Final Project: Sensor Fusion and Object Tracking

I have attached my code in the submitted file with this zip folders including loop_over_dataset.py

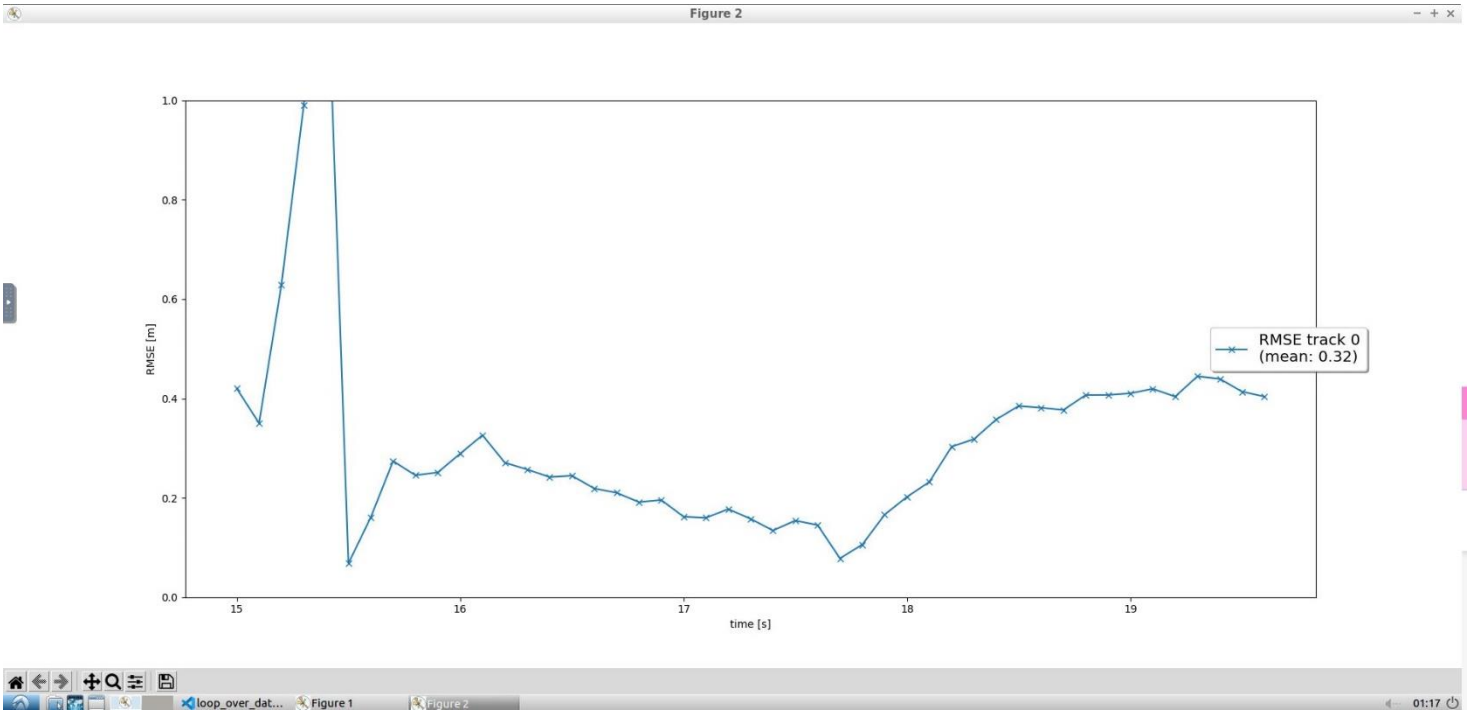
Please find the attached pictures below for the Step 1-4 of the project and RMSE values at each step.

Step 1: **Tracking**

Within the "filter.py" file, the Extended Kalman Filter (EKF) is put into use.

To start, we establish the system states as [x, y, z, vx, vy, vz], construct the process model, and specify the constant velocity model. Following this, we proceed to deduce the system matrix for the 3D process models, considering the presence of constant velocity and noise covariances. This phase is pivotal for the computation of both the state function denoted as h(x) and the Jacobian matrix referred to as H.

