I have started working on this project based on the mid-term 3d object detection project. For final project I used fpn_resnet model. Pre-computed results for fpn_resnet are in results directory of fpn_resnet folder. Some changes have done in loop_over_dataset to get the lidar detection and tracking loop result properly.

There are 4 steps in the project.

1. Tracking, track objects over time with a Kalman Filter

2. Track Management, initialize, update and delete tracks

3. Data Association, associate measurements to tracks with nearest neighbor association

4. Sensor Fusion, SWBAT fuse measurements from lidar and camera

Track objects over time with an Extended Kalman Filter is the first step.

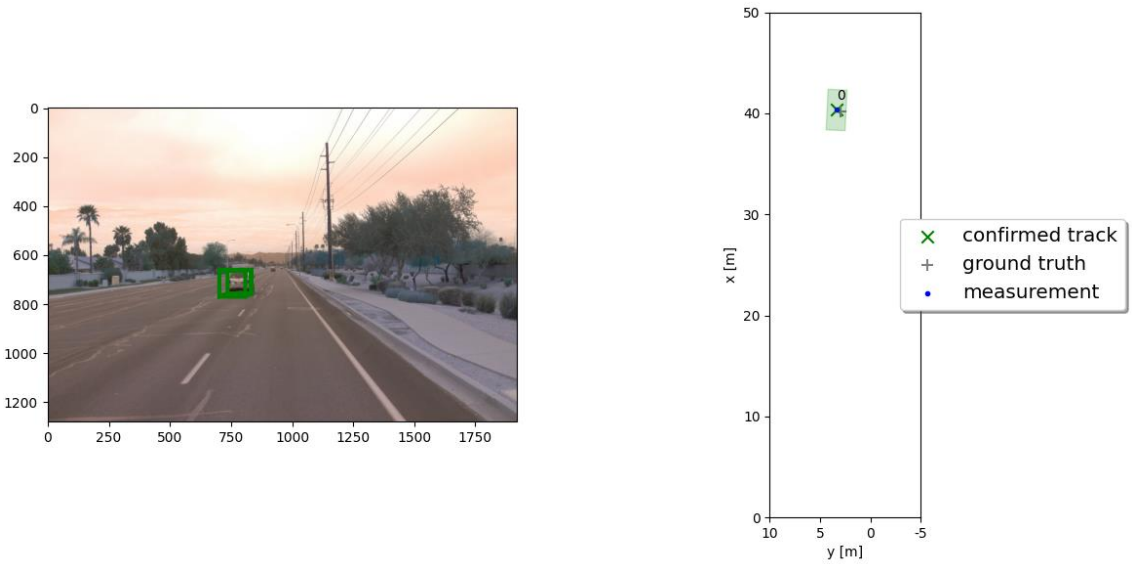
Wrote code within the file student/filter.py to implement this step.

1. In this step implemented predict () function for an EKF.

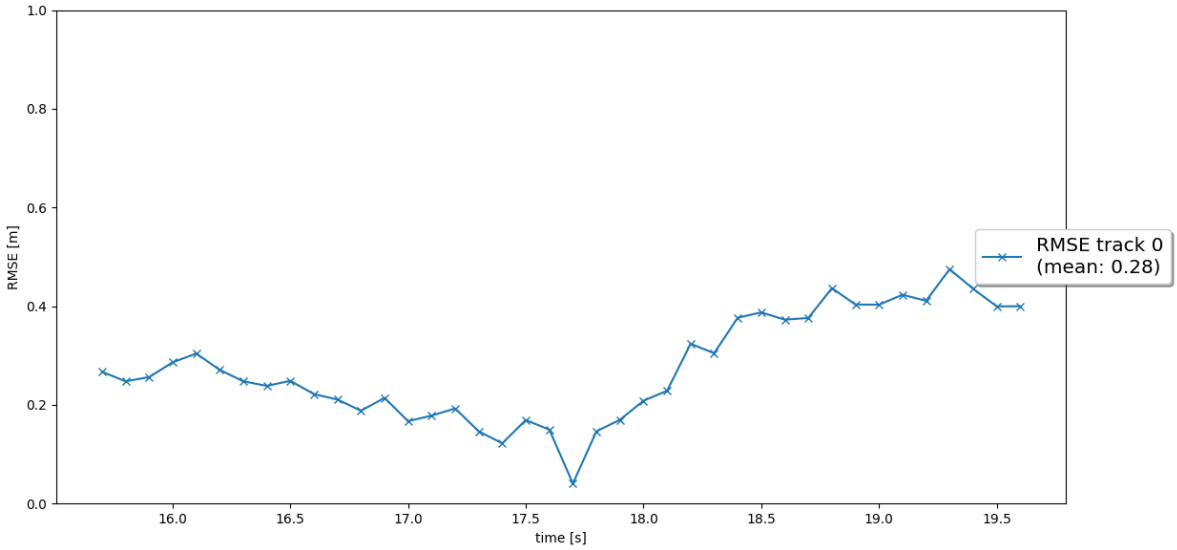
2. Calculated a system matrix for constant velocity process model in 3D using F() and Q() functions.

