Project 3: Sensor Fusion and Tracking: Final

Vinayak Deshpande: vinayak.desh2@gmail.com

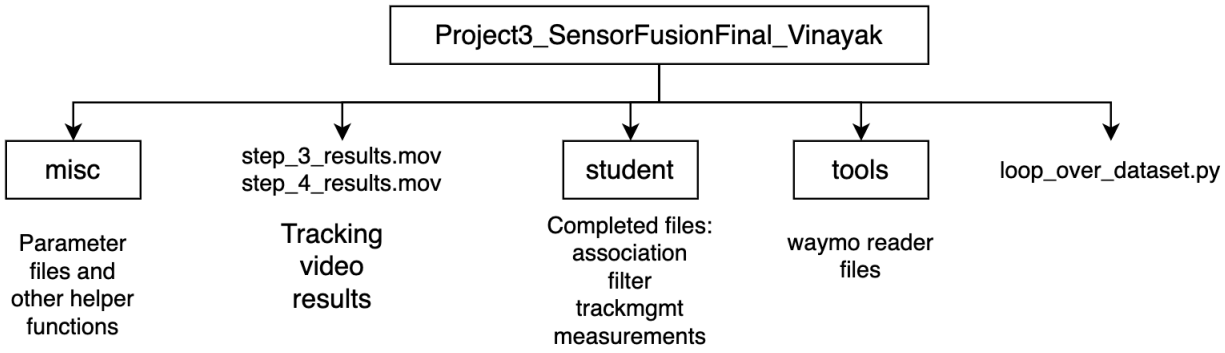
January 13, 2024

Abstract

This report contains the summary and findings of the sensor fusion and tracking final project for the Udacity Self Driving Car Nanodegree. Let $\mathcal{R}^{n\times m}$ be the set of all $n \times m$ real matrices and $\mathcal{R}^n$ be the set of all real $n \times 1$ column vectors.

1 Project Folder Structure

The attached zip submitted on the Udacity website contains the following files:



2 Kalman Filter

The objective of this section was to integrate a Kalman filter to estimate the target position $\hat{x}$, given noisy measurements over time. There are 6 states $x = [p_x, p_y, p_z, v_x, v_y, v_z]^T$, and 3 measurements $z = [p_x, p_y, p_z]$. Given a sample time of Δt, and process noise covariance q, and a discrete time state space representation $x_{k+1} = Fx_k$, the equations are:

2.1 Relevant Equations

$$F \in \mathcal{R}^{6\times 6} : \begin{bmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

$$Q \in \mathcal{R}^{6\times 6} : \begin{bmatrix} q_1 & 0 & 0 & q_2 & 0 & 0 \\ 0 & q_1 & 0 & 0 & q_2 & 0 \\ 0 & 0 & q_1 & 0 & 0 & q_2 \\ q_2 & 0 & 0 & q_3 & 0 & 0 \\ 0 & q_2 & 0 & 0 & q_3 & 0 \\ 0 & 0 & q_2 & 0 & 0 & q_3 \end{bmatrix} where \rightarrow q_1 = \frac{q\Delta t^3}{3}, q_2 = \frac{q\Delta t^2}{2}, q_3 = q\Delta t \tag{2}$$

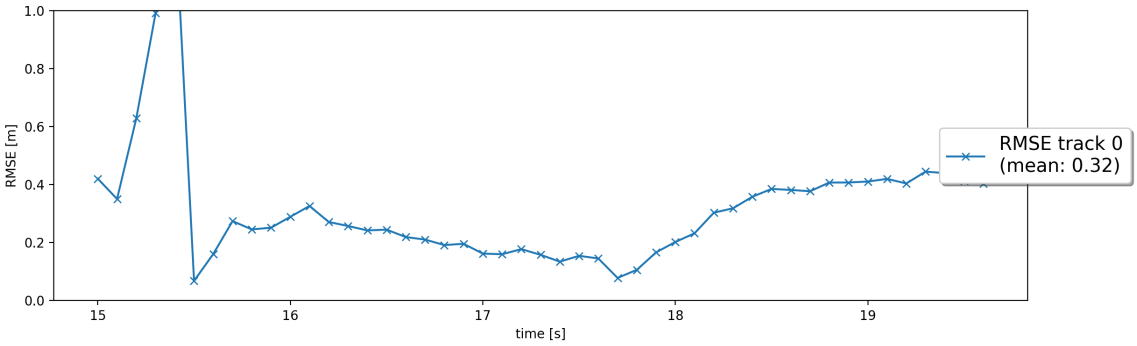
As the LIDAR measurements are in vehicle coordinates already, the measurement matrix is:

$$H \in \mathcal{R}^{3\times 6} : \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{3}$$

The initial track estimate was $\hat{x} = [49.5, 3.41, 0.917]^T$. The sampling time was $\Delta t = 0.1s$ along with a process noise covariance term $q = 3$, representing sudden changes in velocity in this constant velocity state space model. 50 frames were processed which contained only a single measurement, thus representing a time of $5s$. Each frame represented a moment in time of the vehicle driving, and the Kalman Filter predicted and updated at all time steps.

2.2 Result

The root mean square error plot is shown here, it averaged at 0.32, showing good performance.



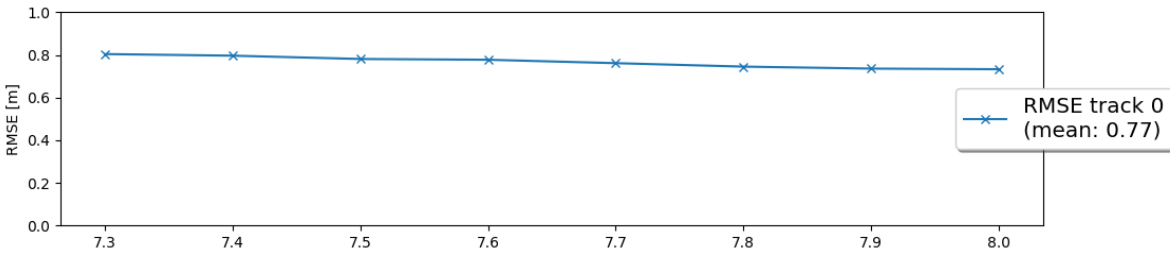
2.3 Tuning Filter and Single Target Tracking

Firstly, the Kalman filter initial state estimate $\hat{x}$ and first three rows of error covariance P were updated to the first set of noisy measurements, to provide better performance. The rotation matrix $M_R \in \mathcal{R}^{3\times3}$ represents the rotation from vehicle to sensor frame. It is the identity matrix in the LIDAR measurement scenario. The first set of noisy measurements is $z_0 \in \mathcal{R}^3$. The measurement noise covariance is R. The last three rows of P corresponding to the velocity covariance were updated from the parameters provided. The track state was set to "initialized" and score was updated to a small value as well.

