This is a template submission for the final project: Sensor fusion and object tracking.

Sensor Fusion and Object Tracking

We have used the Waymo dataset real-world data and used kalman filter over time to track the multi-object by sensor fusion.

- Before implementation kalman filter , it need to configure fpn_resnet model under objdet_detect.py
- EKF is implemented including appropriate system matrix F and process noise Q for constant velocity motion model under method filter.py to obtain simple signle target scenerio with only lidar and RMSE value which help to track single object.
- Under trackmanagement.py,initializing,scoring ,updating and deleting from visible range of is perform and obtain RMSE value that help to manage track of objects.
- Inside STEP3, Associate measurements to tracks with nearest neighbor associationwith RMSE plots.
- STEP4, combining sensor (lidar) and camera to measure tracks of vehicles or objects.

The project can be run by running

python loop_over_dataset.py

All training/inference is done on udacity workspace.

Step-1: Tracking

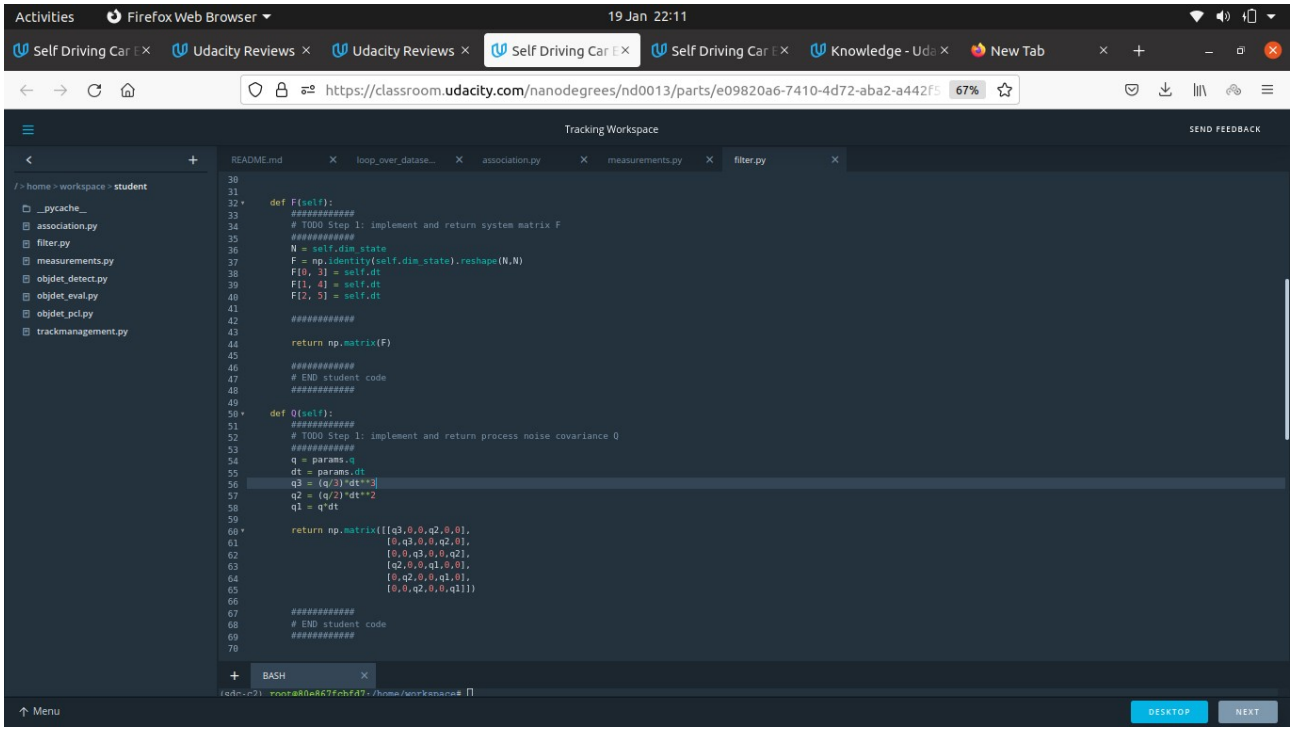
In this we are first previewing implement an EKF to track a single real-world target with lidar measurement input over time including appropriate system matrix F and process noise Q for constant velocity motion model.

- Implement Extended Kalman Filter applied to a simple single-target scenario with lidar only.
- Prediction and updation of state x
- Plot RMSE values in graph

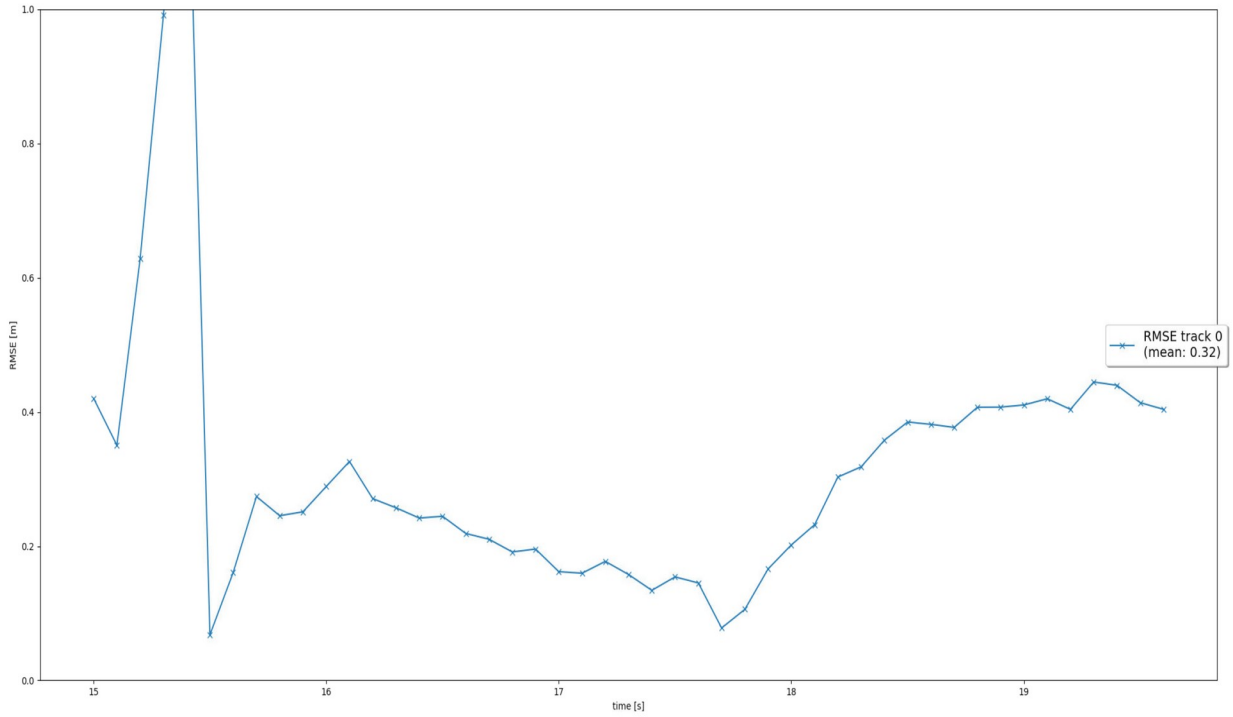
The changes are made in 'loop_over_dataset.py'

