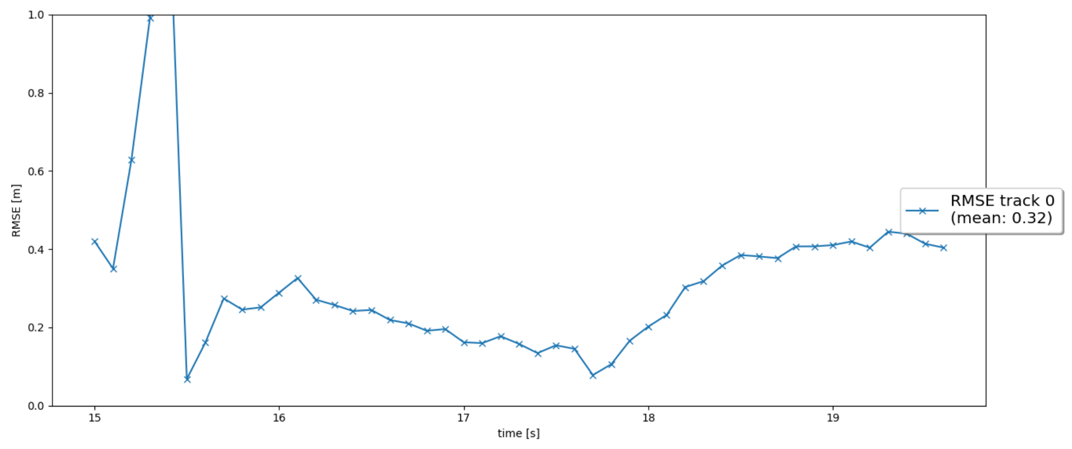
Final Project: Sensor Fusion and Object Tracking

https://github.com/LimHaeryong/Sensor-Fusion-and-Object-Tracking

Project Instructions Step 1

In this part, I implemented predict and update function for an Extended Kalman Filter within the file "filter.py". Also I implemented F, Q, gamma, S functions which are needed to predict and update step. As a result, the RMSE plot shows a mean RMSE of 0.32. RMSE value means the residual between estimated state and ground truth state.



Project Instructions Step 2

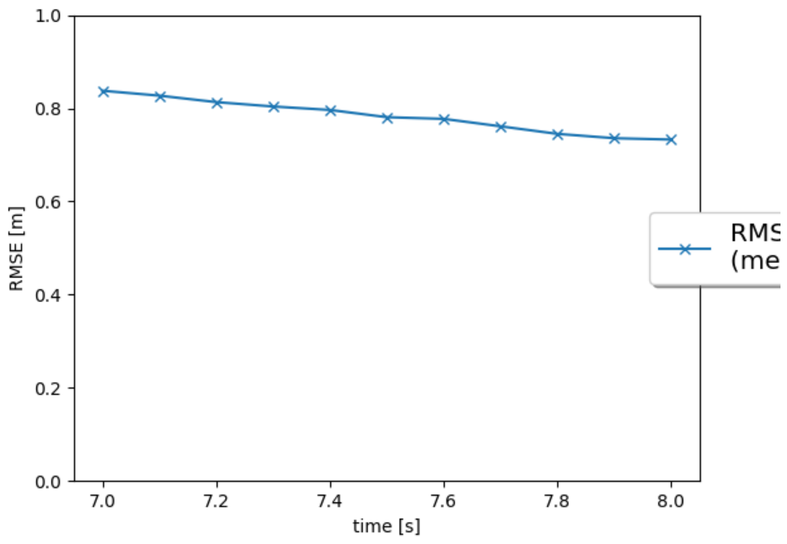
I implemented track initialization, manage_tracks and handle_updated_track functions within the file "trackmanagement.py".

In the track initialization part, initialized state vector track.x and error covariance matrix track.P based on the unassigned measurement which needs to be transformed vehicle coordinates. Then initialized track.state with 'initialized' and track.score with 1/window.

In the manage_tracks function, decreased track.score by 1/window for unassigned tracks. And deleted tracks if (confirmed but score is lower than delete_threshold) or (one of the position x and y values in the track.P is bigger than max_P value) or (score is lower than 0.05).

In the handle_updated_track function, increased score for all updated track by 1/window. If score is bigger than confirmed_threshold, assigned state as 'confirmed'. Else, assigned 'tentative'.

The result RMSE plot shows one single track without track losses.



Project Instruction Step 3

I implemented associate, get_closest_track_and_meas, MHD and gating function within the file "association.py".

MHD function returns mahalanobis distance between a track and a measurement. It requires gamma and S values which are implemented in the step 1.