$$\hat{x}_0 = M_R z_0 \tag{4}$$

$$P_0^-[0:3, 0:3] = M_R(R)M_R^T \tag{5}$$

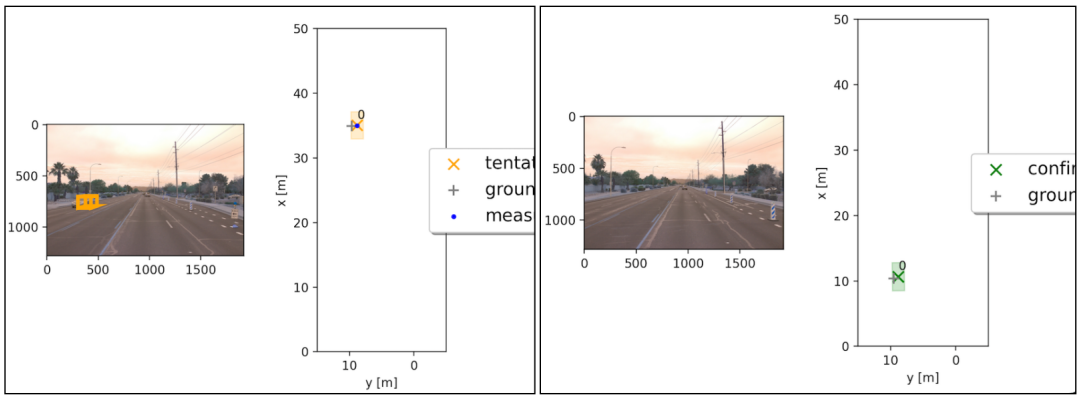
Next, the track management module was updated. The *manage_tracks* method was updated to decrement the track score by 0.1 for an unassigned track with a measurement. Then logic was added to iterate through the list of tracks, and delete any if their score dropped below a certain value if they were confirmed tracks. In addition, a track was commanded to be deleted if it was initialized or tentative with a large error covariance for the x, y position. Lastly, the method *handle_updated_tracks* was configured to increment the track score by 0.1, and to set a track to tentative or confirmed based on its score. A total of 35 frames were used (Frame: $65 \rightarrow 100$) indicating 35 measurements over a timestep of $0.1s$ The RMSE plot is shown here and stays roughly constant at 0.77. Note that the 0.1 score delta is chosen based on the window size of 6 frames, it is close to the value $1/6 = 0.166$. Lastly, once the track was confirmed, its score was capped to prevent it from perpetually increasing.



In particular, only a single track was used, and assigned to a single measurement. It was observed that from frames $78 \rightarrow 100$, there were no incoming measurements, meaning the single track was unassigned. The following logic was used to decrease the track score. However it never executed because *meas_list* was always empty when there was an unassigned track present.

```
# decrease score for unassigned tracks
for i in unassigned_tracks:
    track = self.track_list[i]
    # check visibility
    if meas_list: # if not empty <-- always returned False
        if meas_list[0].sensor.in_fov(track.x):
            # your code goes here
            track.score = track.score - 0.1
```

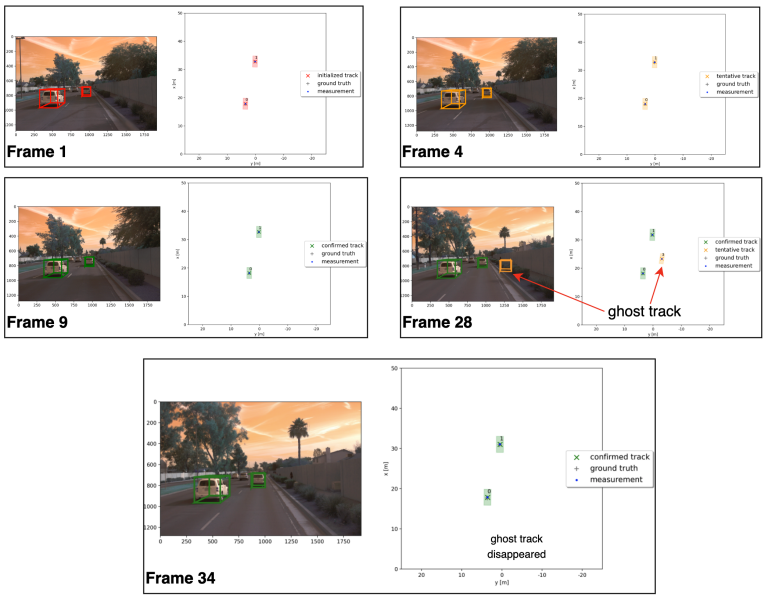
The measurement and track states for a couple of frames (left = tentative track, right = confirmed track) are shown here. Notice on the confirmed track there is no measurement (the vehicle is not in the field of view), as a result it remains unassigned. With an empty measurement list, the score remains constant.



