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 feedback controllers. Empirical analysis of runtimes and robustness.

1.1 Setup Guideline

As the assignment is python 2.7, we recommend using a virtual environment to install these packages. Below are the required libraries to get everything up and running. Note, use your own python2.7 path.

```
virtualenv -p /usr/local/lib/python2.7/bin/python
source py_2_7/bin/activate
pip install numpy
pip install pygame
pip install vispy==0.6
pip install opencv-python==3.4.0.12
pip install imutils
```

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 a tracker that estimates the position of a ball as it moves across the scene. It does this using the simulated ball environment we provide. For this assignment, you only need to modify the opencv_ball_tracker.py and kalman_ball_tracker.py files.

2.1 Environment

In this assignment, the environment consists of a red ball whose position can be controlled by fans. There are several variables you can modify; the target positions and PID variables. The goal of the assignment is to provide

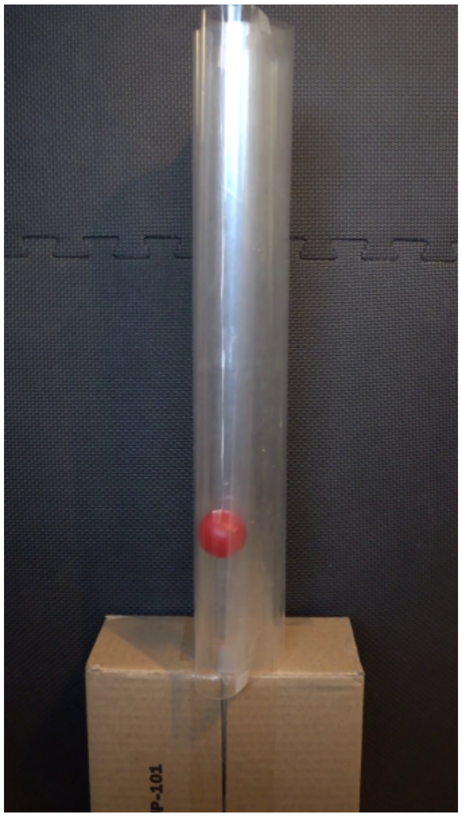

Figure 1: The simulated environment

an accurate current position for the PID controller to move the ball to the target position.

The environment consists of several modes

1. Experimental Mode: Manually control the target position and variables.
2. Validation mode: The target position is randomly sampled.
3. Noisy version of the aforementioned modes; the sensor images suffers from heavy pixel noise.

2.2 Problem 1 - Blob Tracker

Implement a blob tracker using OpenCV to extract the position of the red ball in the simulated environment. Plot the position of the blob as a function of time with the validation mode. Adjust the parameters of your extractor for good results. Note that depending on the validation's target positions, it can be difficult to obtain perfect estimations.

For each of the input videos provides 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.

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

python opencv_ball_tracker.py

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

2.3 Problem Two - Ball Tracker vis KF

Implement a Kalman Filter to track the position of the ball. The Kalman Filter consists of two parts, a transition model(which we assume to be linear) and an observation model which takes as input the sensor image. For the transition model, you can use the past positions to estimate a linear movement of the ball. For the observation model, a simple approach such as finding the area of the image that is reddest could work.

Your command should be: **python kalman_ball_tracker.py**

What should your covariance be? For the noiseless mode try the value 1 (one). For the noisy mode, try several values such as 4 for the sensor variance.

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

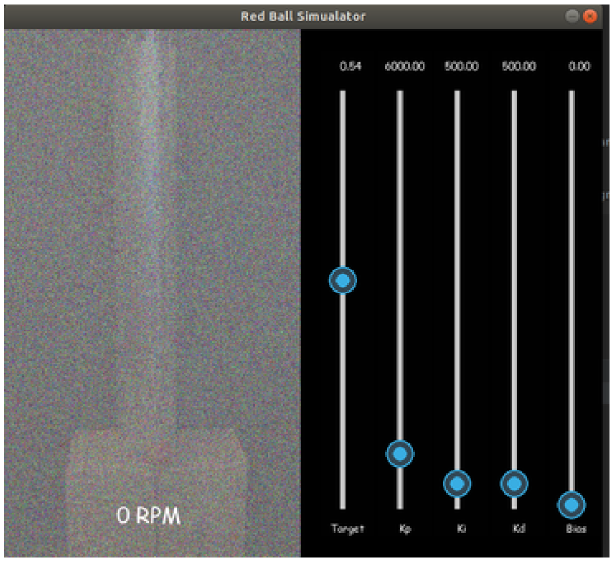

Figure 2: The simulated environment with noise

3 Submission

Submit a report detailing your implementation of the ball tracker for both the Opencv version and Kalman filter version. For both the noiseless and noisy validation mode, please submit the plots, numerical results for at least 3 runs.

Describe any problems or errors with your code, or tricks you used that are unusual. A zipped file with all the .py files(including the ones you don't have to change) and the PDF report 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 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.