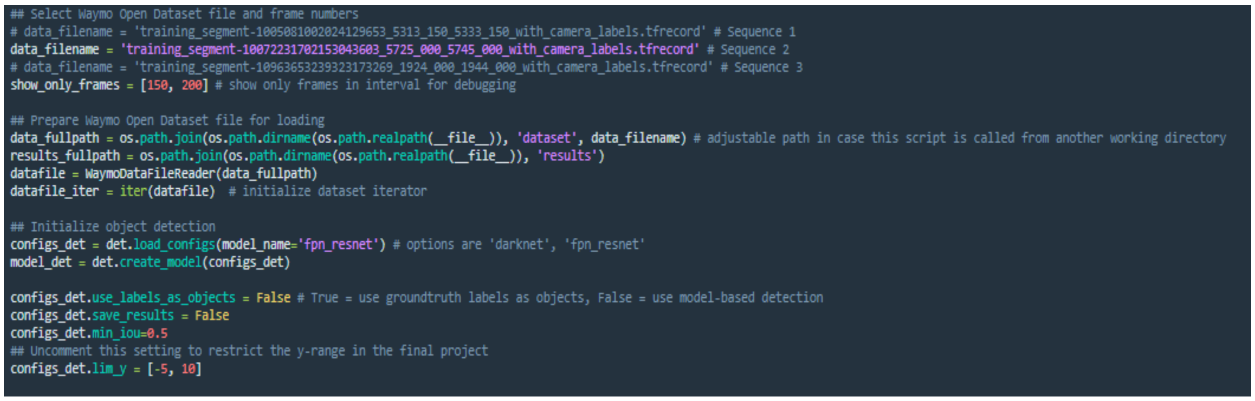
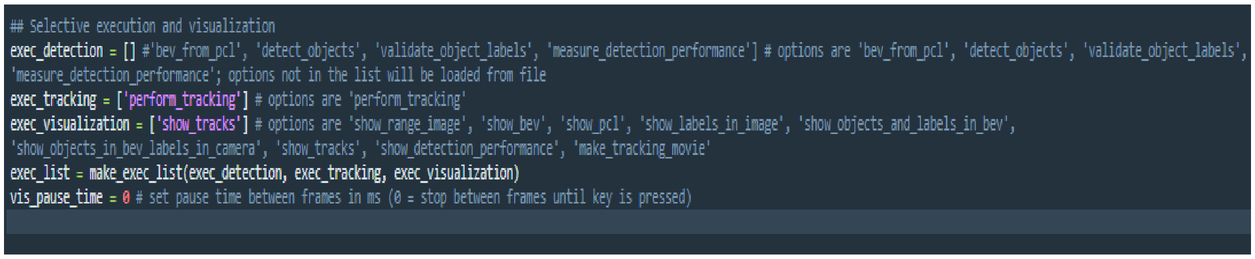
1-Track objects over time with a Kalman Filter:

- EKF is implemented including appropriate system matrix F and process noise Q for constant velocity motion model.
- EKF is applied to a simple single-target scenario with lidar only.
- The mean RMSE is 0.35 or smaller. Please upload the RMSE plot as png or pdf file.

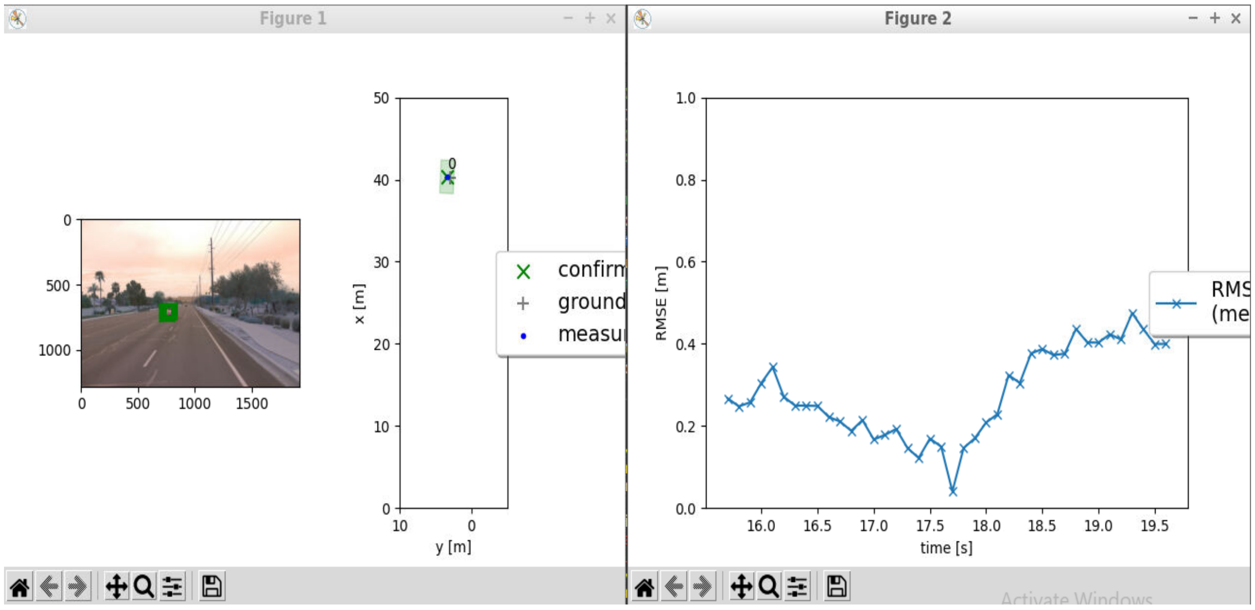


```
## Select Waymo Open Dataset file and frame numbers
# data_filename = 'training_segment-1005081002024129653_5313_150_5333_150_with_camera_labels.tfrecord' # Sequence 1
data_filename = 'training_segment-10072231702153043603_5725_000_5745_000_with_camera_labels.tfrecord' # Sequence 2
# data_filename = 'training_segment-10963653239323173269_1924_000_1944_000_with_camera_labels.tfrecord' # Sequence 3
show_only_frames = [150, 200] # show only frames in interval for debugging

## Prepare Waymo Open Dataset file for loading
data_fullpath = os.path.join(os.path.dirname(os.path.realpath(__file__)), 'dataset', data_filename) # adjustable path in case this script is called from another working directory
results_fullpath = os.path.join(os.path.dirname(os.path.realpath(__file__)), 'results')
datafile = WaymoDataFileReader(data_fullpath)
datafile_iter = iter(datafile)  # initialize dataset iterator

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

configs_det.use_labels_as_objects = False # True = use groundtruth labels as objects, False = use model-based detection
configs_det.save_results = False
configs_det.min_iou=0.5
## Uncomment this setting to restrict the y-range in the final project
configs_det.lim_y = [-5, 10]
```



```
## Selective execution and visualization
exec_detection = [] #'bev_from_pcl', 'detect_objects', 'validate_object_labels', 'measure_detection_performance'] # options are 'bev_from_pcl', 'detect_objects', 'validate_object_labels',
'measure_detection_performance'; options not in the list will be loaded from file
exec_tracking = ['perform_tracking'] # options are 'perform_tracking'
exec_visualization = ['show_tracks'] # 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'
exec_list = make_exec_list(exec_detection, exec_tracking, exec_visualization)
vis_pause_time = 0 # set pause time between frames in ms (0 = stop between frames until key is pressed)
```

