Extended Kalman Filters

Compiling

CRITERIA	MEETS SPECIFICATIONS	STUDENT COMMENTS
Your code should compile.	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.	The code compiles and the executable(ExtendedKF) is created.

Accuracy

CRITERIA	MEETS SPECIFICATIONS	STUDENT COMMENTS
px, py, vx, vy output coordinates must have an RMSE <= [.11, .11, 0.52, 0.52] when using the file: "obj_pose-laser-radar-synthetic-input.txt" which is the same data file the simulator uses for Dataset 1.	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].	The RMSE values ends with [0.097, 0.085, 0.451, 0.439] We can clearly see algorithm converging with ground truth as we get more measurements.

Follows the Correct Algorithm

CRITERIA	MEETS SPECIFICATIONS	STUDENT COMMENTS
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.	Care is taken to follow the flow suggested in slide 6 of the project lesson. https://classroom.udacity.com/nanodegrees/nd013/parts/40f38239-66b6-46ec-ae68-03afd8a601c8/modules/0949fca6-b379-42af-a919-ee50aa304e6a/lessons/f758c44c-5e40-4e01-93b5-1a82aa4e044f/concepts/15b78db8-924a-47a0-bee2-9ea135651d63
Your Kalman Filter algorithm handles the first measurements appropriately.	Your algorithm should use the first measurements to initialize the state vectors and covariance matrices.	Initializations are done in lines 34-64 in fusionEKF.cpp file. The first measurement is handled in lines fusionEKF 79-144 when the filter is not initialized.

CRITERIA	MEETS SPECIFICATIONS	STUDENT COMMENTS
Your Kalman Filter algorithm first predicts then updates.	Upon receiving a measurement after the first, the algorithm should predict object position to the current time step and then update the prediction using the new measurement.	Prediction is handled in lines 145-177 in fusionEKF.cpp Update is handled in 189 – 210 in fusionEKF.cpp
Your Kalman Filter can handle radar and lidar measurements.	Your algorithm sets up the appropriate matrices given the type of measurement and calls the correct measurement function for a given sensor type.	During initialization and update we check the sensor type and call the respective update function.

Code Efficiency

CRITERIA	MEETS SPECIFICATIONS	STUDENT COMMENTS
Your algorithm should avoid unnecessary calculations.	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.	The Tips provided in slides 7 of the project are followed to: 1. Initialize the state vector 2. Calculate y = z – H * x’ 3. Normalize angles 4. Avoid divide by zero etc.