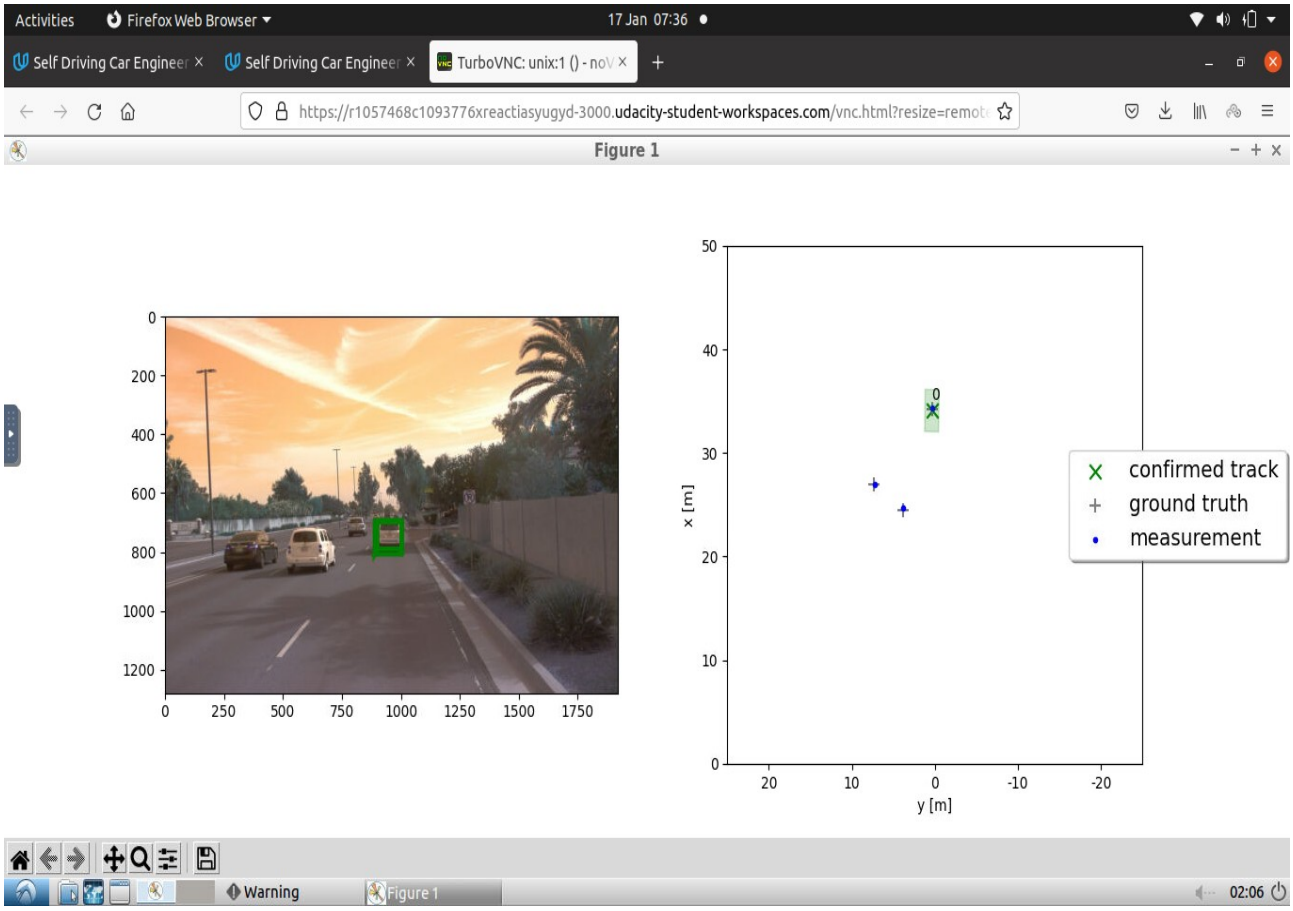
```
79
80 ## Selective execution and visualization
81 exec_detection = [] #'bev_from_pcl', 'detect_objects', 'validate_object_labels', 'measure_detection_performance'] # options are 'bev_
   'validate_object_labels', 'measure_detection_performance'; options not in the list will be loaded from file
82 exec_tracking = ['perform_tracking'] # options are 'perform_tracking'
83 exec_visualization = ['show_tracks'] # options are 'show_range_image', 'show_bev', 'show_pcl', 'show_labels_in_image', 'show_objects_
   'show_objects_in_bev_labels_in_camera', 'show_tracks', 'show_detection_performance', 'make_tracking_movie'
84 exec_list = make_exec_list(exec_detection, exec_tracking, exec_visualization)
85 vis_pause_time = 0 # set pause time between frames in ms (0 = stop between frames until key is pressed)
86
```

The changes are made in "filter.py"



The output of EKF sample(step1):





By seeing above figure, we have concluded fro STEP1,Tracking 3D object with a constant velocity model including height estimation, so F and Q will be 6D matrices and has implemented in above code section. This has provided RMSE graph with value of 0.21 which is less than 0.35.

Step-2: Track Management

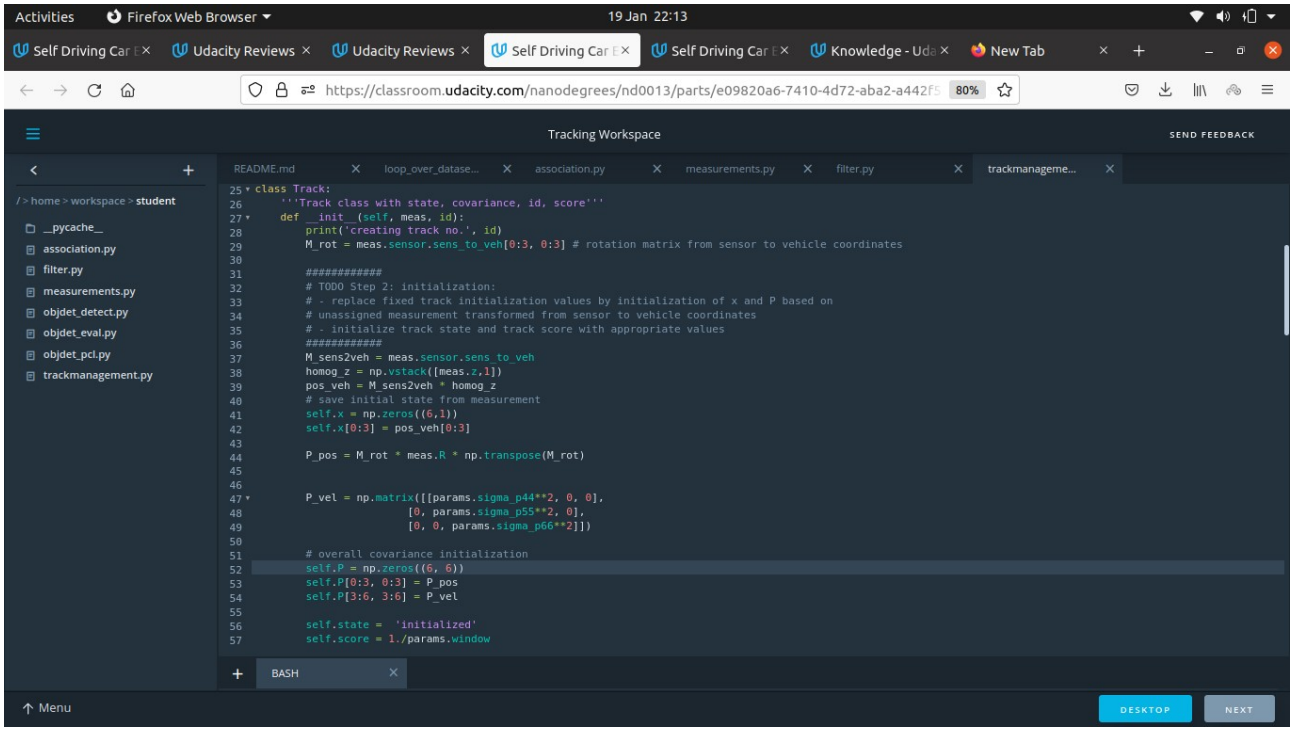
In this case, we are:

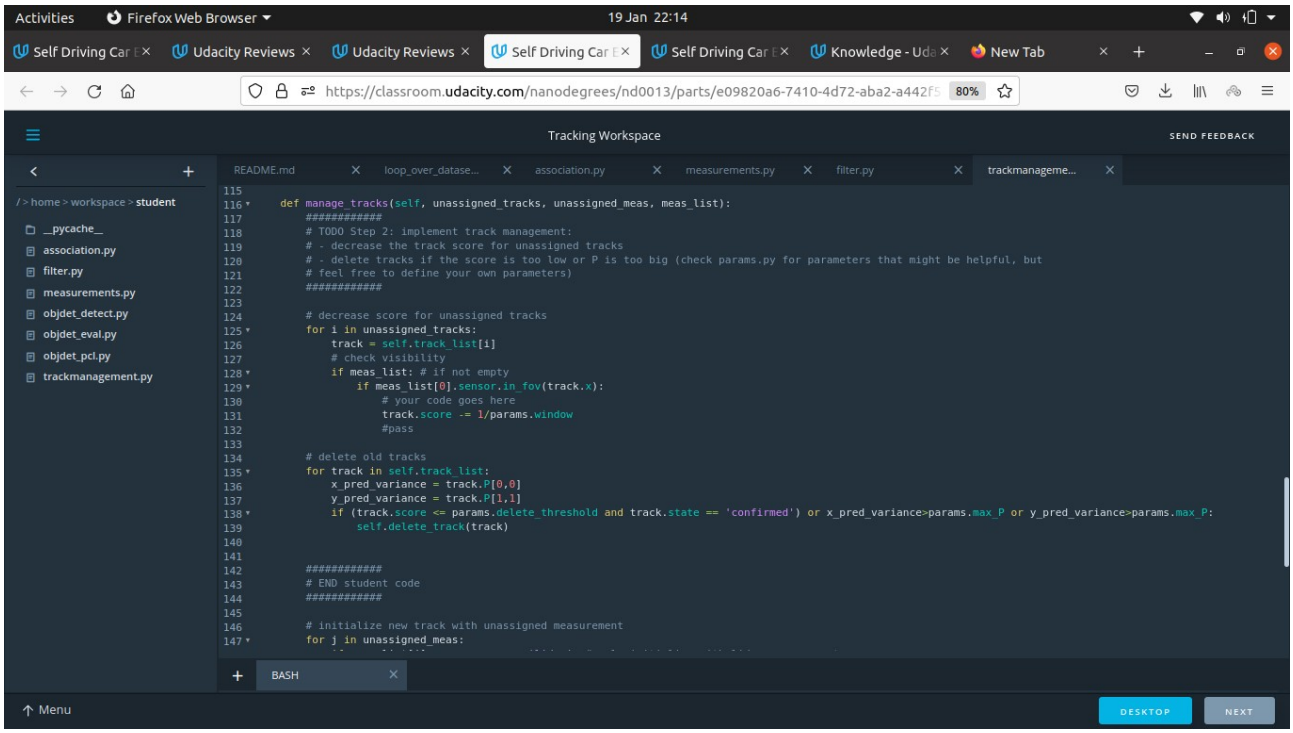
- Track initialization from unassigned measure, is implemented
- Track state and score are defined and implemented
- Old tracks are deleted for not updated tracks
- Plotation of RMSE

The changes are in the 'loop_over_dataset.py'

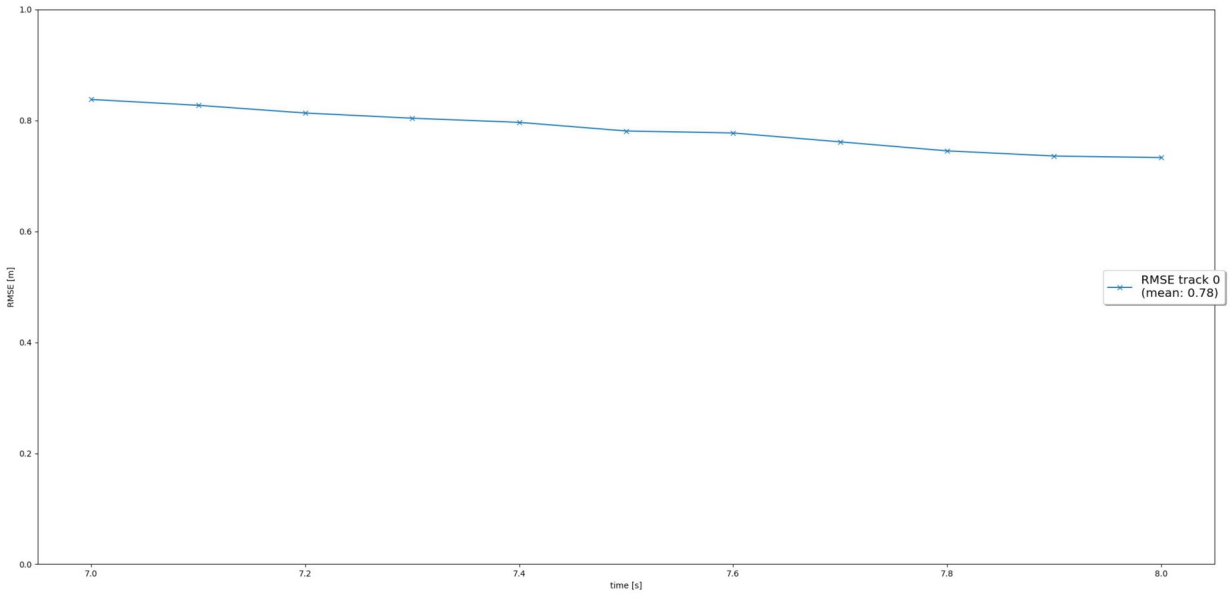
```
79
80  ## Selective execution and visualization
81  exec_detection = [] #'bev_from_pcl', 'detect_objects', 'validate_object_labels', 'measure_detection_performance'] # options are 'bev_
    'validate_object_labels', 'measure_detection_performance'; options not in the list will be loaded from file
82  exec_tracking = ['perform_tracking'] # options are 'perform_tracking'
83  exec_visualization = ['show_tracks'] # options are 'show_range_image', 'show_bev', 'show_pcl', 'show_labels_in_image', 'show_objects_
    'show_objects_in_bev_labels_in_camera', 'show_tracks', 'show_detection_performance', 'make_tracking_movie'
84  exec_list = make_exec_list(exec_detection, exec_tracking, exec_visualization)
85  vis_pause_time = 0 # set pause time between frames in ms (0 = stop between frames until key is pressed)
86
```

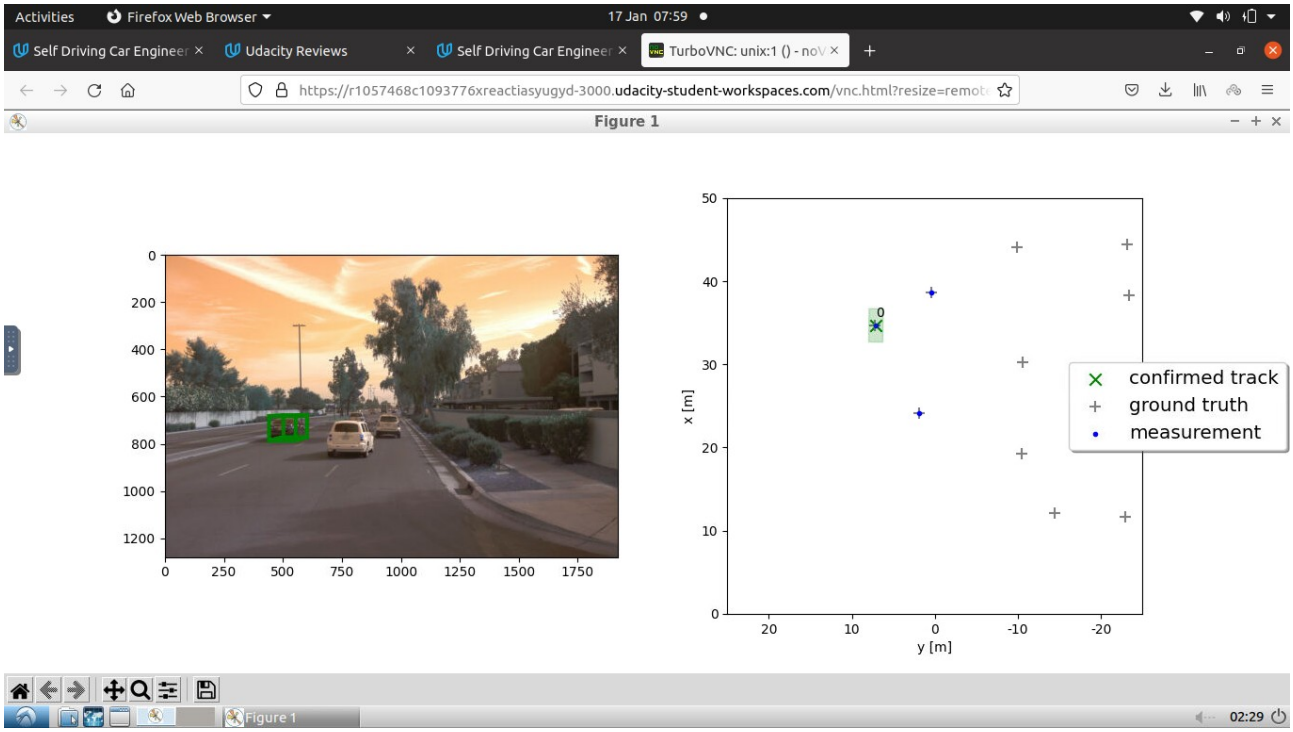
The changes are also in the "trackmanagement.py"