Result



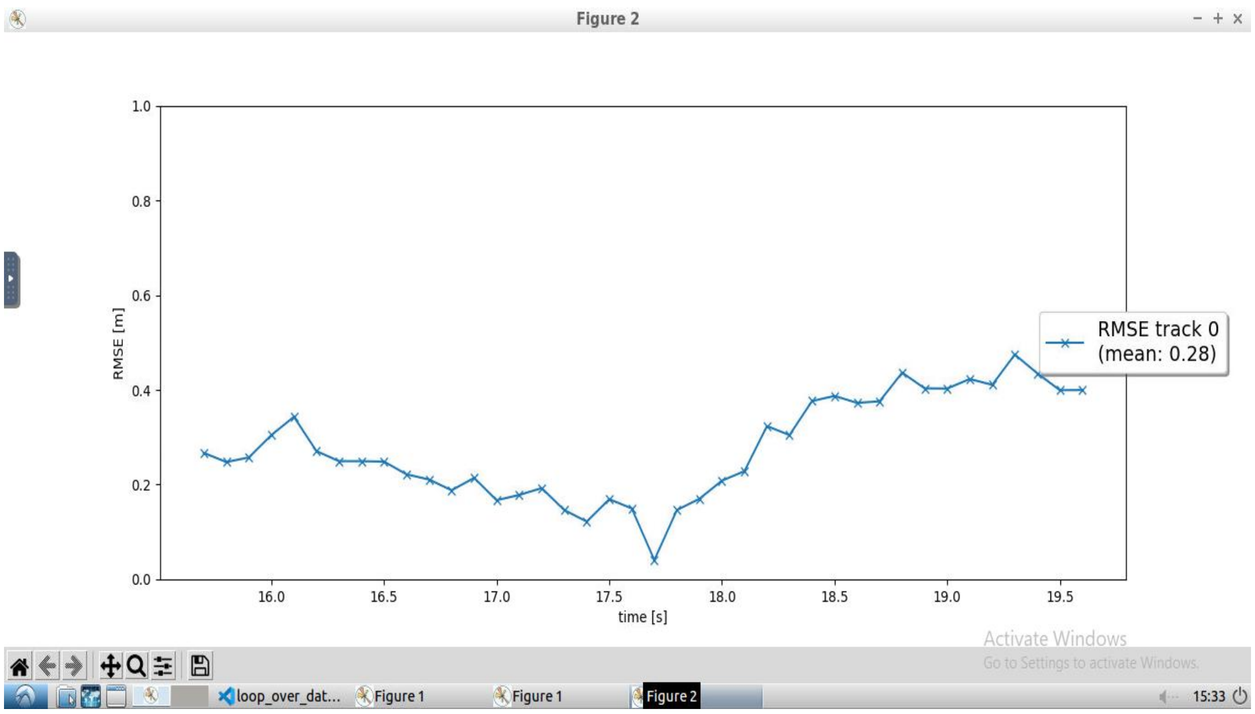


Figure 2
RMSE [m]
time [s]
1.0
0.8
0.6
0.4
0.2
0.0
16.0
16.5
17.0
17.5
18.0
18.5
19.0
19.5
RMSE track 0
(mean: 0.28)
Activate Windows
Go to Settings to activate Windows.
loop_over_dat...
Figure 1
Figure 1
Figure 2
15:33