To ascertain the current state, we assess the h(x) function along with its corresponding Jacobian matrix, H. The Kalman gain is subsequently determined and used to update both the state itself and its associated covariance.



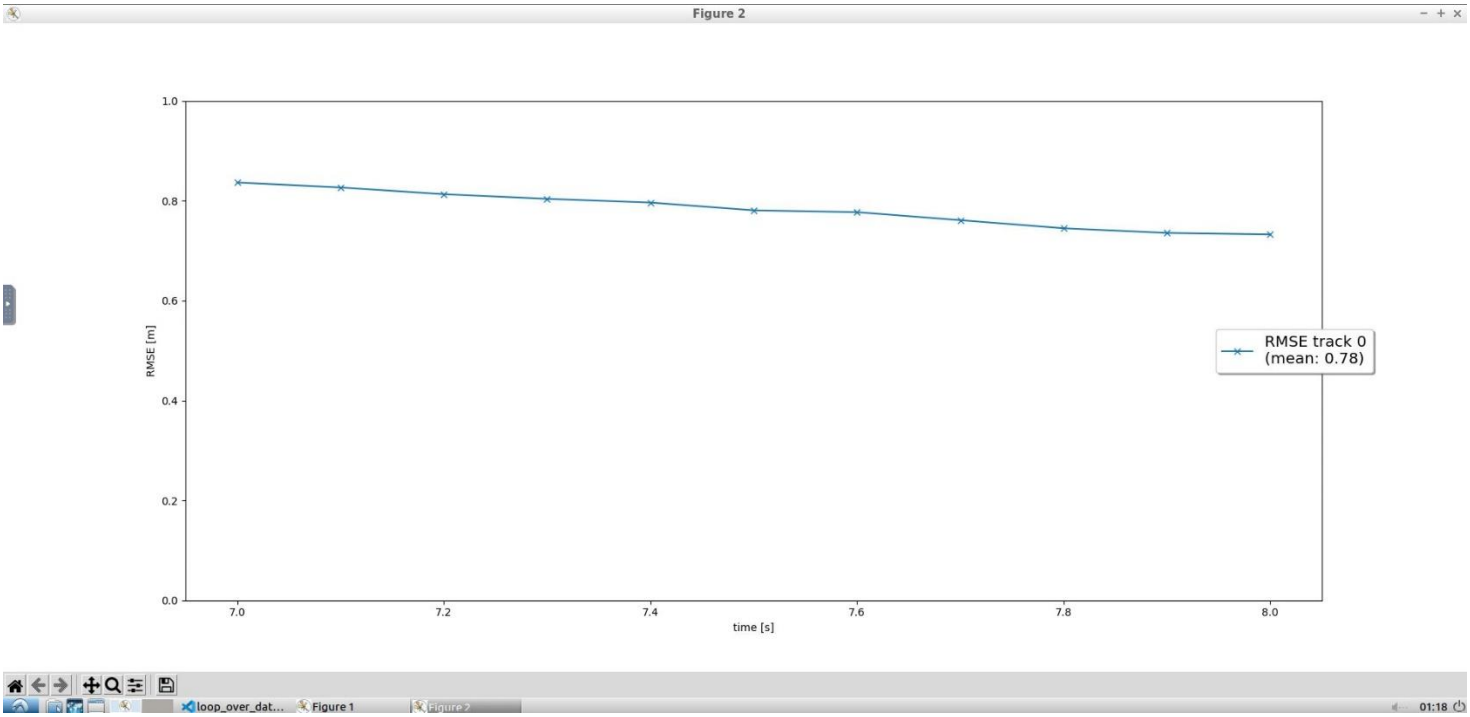
Step 2: **Track Management**

We now turn our attention to the field of track management, where the tracklist is equipped to manage multiple objects simultaneously. In our device architecture, we represent each object as a track. To ensure these tracks remain up-to-date, we transfer both track information and measurement data to the Kalman filter, which is responsible for managing the specific task associated with each track.

Initially, the track is initialized with lidar calculations that have not yet been assigned.

When the track's scores are in alignment with the measurements, the corresponding scores are increased, and vice versa. We employ a track ranking system to adjust the track's conditions.

