COMP 417 Assignment 3

Due: Nov 30, 2018 at 2:30pm update

1 Problem Statement

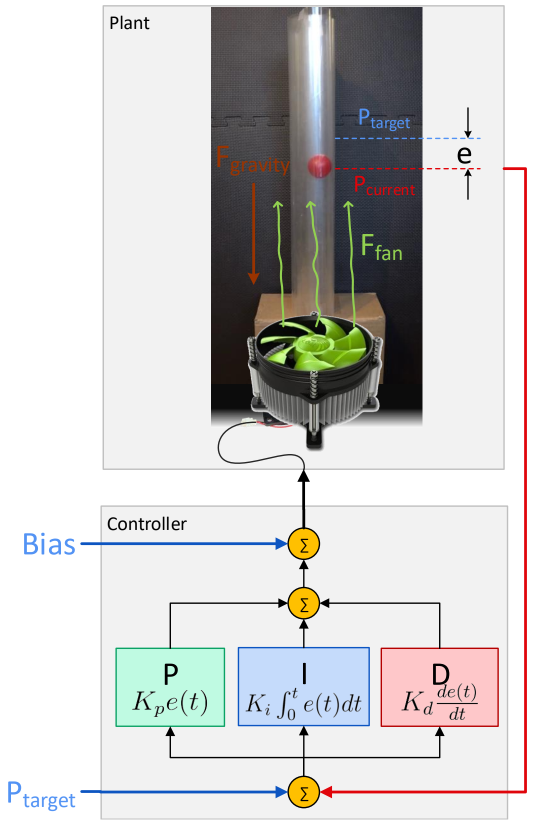

Figure 1: Plant and PID control loop

The goal of this assignment is to understand the PID controller in the context of balancing a ping-pong ball using a fan as shown in figure 1. The position of the ball is detected using computer vision (similar to assignment 2). You will also need to familiarize yourself with the steps required to tune the PID controller to give reasonable results.

Your code will implement the ball detector and PID controller. The assignment is graded on the implementation of the PID controller and validated using several metrics of its performance. You will need at least two new python packages in addition to those used in the previous assignment, *pygame* and *vispy*. You can install these using *pip*.

1.1 Problem One - PID Controller

You are provided started code that the simulates a ping-pong ball inside a tube with a fan underneath. Your task is to detect the ball and implement a PID controller to reach a target height. You may modify any piece of code during your development, but you will **ONLY** be evaluated on your implementation of *pid.py*. Therefore, it is strongly recommended that you check your implementation against the original code when complete.

The simulator is run using the command: `Python main.py` Afterwards, you can type 'e' to run the simulator in experimental mode (normal mode of operation), or 'v' [target height] (ie 'v 0.5') for evaluation mode where the program will run a step response for several seconds and then show a plot with the results. The validation mode will be used to to mark your assignment.

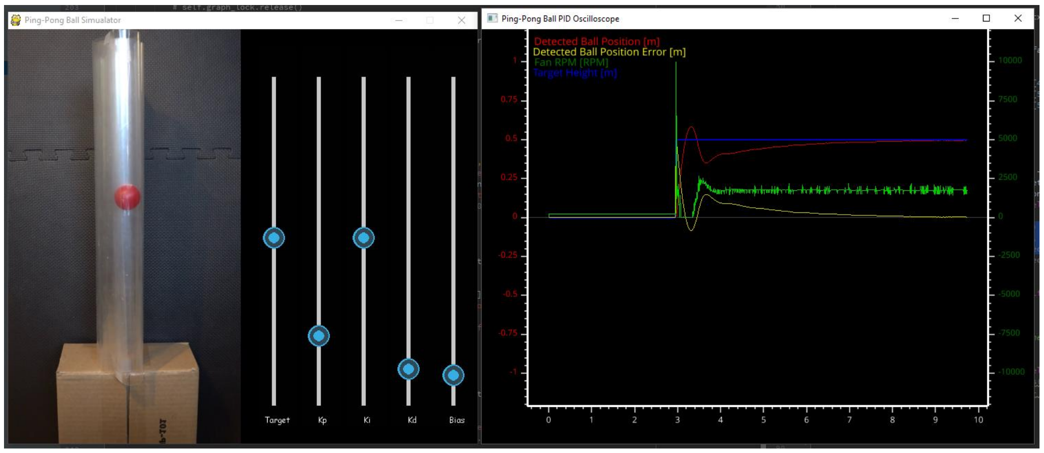
Ping-Pong Ball Simulator
Target
Kp
Ki
Kd
Bias
Ping-Pong Ball PID Oscilloscope
Detected Ball Position [m]
Detected Ball Position Error [m]
Fan RPM [RPM]
Target Height [m]

Figure 2: Simulator and graphing user interface

Two windows should appear as shown in figure 2. The left window is the simulator. The four sliders adjust the *target height*, *Kp*, *Ki*, *Kd* and *bias* parameters of the pid controller. You can manually control the output of the fan by pressing the left mouse button inside the image area and slide the cursor up and down to adjust the fan rpm. Some additional commands are:

- 'r' - reset the simulation, the target height is set to zero and the ball is reinitialized to the starting position.
- 'g' - show a plot of the target height, fan output, detected ball position, and real output.
- numbers '0' through '9' - sets the target height of the ball. This is useful to simulate the step response of the PID controller.

