Writeup: Track 3D-Objects Over Time

1. Write a short recap of the four tracking steps and what you implemented there (filter, track management, association, camera fusion). Which results did you achieve? Which part of the project was most difficult for you to complete, and why?

1. filter.py : In this step, I implement the state transition function F and process noise covariance Q for the Extended Kalman Filter for the tracking car

   - State Prediction: For 3D motion and velocity, the constant velocity system matrix is:

$$\mathbf{F}(\Delta t) = \begin{bmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

     The state transition equation in 3D space is:

$$\begin{bmatrix} p_x \\ p_y \\ p_z \\ v_x \\ v_y \\ v_z \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z \\ v_x \\ v_y \\ v_z \end{bmatrix} + \begin{bmatrix} v_{p_x} \\ v_{p_y} \\ v_{p_z} \\ v_{v_x} \\ v_{v_y} \\ v_{v_z} \end{bmatrix}$$

   - Process Noise Covariance: If we assume the noise through acceleration in x and y and z to be equal, $v_x = v_y = v_z$, the continuous process noise covariance Q can be modelled as:

$$Q = E[vv^t] = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & E[v_x^2] & 0 & 0 \\ 0 & 0 & 0 & 0 & E[v_y^2] & 0 \\ 0 & 0 & 0 & 0 & 0 & E[v_z^2] \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & q & 0 & 0 \\ 0 & 0 & 0 & 0 & q & 0 \\ 0 & 0 & 0 & 0 & 0 & q \end{bmatrix}$$

     Discretizing this continuous model leads to a formula for Q depending on Δt as follows:

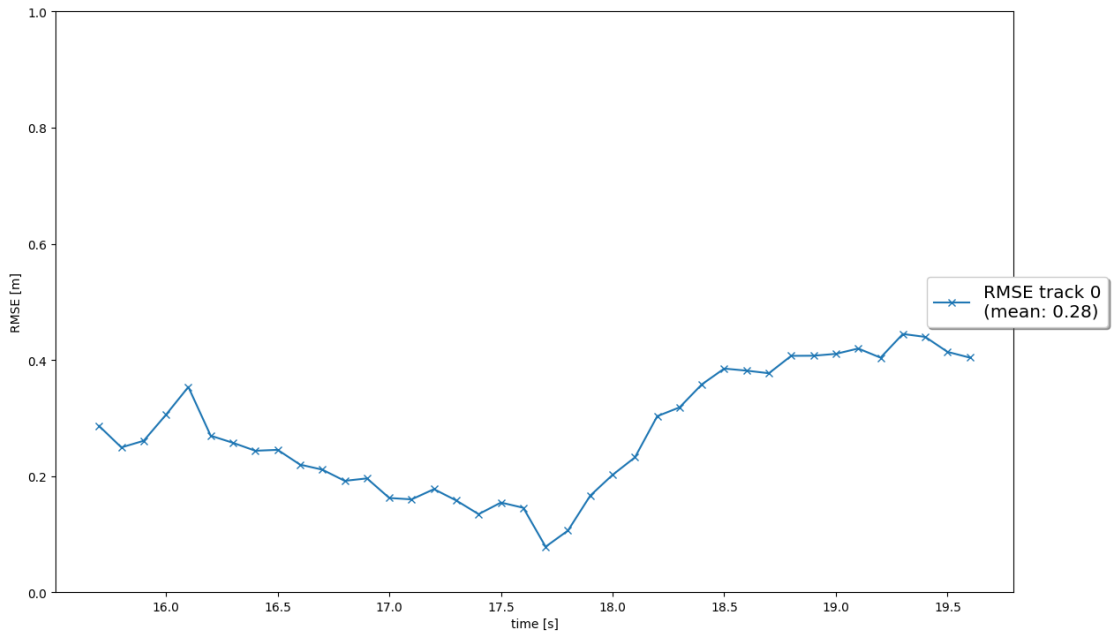
$$Q(\Delta t) = \int_0^{\Delta t} F(t) Q F(t)^T dt =$$

$$\int_0^{\Delta t} \begin{bmatrix} 1 & 0 & 0 & t & 0 & 0 \\ 0 & 1 & 0 & 0 & t & 0 \\ 0 & 0 & 1 & 0 & 0 & t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & q & 0 & 0 \\ 0 & 0 & 0 & 0 & q & 0 \\ 0 & 0 & 0 & 0 & 0 & q \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ t & 0 & 0 & 1 & 0 & 0 \\ 0 & t & 0 & 0 & 1 & 0 \\ 0 & 0 & t & 0 & 0 & 1 \end{bmatrix} dt$$

