Extended Kalman Filters

In this project I will utilize a kalman filter to estimate the state of a moving object of interest with noisy lidar and radar measurements. This project involves the Term 2 Simulator which can be downloaded here. More information and dependancies in the link.

The video of testing is in the link.

Project Basics

In the project, I used the C++ to complete the below:

- In src/tools.cpp, fill in the functions that calculate root mean squared error (RMSE) and the Jacobian matrix.
- Fill in the code in src/FusionEKF.cpp. Initialize the Kalman Filter, prepare the Q and F matrices for the prediction step, and call the radar and lidar update functions.
- In src/kalman_filter.cpp, fill out the Predict(), Update(), and UpdateEKF() functions.

Build instructions

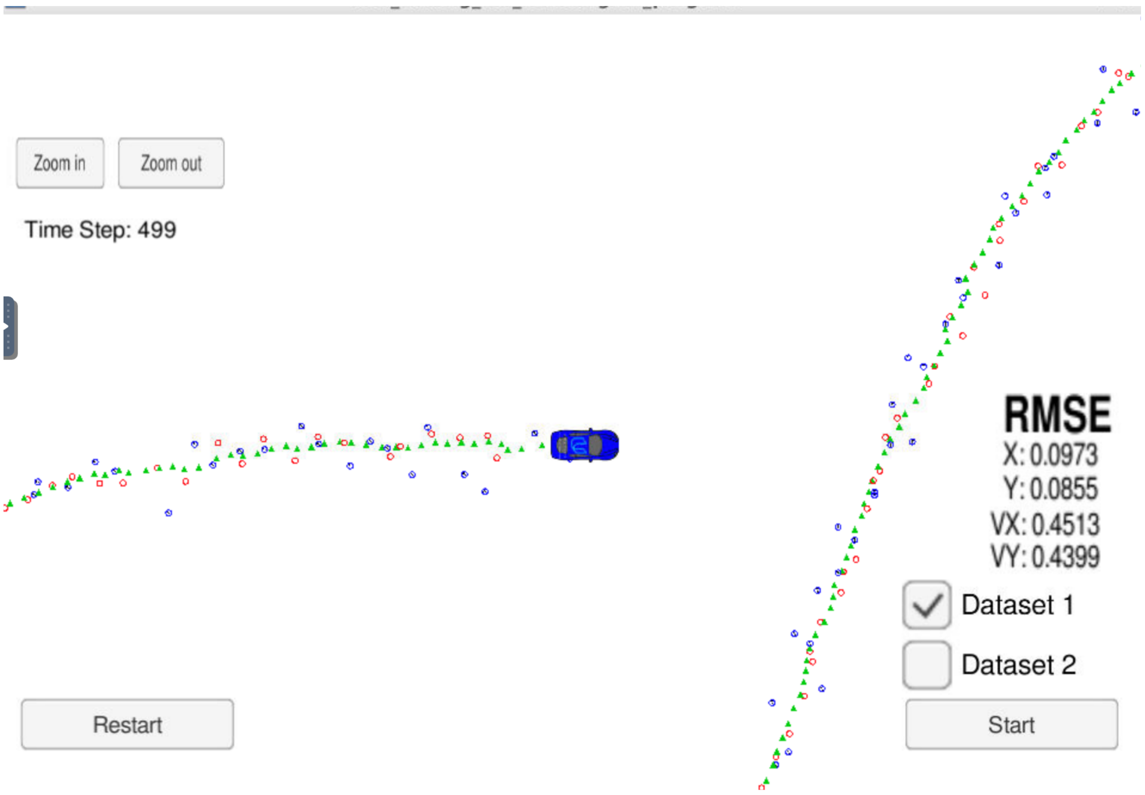
Assuming you have all important dependancies, the main program can be built and run by doing the following from the project top directory:

- mkdir build && cd build
- cmake .. && make
- ./ExtendedKF

Test Result

The RMSE baseline of test result is that RMSE should be less than or equal to the values [.11, .11, 0.52, 0.52].