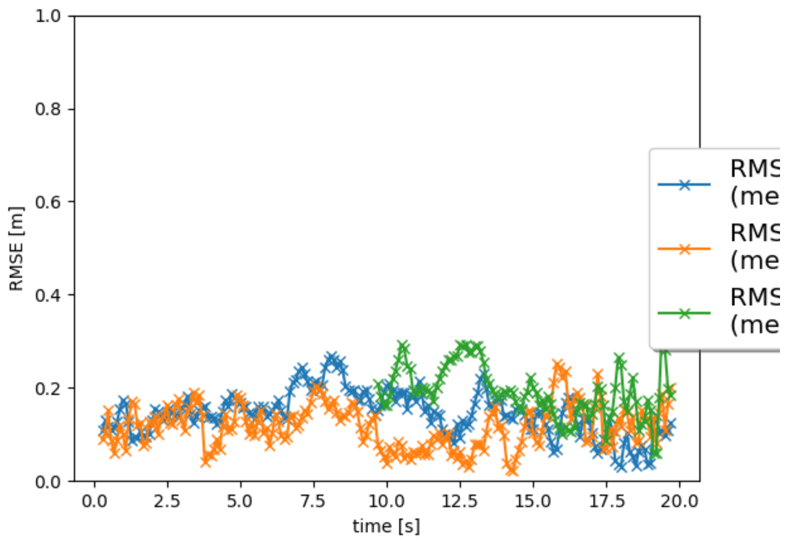
gating function returns True if MHD is smaller than a limit. The limit is chi square critical value. The significance level p is 0.995 and dof is same as measurement dim.

associate function creates association matrix size of NxM. N = length of track list, M = length of meas list. Using MHD and gating functions, at each element of this matrix, if gating returns False, assign np.inf else assign MHD value. Then update the list of unassigned meas and unassigned tracks based on all meas list and track list.

In the get_closest_track_and_meas function, find the min value and index in association matrix and delete corresponding row and column. If min value if np.inf, just return (np.nan, np.nan) else return the association pair (matched track, matched meas).

After matched nearest track and measure, remove corresponding track and measure from unassigned tracks and unassigned meas.

The result RMSE plot shows no confirmed ghost tracks and every RMSE values are lower than 0.4.



Project Instruction Step 4

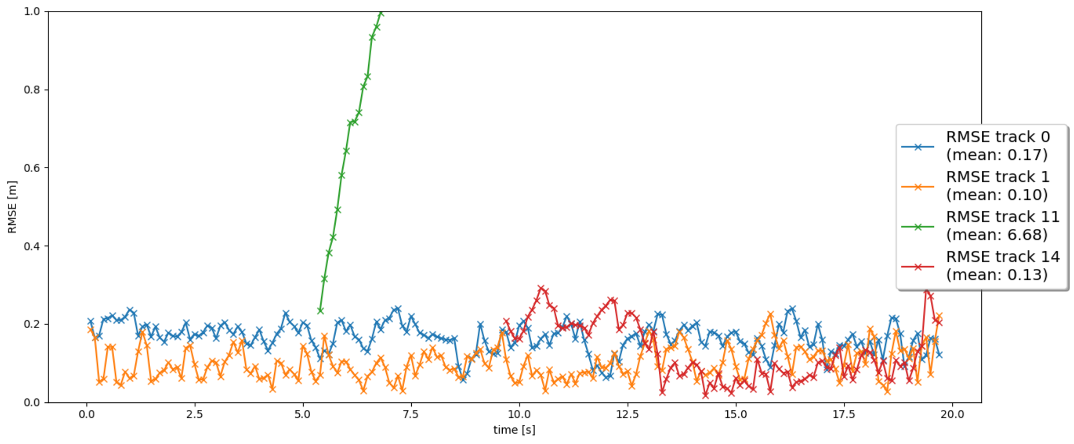
In this step, I implemented in_fov and get_hx functions and initialization of camera measurement objects z(measurement vector), R(measurement noise matrix) and sensor object.

in_fov function checks if the state x can be seen by this sensor or not.

In the get_hx function, I implemented codes which calculate h(x) value when sensor is camera.

The result RMSE plot shows two confirmed tracks are tracked from beginning to end without track loss. Also the mean RMSE values are 0.17 and 0.10.

There are more two tracks. One of them is tracked from alsomot 10(s) to end. And another one is tracked from 5(s) but its RMSE diverged and deleted. If watch the my_tracking_results video uploaded to github repository, the track 11 appear and move away slowly. So it seems reasonable.



Conclusion

Completing this project, implementing EKF was easier than I thought. I spent most time to implement the file trackmanagement.py. As you see in the step 4 RMSE plot, one track RMSE diverged and its mean value is 6.68. I was confused by this result.

Decreasing the unassigned track score by 2/window instead of 1/window or Changing the confirmed threshold, the diverged track is not confirmed and I could get the 'clean' RMSE plot same as example.

However as I mentioned above, this diverged track is ok. Because there is no 'confirmed ghost' and track loss.

Camera-lidar fusion has several advantages. Lidar measurement intensity varies with the reflectivity of surface. So using camera data helps to sensing low intensity objects. And camera measurement needs lidar measurement because camera have weakness to determine depth.

Also using more sensors is helpful to estimate state more accurately. In my concrete results, after using camera-lidar fusion, many ghost tracks(not confirmed but frequently occurred) improved.

In real-life scenarios, sensor performance varies with many environments like weather, time. In this project, we used constant measurement noise matrix.

Using variable measurement noise matrix, sensor performance should be considered in real-time.

Also using better performed detection model would be helpful.