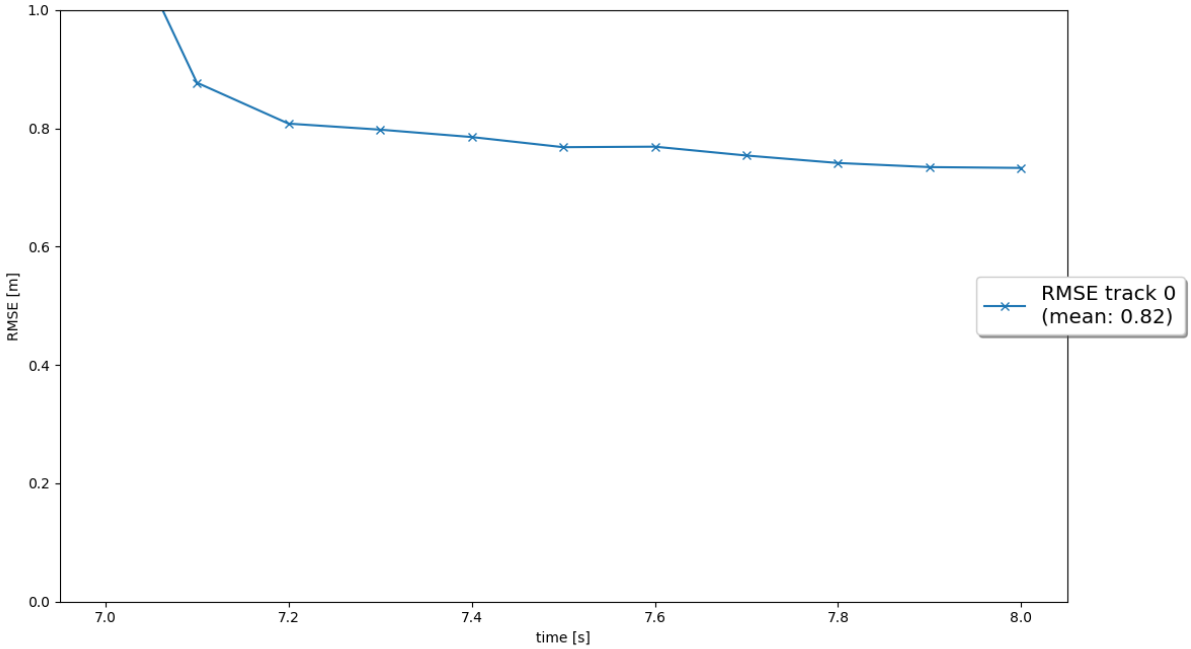
$$= \int_0^{\Delta t} \begin{bmatrix} t^2 q & 0 & 0 & tq & 0 & 0 \\ 0 & t^2 q & 0 & 0 & tq & 0 \\ 0 & 0 & t^2 q & 0 & 0 & tq \\ tq & 0 & 0 & q & 0 & 0 \\ 0 & tq & 0 & 0 & q & 0 \\ 0 & 0 & tq & 0 & 0 & q \end{bmatrix} dt =$$

$$\begin{bmatrix} \frac{1}{3}(\Delta t)^3 q & 0 & 0 & \frac{1}{2}(\Delta t)^2 q & 0 & 0 \\ 0 & \frac{1}{3}(\Delta t)^3 q & 0 & 0 & \frac{1}{2}(\Delta t)^2 q & 0 \\ 0 & 0 & \frac{1}{3}(\Delta t)^3 q & 0 & 0 & \frac{1}{2}(\Delta t)^2 q \\ \frac{1}{2}(\Delta t)^2 q & 0 & 0 & tq & 0 & 0 \\ 0 & \frac{1}{2}(\Delta t)^2 q & 0 & 0 & tq & 0 \\ 0 & 0 & \frac{1}{2}(\Delta t)^2 q & 0 & 0 & tq \end{bmatrix}$$

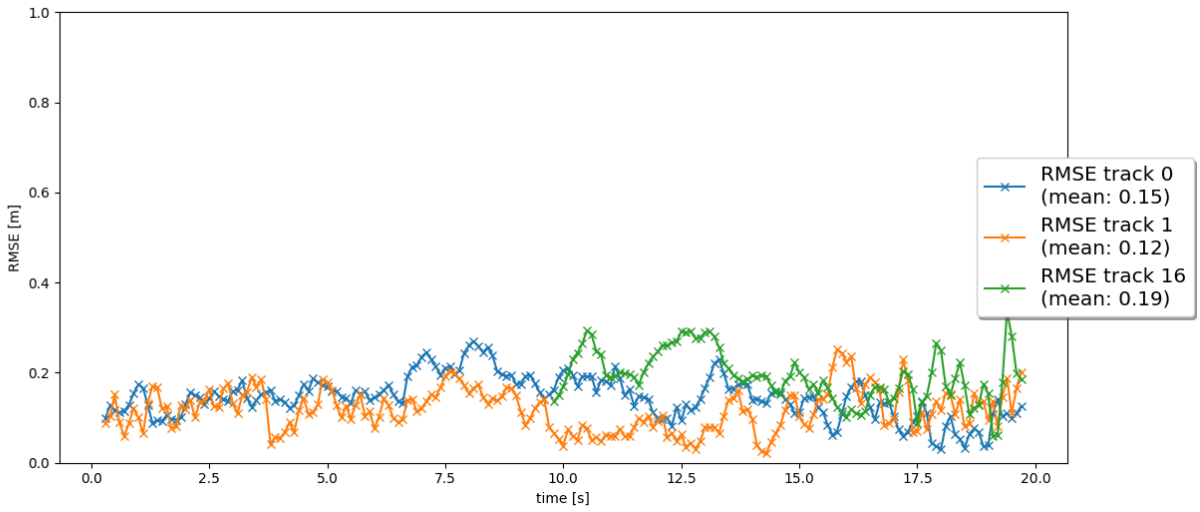
- Result: After applying the above F and Q formula to the EKF code, I can see that RMSE is below 0.35 which means it works correctly.