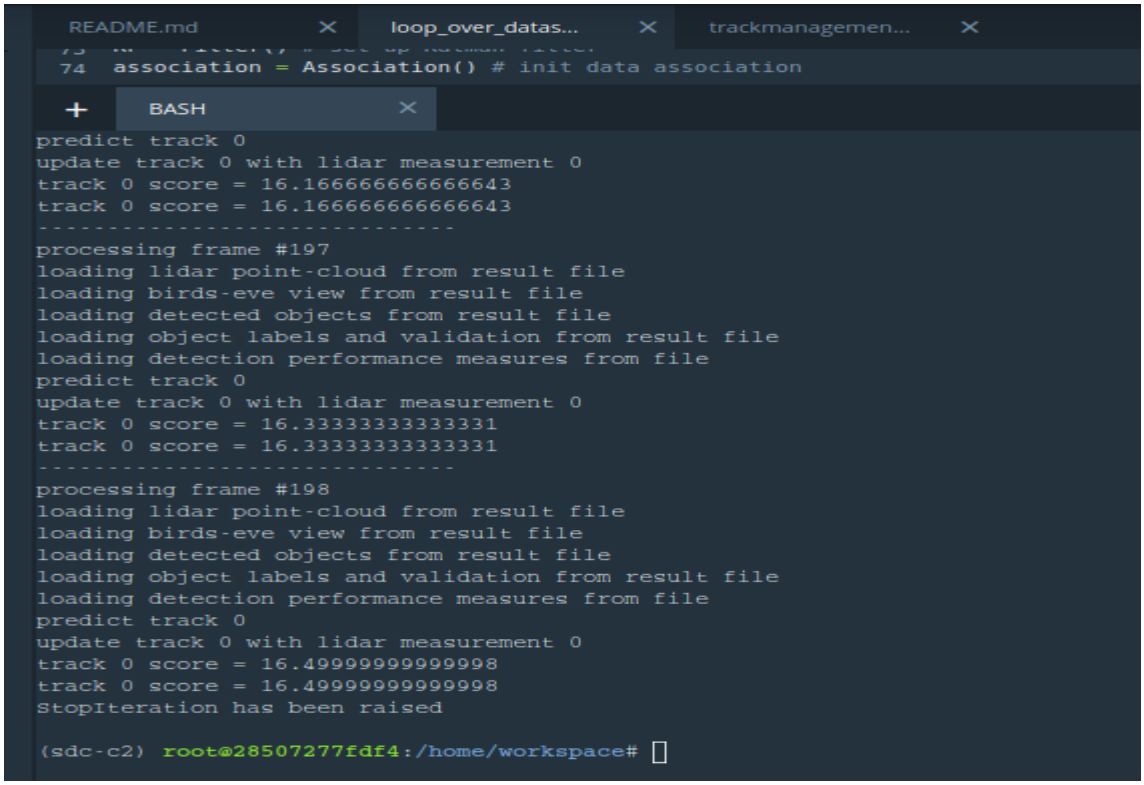




A sample preview of the Trackmanagement:







Step-3: Data Association

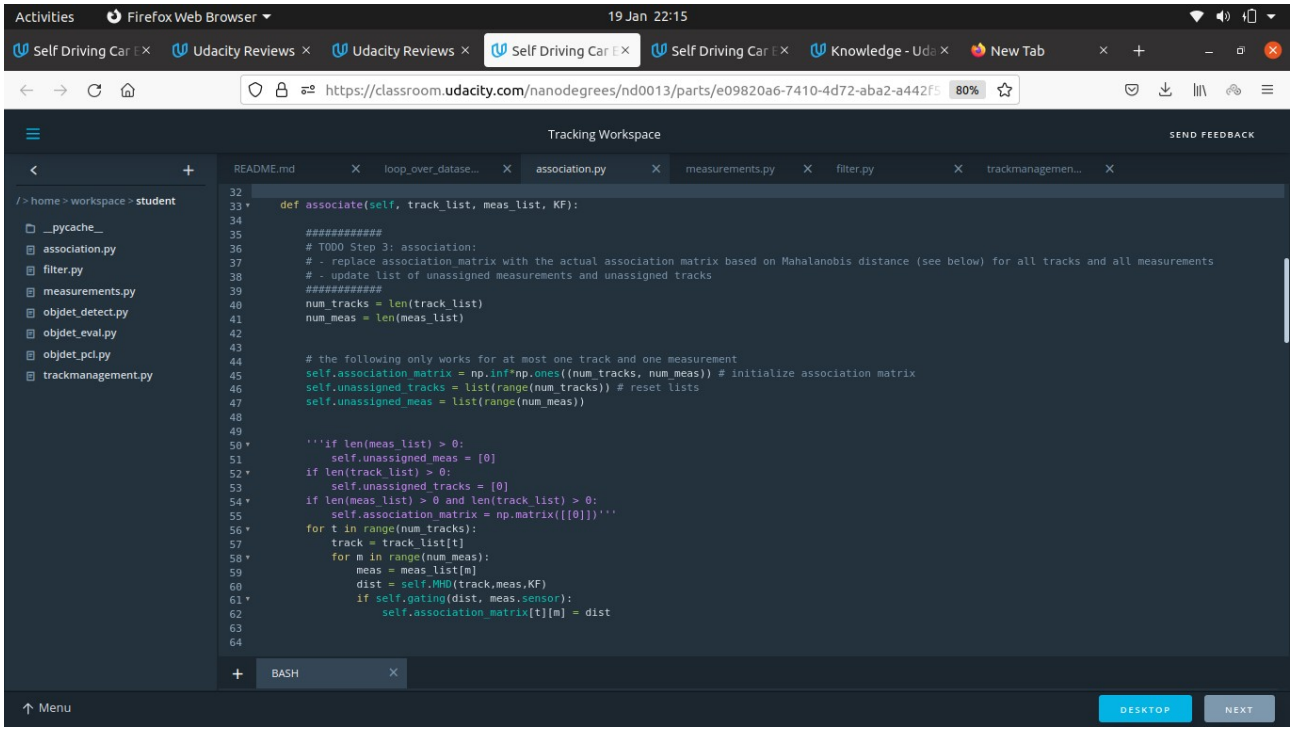
Here, particularly implementing a single nearest neighbor data association to associate measurements to tracks where output shows that each measurement is used at most once and each track is updated at most once. The visualization has shown that there are no confirmed “ghost tracks” that do not exist in reality. There may be initialized or tentative “ghost tracks” as long as they are deleted after several frames

- Nearest neighbor data association including association matrix is implemented with returns method for nearest track and measurement.
- Gating method with chi-square-distribution is implemented to reduce complexity.
- RMSE plot

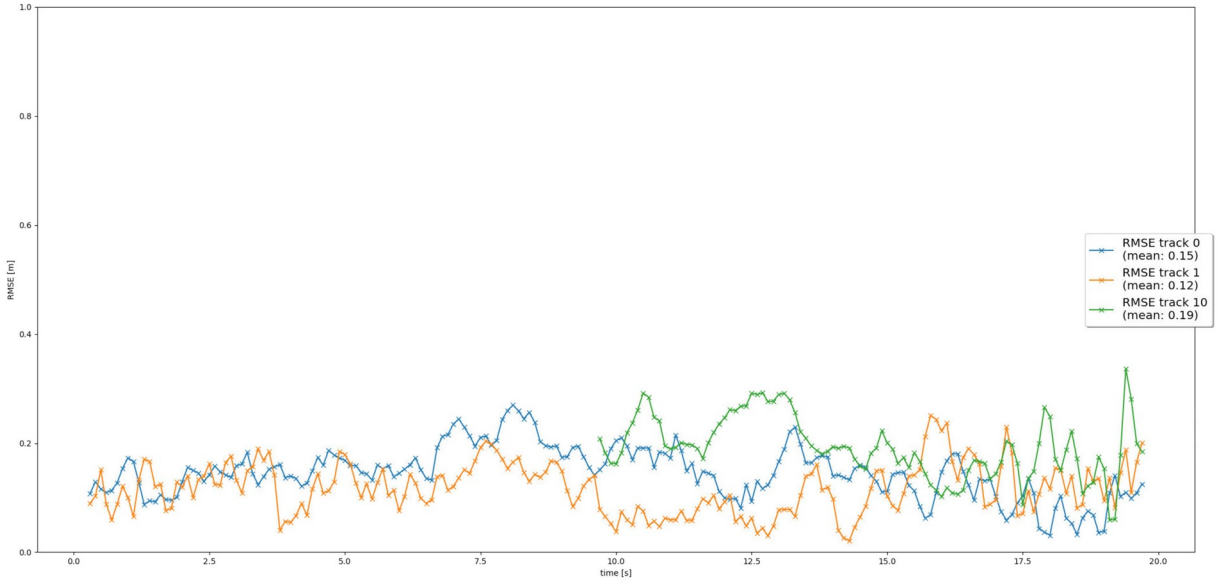
The changes are in "loop_over_dataset.py"