3. dt(delta time) is fixed and loaded it from misc/params.py file to calculate the corresponding process noise covariance depending on the current timestep dt.

4. Implemented the update() function as well as the gamma() and S() functions for residual and residual covariance. Implemented this step with lidar measurements model.



Single target tracking results



The mean RMSE is 0.28 which is smaller than 0.35.

Track Management

Wrote code within the student/trackmanagement.py file to implement this step.

1. Replaced the fixed track initialization values by initialization of track.x and track.P based on the input meas, it is an unassigned lidar measurement object of type Measurement.

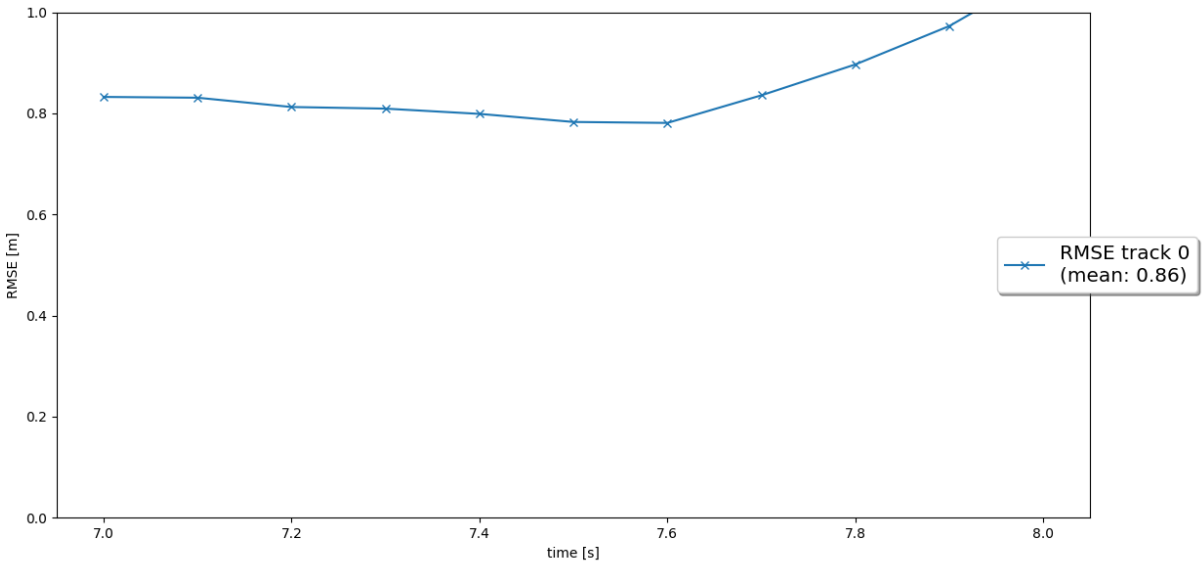
2. Transformed the unassigned measurement from sensor to vehicle coordinates with the sens_to_veh transformation matrix implemented in the Sensor class.

3. Initialized the track state with 'initialized' and the score with 1./params.window.

4. In the Trackmanagement class, implemented the manage_track to decrease the track score for unassigned tracks and delete tracks if the score is too low or P is too big.

5. In the Trackmanagement class, implemented the handle_updated_track() function to increase the track score for the input track and set the track state to 'tentative' or 'confirmed' depending on the track score.

The visualization shows that a new track is initialized automatically where unassigned measurements occur, the true track is confirmed quickly, and the track is deleted after it has vanished from the visible range.



RMSE plot

Data Association

Wrote code within the student/association.py file to implement this step.

1. In the Association class, implemented the associate() function where replaced association_matrix with the actual association matrix based on Mahalanobis distances for all tracks in the input track_list and all measurements in the input meas_list.