If a track's score drops below a specific three-point threshold and the state balance surpasses a certain level, the track is retained for further consideration and is not eliminated.



Step 3: **Data Association**

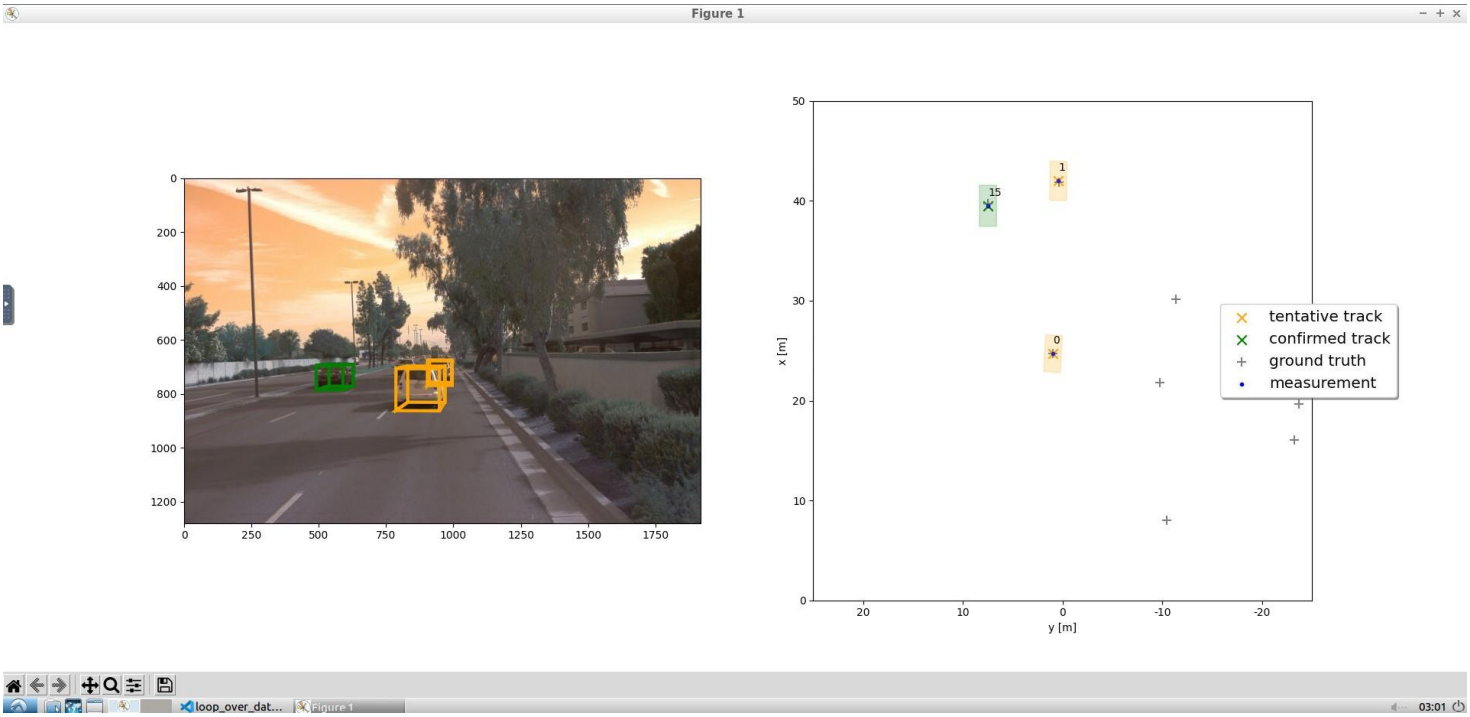
In this phase, the nearest neighbor association process effectively matches multiple measurements with multiple tracks. Within the association.py file, the concept of data association is introduced, and the procedure proceeds as outlined below.