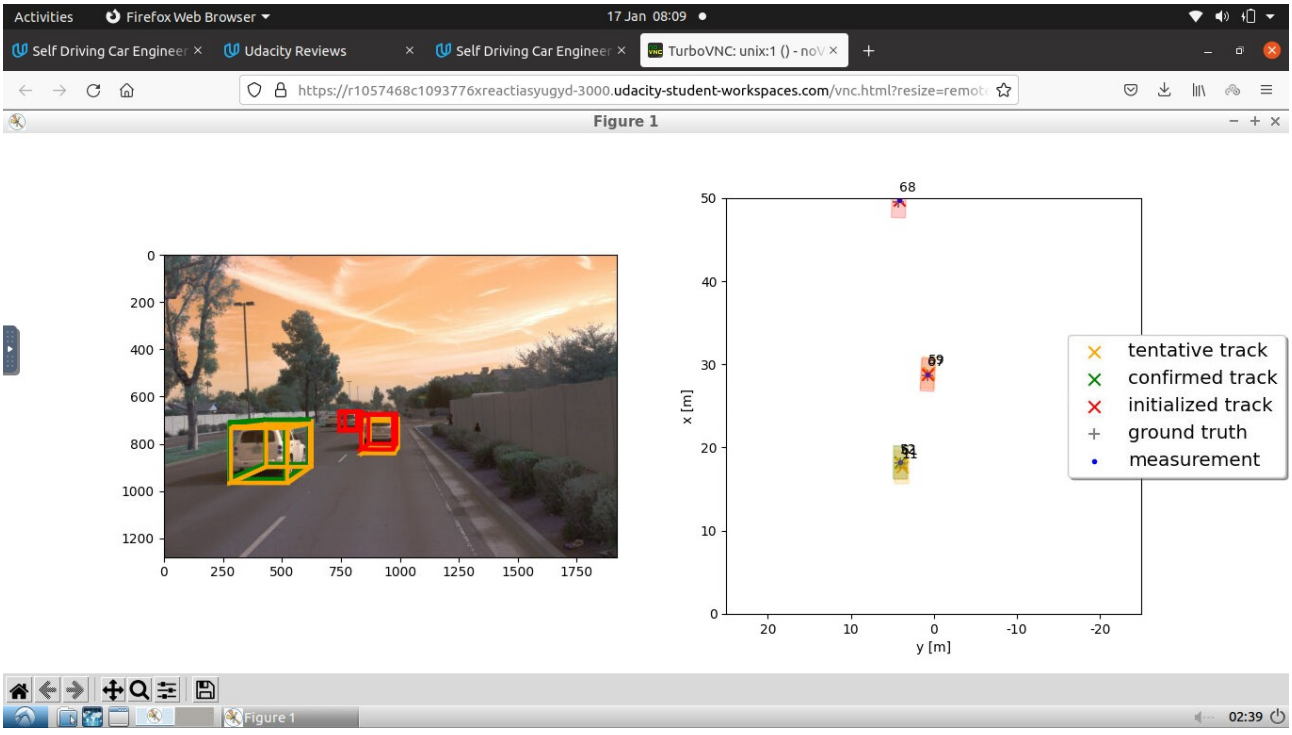
```
79
80  ## Selective execution and visualization
81  exec_detection = [] #'bev_from_pcl', 'detect_objects', 'validate_object_labels', 'measure_detection_performance'] # options are 'bev_
    'validate_object_labels', 'measure_detection_performance'; options not in the list will be loaded from file
82  exec_tracking = ['perform_tracking'] # options are 'perform_tracking'
83  exec_visualization = ['show_tracks'] # options are 'show_range_image', 'show_bev', 'show_pcl', 'show_labels_in_image', 'show_objects_
    'show_objects_in_bev_labels_in_camera', 'show_tracks', 'show_detection_performance', 'make_tracking_movie'
84  exec_list = make_exec_list(exec_detection, exec_tracking, exec_visualization)
85  vis_pause_time = 0 # set pause time between frames in ms (0 = stop between frames until key is pressed)
86
```

The changes are inside the "association.py" file:

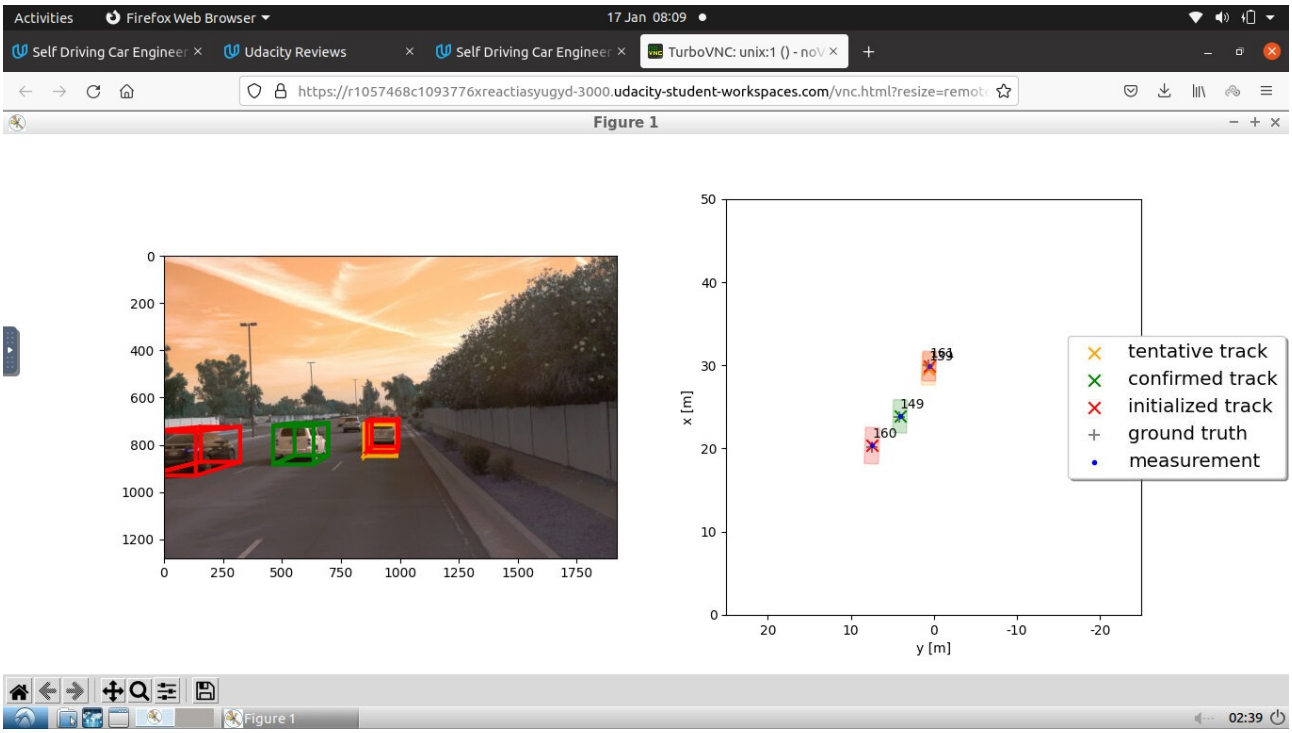


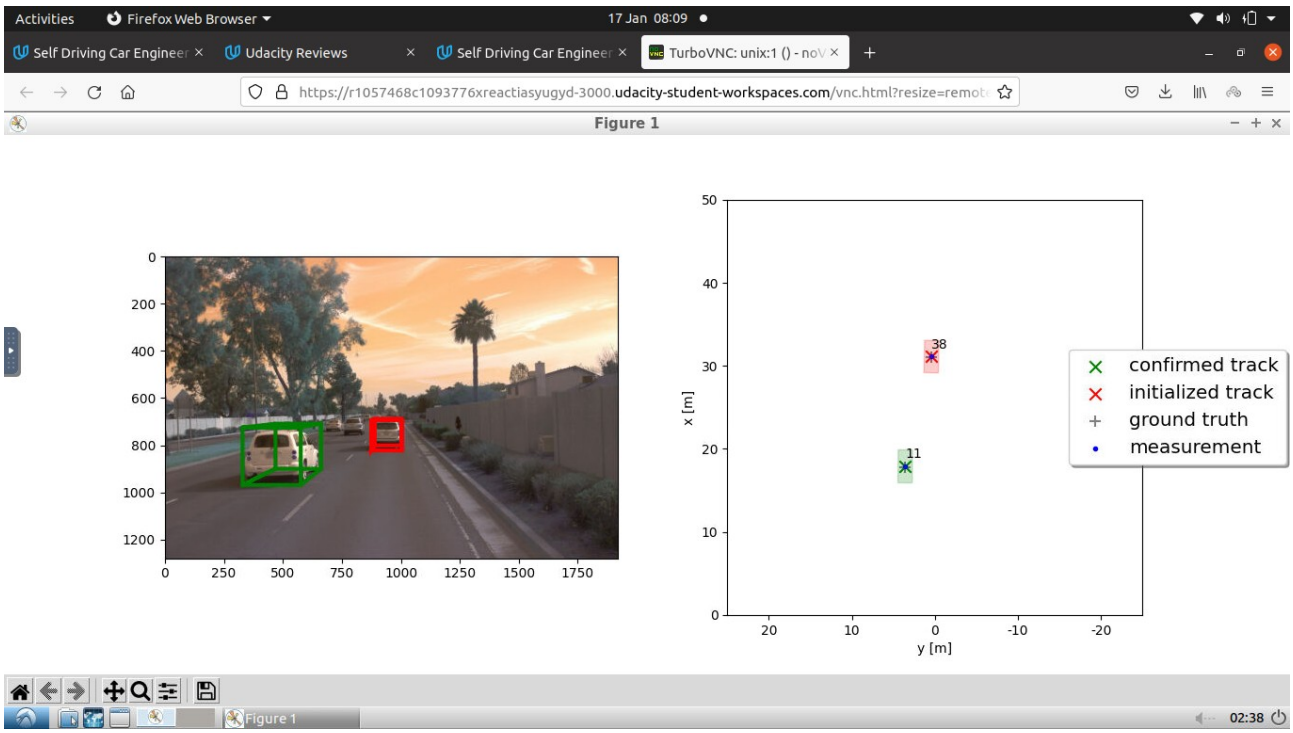
A sample preview of the association measurement:

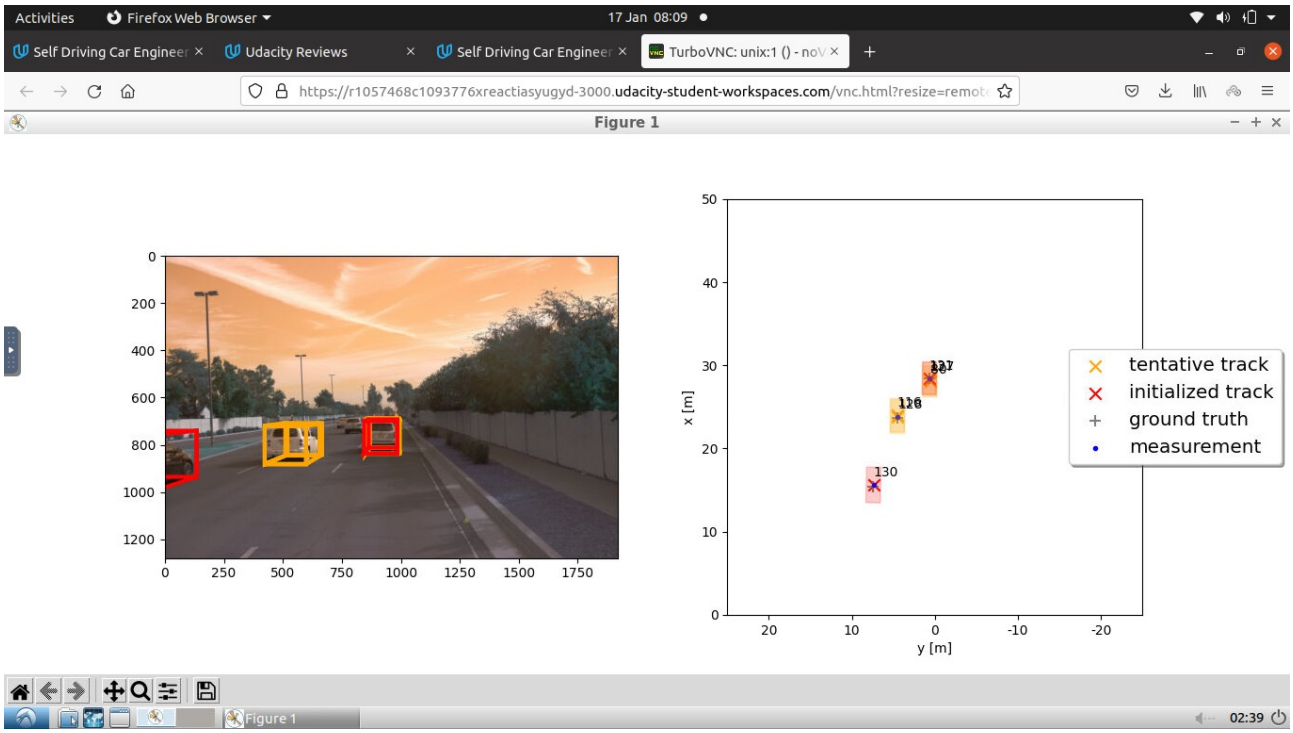


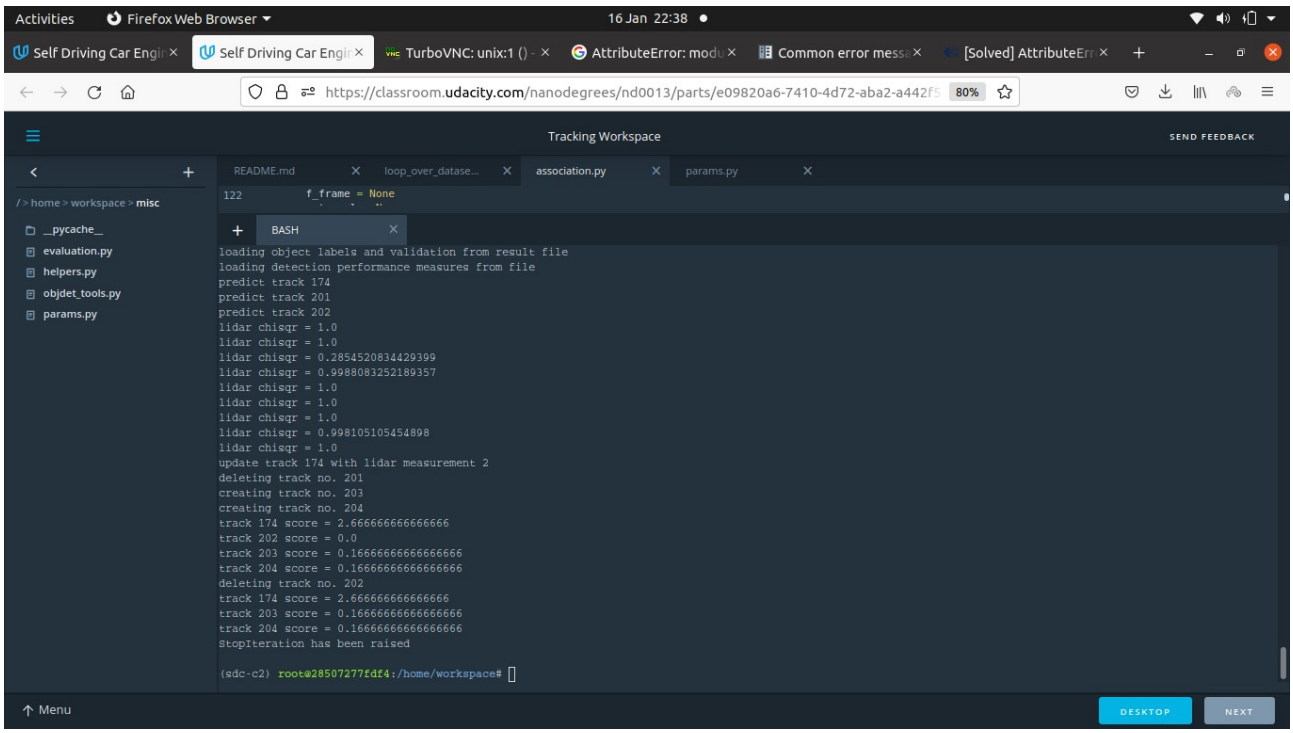



Activities
Firefox Web Browser
17 Jan 08:09
Self Driving Car Engineer
Udacity Reviews
Self Driving Car Engineer
TurboVNC: unix:1 () - noV
https://r1057468c1093776xreactiasyugyd-3000.udacity-student-workspaces.com/vnc.html?resize=remot
Figure 1
tentative track
confirmed track
initialized track
ground truth
measurement
x [m]
y [m]
78
71
Figure 1
02:39