2. In MHD()function implemented the Mahalanobis distance between a track and a measurement

3. Used the gating_ok() function with chi-square-distribution to check if a measurement lies inside a track's gate. If not, the function shall return False and the entry in association_matrix set to infinity.

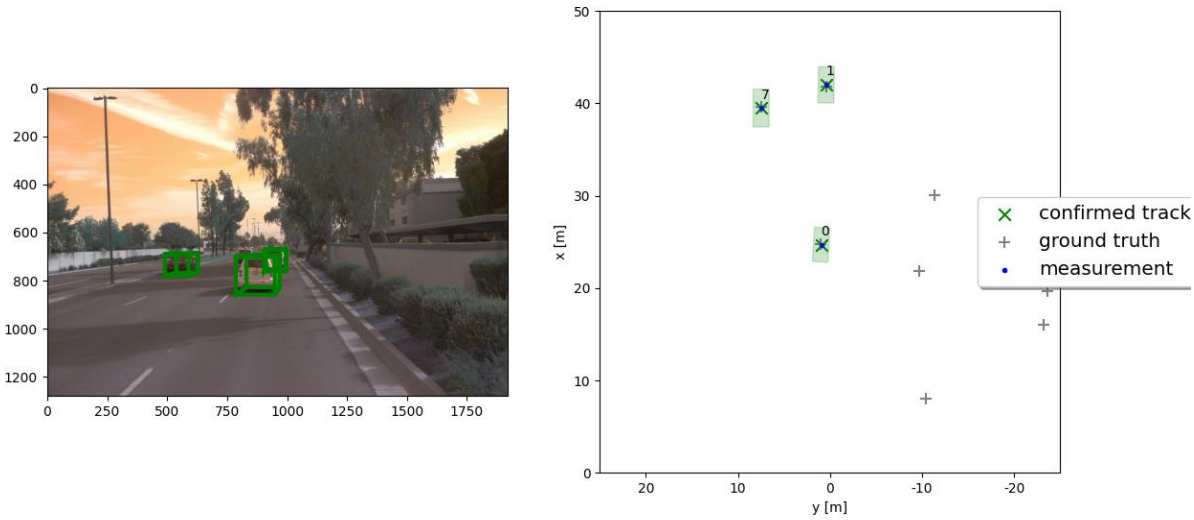
4. Updated unassigned measurements and unassigned tracks.

5. Found the minimum entry in association_matrix, deleted corresponding row and column from the matrix.

6. Removed the corresponding track and measurement from unassigned_tracks and unassigned_meas.

7. Between track and measurement returned numpy.nan for the track and measurement

The visualization shows that there are no confirmed “ghost tracks” that do not exist in reality. There are initialized and tentative “ghost tracks” as long as they are deleted after several frames.



Tracking result with three tracks from Sequence 1

Sensor Fusion of LiDAR and camera

Wrote code within the student/measurements.py file to implement this step.

1. First transformed the vehicle to sensor coordinates. In the Sensor class, implemented the function in_fov() that checks if the input state vector x of an object can be seen by this sensor. The function returns True if x lies in the sensor's field of view, otherwise False.

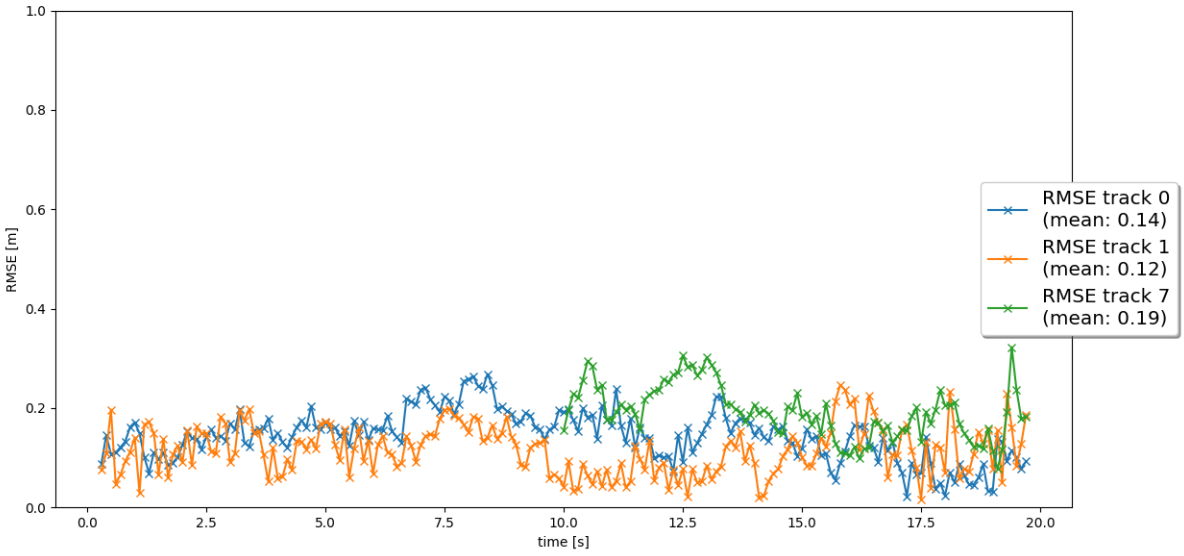
2. Implemented the function get_hx() with the nonlinear camera measurement function . transformed position estimate from vehicle to camera coordinates, projected from camera to image coordinates and it is not dividing by zero so returned h(x).