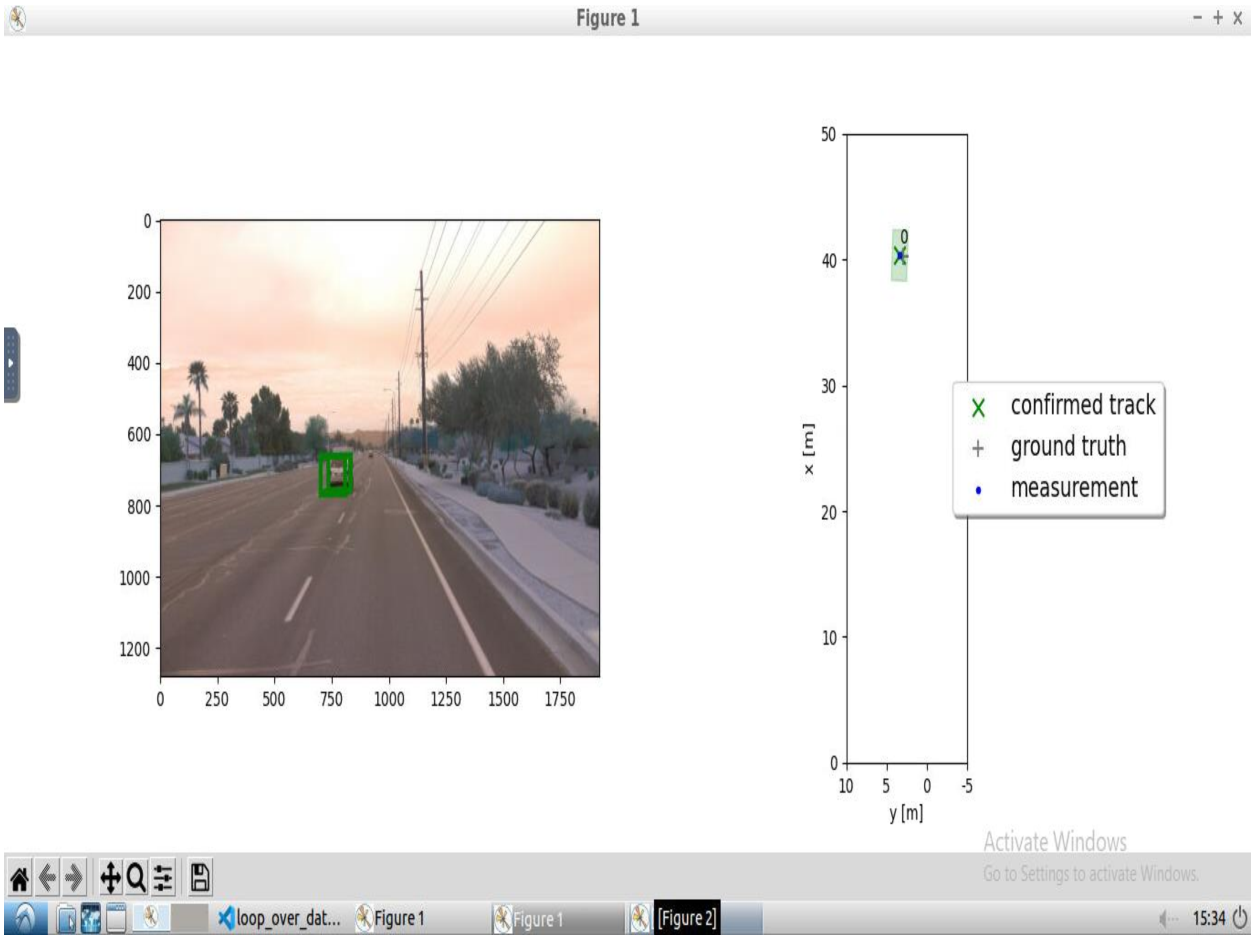
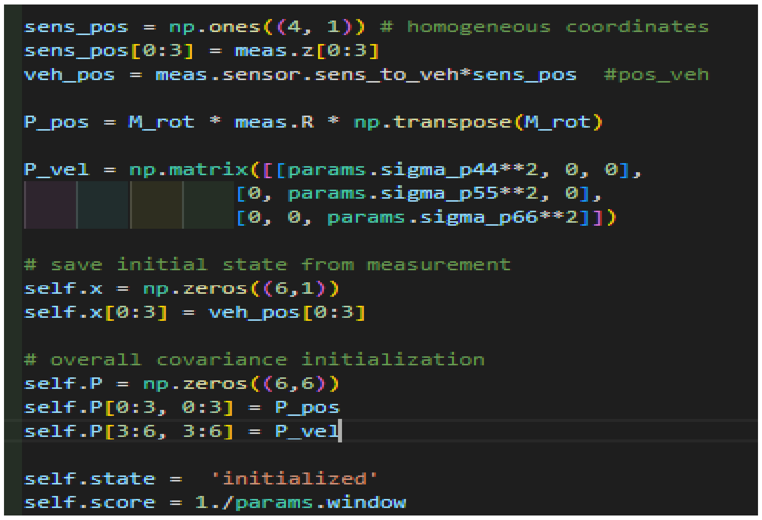