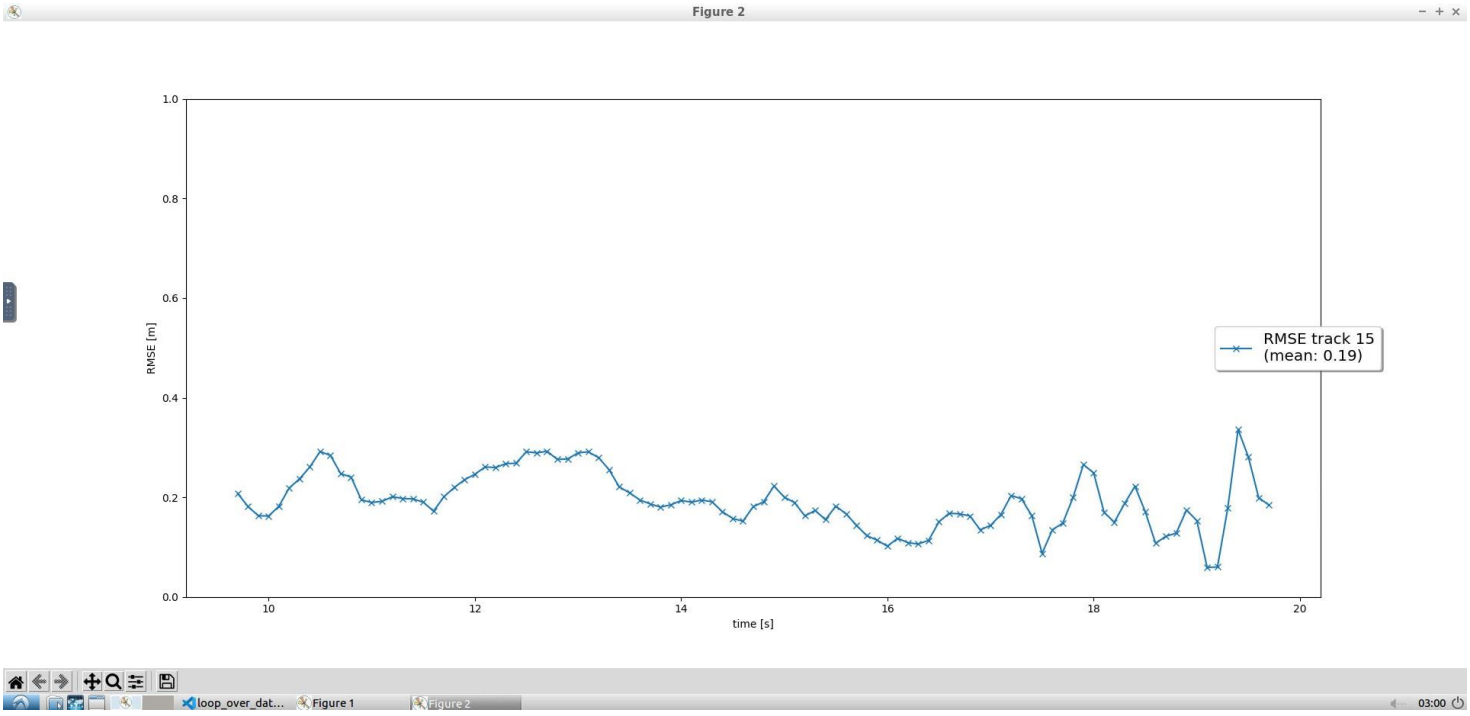
We generate a matrix that encompasses all tracks and available observations.

We calculate the Mahalanobis Distance for each measurement in relation to each track.

A Chi-Square hypothesis test is employed to filter out unlikely track-measurement pairs.

