Unscented Kalman Filter Project

This project utilized an Unscented Kalman Filter to estimate the state of a moving object of interest with noisy lidar and radar measurements. Passing the project requires obtaining RMSE values that are lower that the tolerance outlined in the project rubric.

Compiling

This project made a subfolder '/build' in the project folder, and start compiling by doing the following:

cmake ..

make

The C++ code can be compiled without errors.

Accuracy

To start the simulator (located at /build/term2_sim_windows/term2_sim.exe), launched it and select the EKF/UKF project, running command:

./UnscentedKF

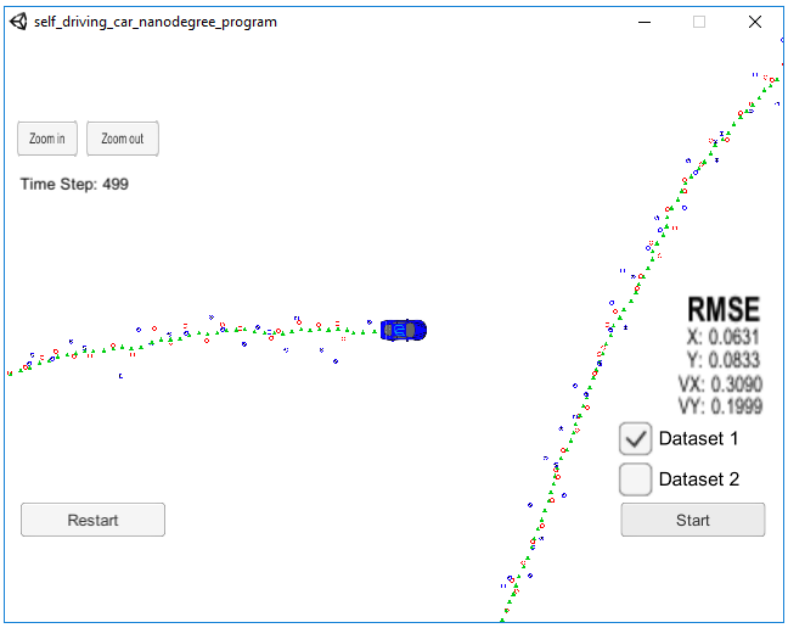
Then, click start button in the simulator.

The algorithm is run against both Dataset 1 and Dataset 2 in the simulator. We collected the positions that the algorithm outputs and compare them to ground truth data. the value of RMSE is as following:

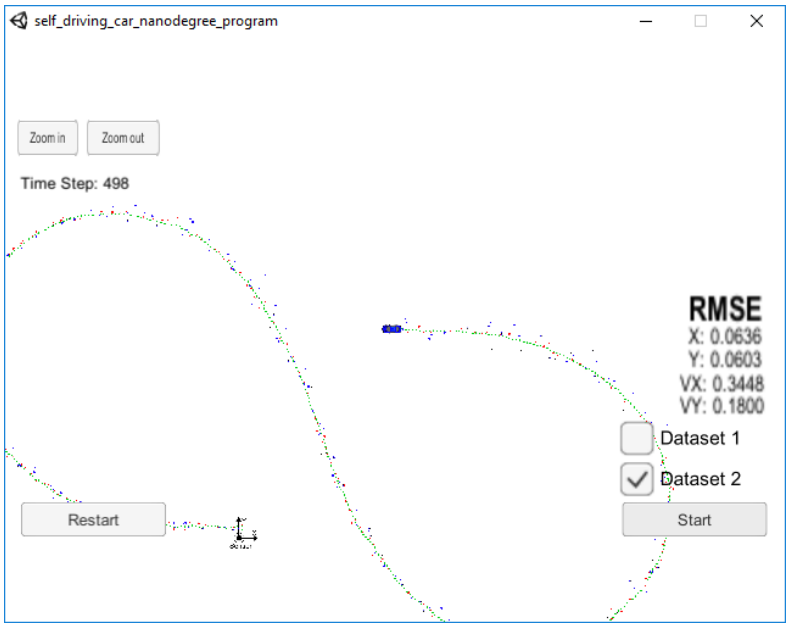
(1) For Dataset 1 (see Picture 1 below): px=0.0631, py=0.0833, vx=0.3090, and vy=0.1999
(2) For Dataset 2 (see Picture 2 below): px=0.0636, py=0.0603, vx=0.3448, and vy=0.1800

For both datasets, the RMSE value is less than or equal to the values [.09, .10, 0.40, 0.30] – Meet the project specifications.

Picture 1: Output with RMSE for Dataset 1



Picture 2: Output with RMSE for Dataset 2



Unscented Kalman Filter VS Extended Kalman Filter

In the previous project on Extended Kalman Filter, the same data (as Dataset 1) is used to calculate the value of RMSE on px, py, vx, and vy. It's time to do the comparison on RMSE between Unscented Kalman Filter and Extended Kalman Filter. The Table 1 shows the accuracy comparison in RMSE by using EKF and UKF with both lidar and radar measurements. The lidar and radar measurements are included in the txt file under the /data folder.

Table 1: RMSE by using EKF and UKF

State	**UKF**	EKF
px	**0.0631**	0.0969
py	**0.0833**	0.0854
vx	**0.3090**	0.4255
vy	**0.1999**	0.4392

Follows the Correct Algorithm

The project's C++ programs that need to be written are in subfolder /src, and the names of file are ukf.cpp, and tools.cpp. The detail of the works in each program are:

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

- Completed initialization -- Initialized other variables in UKF::UKF() besides the ones given in the ukf.cpp template.
- Completed function UKF::ProcessMeasurement() – This function return the latest measurement data and switch between lidar and radar measurements.

- Completed function UKF::Prediction() – In this function, estimated the object's location, modify the state vector, and predict sigma points, the state, and the state covariance matrix.
- Completed function UKF::UpdateLidar() – In this function, updated the state, the state covariance matrix by using Laser measurement, and calculated NIS value.
- Completed function UKF::UpdateRadar() – In this function, updated the state, the state covariance matrix by using Radar measurement, and calculated NIS value.

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

- Calculated the RMSE in function Tools::CalculateRMSE()

Code Efficiency

In this project, the C++ code with algorithm has avoided unnecessary calculations. And the code maintained good practice with respect to calculations.

Conclusion

As seen above, the UKF does an excellent job of estimating both the object position, velocity and more impressively, the yaw angle which is not measurement directly by either the RADAR or LIDAR. Additionally, the RMSE values obtained for position and velocity are lower than those of an Extended Kalman Filter for the same dataset. This is attributed to the UKF's ability to better handle non-linear motion of the tracked object compared to an EKF as well as the use of the CRTV model as opposed to a constant velocity model used in the EKF.

This project completed the Unscented Kalman Filter algorithm in C++ program, and tested the Unscented 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