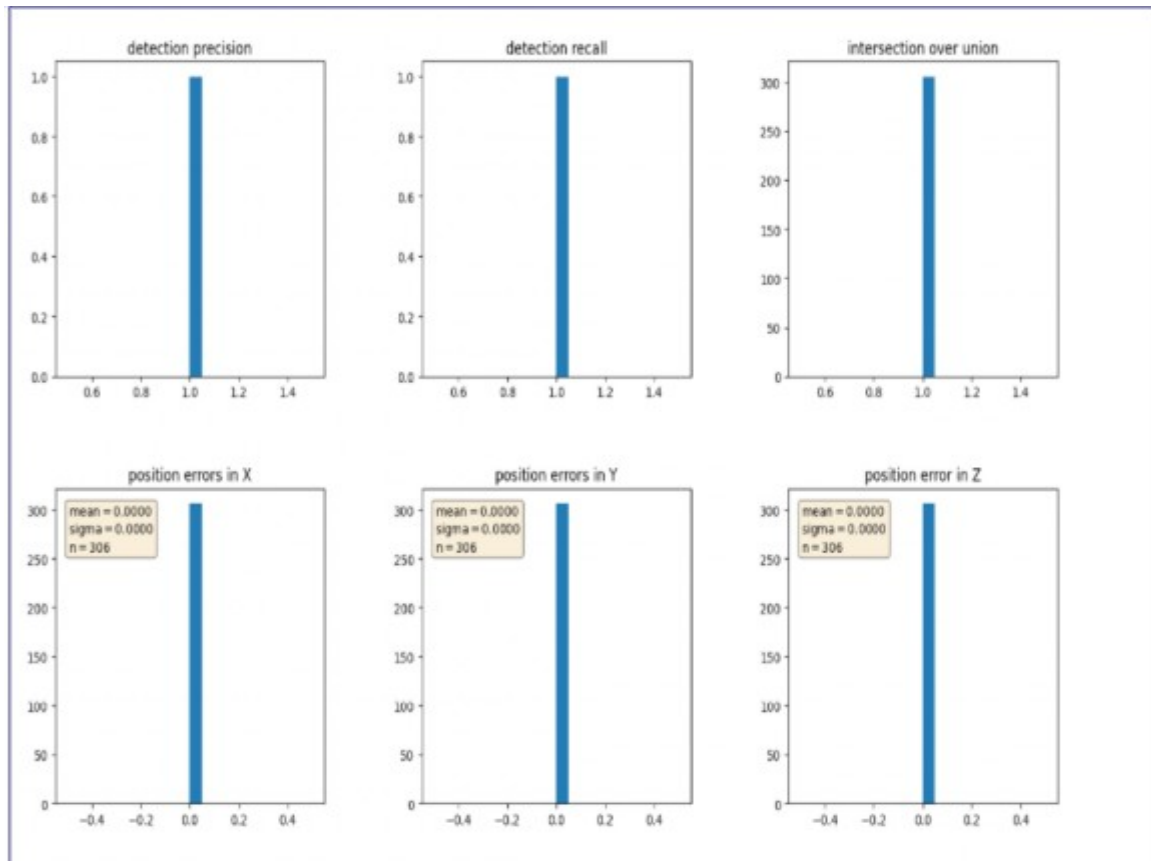
```

```

3D Object Detection workspace
README.md X loop_over_data... X objdet_eval.py X
50
51 ## step 1 : extract the four corners of the current label bounding-box
52 box=label.box
53 box_l=tools.compute_box_corners(box.center_x, box.center_y, box.width, box.length, box.heading)
54
55 ## step 2 : loop over all detected objects
56 for objects in detections:
57
58     ## step 3 : extract the four corners of the current detection
59     _id, x, y, z, _h, w, l, yaw= objects
60     box_2=tools.compute_box_corners(x, y, w, l, yaw)
61
62     ## step 4 : compute the center distance between label and detection bounding-box in x, y, and z
63     x_dist=np.array(box.center_x-x).item()
64     y_dist=np.array(box.center_y-y).item()
65     z_dist=np.array(box.center_z-z).item()
66
67     ## step 5 : compute the intersection over union (IOU) between label and detection bounding-box
68     rect1=Polygon(box_l)
69     rect2=Polygon(box_2)
70     intersection=rect1.intersection(rect2).area
71     union=rect1.union(rect2).area
72     iou=intersection/union
73
74
