3D Object Detection Final Project Writeup

Report by: **Nolan Lunscher**

Report Submitted: **2021-10-16**

Repo: https://github.com/nlunscher/nd013-c2-fusion-starter (branch: devel-nl-final)

Introduction

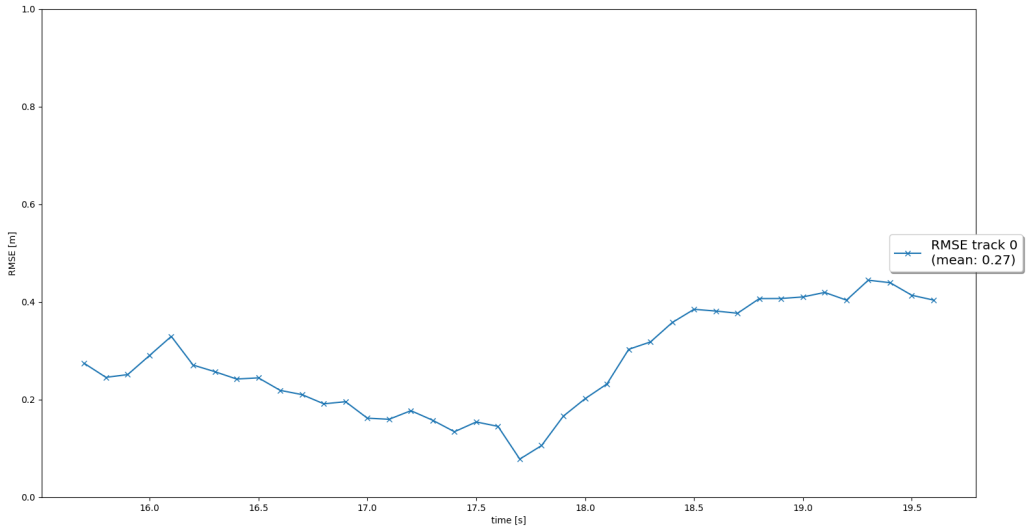
In this project we build upon the task of the midterm project, by applying sensor fusion and multiple object tracking. Here we use an EKF to fuse vehicle detections from a 3D lidar detector and a 2D image detector, and track them over time.

Part 1 EKF Tracking

Code for this part is found in **filter.py**

In this section we are tasked with adding the EKF prediction and update steps, which includes the system matrix F and process noise covariance Q, as well as the residual Gamma and covariance of residual S. With these two steps completed, we are able to fuse lidar detections over time into a single measurement that contains more information than any single detection such as velocity.

After running this section, the resulting RMSE of the track was 0.27 as shown below.

