1 Pedagogical objectives


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

Implement a blob tracker using openCV to extract the position of the red ball in the video. Plot the position of the blob as a function of time. Adjust the parameters of your extractor for good results on the first (easiest) video. Note that on the difficult videos you should not expect a perfect result.

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

Startup code for video processing can be found at:

Sample test videos are available at:
  https://goo.gl/haknLf

2.2 Problem Two - Ball Tracker vis KF

Use a Kalman Filter to track the position of the ball in one dimension (x axis). 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.

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

Optional: compute the true variance for the sensor using an approach and assumptions that you describe.

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.
Our assignment deadline policy for this course is that if you attempt to submit code which is minimally functional by the deadline, you are allowed to consult with other students, the TA or the instructors in order to gain an understanding of what you missed at first. You can submit an improved solution within one week of the initial date. For this assignment, minimal functionality means your code must be a proper ROS node that compiles and runs, it must at least move the robot roughly in the right directions initially, but it does not have to reach the goal.