Activities
Firefox Web Browser
17 Jan 08:09
Self Driving Car Engineer
Udacity Reviews
Self Driving Car Engineer
TurboVNC: unix:1 () - noV
https://r1057468c1093776xreactiasyugyd-3000.udacity-student-workspaces.com/vnc.html?resize=remot
Figure 1
tentative track
confirmed track
initialized track
ground truth
measurement
x [m]
y [m]
161
159
149
160
Figure 1
02:39


Activities
Firefox Web Browser
17 Jan 08:09
Self Driving Car Engineer
Udacity Reviews
Self Driving Car Engineer
TurboVNC: unix:1 () - noV
https://r1057468c1093776xreactiasyugyd-3000.udacity-student-workspaces.com/vnc.html?resize=remot
Figure 1
0
200
400
600
800
1000
1200
0
250
500
750
1000
1250
1500
1750
50
40
30
20
10
0
x [m]
20
10
0
-10
-20
y [m]
38
11
confirmed track
initialized track
ground truth
measurement
Figure 1
02:38


Activities
Firefox Web Browser
17 Jan 08:09
Self Driving Car Engineer
Udacity Reviews
Self Driving Car Engineer
TurboVNC: unix:1 () - noV
https://r1057468c1093776xreactiasyugyd-3000.udacity-student-workspaces.com/vnc.html?resize=remot
Figure 1
0
200
400
600
800
1000
1200
0
250
500
750
1000
1250
1500
1750
50
40
30
20
10
0
x [m]
20
10
0
-10
-20
y [m]
130
tentative track
initialized track
ground truth
measurement
Figure 1
02:39

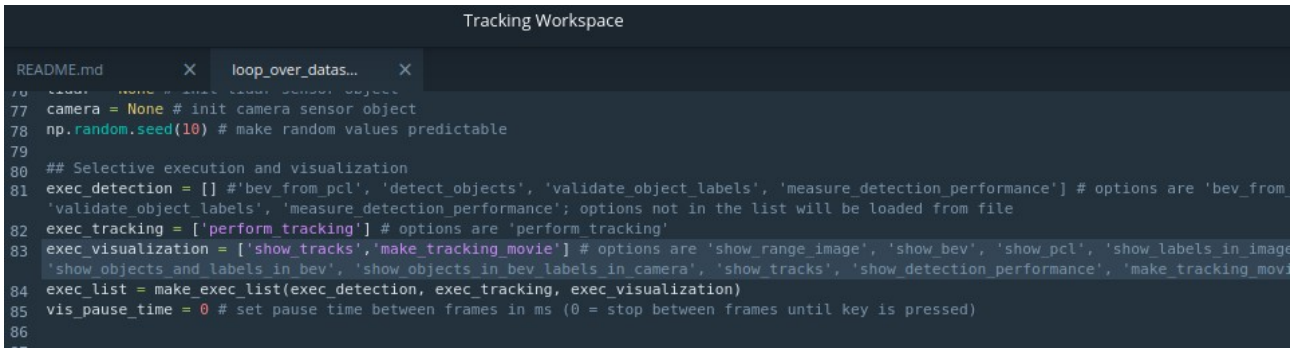


Step-4: Sensor Fusion

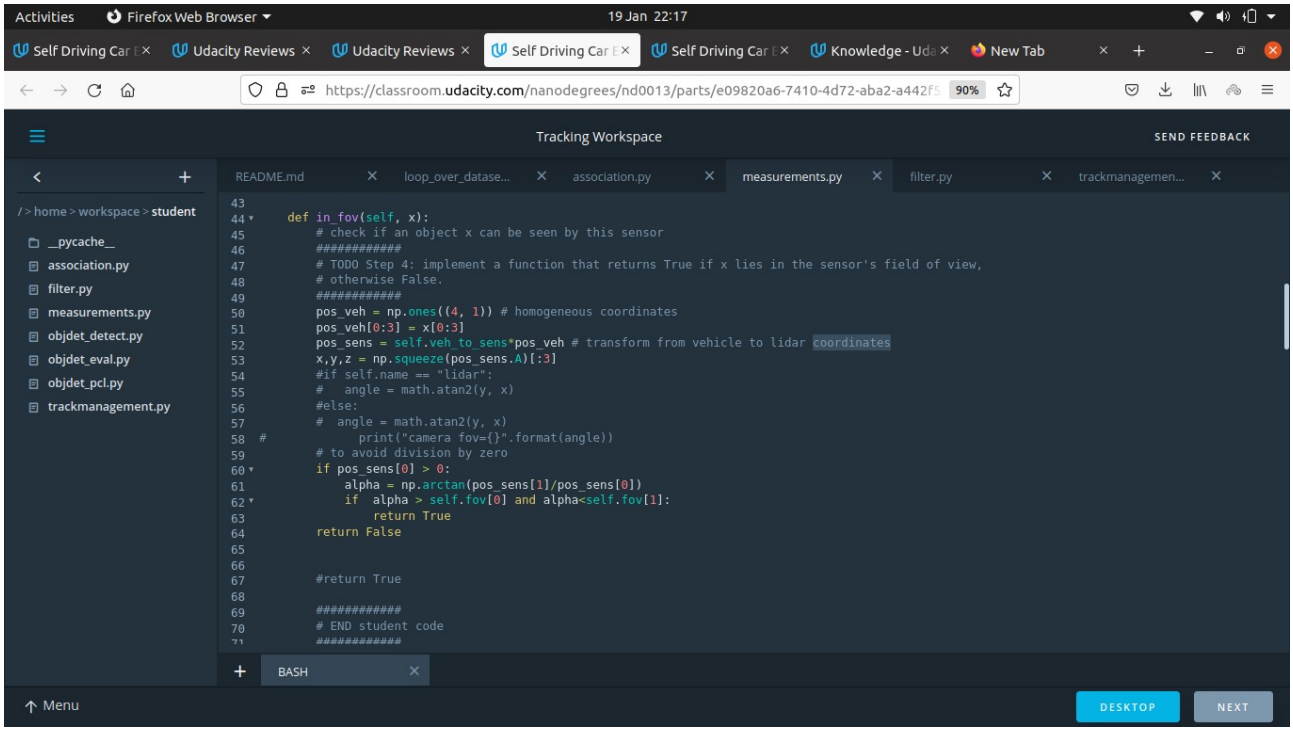
In this step, implementation the nonlinear camera measurement model and then finally combining the sensor fusion module for completing camera-lidar fusion where output shows lidar updates followed by camera updates.