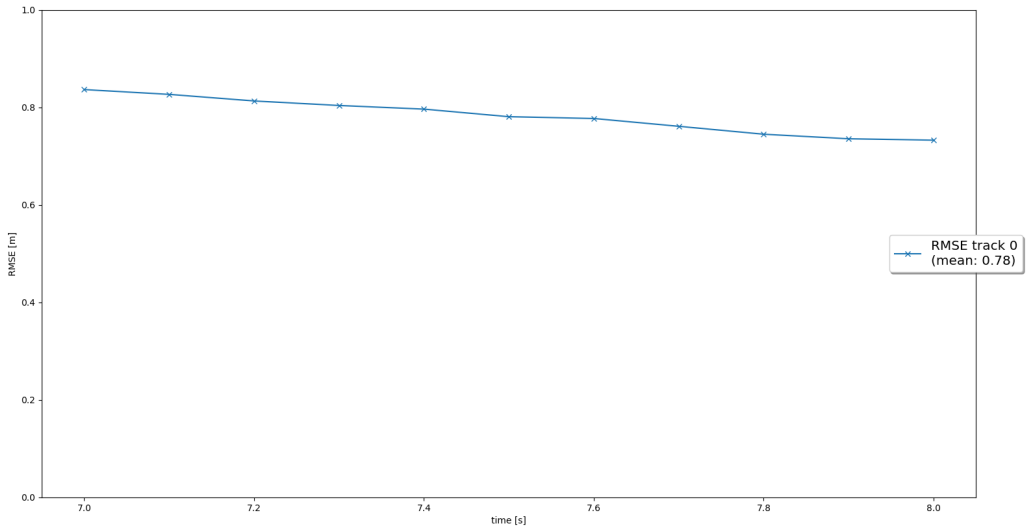


Part 2 Track Management

Code for this part is found in **trackmanagement.py**

In this section we are tasked with completing the Track and Trackmanagement classes responsible for tracking vehicle instances over time. This included initializing tracks with measurements as well as coding criteria for scoring tracks over time in order to confirm or delete them when needed.

After running this section, the resulting RMSE of the single track was 0.78 as shown below.

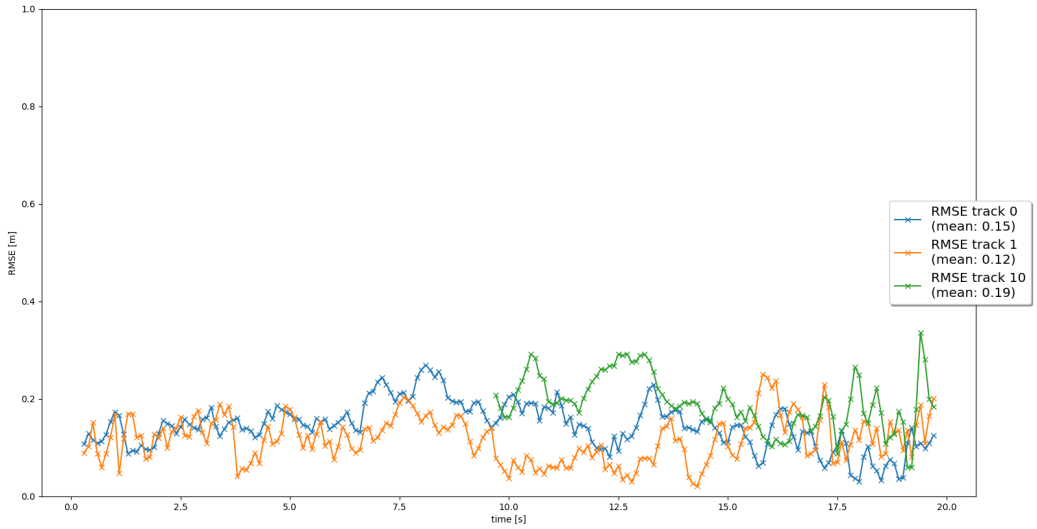


Part 3 Data Association

Code for this part is found in **association.py**

In this section we are tasked with completing the Association class that is responsible for determining which new measurement belongs to which existing track or if a new track is needed. Here we create the association matrix using mahalanobis distances, which incorporates covariance information. We also add a gating condition to filter away unlikely associations, which helps speed up the association algorithm.

After running this section, no ghost tracks occur and the resulting RMSE of tracks are all below 0.2 as shown below.

