SENSOR FUSION FINAL PROJECT

HANS FRANZEN

The tracking exercise consisted of four steps:

(1) Implement an extended Kalman filter. This was done in the file `filter.py`. The state transition matrix F, the process noise covariance matrix Q, the residual γ and the matrix S were implemented, along with the prediction and update functions. The RMSE is in Figure 1.
(2) Manage tracks. The respective code can be found in `trackmanagement.py`. New tracks are being initialized, a track score is maintained, a track status (initialized, tentative, confirmed) is set and updated, and tracks are being deleted once the track score drops below a threshold or the positional uncertainty is too big. The RMSE can be found in Figure 2.
(3) Associate tracks to measurements. This happens in the file `association.py`. The Mahalanobis distance between a track and a measurement is computed and an association matrix is set up accordingly. A gating threshold is applied. Then an association is made using the simple nearest neighbor algorithm. The RMSE is depicted in Figure 3.

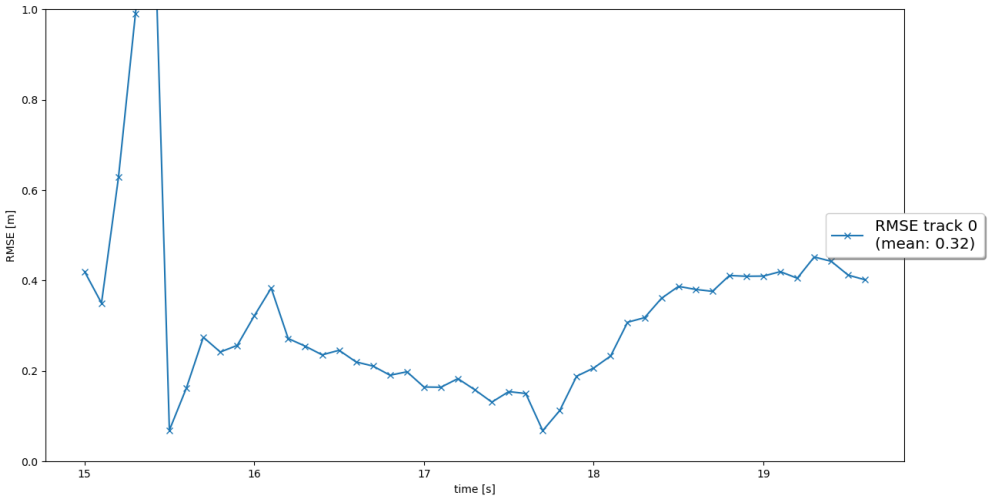

FIGURE 1. RMSE in Step 1

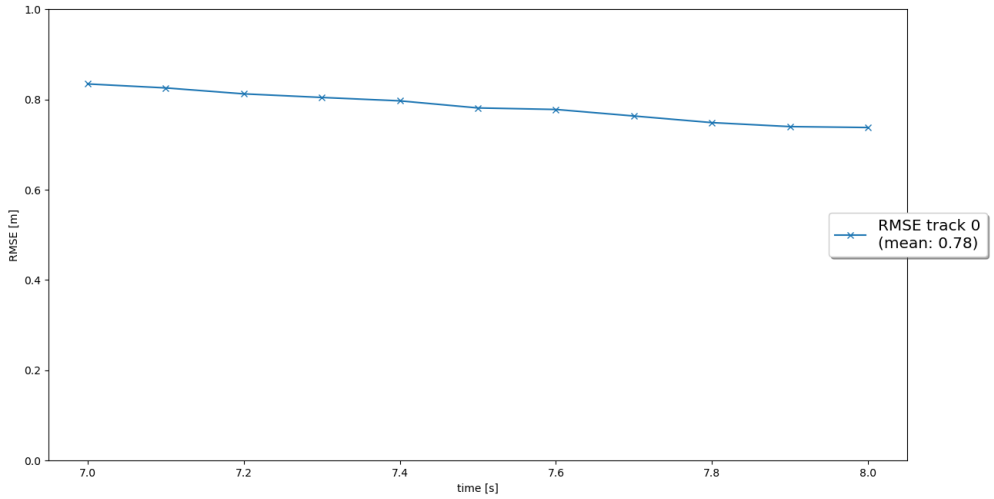

FIGURE 2. RMSE in Step 2

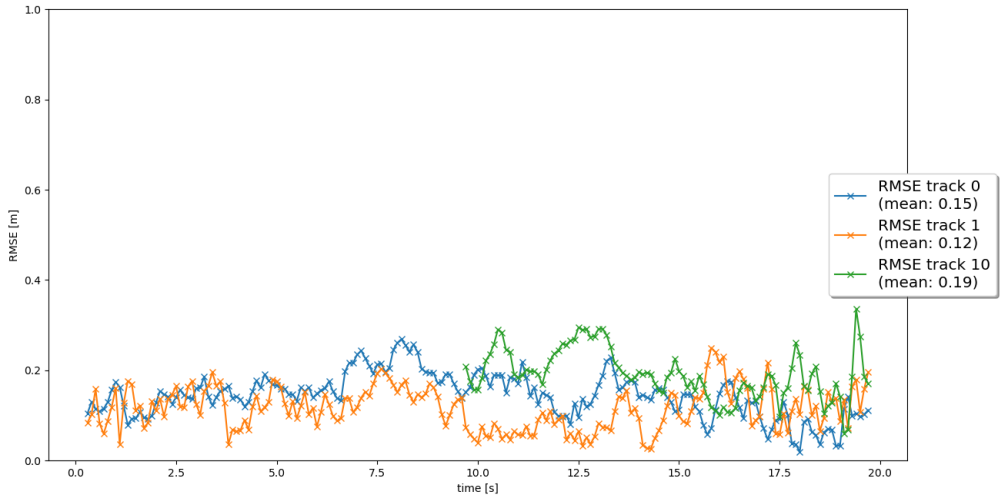

FIGURE 3. RMSE in Step 3

