Final Project: Sensor Fusion and Object Tracking

Introduction

This project deals with implementing sensor fusion system that is able to track vehicles over time with real world camera and LiDAR measurements. Following are the main steps as mentioned in the course introduction

Step 1: Implement an extended Kalman filter

Step 2: Implement track management including track state and track score, track initialization and deletion

Step 3: Implement single nearest neighbour data association and gating

Step 4: Apply sensor fusion by implementing the nonlinear camera measurement model and a sensor visibility check

---

Step 1:

The objective in Step 1 is to implement an Extended Kalman Filter (EKF) to track a single real-world target with LiDAR measurement input over time. A simple single-target-scenario is already prepared. Mid- term project solution code is copied in student/objdet_detect.py.

Parameters:

File: loop_over_dataset.py

training_segment-10072231702153043603_5725_000_5745_000_with_camera_labels.tfrecord

show_only_frames = [150, 200]

configs_det = det.load_configs(model_name='fpn_resnet')

configs_det.lim_y = [-5, 10]

exec_detection = []

exec_tracking = ['perform_tracking']

exec_visualization = ['show_tracks']

After setting the parameters in loop_over_dataset.py code was written in student/filter.py.

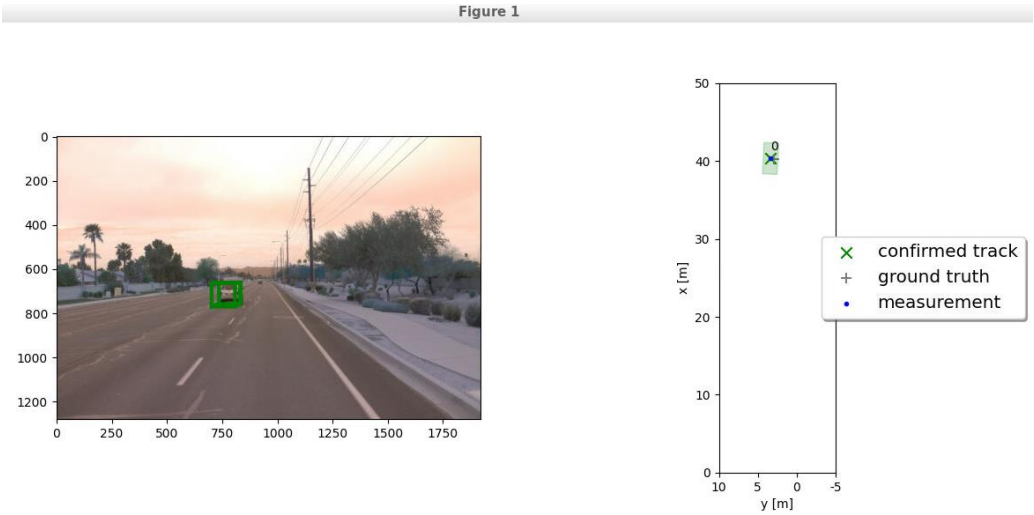

Fig. 1 Single object tracking using extended Kalman Filter. Here we can observe confirmed track.