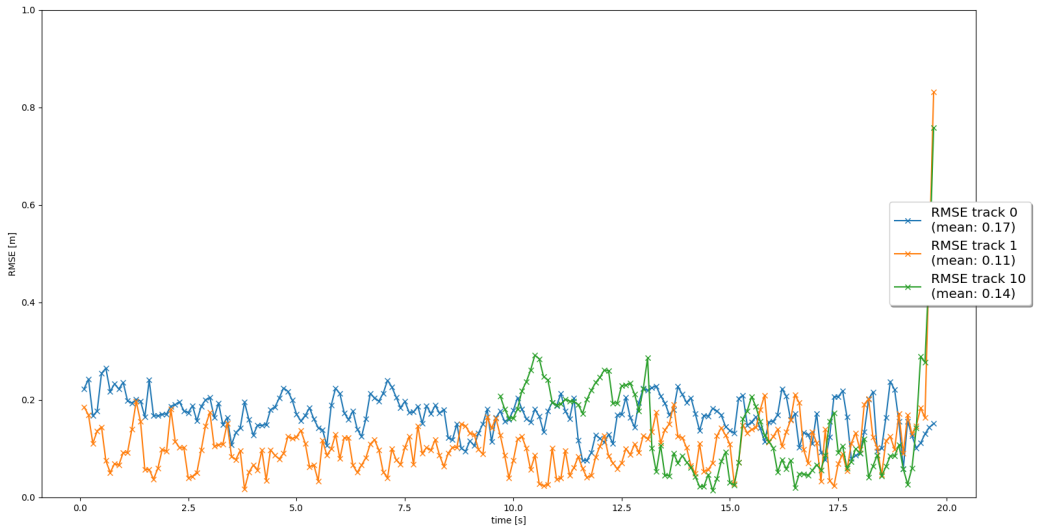


Part 4 Camera Measurement Fusion

Code for this part is found in **measurements.py**

In this section we are tasked with implementing the camera specific elements to the measurement model. These are necessary because the projection model of the camera introduces nonlinearity into the model that needs to be incorporated correctly. Due to the camera not measuring depth, this also introduces a change from lidar, where the Z coordinate of the detection is not measured.

After running this section, no ghost tracks occur and the resulting RMSE of tracks are all below 0.2 as shown below.



Most Difficult Parts of the Project

I found the most difficult part of the project to be expanding the EKF equations into 3D. The course went over the 2D version of the filter, but it wasn't immediately obvious how to apply the Z axis, especially for Q. After doing more thinking about how the matrix elements are applied during the various matrix multiplications it became more clear. After completing the filter component, it looks more obvious to me now, how the equations are set up.

Camera-Lidar Fusion Benefits

Object detections in cameras and lidars should be very complimentary since each modality excels at measuring different things. Cameras for example are great at capturing information for classification, such as high resolution textures. This information could be useful assigning motion models or other behaviours (e.g. vehicles and pedestrians may have different motion models). Lidars on the other hand are very good at measuring the 3D position of an object, but less so the finer details that may be needed to understand the object. Beyond this, using multiple sensors allows for more measurements of each object, which helps reduce the filter covariance since new measurements decrease the covariance when added to the filter. This can be seen where the RMSE dropped from 0.12-0.19 in Part 3 to 0.11-0.17 in Part 4 after the camera was added.

Real World Challenges of Sensor Fusion

One of the challenges I foresee when utilizing sensor fusion in the real world might be in determining accurate covariances for different situations. For example not every sensor manufacturer will provide this information, and for different use cases, that covariance may be different even if they did. For example, the confidence of camera detections would likely be different between sunny clear days, and snowy nights. In the project this issue was largely ignored, as the measurement noise covariances were provided.

Another challenge may be dealing with sensor faults. Sensor fusion has the benefits of allowing for redundancy, but it may be challenging to tune the system if those sensors don't agree, such as if the cameras and lidars are detecting very different objects in the same FOV (e.g. faulty sensor or faulty algorithm) . In these cases it may be difficult to determine which sensor to trust.

Possible Improvements

One thing that could be done to improve the tracking results would be to add more measurements into the tracking model. This way more information is added to the tracker rather than relying on the update model. There are two main ways this can be done. One way would be to use sensors and detectors that run at a faster frame rate, this would reduce the amount of time between measurements where the motion model is active. Another way might be to add more sensors and modalities that can further complement the existing ones. Either of these solutions are likely to add cost to the overall project however, but would likely result in better results.

Another possible improvement might be to incorporate semantic information into the motion model. For example different classes of vehicles could have their own motion models, or the motion model could incorporate different types of measurements, such as whether or not a turn signal is detected. This could potentially better allow for tracking and prediction of objects in more complex scenarios.