COMP 417 Assignment 2

Due Date: Nov. 25th 2021, 11:59PM Eastern Standard Time

1 Pedagogical objectives

Experience with practical implementations of sensor-based path planning algorithms. Simple feedback controllers. Empirical analysis of runtimes and robustness. Understand the Kalman filter. Understand the role of the variances. Learn to develop vision-based sensing to close the Kalman filter loop.

1.1 Setup Guideline

We currently support two versions, Python 2.7 and Python 3.+. We recommend using the python 3.+ version as it is easier to install. If for some reason your python 3.+ version is not working, the python 2.7 version has a docker image which should work but is more tricky to use. Required libraries are as follows:

```
pip install NumPy
pip install pygame
pip install vispy
pip install opencv-python
pip install imutils
```

Furthermore, you will also need a graphic library from one of the following, ['PyQt4', 'PyQt5', 'PyQt6', 'PySide', 'PySide2', 'PySide6', 'Pyglet', 'Glfw', 'SDL2', 'wx', 'EGL', 'osmesa', 'tkinter']

2 Problems

2.1 Environment

In this assignment, the environment consists of a red ball whose position can be controlled by fans. The ball moves up and down depending on the speed of the fans. The state of the environment is provided as an RGB image. Note that the ball can fly outside the line of sight of the camera if the fan rpm is set high enough.

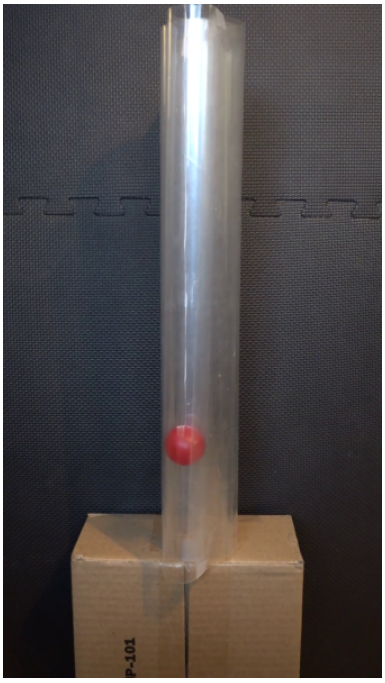

Figure 1: The simulated environment

There are 3 modes, validation, validation noisy, and validation save; each lasts 15 seconds. Validation mode is the standard mode; the target position slider will bob up and down. The noisy mode adds significant Gaussian noise to the RGB image provided, making it significantly more difficult to detect the ball. Finally, validation save is the standard validation mode but saves all RGB images in a NumPy matrix in the *../output/* directory.

2.2 Task

Your goal in this assignment is to first detect the ball's location. Then, using the ball's location, compute the proper fan rpm to move it closer to the target position. For this assignment, you only need to modify the opencv_ball_tracker.py and kalman_ball_tracker.py files. More specifically, the functions detect_ball, get_fan_rpm, and for the Kalman version, get_kalman_filter_estimation. Note there are several variables you can modify; PID variables, bgr_color, thresh which can improve performance.

2.3 Problem 1 - Blob Tracker

Implement a blob tracker and controller 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 detect_image for good results. Note that depending on the randomness of the validation mode's target positions, it can be difficult to obtain perfect estimations.

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.

2.4 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, you may reuse the model from the previous section.

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.

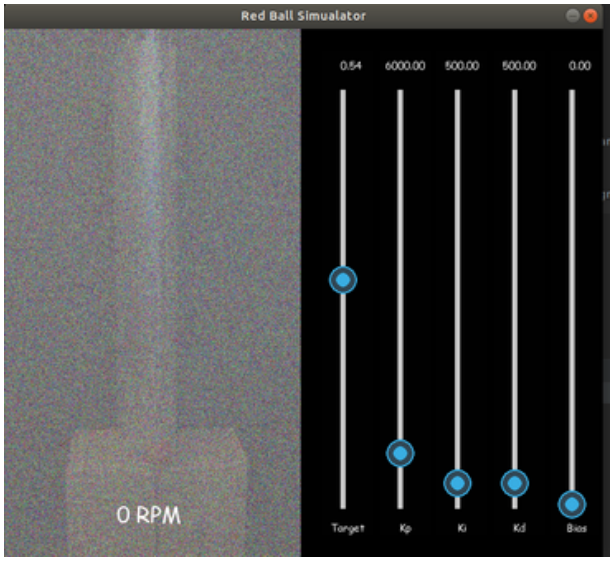

Figure 2: The simulated environment with noise

3 Submission

Submit a report detailing your implementation of the ball tracker for both the Opencv version and Kalman filter version. For both Opencv and Kalman versions, provide the experiment results for the noiseless and noisy validation mode. For each mode and algorithm, Show the plots of the position of the ball and estimation error 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.