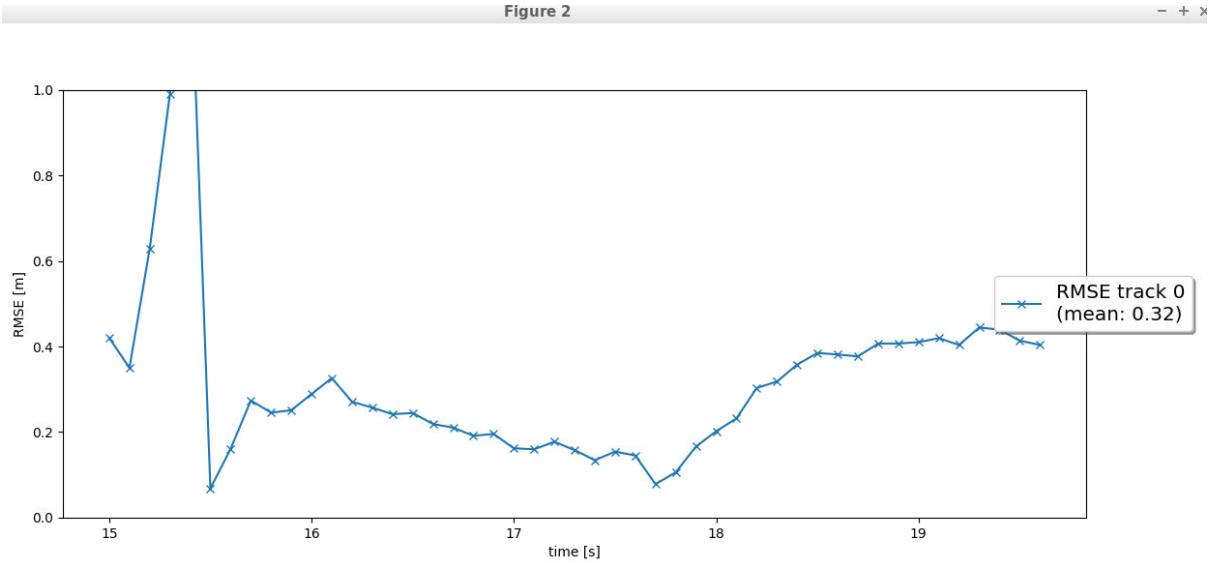

Fig. 2 RMSE plot with mean 0.32

Step 2:

The objective is to implement track management to initialize and delete tracks, set a track state and a track score. Fixed track initializations have been replaced with initializations based on LiDAR measurement. The measurements are then transformed into vehicle coordinates. Following functionalities are implemented

a. Track score decrease for unassigned tracks
b. Deletion of tracks if score is too low
c. Track score increments for input track
d. Setting track state to tentative or confirmed based on track score

Parameters:

File: loop_over_dataset.py

show_only_frames = [65, 100]

configs_det.lim_y = [-5, 15]

File: params.py

Delete_threshold = 0.3

After setting the parameters, code was written in student/filter.py.

Observation: After the implementation, the visualization mentioned below 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.