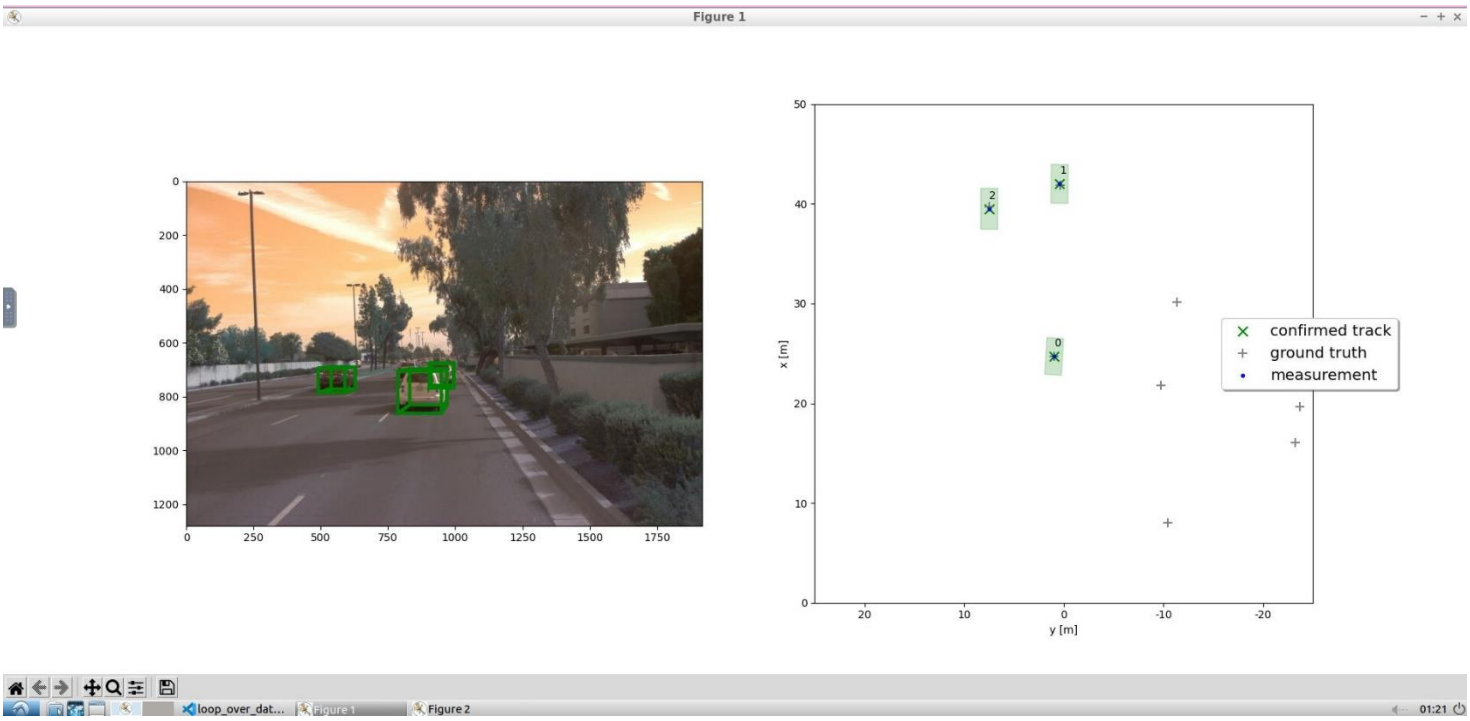
The pair with the smallest Mahalanobis Distance is chosen and utilized to update the Kalman Filter. Subsequently, the corresponding row and column in the relationship matrix are eliminated.

