Final Project: Sensor Fusion and Object Tracking

Self-Driving Car Engineer Nanodegree

Submitted by:
Abhishek U H

- **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? Which part of the project was most difficult for you to complete, and why?**

1. Extended Kalman Filter

The Kalman filter is used to predict the states of the system and update internal parameters when the new measurement is seen. In project we want to predict the position and velocity of the dynamic vehicles. The whole Kalman filtering implementation revolves around 2 steps i.e. Prediction step and Update step

Prediction: For linear dynamic system for 1D motion , position and velocity can be predicted using the below given equation.

$$x_k = x_{k-1} + v_{k-1} * \Delta t + 0.5 * a_{k-1} * \Delta t^2$$

$$v_k = v_{k-1} + a_{k-1} * \Delta t$$

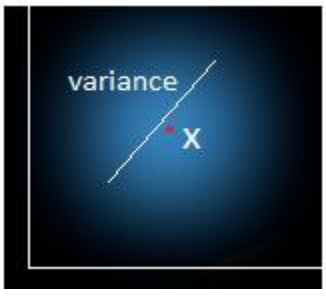
Since we need to predict in 3D motion the same equation can be extended to matrix addition and multiplication

$$X_k = AX_{k-1}$$

$$A = \begin{bmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \text{ and } X_k = \begin{bmatrix} x \\ y \\ z \\ Vx \\ Vy \\ Vz \end{bmatrix}$$

Note: The acceleration in the project is ignored while predicting the states of vehicle

Also we assume that all the measurement , noise and prediction are Gaussian in nature so we assume the prediction as a Gaussian blob with mean X_k. To illustrate pictorially see below.



So, the predicted Covariance Matrix has the below equation, we add a gaussian process white noise $(0, \sigma^2)$.

$$P_k = AP_{k-1}A^T + Q$$

$$\text{where Q} = \begin{bmatrix} q3 & 0 & 0 & \text{q2} & 0 & 0 \\ 0 & q3 & 0 & 0 & \text{q2} & 0 \\ 0 & 0 & q3 & 0 & 0 & \text{q2} \\ q2 & 0 & 0 & q1 & 0 & 0 \\ 0 & q2 & 0 & 0 & q1 & 0 \\ 0 & 0 & q2 & 0 & 0 & q1 \end{bmatrix}$$

$q3 = (1/3)*q*\Delta t^3$, $q2 = (1/2)*q*\Delta t^2$, $q1 = q*\Delta t$

Update: When there is a new measurement from sensors, we need to update the predicted value with measurement value. The way to do it is given in the below equations

$\gamma = \mathbf{z} - \mathbf{Hx}$ # get the difference between measured and predicted

$\mathbf{S} = \mathbf{H} * \mathbf{P}^- * \mathbf{H}^T + \mathbf{R}$ # Kalman Gain to decide how much to put weights on new measurement

$\mathbf{K} = \mathbf{P}^- * \mathbf{H}^T * \mathbf{S}^{-1}$

$\mathbf{X}^+ = \mathbf{x}^- + \mathbf{K} * \gamma$ # Based on Kalman gain updated the predicted value with delta

$\mathbf{P}^+ = (\mathbf{I} - \mathbf{K} * \mathbf{H}) * \mathbf{P}^-$ # Update Covariance Matrix

2. Track Management

In Project, the detected object from the fpn-resnet object detection algorithm for Lidar point cloud data is used as a measurement(input) in the track management. These measurement is used for initializing the tracks, updating the track, increasing and decreasing the score of track and finally deleting the tracks.

Track Initialization : We try to associate the measurement with existing tracks list if the measurement is not associated with already available track in the list then we initialize with a new track with score 1/num_of_frames (this track only gets confirmed if we see it for num_of_frames)

Update: We use the Kalman filter for updating the predicted state and the covariance matrix of the tracks that was associated with the measurement.