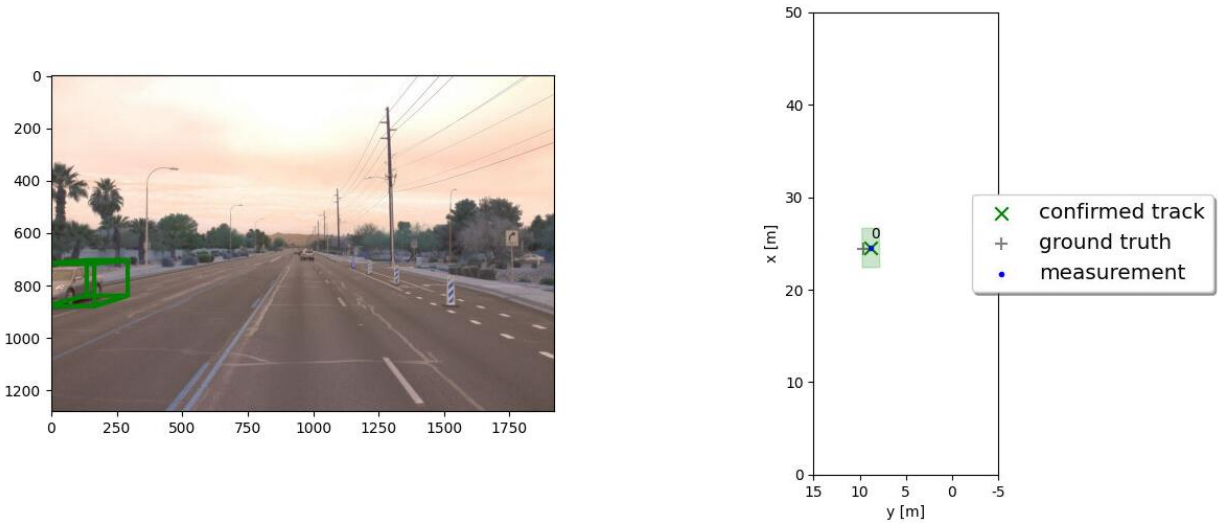

Fig. 3 Single object tracking based on measurement with track management

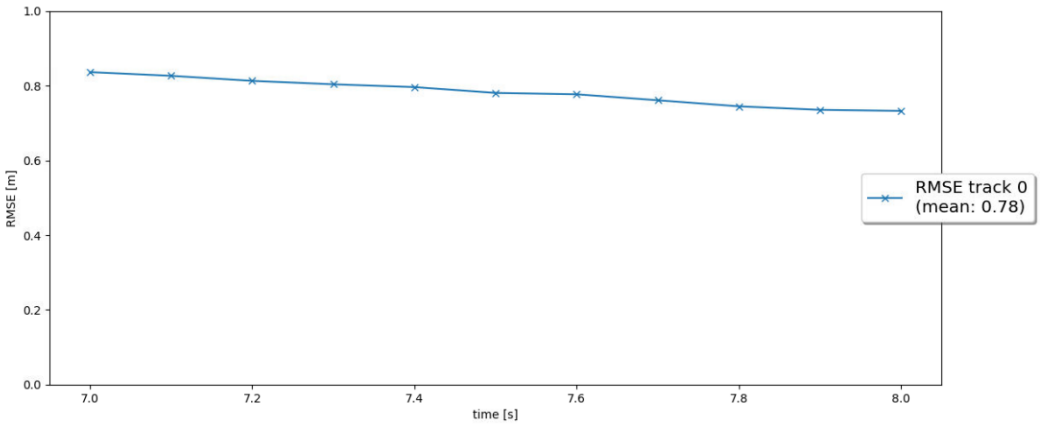

Fig. 4 RMSE plot with mean 0.78

Step 3:

This step is about implementing a single nearest neighbour data association to associate measurements to tracks.

Parameters

File: loop_over_dataset.py

training_segment-10050810020241296 53_5313_150_5333_150_with_camera_labels.tfrecord

show_only_frames = [0, 200]

configs_det.lim_y = [-25, 25]

File: params.py

Delete_threshold = 0.3

After setting the parameters in loop_over_dataset.py code was written in student/association.py. Main steps for coding are

a. Replacing association_matrix with the actual association matrix based on Mahalanobis distances for all tracks in the input
b. Updating list of unassigned measured and unsassigned tracks
c. Implementing get_closest_track_and_meas() in Association class