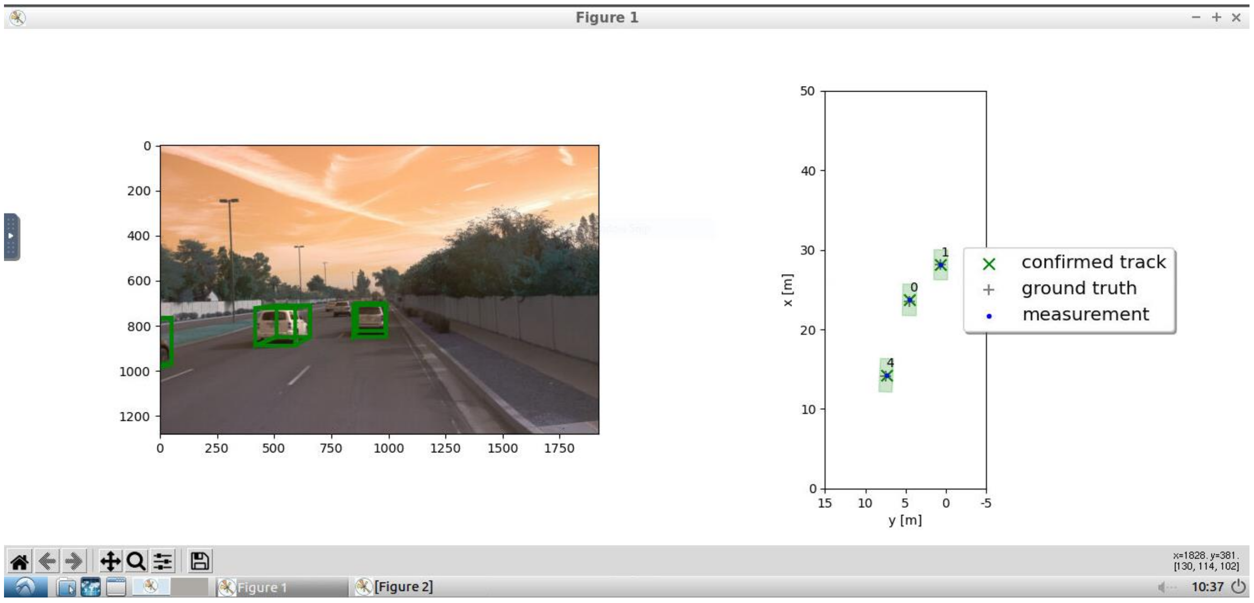
Figure 1
0
200
400
600
800
1000
1200
0
250
500
750
1000
1250
1500
1750
50
40
30
20
10
0
x [m]
10
5
0
-5
y [m]
0
confirmed track
ground truth
measurement
Activate Windows
Go to Settings to activate Windows.
loop_over_dat...
Figure 1
Figure 1
[Figure 2]
15:34

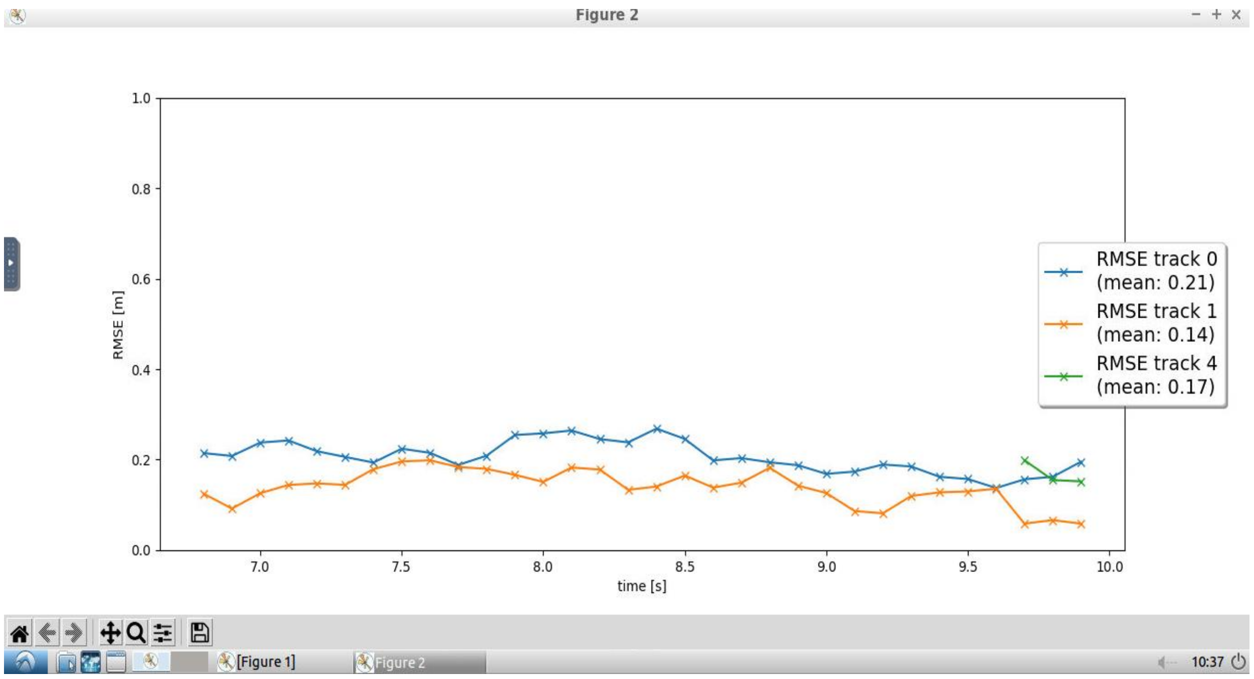
2-Initialize, update and delete tracks:

- Track initialization from unassigned measurements is implemented.
- A track score is defined and implemented.
- Track states are defined and implemented, e.g., “tentative”, “confirmed”.
- Old tracks are deleted for not updated tracks.
- The tracking works properly if you see the following results: After applying the track management to a new sequence (see instructions), the visualization shows that a new track is initialized automatically where unassigned measurements occur, the true track is confirmed quickly, and the track is deleted after it has vanished from the visible range. There is one single track without track losses in between. Please upload the RMSE plot as png or pdf file.



```
sens_pos = np.ones((4, 1)) # homogeneous coordinates
sens_pos[0:3] = meas.z[0:3]
veh_pos = meas.sensor.sens_to_veh*sens_pos  #pos_veh

P_pos = M_rot * meas.R * np.transpose(M_rot)

P_vel = np.matrix([[params.sigma_p44**2, 0, 0],
                   [0, params.sigma_p55**2, 0],
                   [0, 0, params.sigma_p66**2]])

# save initial state from measurement
self.x = np.zeros((6,1))
self.x[0:3] = veh_pos[0:3]

# overall covariance initialization
self.P = np.zeros((6,6))
self.P[0:3, 0:3] = P_pos
self.P[3:6, 3:6] = P_vel

self.state =  'initialized'
self.score = 1./params.window
```





3-Associate measurements to tracks with nearest neighbor association:

- Nearest neighbor data association including association matrix is implemented.
- A method that returns nearest track and measurement for association is implemented.
- Gating method with chi-square-distribution is implemented to reduce complexity.
- The association works properly if you see the following results: After applying the data association to a new sequence with multiple targets, multiple tracks are updated with multiple measurements. The console output shows that each measurement is used at most once and each track is updated at most once.
- The visualization should show 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. Please upload the RMSE plot as png or pdf file.