Update Score: We need to update the score of the track each time we see the measurement from sensor and update the score and also the status as well, in case if the measurement is not available for the existing track we decrease the score and the method we follow for increasing or decreasing the tracks is as below

if (*in_fov() and not_associated* **):**

score = score -1/num_of_frames

Delete Track: If the score of the track is very less or the variance matrix of the track is huge we delete the track

3. Data Association:

In data association we try to associate the measurement with tracks. The logic that we followed in project is as below

1. Suppose, we have M measurement and N tracks, then we will create the matrix with size NxM

$$\begin{bmatrix} (0,0) & (0,1) & (0,2) & (0,3) & .. & .. & .. & (0,M-1) \\ (1,0) & (1,1) & (1,2) & (1,3) & .. & .. & .. & (1,M-1) \\ . & . & . & . & .. & .. & .. & . \\ . & . & . & . & .. & .. & .. & . \\ . & . & . & . & .. & .. & .. & . \\ . & . & . & . & .. & .. & .. & . \\ . & . & . & . & .. & .. & .. & . \\ . & . & . & . & .. & .. & .. & . \\ (N,0) & (N,1) & (N,2) & (N,3) & .. & .. & .. & (N-1,M-1) \end{bmatrix}$$

2. Initially, all the matrix element is initialized with inf

$$\begin{bmatrix} \infty & \infty & \infty & \infty & .. & .. & .. & \infty \\ \infty & \infty & \infty & \infty & .. & .. & .. & \infty \\ . & . & . & . & .. & .. & .. & . \\ . & . & . & . & .. & .. & .. & . \\ . & . & . & . & .. & .. & .. & . \\ . & . & . & . & .. & .. & .. & . \\ . & . & . & . & .. & .. & .. & . \\ . & . & . & . & .. & .. & .. & . \\ \infty & \infty & \infty & \infty & .. & .. & .. & \infty \end{bmatrix}$$

3. We calculate Mahalanobis distance between track and measurement and if it passes the gating then we store the value to above matrix. This will be repeated for all measurement with all tracks.
4. Next, we find the minimum value in matrix , then remove that measurement and track from unassociated buffer and also delete the column and row in the matrix and we repeat this until all the tracks are associated or there is nothing left for association. For example, below column is deleted

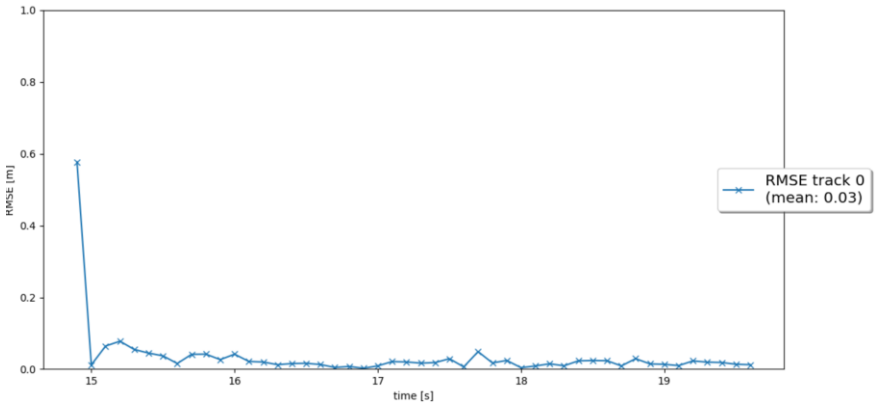
$$\begin{bmatrix} 4.6 & 12.1 & 14.5 & 10.0 & .. & .. & .. & \infty \\ \infty & 2.0 & 16.0 & 22.2 & .. & .. & .. & \infty \\ . & . & . & . & .. & .. & .. & . \\ . & . & . & . & .. & .. & .. & . \\ . & . & . & . & .. & .. & .. & . \\ . & . & . & . & .. & .. & .. & . \\ . & . & . & . & .. & .. & .. & . \\ . & . & . & . & .. & .. & .. & . \\ \infty & \infty & \infty & \infty & .. & .. & .. & \infty \end{bmatrix}$$