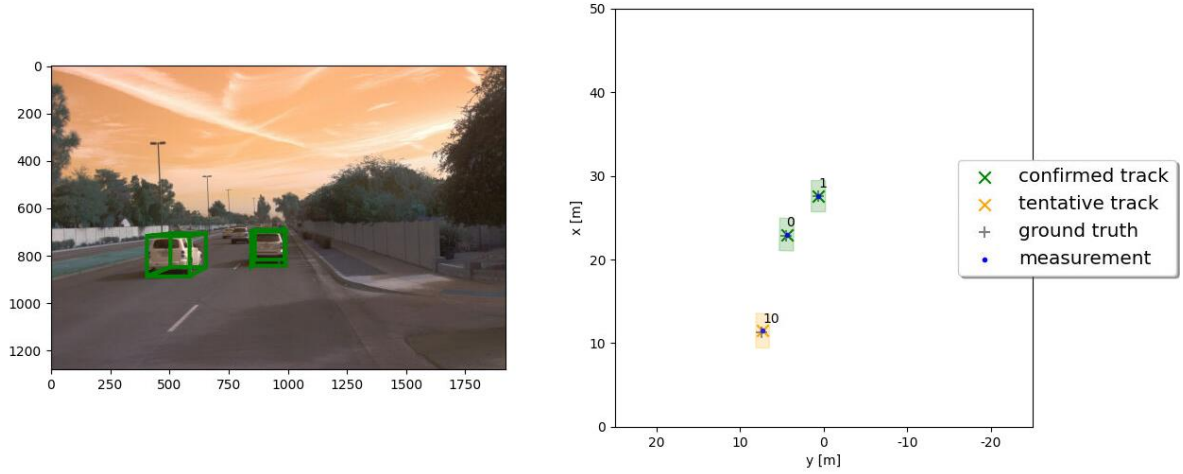

Fig. 5 Multi object tracking

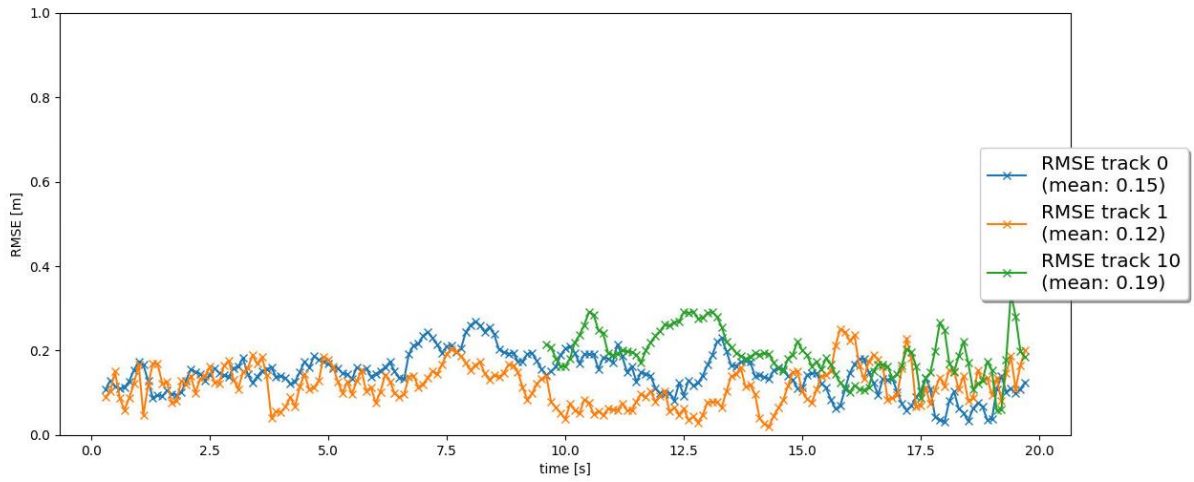

Fig. 6 RMSE plot for various tracks

Step 4:

The objective is to implement non-linear camera measurement model

Parameters: (same as in Step 3)

