COMP 417 Assignment 2

Due: Nov 14, 2018 at 2:30pm

1 Pedagogical objectives

Experience with practical implementations of sensor-based path planning algorithms. Simple feed-back controllers. Empirical analysis of runtimes and robustness.

2 Problems

Understand the Kalman filter. Understand the role of the variances. Learn to develop vision-based sensing to close the Kalman filter loop.

Your code will implement tracker that estimates the position of a ball as it moves across the scene. It does this using video footage we provide to you.

2.1 Problem One - Blob Tracker

1) Examine the *detect ball* method and modify the parameters to provide the best detection of the red ball. For each video, provide the parameters you used for:

- bgr-color
- color-threshold
- erode and dilate iterations
- minimum radius size

2) For each on the input videos provide a plot of horizontal position vs time on a scale of absolute pixels, and as a fraction of the total path length (0,1). Now plot the position in 1D as a function of time, simply taking the horizontal position component of the blob position as returned by the sensor.

You can use any programming language you want, but your code must function on a linux-based computer.

You code should be runnable using a command of the form:

python ballestimator.py VIDEO_PATHNAME

Your solution may use any logic that you like for this component as long as you manage to follow balls successfully.

Starter code for video processing and video files can be found on the course homepage.

2.2 Problem Two - Ball Tracker via KF

Use a Kalman Filter to track the position of the ball in one dimension (x axis), i.e. the x-position is the state.

If the ball moves vertically, project it's position onto the horizontal axis. For the easy movie, assume the transition model for the ball is a simple constant motion which can be computed by taking the total distance travelled divided by the number of video frames (you can use either frames or seconds as the time index, so long as you are consistent and document your assumption),

For the sensor model, simply take the horizontal position component of the blob position as returned by the sensor.

Your command should be:

python ballestimator.py VIDEO_PATHNAME

What should your covariance be? For the "clean" video use the value 1 (one) for each variance.

Your filter produce, and you should **plot the estimated horizontal position** of the ball at each frame (x vs t), as well as the **associated variance** (v vs t).

For the noisy video, try using 4 for the sensor variance.

Next step: compute the true variance for the sensor using an approach and assumptions that you describe. Plot this and re-run the filter with this variance.

3 Submission

Submit a description of your process and your code (presumably in Python).

For **each test movie**, submit the plots and numerical results for each the two questions. Describe any problems or errors with your code, or tricks you used that are unusual. A PDF file is the preferred submission format.

4 Tips and warnings

Submit your code and assure it runs on Linux (the Schools' machines). You are allowed to develop your code on Windows, but be sure to test it on Linux before final submission.

Do not count on us necessarily running your code, but provide it "just in case."

Work independently. You can discuss the nature of your solutions and approach, but write your own code and do your own data analysis.

The sample code provided to you uses Python 2 although it is trivial to get it running on Python 3.

To install OpenCV you can use
pip install opencv-python

On MacOS, there are several additional approaches such as using "homebrew" followed by "brew install opencv".