Project 5 Extended Kalman Filters

July 24, 2024

1 Project Goals

The goals of this project are the following:

1. Estimate 2-D car position with sensor data and Kalman Filter.
2. Implement Kalman Filter on Lidar data and Extended Kalman Filter on Radar Data.
3. Compute Root Mean Squared Error to measure the difference between estimate and actual position.

2 Write up

In this section I will briefly go over the C++ code and Environment setup.

2.1 Kalman Filter with C++

Three main files edited are `kalman_filter.cpp`, `FusionEKF.cpp` and `tools.cpp`.

1. `kalman_filter.cpp` follows the calculations in Figure 1, defines a Kalman Filter calculation object `KalmanFilter` and several main function:
   (a) `Init`: initializes Kalman Filter
   (b) `Predict`: predicts the state and state covariance
   (c) `Update`: update the state using standard Kalman Filter equations.
   (d) `UpdateEKF`: update the state using Extended Kalman Filter equations.
   (e) `UpdateCal`: includes the same equations used in both Kalman Filter and Exteneded Kalman Filter
2. `FusionEKF.cpp` reads the data from Lidar and Radar, calls Kalman Filter calculation object (`ekf_` in the code) to get the Kalman Filter estimate Position. The overall processing flow is shown in Figure 2.
3. `tools.cpp` includes supporting function, Jacobian Matrix calculation and RMSE calculation.

Kalman Filter (used for laser)

- **Prediction**

$$x' = Fx + u$$
$$P' = FPF^T + Q$$

- **Measurement Update**

$$y = z - Hx'$$
$$S = HP'H^T + R$$
$$K = P'H^T S^{-1}$$
$$x = x' + Ky$$
$$P = (I - KH)P'$$

Extended Kalman Filter (used for radar)

- **Prediction**

$$x' = f(x,u)$$
$$P' = F_j P F_j^T + Q$$

- **Measurement Update**

$$y = z - h(x')$$
$$S = H_j P' H_j^T + R$$
$$K = P'H_j^T S^{-1}$$
$$x = x' + Ky$$
$$P = (I - KH_j)P'$$

Figure 1: Kalman Filter Formulas

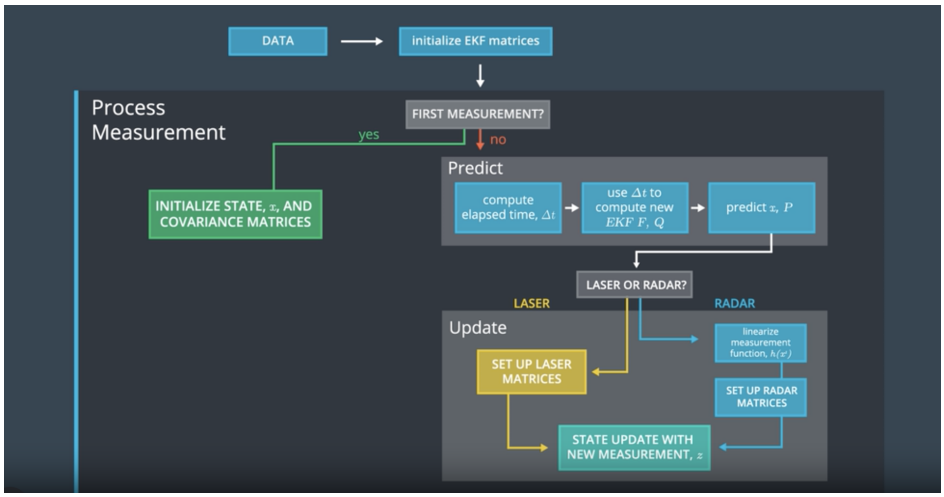
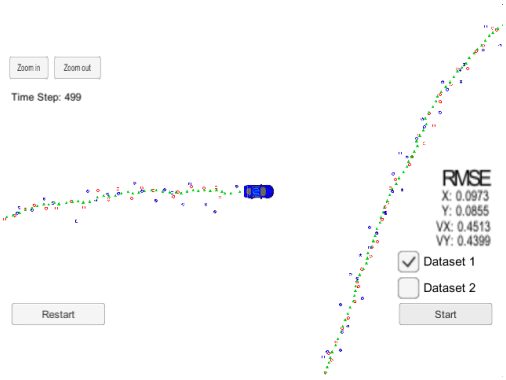


Figure 2: Sensor Fusion Process

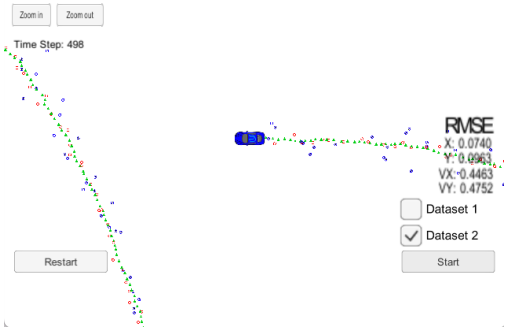
2.2 Environment setup

The setup in Mac is relative straightforward, download and extract term2 simulator and follow the instructions to compile the C++ code and connect the code to the simulator.

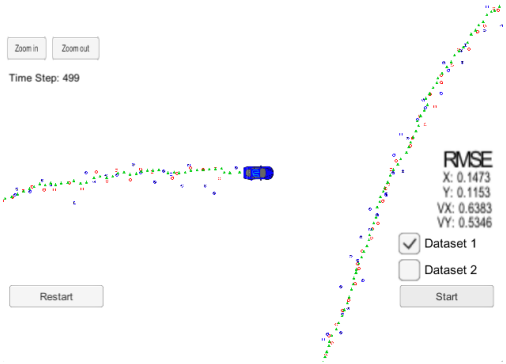
- `mkdir build`
- `cd builf`
- `cmake ..`
- `make`
- `./ExtendedKF`



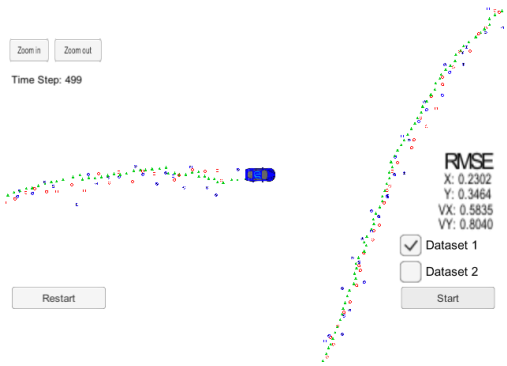
(a) RMSE Dataset 1



(b) RMSE Dataset 2



(a) Turn off Radar



(b) Turn off Lidar

2.3 Result

Final Result is shown in Figure 3a and Figure 3b. The px, py, vx, and vy RMSE for Dataset 1 and 2 is less than [.11, .11, 0.52, 0.52].

3 Discussion

A bug I wrote is modifying P matrix calculation in the predict process again, P matrix is inherited from the last Kalman Filter iteration and there is no need to manually input it again.

A simple test is done by turning off Radar or Lidar calculation, from the RMSE result we can see estimate from Lidar is more accurate, and by fusing the data from Radar the accuracy is further improved.