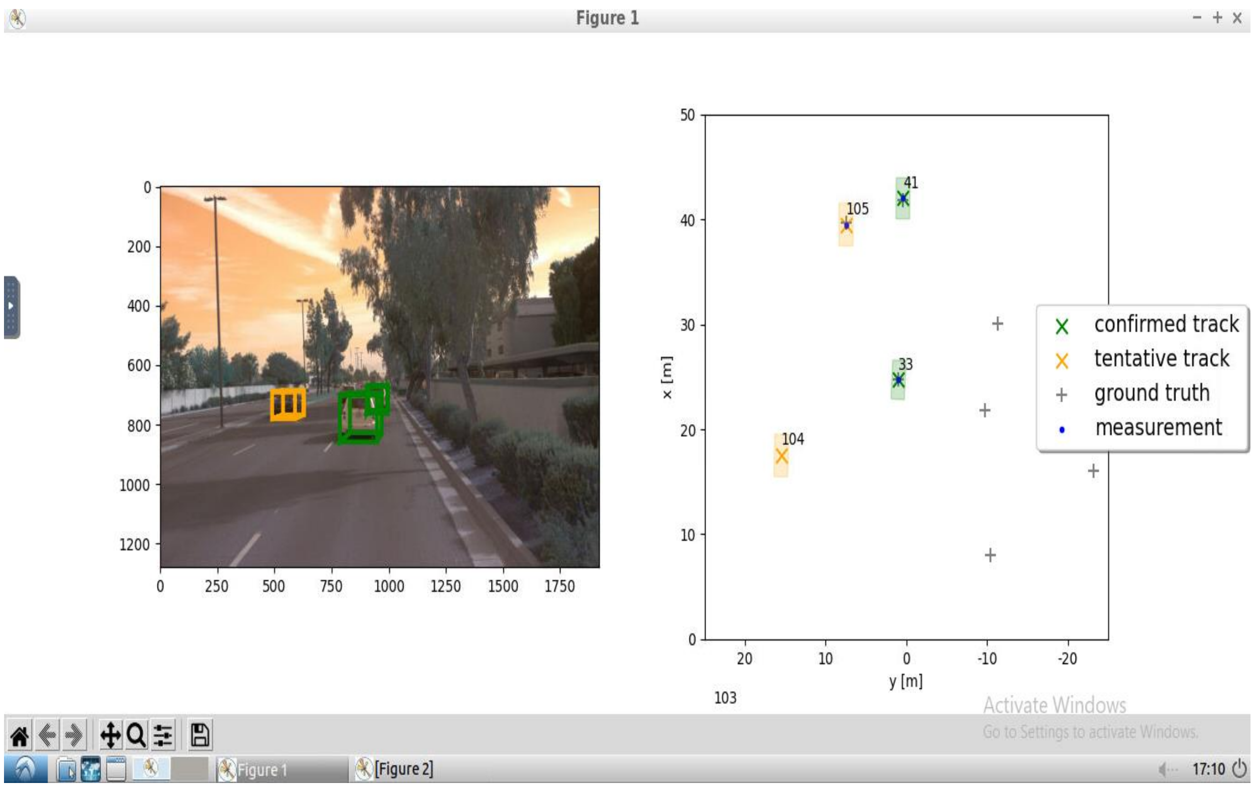


Figure 1
confirmed track
tentative track
ground truth
measurement
x [m]
y [m]
41
105
33
104
103
Activate Windows
Go to Settings to activate Windows.
Figure 1
[Figure 2]
17:10