The right window is a real-time graph of the state of the system (Note. that at this time, this does not work on macOS due to a threading limitation by the OS, however it is not necessary for the assignment). You can manipulate the the graph using the mouse buttons and scroll wheel and the keyboard commands are:

- 'r' - reset the graph to show whole plot.
- 's' - auto-scroll

Implement the PID control loop

Implement pid.py. The simulator calls the *get_fan_rpm* method (passing the image frame as an argument) at each time step and expects the rpm output and ball position (in meters) to be returned. The target position of the ball is determined by the *target_pos* variable and this variable can change over time. You should first implement a *detect_ball* method and you can modify the code used in the previous assignment to detect the ball. This function should output the position of the ball in meters where the bottom of the tube (lowest point the ball reaches) is 0 m and the top is 1.0 m (the highest point the ball is detected). You then need to implement the logic of the PID controller, and use the Kp, Ki, Kd and bias variables to set the parameters of the controller.

Your file needs to be called 'pid.py' and be included the source directory. You must use the template provided below:

```python
class PIDController:
    def __init__(self, target_pos):
        self.target_pos = target_pos
        self.Kp = 0.0
        self.Ki = 0.0
        self.Kd = 0.0
        self.bias = 0.0
        return

    def reset(self):
        return

    def get_fan_rpm(self, image_frame=None):
        output = 0.0
        vertical_ball_position = 0.0
        return output, vertical_ball_position
```

Tune your PID controller (with **zero** bias term) to achieve the following performance to a **step response of 0.5 m** :

- Minimum average error (difference between target and desired position)
- Minimum settling time to steady state
- Minimum overshoot (maximum excursion of the ball up the tube)

Do this for two cases:
1) a ball of mass of 0.0027kg (2.7g, current default in sim_env.py) and
2) a second case with a ball mass of 0.01 kg.

1.2 Tune the PID Controller

In your report explain the affects of your controller, and its limitations on the following:

- The maximum overshoot
- The length of time till the error within 5 cm

2 Submission

Submit one .pdf file of your report and your modified 'pid.py' file.

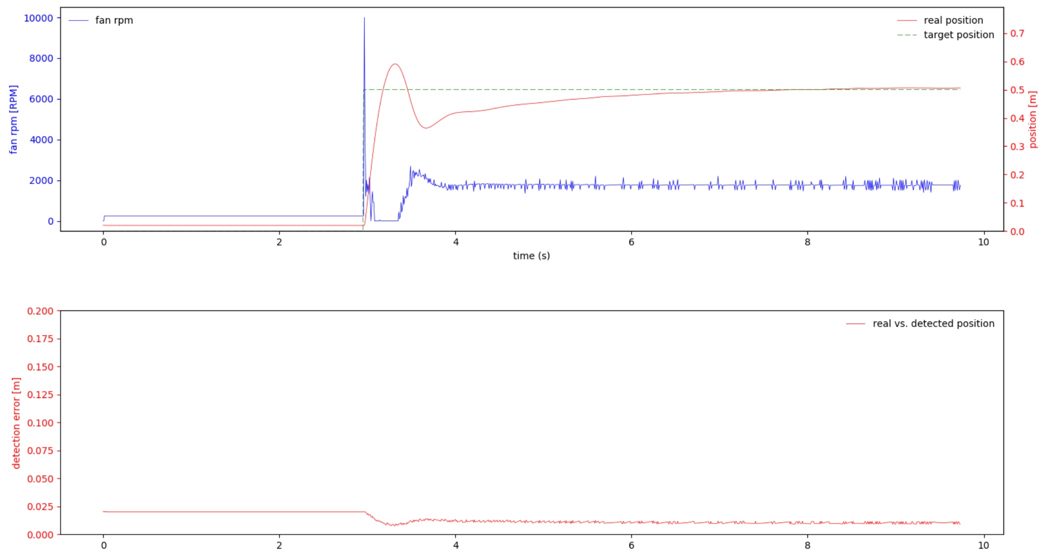

Figure 3: Step response

In your report, include a plot of each case of the position as a function of time (non optimized example shown in figure 3), and indicate the settling time and overshoot (you can just draw this over the plot).

Again, your pid.py must run using the originally provided source code, as this will be used to evaluate its performance.

3 Tips and warnings

Submit your code and assure it runs on Python 2.7

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.