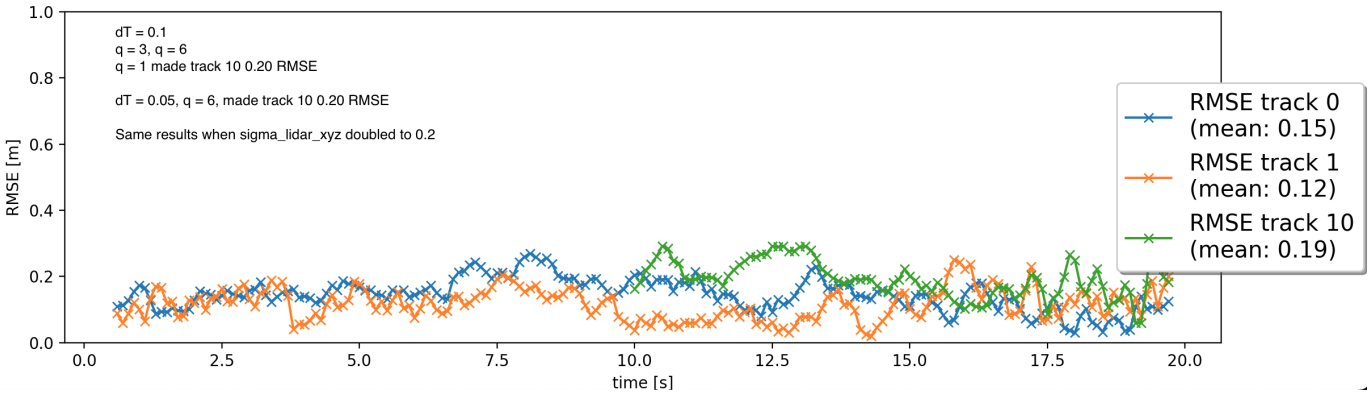
3 Multi Target Tracking

3.1 LIDAR only

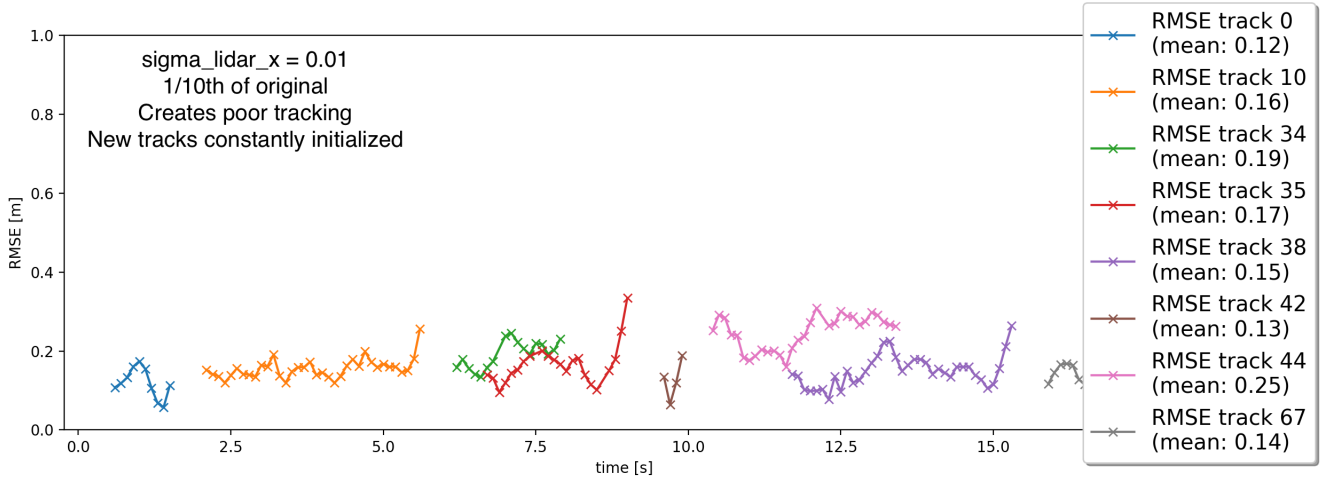
Moving on to multi target tracking, a sequence of 200 frames was selected each having a varying number of LIDAR measurements. The association class was updated to calculate the association matrix at each timestep (frame) between each track and measurement, based on the number of tracks and measurements available. At the very first timestep, 2 measurements were available, however there were no tracks as they were not yet initialized. As a result, the prediction step was skipped, and there was no association matrix. However, from the second timestep, a $2x2$ association matrix was created and the track score kept increasing gradually, as both vehicles were in the FOV. Later on, some ghost tracks (initialized / tentative) momentarily appeared and disappeared as their score dropped to a low value.