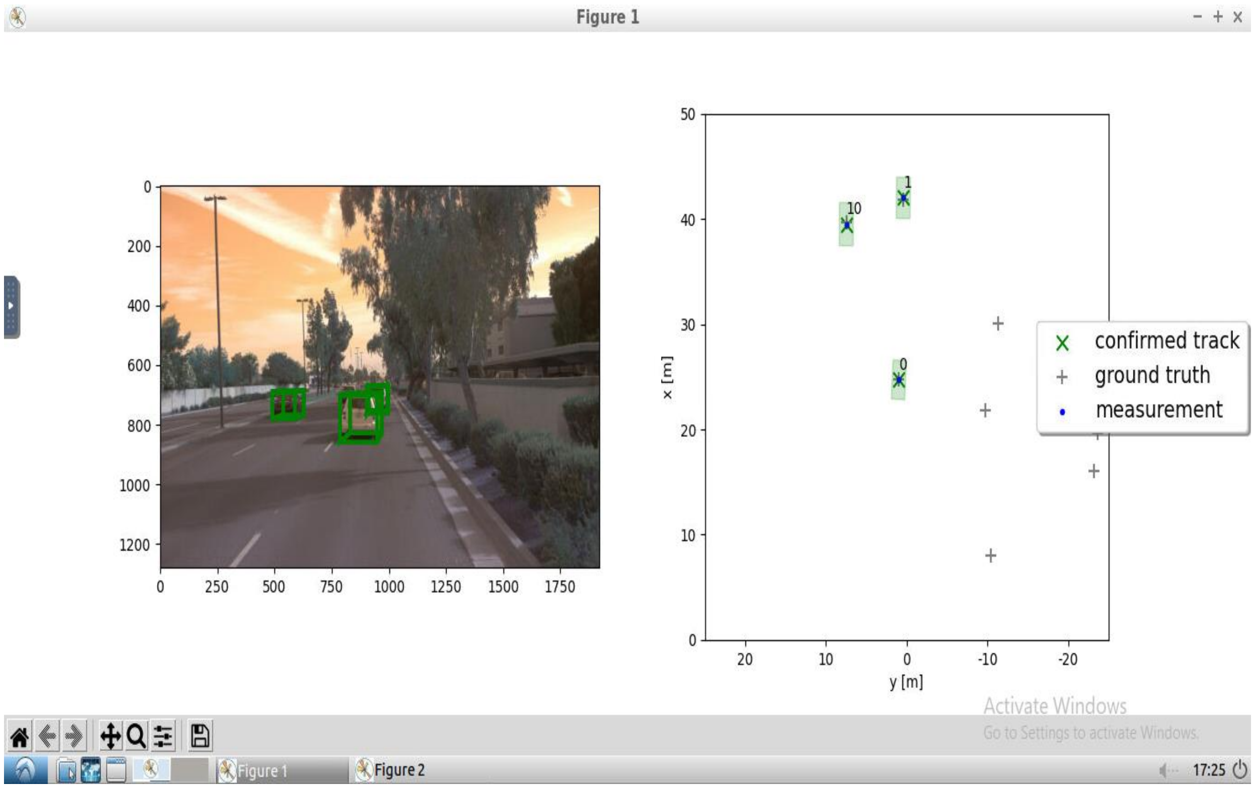
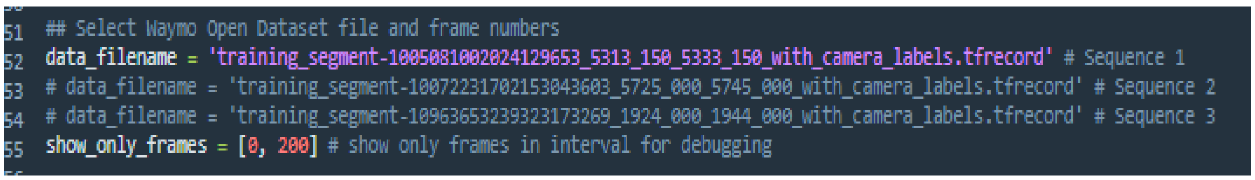
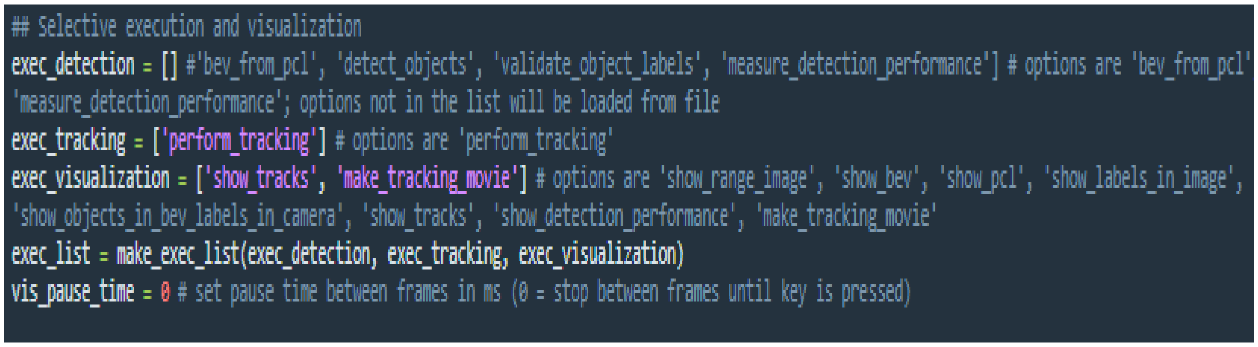


Figure 1
confirmed track
ground truth
measurement
x [m]
y [m]
1
10
0
Activate Windows
Go to Settings to activate Windows.
Figure 1
Figure 2
17:25

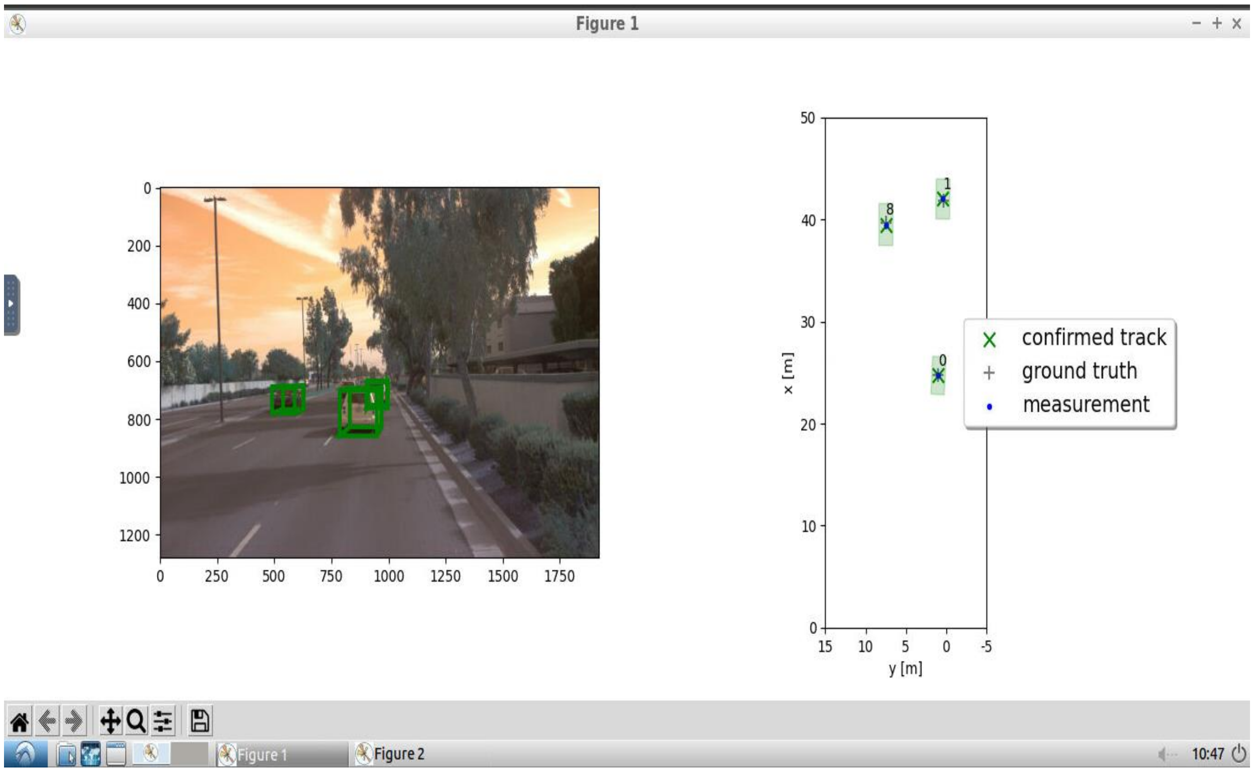
4-SWBAT fuse measurements from lidar and camera