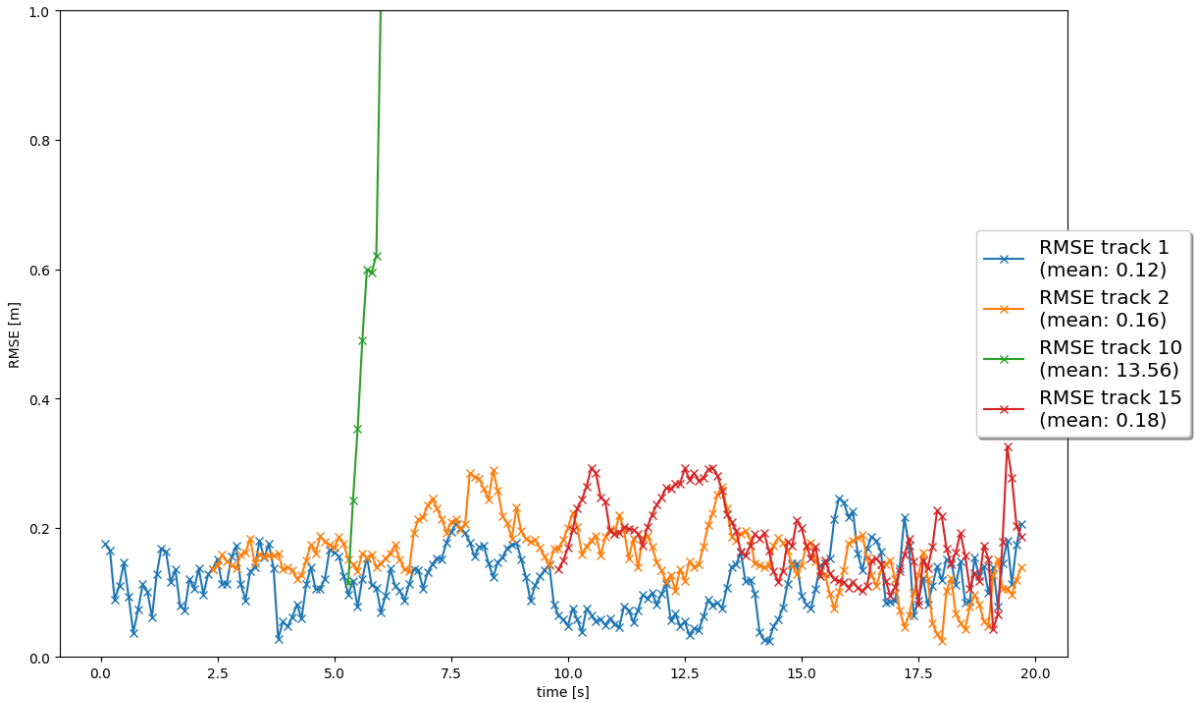
2. trackmanagement.py: In this step, I implenment the code for mangaing score of track. When the sensor detects the car, the Track class is initiaized as the default score. The track state will be changed to confirmed when the score reached the threashold value. When car is away from visible area of sensor, the score will be decreased. The confirmed track which has low score will be deleted.



3. association.py: In this step, I implement the code for situation where there are multiple track and measurement. For that, I calculate the Mahalanobis distance between tracks and measurements and update them in the association matrix. The unassigned track that is found from that process lose the score by the step 2.



4. measurements.py: In this step, I implement the code for receving and processing the camera frame image. The jacobian formula is used to transform the 2D point to 3D point. Additionally, camera measurement parameters are also initialized separately from lidar measurement.



   - Result: Please check the video file of workspace folder.

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

Actually, it is difficult to see the benefit of using the sensor fusion in the provided three dataset. The lidar-only tracking even shows more stable performance. I assume the reason for such result as follows. In my understand, the sensor fusion methos is only useful when the environment around the car obstructs the one of sensor's ability to perceive the state well. I think there is no such obstacle in provided dataset that prevents the lidar from working properly such as the reflective surface, heavy sunlight.

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

I can see that the sensor fusion result depend on the covariance value of sensor. It means we need to spend amount of time to find the exact value of each sensor becasue every sensors are

slighly different even if they are manufactured from same factory.

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

One thing I can easily is finding more good object detection model. Maybe, I use the Neural Network which has more deep layer or high resolution input.