4. Camera-Lidar Fusion

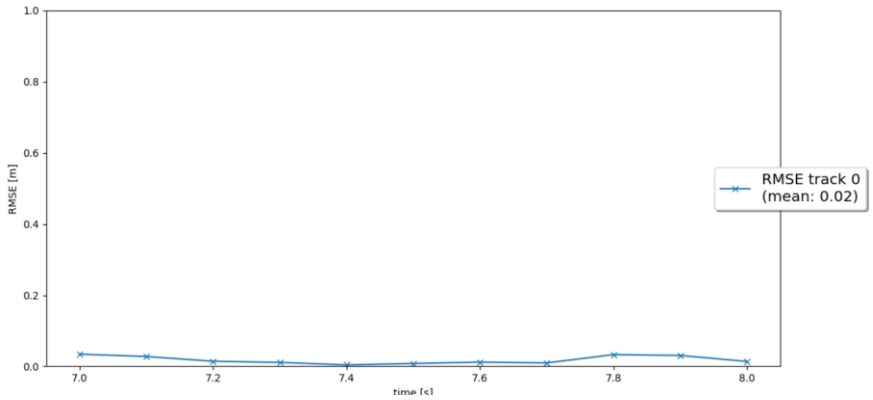
In the Fusion steps, the measurements from camera are used to update the track created from lidar measurement but no new tracks are created for the camera measurement. The measurement from camera has only x, y in image co-ordinate that needs to be converted to sensor co-ordinate for this we need a H matrix or a Jocobian Matix for co-ordinate conversion.

RMSE Plots:

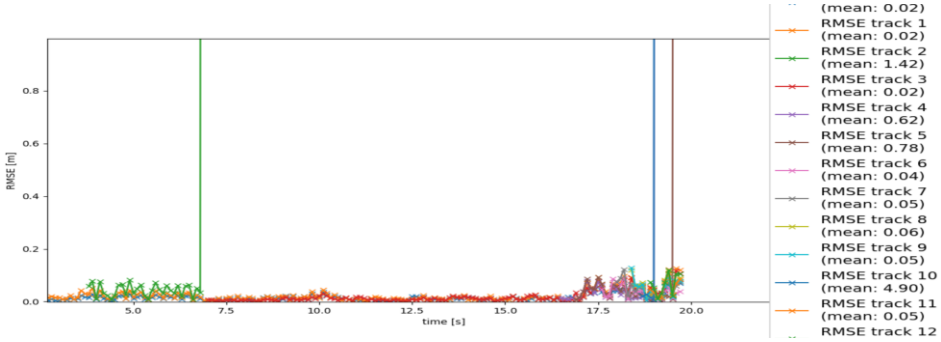
Step 1: EKF



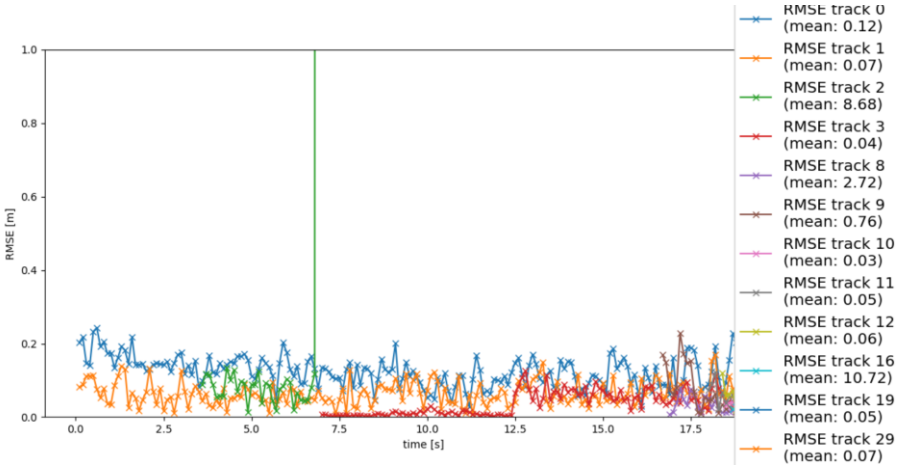
Step 2: trackmanagement



Step 3: Association



Step 4: Measurement



Difficult part of Project:

Understanding the camera H matrix and also the derivation of the Q matrix is bit tricky. Since this required understanding of probability distribution and other concept.

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

  Yes, there are benefit of adding camera lidar fusion. Reason

  1 The camera measurement assist the lidar tracks to confirm it early as the track score increases.
  2 There might be a chances where the track goes out of FOV of one sensor and the it helps in keeping the track alive for longer duration

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

  The sensor fusion might under perform when in curved scenario and sudden braking and accelerating scenarios die to the assumption that we had made as constant velocity and acceleration as noise. Other scenario which we could think of is the weather condition, both lidar and camera use visible light so they might underperform when in foggy or rainy condition where the view is not visible.

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

  We can tune with the initial assumption that we made while designing the KF and also adding radar sensor to nullify the weather effect.