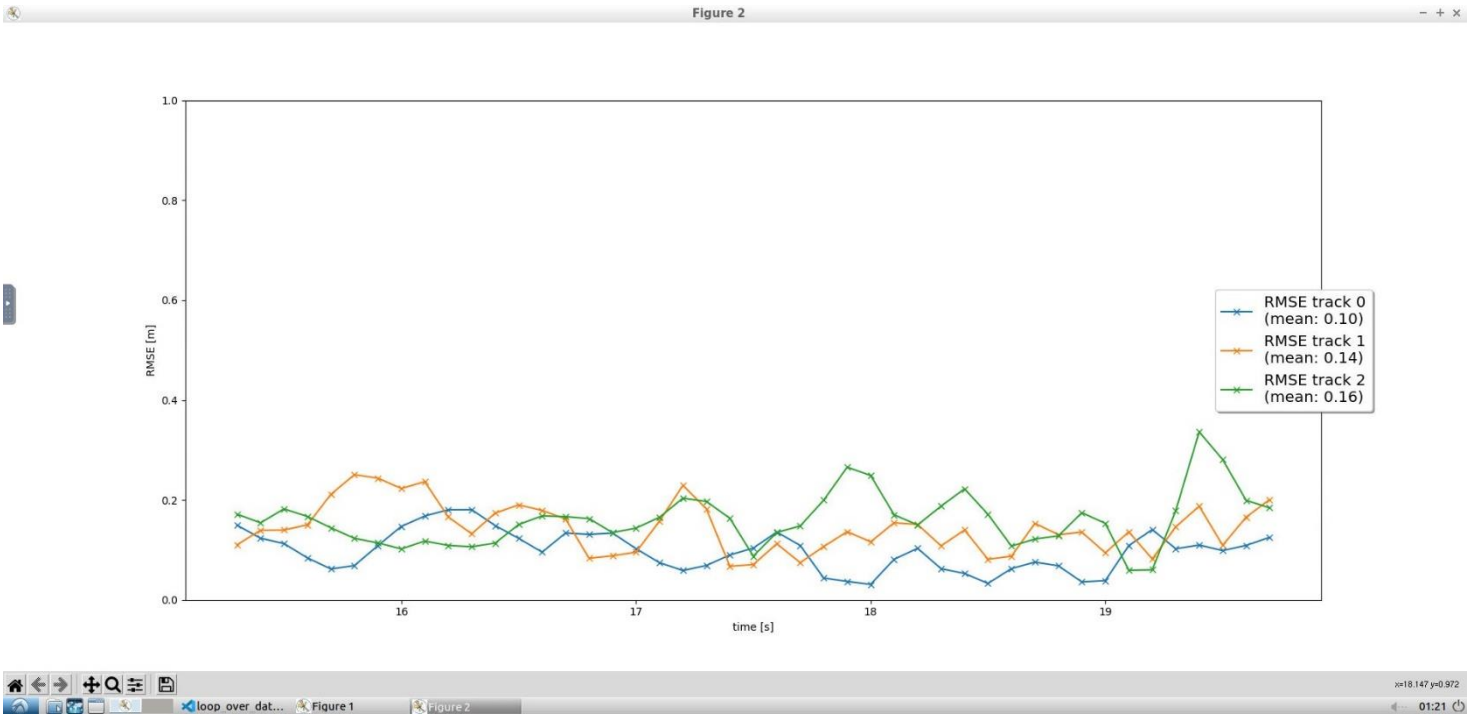




Step 4: **Sensor Fusion**

We will now introduce an enhancement to the Kalman filter. The fundamental assumption in this context is that the center of a car's 3D space bounding box corresponds to the center of the vehicle's 2D imagery. While this assumption is generally valid, it may not always hold true, especially for a front-facing camera. The implementation incorporates a projection matrix that converts points from 3D space into 2D image coordinates. We utilize partial derivatives (x, y, z) to map these points to model parameters (u, v) while also considering noise (R). If the tracking status is within the Field of View (FOV), we accept the measurement-track pair; otherwise, it is rejected.





- ## Difficulties Faced in Project:

  There were some difficulties faced during the project such as project files and some parameters were hard to find from the library packages. I had to reverse engineer some parts of the code given to understand the implementation. Also, the workspace on the virtual machine keeps losing the connections which becomes a huge issue for project running. Nevertheless, I would say that some of the algorithms such as EKF, Data Association and Track Management were easy to implement because of the exercises and solutions provided in the course lectures.

- ## Benefits in Camera-Lidar Fusion tracking over Lidar-only tracking:

  The debate for this is still present in the automotive industry leaders, but I personally found that it is a stable and reliable tracking. I think it would be even better if we could add one more sensor other than a camera and a lidar to fuse and eventually gain more confidence in object detection and tracking. LiDAR does a tremendous job in the multi-dimensional projections and tracking the objects. Different deep neural network algorithms such as YOLO, ResNet and DarkNet and their evolved versions can provide enhanced benefit in using Camera-Lidar fusion tracking methods.

- ## Real-life challenges:

  The only real-life challenge that I can think of is, the ambiguity in the decision making from the data in camera and lidar. This can either lead to less confident decision in object tracking or create a false negative detection and eventually a degraded performance. Also, LiDAR has a chance of detection ghost objects more than the camera.