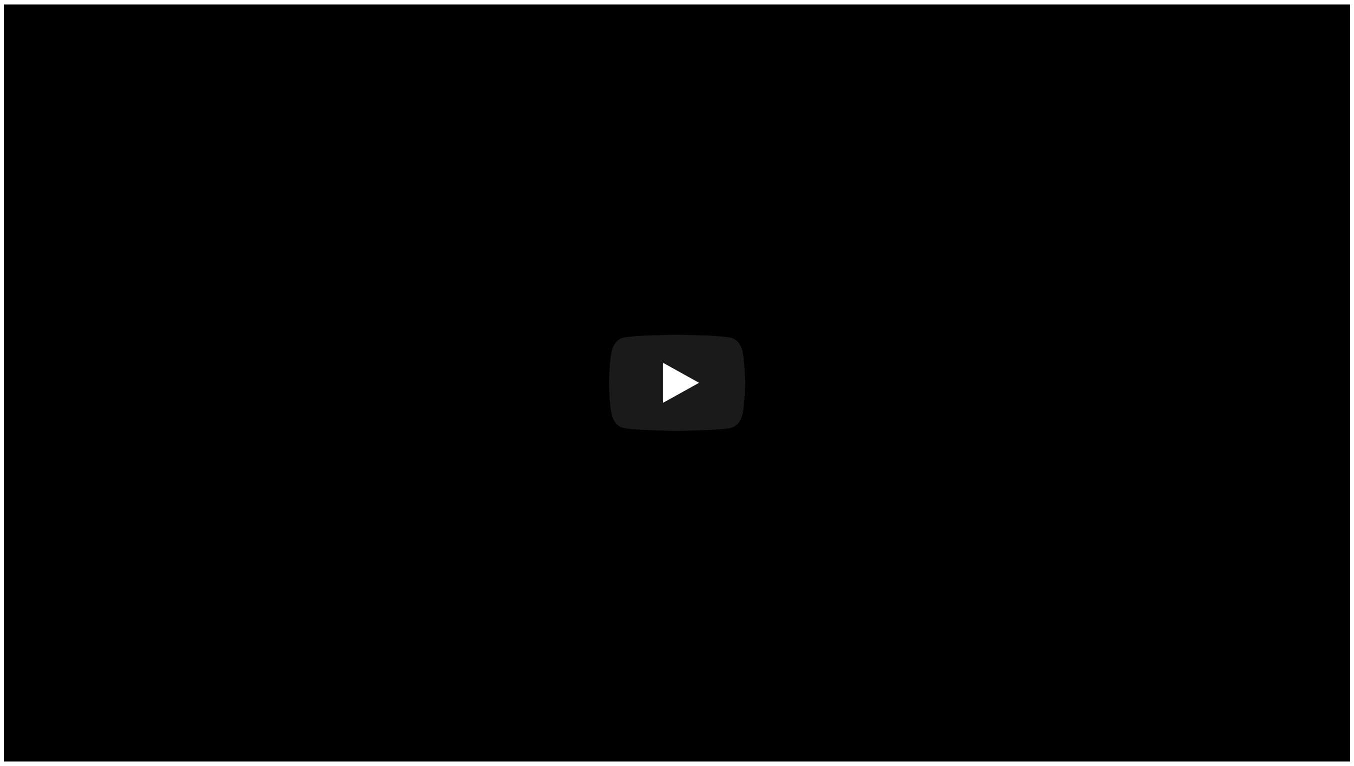
Intro to Extended Kalman Filter Project

Project Introduction

Now that you have learned how the extended Kalman filter works, you are going to implement the extended Kalman filter in C++. We are providing simulated lidar and radar measurements detecting a bicycle that travels around your vehicle. You will use a Kalman filter, lidar measurements and radar measurements to track the bicycle's position and velocity.

The first step is to download the Term 2 simulator, which contains all the projects for Term 2 Self-Driving Car Nanodegree. More detailed instruction about setting up the simulator with uWebSocketIO can be found at the end of this section.

Lidar measurements are red circles, radar measurements are blue circles with an arrow pointing in the direction of the observed angle, and estimation markers are green triangles. The video below shows what the simulator looks like when a c++ script is using its Kalman filter to track the object. The simulator provides the script the measured data (either lidar or radar), and the script feeds back the measured estimation marker, and RMSE values from its Kalman filter.



Download Links for Term 2 Simulator

Term 2 Simulator Release

Running the Program

2. Once the scene is loaded you can hit the START button to observe how the object moves and how measurement markers are positioned in the data set. Also for more experimentation, "Data set 2" is included which is a reversed version of "Data set 1", also the second data set starts with a radar measurement where the first data set starts with a lidar measurement. At any time you can press the PAUSE button, to pause the scene or hit the RESTART button to reset the scene. Also the ARROW KEYS can be used to move the camera around, and the top left ZOOM IN/OUT buttons can be used to focus the camera. Pressing the ESCAPE KEY returns to the simulator main menu.
3. The EKF project Github repository README has more detailed instructions for installing and using c++ uWebScoketIO.

