UDACITY

PROJECT SPECIFICATION

Unscented Kalman Filter Highway Project

Compiling and Testing

CRITERIA	MEETS SPECIFICATIONS
The submission must compile.	The project code must compile without errors using `cmake` and `make`.

Code Efficiency

CRITERIA	MEETS SPECIFICATIONS

CRITERIA	MEETS SPECIFICATIONS
The methods in the code should avoid unnecessary calculations.	Your code does not need to sacrifice comprehension, stability, or robustness for speed. However, you should maintain good and efficient coding practices when writing your functions. Here are some things to avoid. This is not a complete list, but there are 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.

Accuracy

CRITERIA	MEETS SPECIFICATIONS
px, py, vx, vy output coordinates must have an RMSE <= [0.30, 0.16, 0.95, 0.70] after running for longer than 1 second.	The simulation collects the position and velocity values that your algorithm outputs and they are compare to the ground truth data. Your px, py, vx, and vy RMSE should be less than or equal to the values [0.30, 0.16, 0.95, 0.70] after the simulator has ran for longer than 1 second. The simulator will also display if RMSE values surpass the threshold.

Follows the Correct Algorithm

CRITERIA	MEETS SPECIFICATIONS
Your Sensor Fusion algorithm follows the general processing flow as taught in the preceding lessons.	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.

Suggestions to Make Your Project Stand Out!

1. While we're giving this project to you with starter code, you are not actually required to use it! If you think you can organize your Kalman Filter better than us, go for it! Also, this project was templatized in an object-oriented style, however it's reasonable to build a Kalman Filter in a functional style. Feel free to start from scratch with a functional algorithm!

    - Keep in mind that your code *must* compile. If your changes necessitate modifying CMakeLists.txt, you are responsible for ensuring that *any* reviewer can still compile your code given the dependencies listed earlier in the instructions - platform specific errors will *not* be debugged by graders.

2. There is some room for improvement with the Kalman Filter algorithm. Maybe some aspects of the algorithm could be combined? Maybe some could be skipped under certain circumstances? Maybe there are other ways to improve performance? Get creative!

3. Compare your results when only using radar or only using lidar. Sensor fusion should give better results than using only one sensor type.