3. Removed the restriction to lidar in the function generate_measurement() in order to include camera as well.

4. In the Measurement class, initialized camera measurement objects including z, R, and the sensor object sensor.

5. Finally made a movie to showcase tracking results.

The visualization shows that the tracking performs well, again no confirmed ghost tracks or track losses should occur.



RMSE for the three valid tracks in Sequence 1

Difficulty I faced

All step to-do implementation is described in detail but I faced difficulty in step-2 and step-4 which are trackmanagement.py and measurements.py. I faced difficulties with decreasing track score and deleting old track in trackmanagement.py file. Problem was in pre computed results as I used it from midterm project so I changed the pre computed according to final project results also made some changes to loop_over_dataset.py file then overcome it. I also faced difficulty understanding the nonlinear measurement expectation value h(x) it took time but finally implemented nonlinear camera measurement function

Benefits in camera-lidar fusion compared to lidar-only tracking

Theoretical explanation of LiDAR and Camera.

Camera Sensor

- Cameras excel at classifying objects such as vehicles, pedestrians, speed signs and many others. This is one of the prime advantages of camera systems and recent advances in AI emphasize this even stronger.
- Camera systems are the only sensor able to interpret two-dimensional information such as speed signs, lane markings or traffic lights, as they are able to measure both color and light intensity. This is the primary advantage of cameras over the other sensor types.

LiDAR Sensor

- LiDAR have an excellent robustness in darkness, as it is active sensor.
- LiDAR system can detect objects at distances ranging from a few meters to more than 200m.

More specifically, vision-based approaches are more robust and accurate in object detection but fail in estimating the distance of the object accurately. In contrast, LiDAR-based methods are very robust and accurate in measuring the distance of the object but are limited by the object classification ability.

Camera-LiDAR fusion will increase the robustness and accuracy of the tracking system.

In our project when we used Camera-LiDAR fusion which has a wider range of vision that means it can detect more objects accurately.

Challenges will a sensor fusion system face in real-life scenarios

The fusion of light detection and ranging (LiDAR) and camera data in real-time is known to be a crucial process in many applications, especially in the case of autonomous vehicles.

Sensor fusion has been around for some time, but has been evolving at an accelerated rate in recent years. New technologies like multisensory data fusion, big data processing, and deep learning are changing the quality of areas of applications, improving sensors and systems used. This project allows us to handle some typical challenges in real-life sensor fusion and tracking algorithm, including

- It is difficult to get an accurate association between multiple tracks and multiple measurement. A good gating threshold helps to increase accuracy because whenever it perfectly rejects incorrect measurement and track pair tracking works better.
- It is also difficult for measurement noise configuration. In fact, it's better that each measurement should come with its own noise variance, as opposed to setting a fixed noise variance for a sensor

To calibrate the LiDAR and camera sensors and involves the estimation of the extrinsic parameters along with the camera's intrinsic parameters. In real-life project one common challenge is this project does not experience is extrinsic parameters calibration. The camera's intrinsic parameters need to be estimated by using the traditional checkerboard calibration method, and the LiDAR and camera's extrinsic parameters need to be estimated using a planar 3D marker board. A correct configuration for lidar and camera extrinsic parameter is vital to the success of a sensor fusion and tracking project. In this project, these extrinsic parameters are known as we are using a public dataset.

Improvements of tracking results in the future

The improvement of the performance of data association and movement classification can be done by using vision object class and shape information for choosing the method for object detection. As camera measurement model helps to get the good fusion result so based on the distance of the object from the vehicle, a tracking system can be switched between a point and a 3D box models by matching 3d points in lidar point cloud to vehicle object pixels in image.