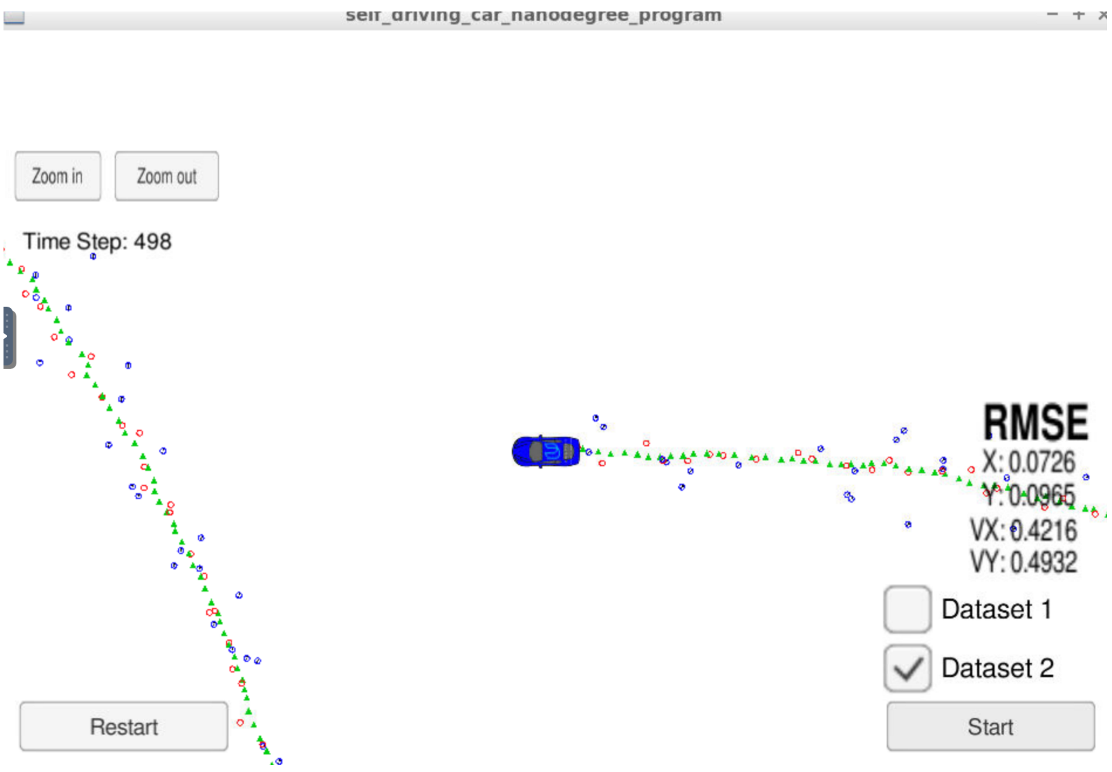
To dataset1, the RMSE value is [0.0973, 0.0855, 0.4513, 0.4399]



Time Step: 499

RMSE
X: 0.0973
Y: 0.0855
VX: 0.4513
VY: 0.4399

To dataset2, the RMSE value is [0.0726, 0.0956, 0.4216, 0.4932]



The key improvement

Before I used the **normalize angle method,** the RMSE value only can get the [0.1401, 0.6662, 0.6045, 1.6253].
After used the **normalize angle method,** the RMSE value can reached the baseline, and the method reference from the link, the 3 lines code:

```
// normalize angle
double width = 2 * M_PI;   //
double offsetValue = y(1) + M_PI;   // value relative to 0
y(1) = (offsetValue - (floor(offsetValue / width) * width)) - M_PI;
```

More In-depth Knowledge of Kalman filters:

- All About Kalman Filters,
- Sensor Fusion and Object Tracking using an Extended Kalman Filter Algorithm — Part 1,
- Estimation of Sideslip Angle Based on Extended Kalman Filter,
- Udacity Self-Driving Car Nanodegree Project 6 - Extended Kalman Filter,
- Extended Kalman Filter for Channel Estimation in Rayleigh Fading Environment and Fadingless Environment.

Below are some links for efficient c++ programming practices:

- 10 most voted C++ best practices
- Principles and Practice Using C++