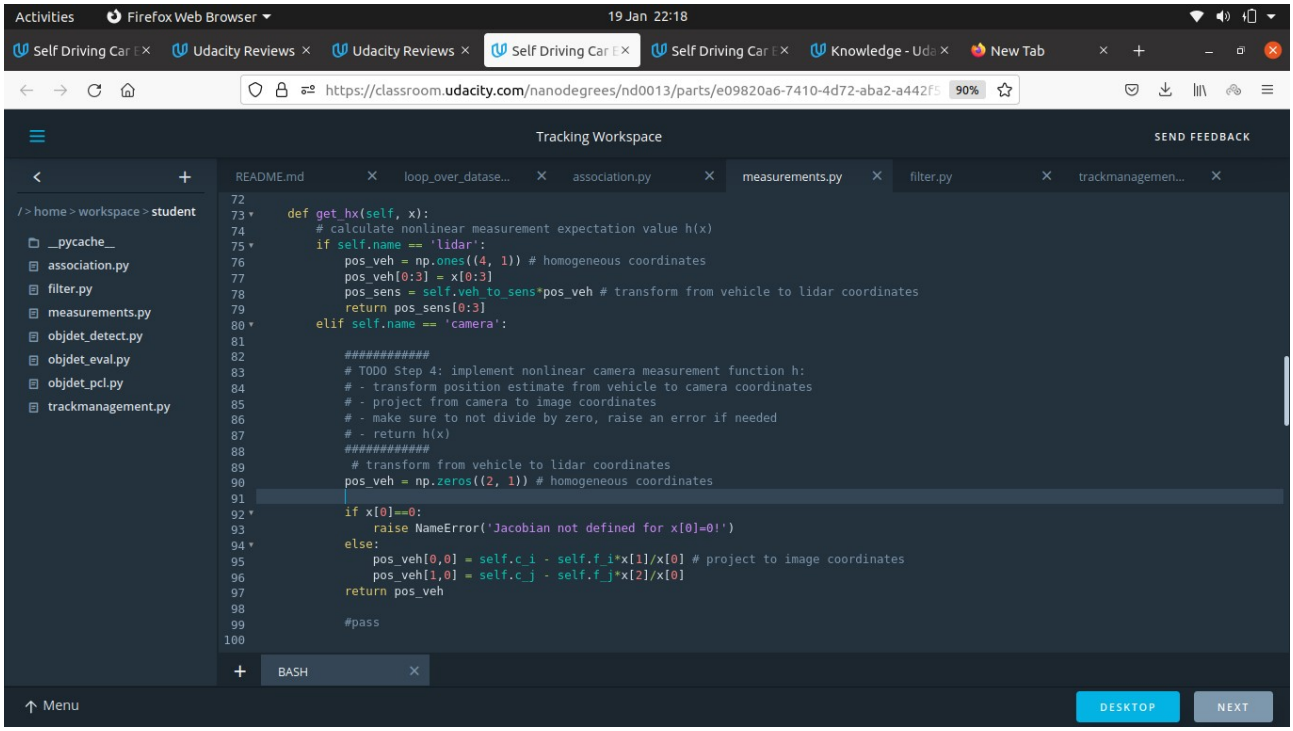
- Camera measurements including appropriate covariance matrix R are implemented.
- Non linear camera measurement model h(x) is implemented.
- A method checked whether an object in range of the camera or outside the field of view is implemented.
- Produce output video using flags ‘make-track_movies’

The changes in the code are:

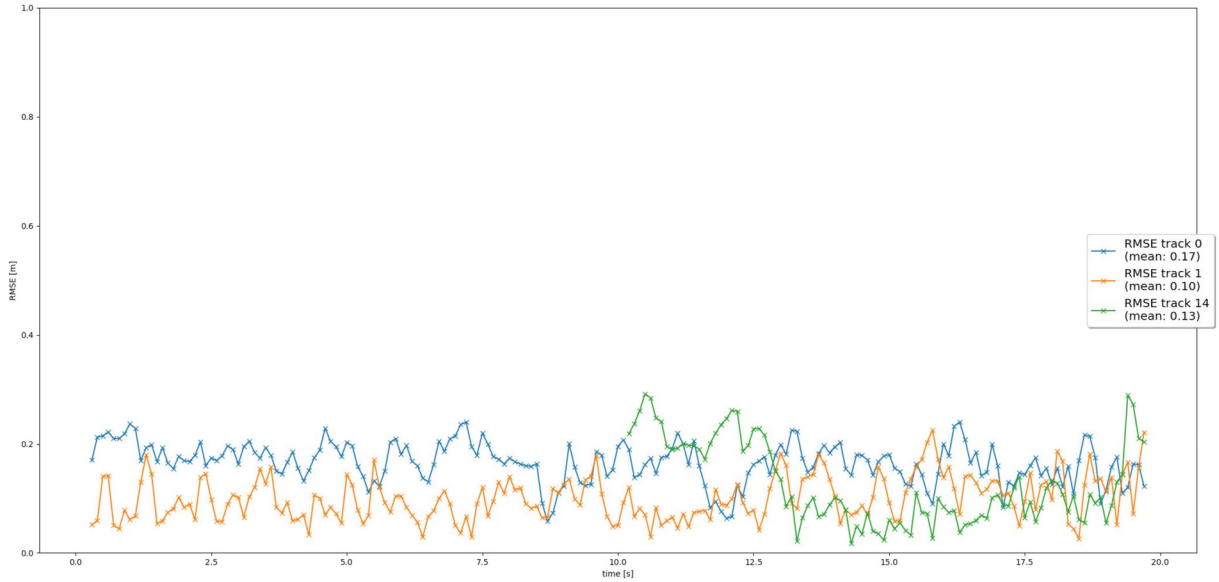


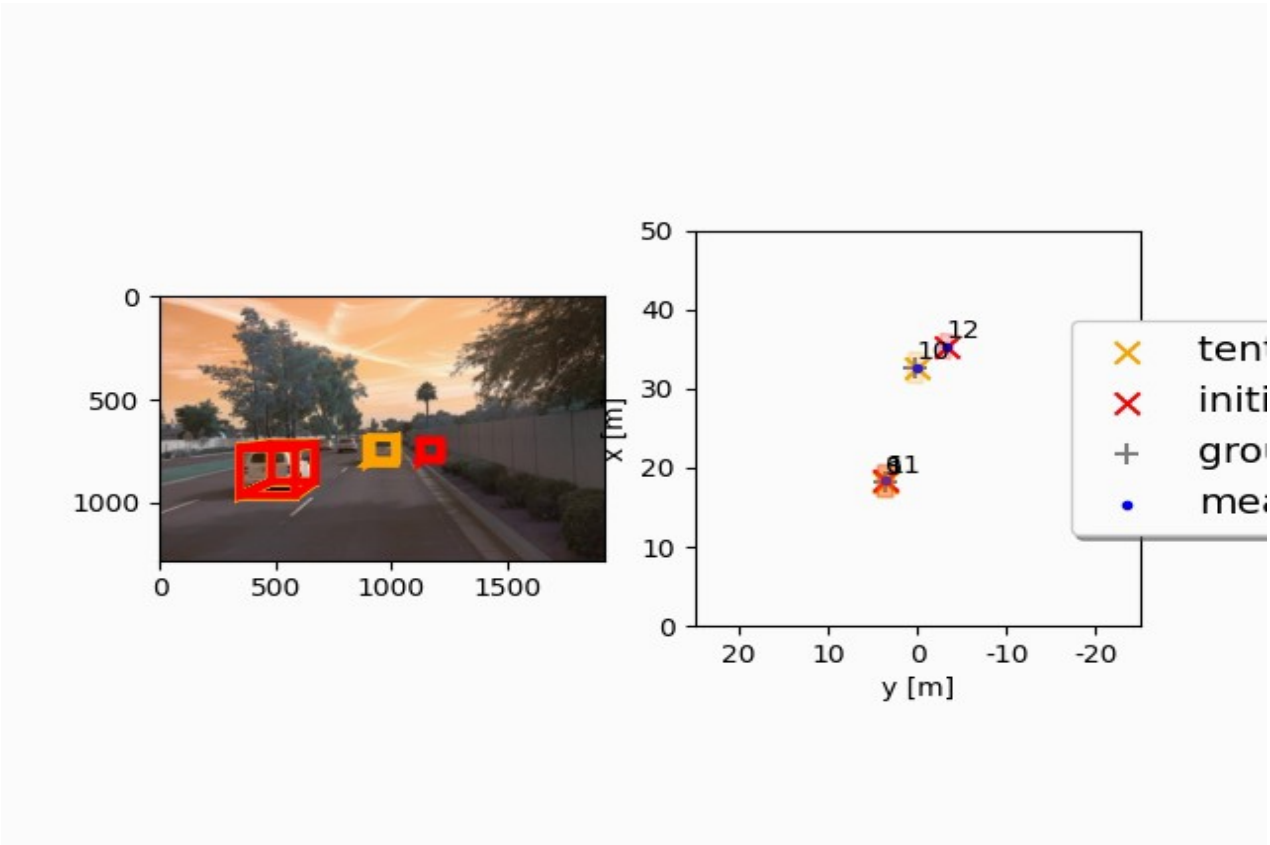
The changes for "measurements.py" :





Results of Measurements.py:





This is video file(click on this image, go to left bottom corner,play button option will seen.)

Summary of Sensor Fusion and Object Tracking

From the project, it is understandable that for a good multi-tracking, lidar and camera should be used . In order to make more efficient these sensor, Extended kalman Filter is necessary for multi tracking under their field view because this algorithm provide more certainty behaviour of having next object around it by updating and predicting future tracks of position and velocity.Along this ,optimization is done through Root Mean Sqaure Error(RMSE) which is equivalent to 0.35 or less than it for better performance and we got 0.21 RMSE value.