As the associate and update steps are run at each timestep, calculating an association matrix can be time consuming in real world scenarios. For tracking several objects $> 5 - 10+$ at once, a large matrix will have to be created, and removing rows and columns can be computationally expensive. A potential way to avoid this issue is to set the row / col to a fixed negative number once a track has been paired with a measurement. This method is feasible because track scores can never be negative. In low level hardware, removing a row and column from a matrix leads to extra memory operations. The RMSE plot is shown here.



Changing the process noise covariance term q did not have a significant effect on the RMSE. Increasing it to 6 from 3 slightly increased track 10 RMSE to 0.20. Furthermore, doubling the 3 σ_{xyz} terms in the error covariance P matrix gave the same results. Reducing the LIDAR measurement covariance σ_{xyz} by a factor of 10 in the R matrix had a major effect on tracking as it greatly reduced performance. More tracks were created despite only 2 measurements available. The tracks kept switching between initialized and tentative. A possible reason is poor tracking scores and the large number of unassigned tracks. Changing the measurement covariance R too much has a major effect on the Kalman gain.



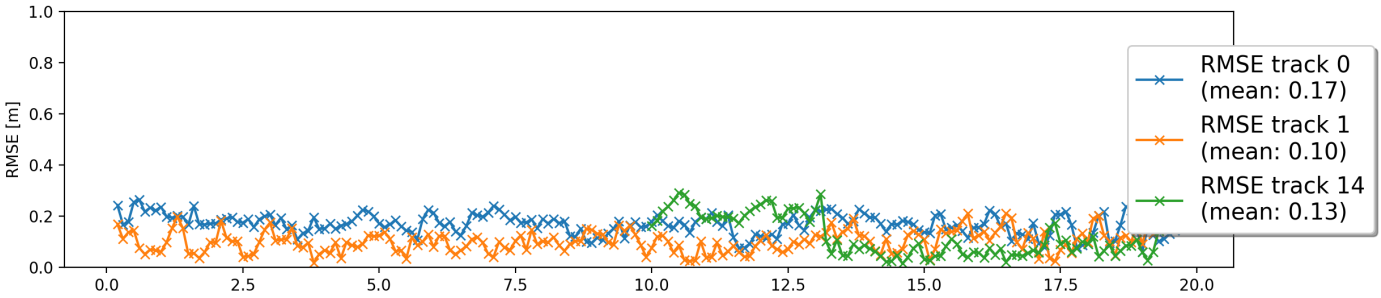
3.2 LIDAR and Camera

The last step involved reconfiguring the track management to also include camera measurements. The following equation converted from camera to image coordinates.

$$h_x = c_i - f_i \frac{p_y}{p_x} \tag{6}$$

$$h_y = c_i - f_j \frac{p_z}{p_x} \tag{7}$$

The calculation for the error in the Kalman Filter had to be updated. Instead of using the *get_H* method, the *get_hx* method was used as the nonlinear camera measurement model was directly calculated here along with all the necessary steps to convert from vehicle to camera sensor coordinaes. For the LIDAR, H could be obtained separately because the track $\hat{x}$ was already in the sensor frame. The resulting RMSE plot is shown here showing good performance.



Two videos are included in the project submission files which show tracking of all frames with and without the camera.