Extended Kalman Filter Project

This project utilized a Kalman Filter to estimate the state of a moving object of interest with noisy lidar and radar measurements. The project obtained the RMSE values that are lower than the tolerance outlined in the project rubric.

Compiling

This project made a ‘build’ directory in the project folder, changed the file ‘CMakeLists.txt’, and start compiling by doing the following:

cmake ..

make

The C++ code will be compiled without errors.

Accuracy

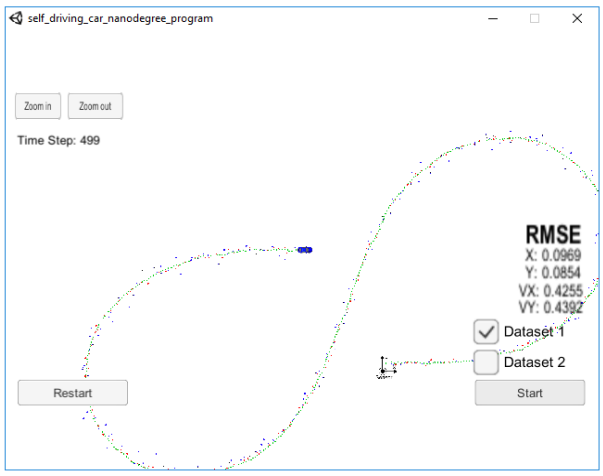
To start the simulator, launched it and select the EKF/UKF project, running command:

./ExtendedKF

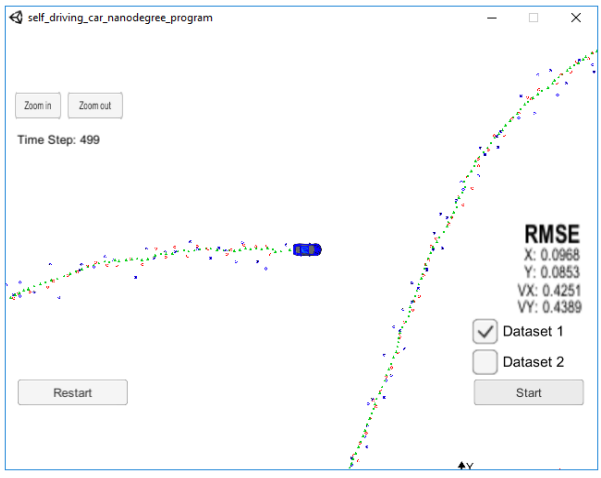
Then, click start in the simulator.

The algorithm is run against Dataset 1 in the simulator which is the same as "data/obj_pose-laser-radar-synthetic-input.txt" in the repository. We collected the positions that the algorithm outputs and compare them to ground truth data. the px, py, vx, and vy RMSE is less than or equal to the values [.11, .11, 0.52, 0.52]. See the picture A and B for the RMSE based on the Dataset 1 as following:

Picture A for Dataset 1

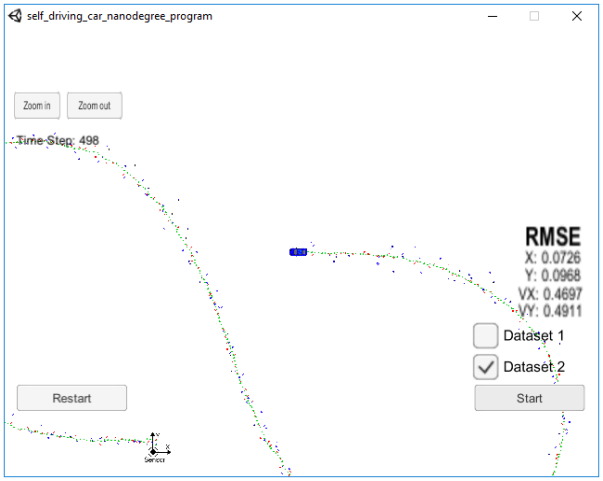


Picture B for Dataset 1

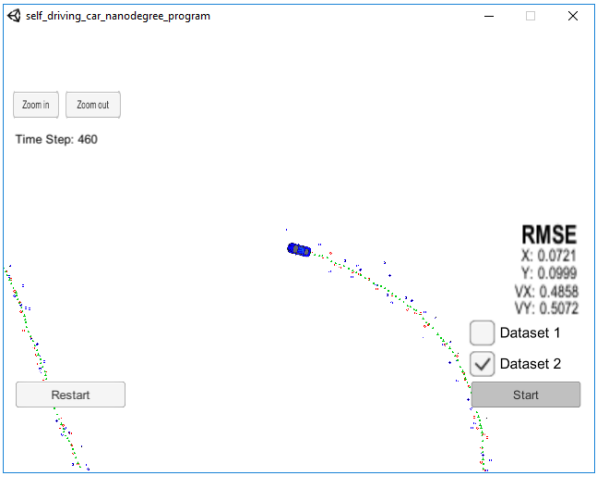


While running with ‘./ExtendedKF’ and selecting the Dataset 2 from the simulator, the px, py, vx, and vy RMSE is still less than or equal to the values [.11, .11, 0.52, 0.52]. See the picture C and D for the RMSE based on the Dataset 2 as following:

Picture C for Dataset 2



Picture B for Dataset 2



Follows the Correct Algorithm

The project's programs that need to be written are in folder /src, and the names of file are as following:

FusionEKF.cpp, kalman_filter.cpp, tools.cpp

(A) Fill in the code for FusionEKF.cpp:

- initialize variables and matrices -- Initalized the F, P & Q matrices, Assign the values for the laser sensor matrix, H_laser, and Set the values for acceleration noise in variable noise_ax and noise_ay.
- Initializing the Kalman Filter – Converted radar from polar to cartesian coordinates and initialize state for both radar and laser.
- Predict and Update Steps – Updated the state transition matrix F, and the process noise covariance matrix Q.

(B) Fill in the code for kalman_filter.cpp:

- Predicted the state in function Predict().
- Updated the state by using Kalman Filter equations in function Update()
- Updated the state by using Extended Kalman Filter equations in function UpdateEKF()

(C) Fill in the code for tools.cpp:

- Calculated the RMSE in function CalculateRMSE()
- Calculated a Jacobian in function CalculateJacobian()

Code Efficiency

In this project, the C++ code with algorithm has avoided unnecessary calculations.

Conclusion

The Extended Kalman Filter (EKF) allow us to use non-linear equations. It uses the Jacobian matrix to linearize non-linear functions. This project completed the Extended Kalman Filter algorithm in C++, and tested the Kalman Filter in the simulator with input data. Ensure that the px, py, vx, and vy RMSE are below the values specified in the rubric.