(4) Fusion with camera sensors. This is done in `measurements.py`. While the above three steps only included lidar measurements, we now use camera measurements

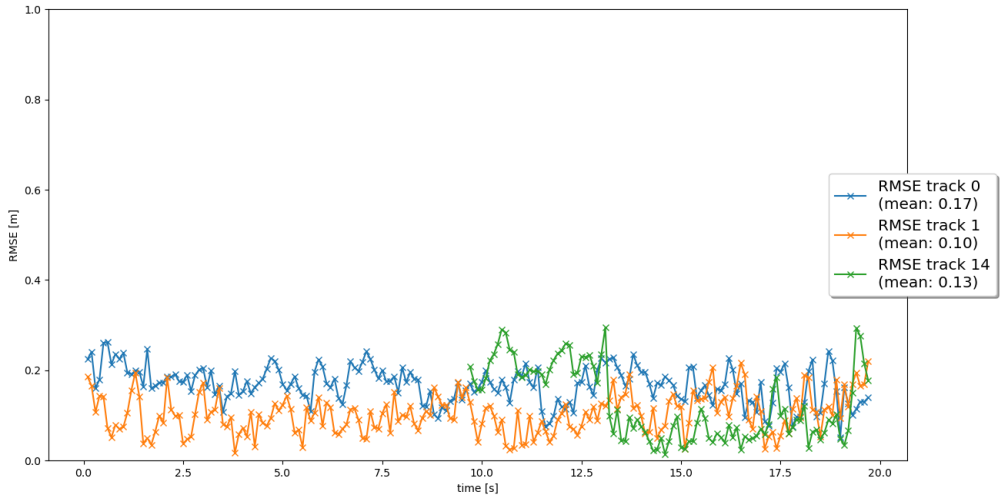

FIGURE 4. RMSE in Step 4

as well. We implement their field of view, the measurement function h and initialize new measurements. The RMSE is as shown in Figure 4.

The step which took most time for me was Step 4. I had a typo in initialization which resulted in very strange results. Due to the many function calls it was very difficult to debug.

Using both camera and lidar measurements had a significant effect on the performance of tracking. Ghost tracks which appeared in the lidar-only tracking in Step 3 disappeared much faster with camera measurements. Also, the RMSE tends to be lower, although just a bit.

Challenges for tracking in real-life scenarios are, e.g., the limitations of the respective sensors (the data for the project was taken on a clear day, which isn't always the case) and a trade-off between model complexity and computational performance. The more complex a model is, the better the results can be but also the more costly the computations become.

The results can be improved by, e.g., using a non-linear model, improving the association algorithm (global nearest neighbor or JPDA), or simply adding more sensors.