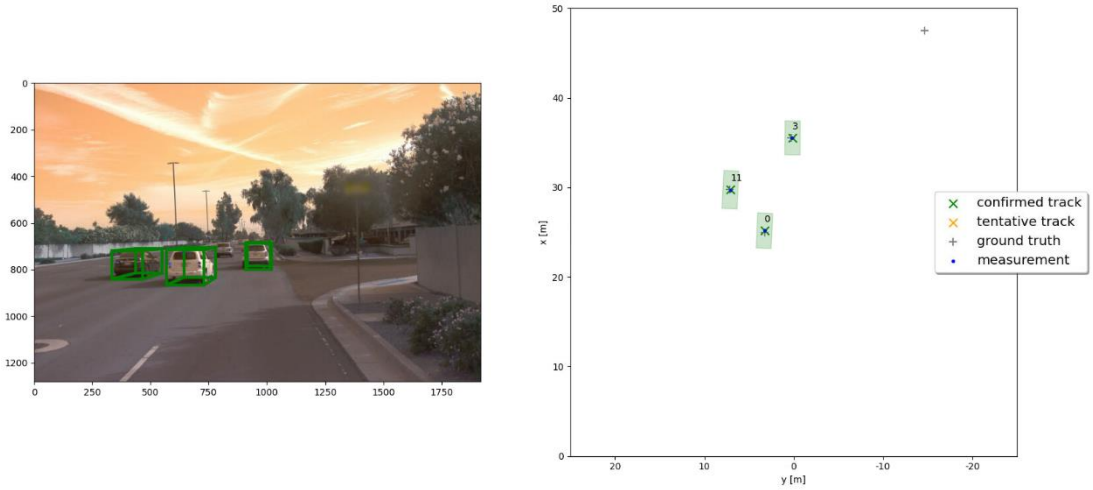

Fig. 7 Multi object tracking after implementation of non-linear camera measurement model

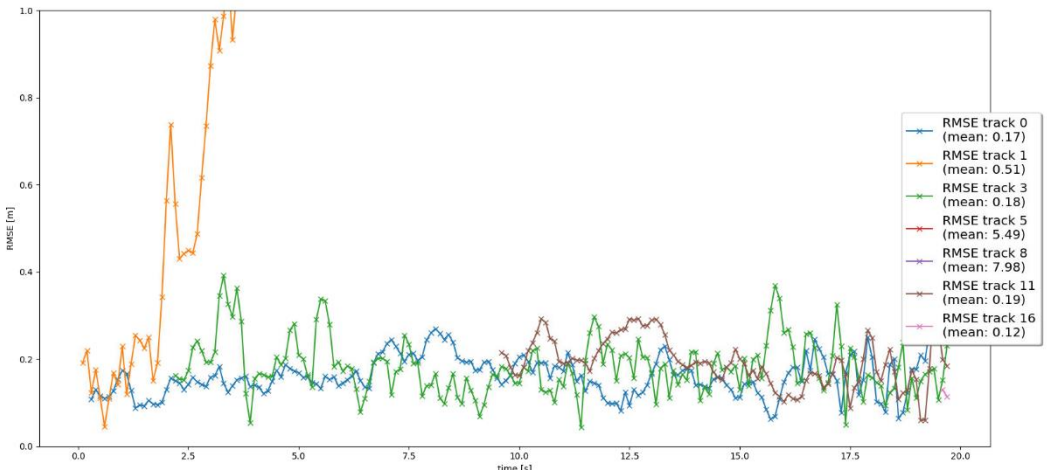

Fig. 8 RMSE plot for various tracks

Questions

a. 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?

Ans: In the track management implementation, I was not able to confirm the tracks. Finally, made change in params.py for delete_threshold and then the RMSE plot was generated.

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

Ans: The combination of camera and LiDAR can solve problems in real time scenario such as detection of vehicle features like number plates, tires, windshield, mirrors etc. Also, it helps in avoiding occlusion, formation of sparse cloud which is generated mainly from LiDAR based detection. On the other hand, the limitation of camera lies in detection during odd weather conditions like rain, fog and dust conditions. Also, camera cannot accurately detect depth of the object which LiDAR can.

Therefore, in principle, camera-LiDAR fusion can address such limitations. As compared to the mid term project which was only LiDAR based, we can observe that object detection are tracking is more efficient.

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

Ans: Following are the challenges

- There are cases when there are many objects and the tracks of vehicles may be crossing each other leading to loss of tracking.
- There are cases of where object detection and tracking of occluded object is difficult
- Shape of the object may vary depending on the direction from which it is viewed. There are cases of missing the track.

d. Can you think of ways to improve your tracking results in the future?

Ans: Following things can be explored

- There can be improvement in detecting false trucks.
- Tracking results and depth estimation can be improved by fusing the data from other sensors like RADAR as it can detect the depth and velocity accurately.
- The tracking results should be checking in rainy and foggy condition.
- If vehicles are coming from both directions then the detection will be complicated.