UDACITY

‹ Return to Classroom

Extended Kalman Filters

REVIEW

CODE REVIEW 2

HISTORY

Meets Specifications

Hi,
This is a very good submission of this project. Well done. Please keep up the good work.

- I'd like to invite you to look at these links to these documents on EKF. I personally use them a lot in order to better understand these filters.
  http://home.wlu.edu/~levys/kalman_tutorial/
  http://biorobotics.ri.cmu.edu/papers/sbp_papers/integrated3/kleeman_kalman_basics.pdf

As you have correctly noted, a writeup is not necessary for this project.

Compiling

Code must compile without errors with `cmake` and `make`.

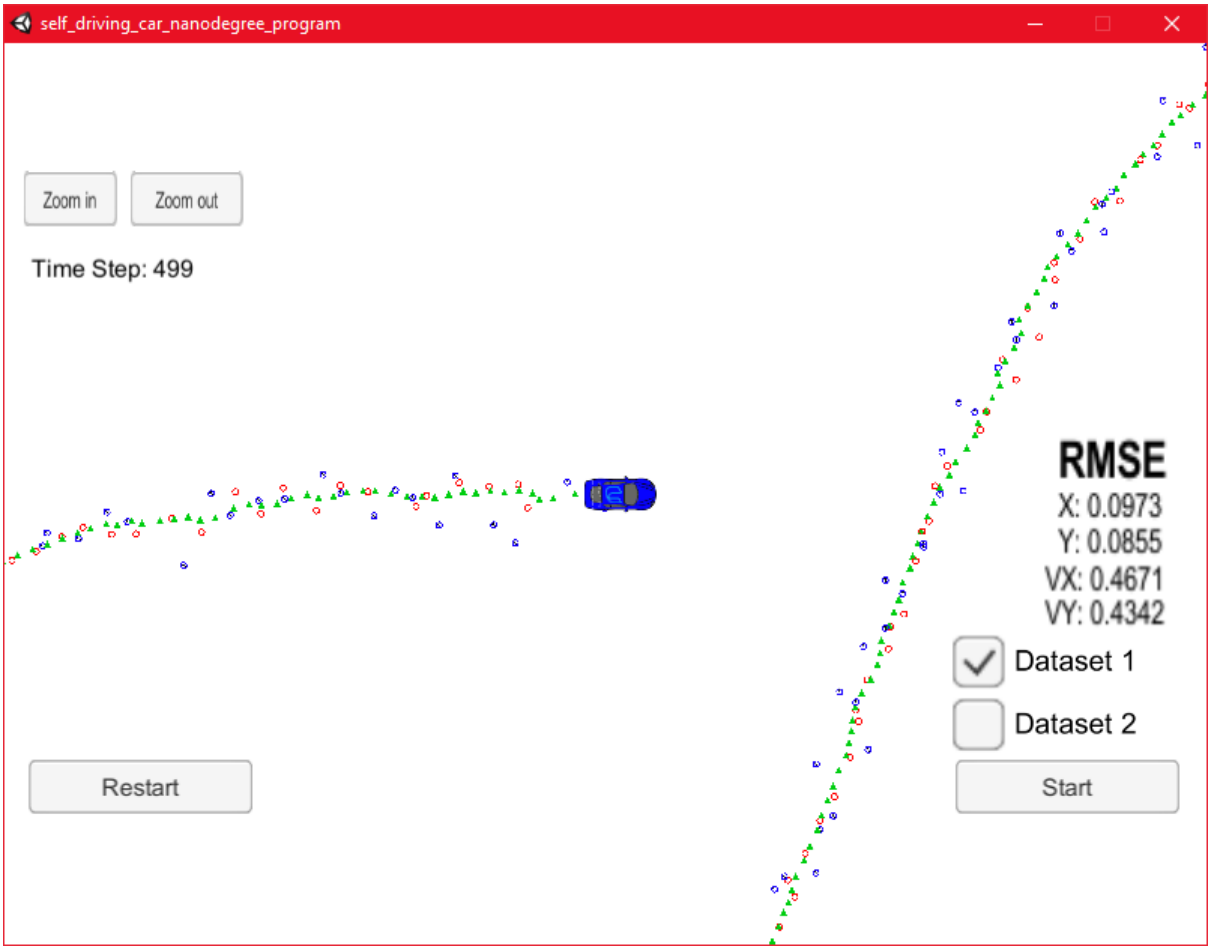
Given that we've made CMakeLists.txt as general as possible, it's recommended that you do not change it unless you can guarantee that your changes will still compile on any platform.

Good work, the code compiles without error.

Accuracy

Your algorithm will be run against Dataset 1 in the simulator which is the same as "data/obj_pose-laser-radar-synthetic-input.txt" in the repository. We'll collect the positions that your algorithm outputs and compare them to ground truth data. Your px, py, vx, and vy RMSE should be less than or equal to the values [.11, .11, 0.52, 0.52].

Great job here, your px, py, vx, and vy RMSE is less than the ground truth. When I run it, these are the results I obtained.



Follows the Correct Algorithm

While you may be creative with your implementation, there is a well-defined set of steps that must take place in order to successfully build a Kalman Filter. As such, your project should follow the algorithm as described in the preceding lesson.
Well done, all the steps required to implement a Kalman Filter has been well defined and correctly implemented

Your algorithm should use the first measurements to initialize the state vectors and covariance matrices.
Excellent work here. Your algorithm effectively uses the first measurements to initialize the state vectors and covariance matrices.

Upon receiving a measurement after the first, the algorithm should predict object position to the current timestep and then update the prediction using the new measurement.
Nice, the algorithm predicts the object position at the current time step and then update it using the new measurement.

Your algorithm sets up the appropriate matrices given the type of measurement and calls the correct measurement function for a given sensor type.

Excellent, the algorithm sets up the appropriate matrices depending on the type of the measurement. It efficiently calls the correct measurement function for each sensor. EKF for radar and Kalman for lasers. Well done 😄

Code Efficiency

This is mostly a "code smell" test. Your algorithm does not need to sacrifice comprehension, stability, robustness or security for speed, however it should maintain good practice with respect to calculations.

Here are some things to avoid. This is not a complete list, but rather a few examples of inefficiencies.

- **Running the exact same calculation repeatedly when you can run it once, store the value and then reuse the value later.**
- **Loops that run too many times.**
- **Creating unnecessarily complex data structures when simpler structures work equivalently.**
- **Unnecessary control flow checks.**

Well done, there is no unnecessary loop that runs too many times, and no unnecessary control flow checks.

DOWNLOAD PROJECT

2 CODE REVIEW COMMENTS ›

RETURN TO PATH

Rate this review

START