75     ## step 6 : if IOU exceeds min_iou threshold, store [iou,dist_x, dist_y, dist_z] in matches_lab_det and increase the TP
76
77 if iou>min_iou:
78     matches_lab_det.append([iou, dist_x, dist_y, dist_z])
79
80 #####
81 ##### ID_S4_EX1 END #####

```

The precision recall curve is plotted showing similar results of precision =0.986 and recall=0.81372

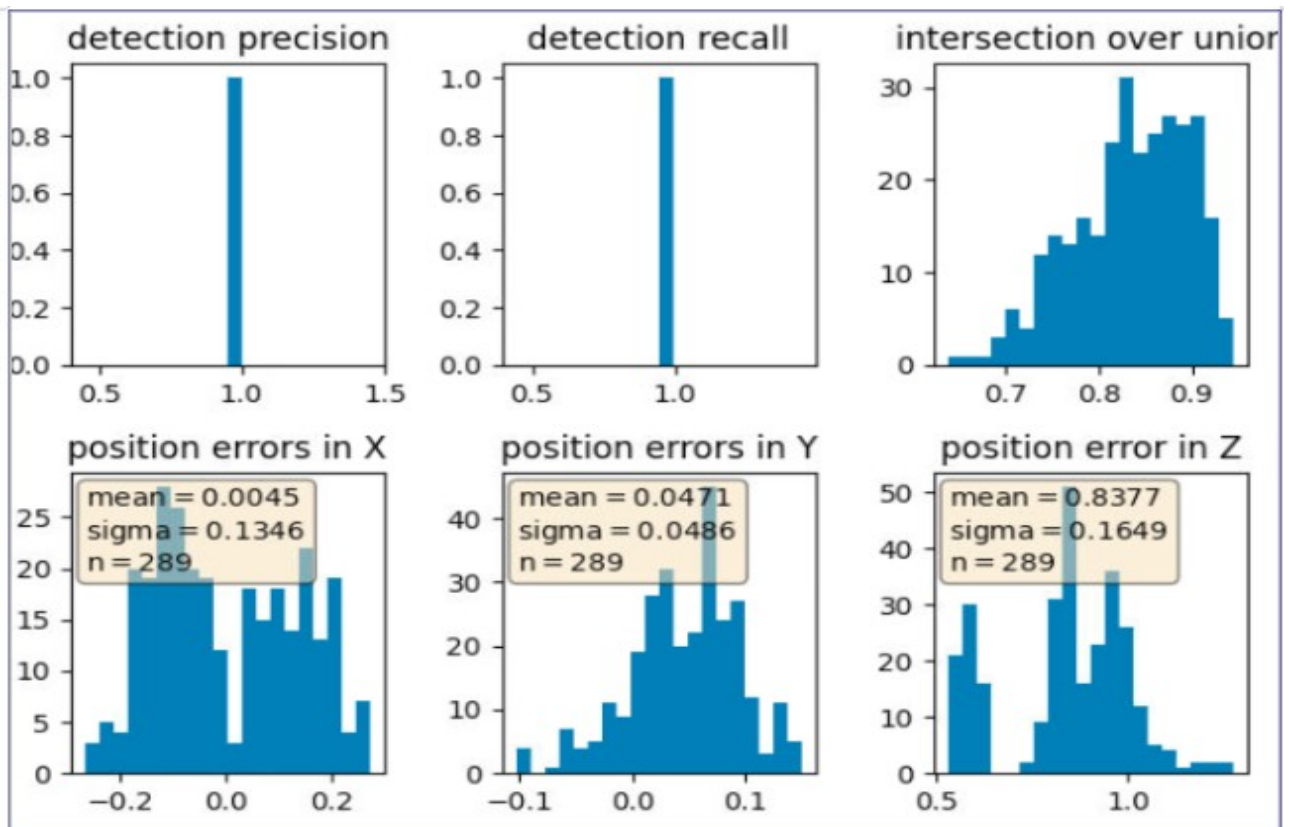


In the next step, we set the

`configs_det.use_labels_as_objects=True`

which results in precision and recall values as 1. This is shown in the following image:





## Summary of Lidar based 3D Object Detection

From the project, it is understandable that for a stabilized tracking, lidar should be used. The conversion of range data to point cloud through spatial volumes, or points (or CNN networks) are important for further analysis. The usage of resnet/darknet and YOLO to convert these high dimensional point cloud representations to object detections through bounding boxes is essential for 3D object detection. Evaluating the performance with help of maximal IOU mapping, mAP, and representing the precision/recall of the bounding boxes are essential to understand the effectiveness of Lidar based detection.