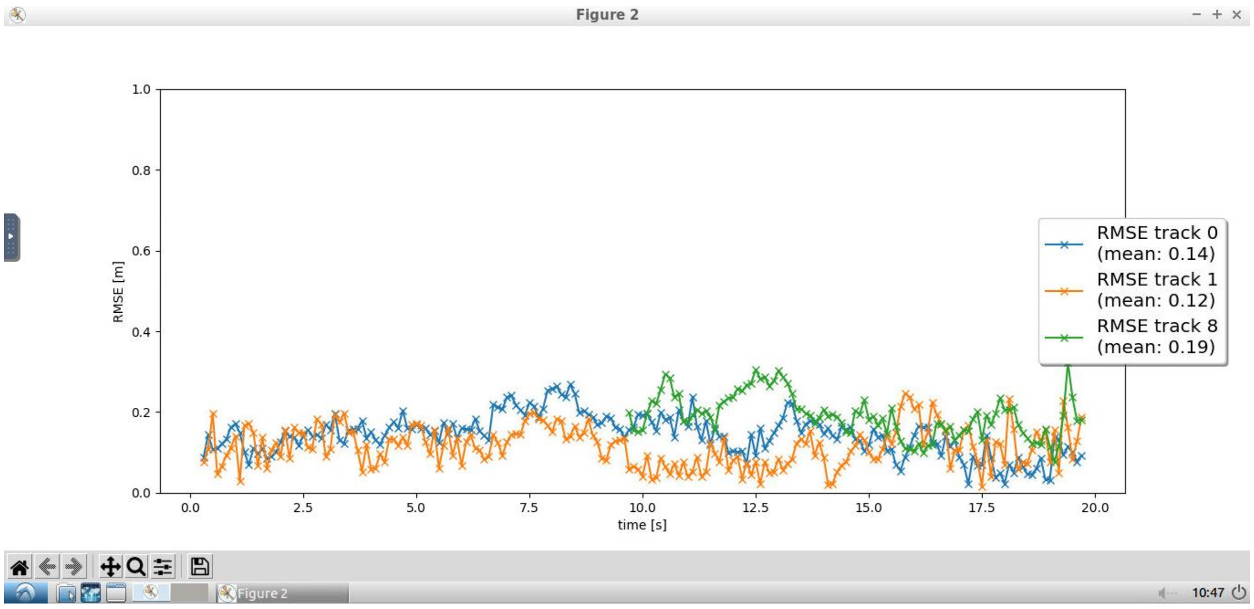


```
## Select Waymo Open Dataset file and frame numbers
data_filename = 'training_segment-1005081002024129653_5313_150_5333_150_with_camera_labels.tfrecord' # Sequence 1
# data_filename = 'training_segment-10072231702153043603_5725_000_5745_000_with_camera_labels.tfrecord' # Sequence 2
# data_filename = 'training_segment-10963653239323173269_1924_000_1944_000_with_camera_labels.tfrecord' # Sequence 3
show_only_frames = [0, 200] # show only frames in interval for debugging
```



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





5-Evaluation and Conclusion

Write a short recap of the four tracking steps and what you implemented there (EKF, track management, data association, camera-lidar sensor fusion). Which results did you achieve?

Step 1. Implementing the Extended Kalman Filter (EKF). I created a full Kalman filter with 6 dimensions: 3 for location (x, y, z) and 3 for velocity (vx, vy, vz). Prof. Thrum believes that this is a fantastic accomplishment because it is the key component of a tracking system with sensor fusion assistance.

Step 2. Track Management: Based on the data from the measurements, I designed a module to add new tracks, remove old tracks, and update the statuses of present tracks. In my updated version, I employed a dictionary whose keys are the frames and whose values are lists that collect the presence or absence of camera measurements and lidar readings.

Step 3. Data Association: The technique used to link measurements and tracks is particularly sophisticated since it finds the closest match between a measurement and a track in the Mahala Nobis Distance (MHD) space, which is a space that has been warped by the statistical predictions of locations and velocities. Up until there are no more pairings to match, the algorithm keeps comparing the closest pairs.

Step 4. The last phase in the sensor fusion process is the camera-lidar sensor fusion. Using the homogeneous transformation matrices, data from two separate sensors with distinct geometries are converted into vehicle coordinates. Similarly, to calculate hx and the EKF's Jacobian, vehicle coordinates are converted into the matching sensor coordinates. I couldn't figure out how to turn on the camera measurements at this phase. because despite fully following the directions, they were inactive.