NOTES:

- Currently hitting Restart or switching between Data sets only refreshes the simulator state and not the Kalman Filter's saved results. The current procedure for refreshing the Kalman Filter is to close the connection, `ctrl+c` and reopen it, `./ExtendedKF`. If you don't do this when trying to run a different Data set or running the same Data set multiple times in a row, the RMSE values will become large because of the the previous different filter results still being observed in memory.
- Students have reported rapid expansion of log files when using the term 2 simulator. This appears to be associated with not being connected to uWebSockets. If this does occur, please make sure you are connected to uWebSockets. The following workaround may also be effective at preventing large log files.
    - create an empty log file
    - remove write permissions so that the simulator can't write to log

What You'll Need to Do

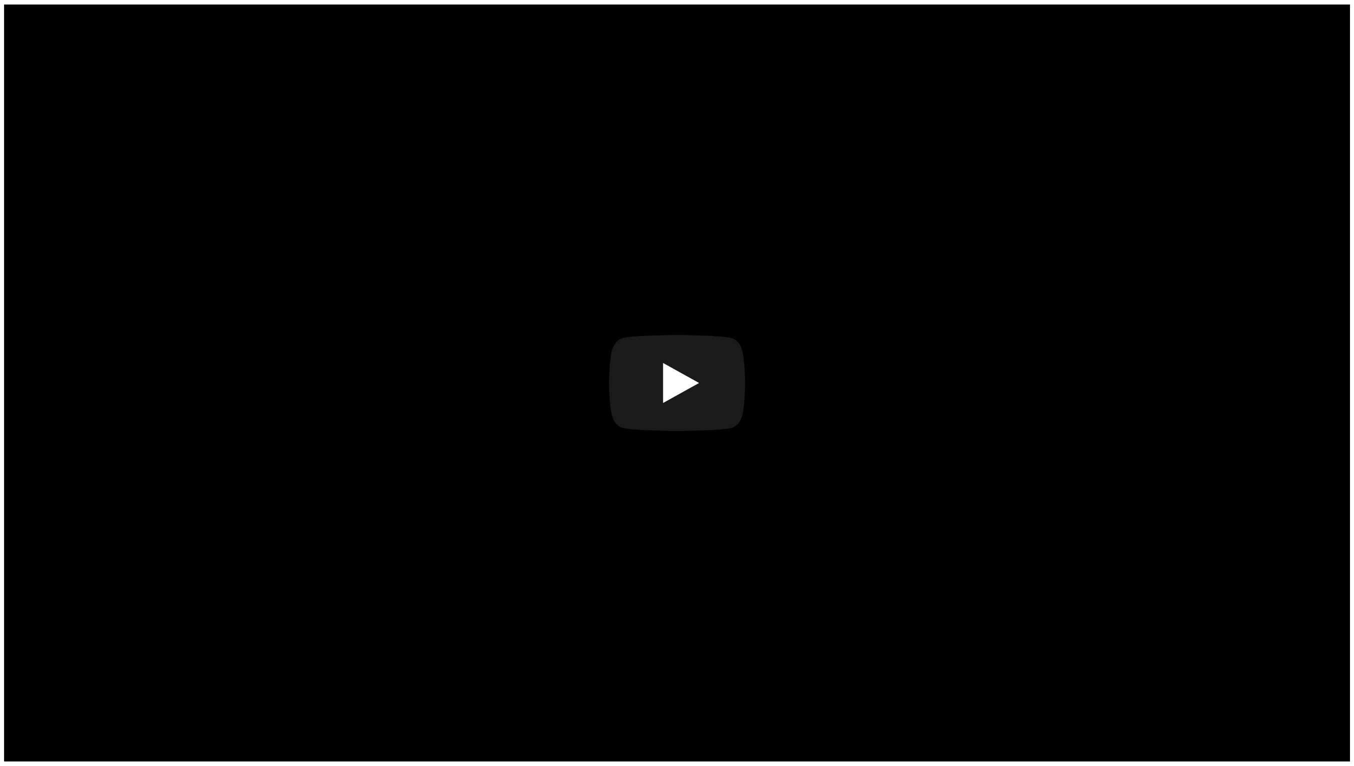
1. Read the repo's README for more detailed instructions.
2. Complete the Extended Kalman Filter algorithm in C++.
3. Ensure that your project compiles.
    1. From the root of the repo:
        1. `mkdir build && cd build`
        2. `cmake .. && make`
        3. `./ExtendedKF`
4. Test your Kalman Filter in the simulator with Dataset 1. Ensure that the px, py, vx, and vy RMSE are below the values specified in the rubric.
5. Submit your project!

Intro to Extended Kalman Filter Project

still be accessed from Github in the branch and for the time being students can submit either version of the project, using either the new simulator interface or the previous text based interface. Running the previous project set up requires `./ExtendedKF path/to/input.txt path/to/output.txt`

Example of Tracking with Lidar

Check out the video below to see a real world example of object tracking with lidar. In this project, you will only be tracking one object, but the video will give you a sense for how object tracking with lidar works:



NEXT