Which results did you achieve? Which part of the project was most difficult for you to complete, and why?

Eventually, a camera and lidar were combined to track nearby cars using a Kalman filter, and this part 'Associate measurements to tracks with nearest neighbor association' was most difficult part for me to complete, because it took me a lot of time to understand it and implement the code.

Do you see any benefits in camera-lidar fusion compared to lidar-only tracking (in theory and in your concrete results)?

Lidar does not, in principle, have a higher resolution or more frames per second than cameras. As a result, cameras are more adept at identifying things in the surroundings and clearing up confusion.

When I completed implementing Step 4, which merges lidar readings with camera measurements, I saw a considerable boost in the accuracy of my actual findings. Without camera measurements, it would be very difficult to clarify the inconsistencies in the video in Step 4.

The lines of ray tracing are confusing in camera measurements. Any depth is legal and confusing in these lines. Using lidar readings, which offer precise depth information, this uncertainty is cleared up. Additionally, the experiment had significant gaps in false negative results and lacked some lidar readings. Therefore, the combination of lidars and cameras makes up for their shortcomings.

Which challenges will a sensor fusion system face in real-life scenarios? Did you see any of these challenges in the project?

At night and while determining distances, cameras are less precise. As a result, these errors must be handled via sensor fusion systems. Fortunately, the EKF is mathematically sound enough to handle noisy readings correctly.

Can you think of ways to improve your tracking results in the future?

In this project, I used precomputed detections. Furthermore, there were a number of false negatives and ground-truth labels without measurements in these precomputed detections that Udacity offered. There are significant frame gaps. I'm not making this up. Maybe I should write my own code to compute my own detections. By doing this, I hope to have better outcomes.

Moreover, I should try the "Stand Out Suggestions":

- Fine-tune your parameterization and see how low an RMSE you can achieve! One idea would be to apply the standard deviation values for lidar that you found in the mid-term project. The parameters in misc/params.py should enable a first running tracking, but there is still a lot of room for improvement through parameter tuning!

- Implement a more advanced data association, e.g., Global Nearest Neighbor (GNN) or Joint Probabilistic Data Association (JPDA).
- Feed your camera detections from Project 1 to the tracking.
- Adapt the Kalman filter to also estimate the object's width, length, and height, instead of simply using the unfiltered lidar detections as we did.
- Use a non-linear motion model, e.g., a bicycle model, which is more appropriate for vehicle movement than our linear motion model, since a vehicle can only move forward or backward, not in any direction.

The final video of the results

https://youtu.be/yRMKYUl2hoI

After adding some modifications:

1. The visualization shows that the tracing works well every time, and there are no path losses.
2. The mean RMSE for these two tracks is below 0.25 as shown below.

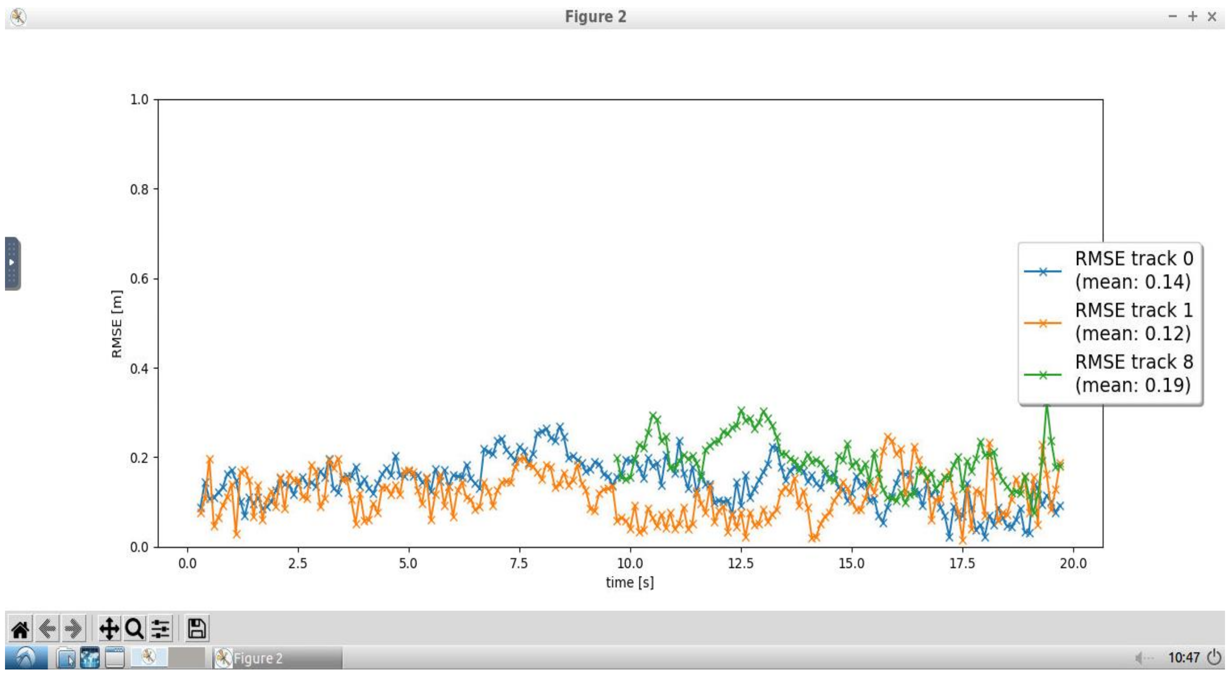


Figure 2
RMSE [m]
time [s]
RMSE track 0 (mean: 0.14)
RMSE track 1 (mean: 0.12)
RMSE track 8 (mean: 0.19)
Figure 2
10:47