REPORT PID Controller and Mimicking Robot

Members:

Shrinand Thakkar 111483215
Amogh Reddy 111493841

Camera Calibration

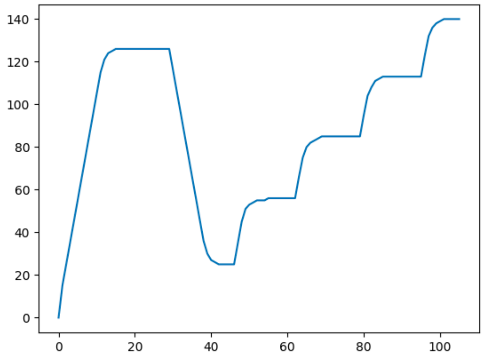
- We have first converted the image to hsv and have used masks for detecting Red and Green color points on the servo.
- To calibrate the camera, we have used 3 different red points and one green point.
- The red points are at 30, 90 and 120 degrees in the reference (A) servo.
- We then used the getPerspectiveTransform function to get the Perspective Projection Matrix.

Mimicking

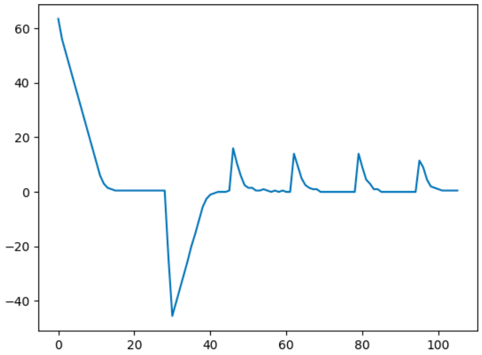
- While Mimicking, we have commanded the reference servo to rotate from 15,150 in the intervals of 30.
- Then we have used the Perspective Projection Matrix in the camera calibration code and got the real coordinates of the reference servo.
- Then we have calculated the angle of reference Servo and computed the Error in the angles.
- And if the error exceeds |10|, we have just cut it down to 10 or -10 resp., otherwise it is kept as it is.
- And then we have sent the error to the B servo.
- And after the servo reacts, we will again read the current angle of B servo and get it back serially in Python.
- Then these things will loop, and it computes the error again and commands the B servo.

PID

- For the **P part**, we have multiplied the error with a constant and returned the error.
- We have checked for constant=0.5

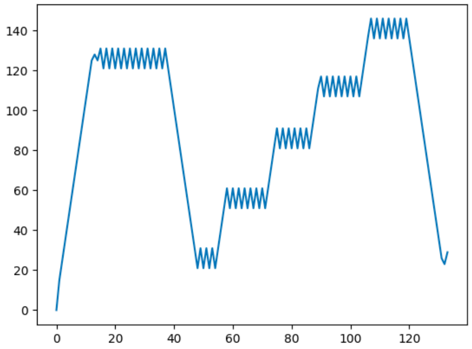


- This graph shows the **angle of servo B at each time step.**
- As we can see, as the error decreases, the change in the angle difference decreases wrt time, as in the angular velocity decreases.



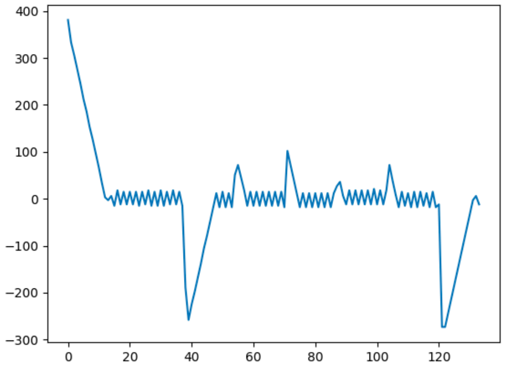
- This graph shows the **errors in servo B at each time step.**

- We have also checked for constant=3



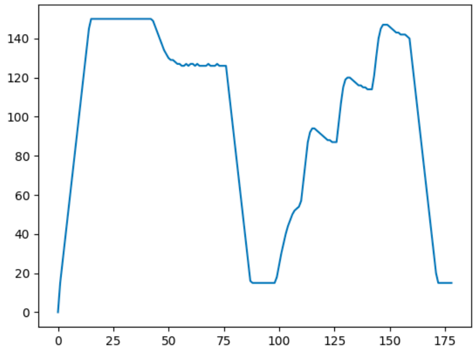
- This graph shows the **angle of servo B at each time step.**

- As the constant value is high it will always overshoot and adjust.
- That shows the ups and downs at the peaks.

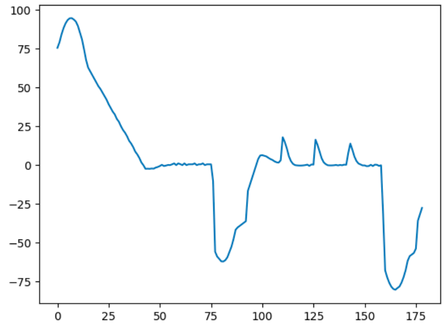


- This graph shows the **errors in servo B at each time step.**

- For the **PI part**, we have multiplied the error with a constant and used a global constant which would keep all the errors and returned the error.
- For Kp=0.5 and Ki=0.1



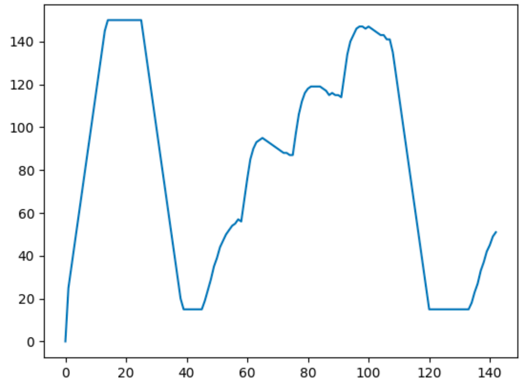
- This graph shows the **angle of servo B at each time step for PI.**



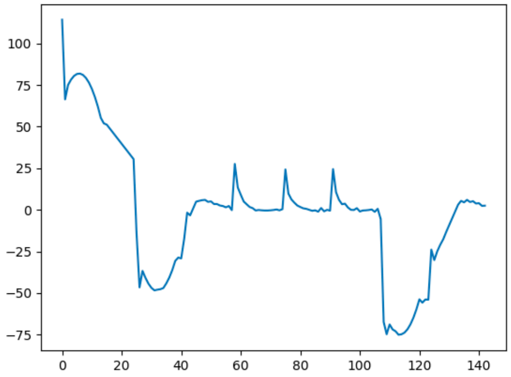
- This graph shows the **errors in servo B at each time step for PI.**
- We can see here that, the errors are not peak pointed, as PI also takes into account the previous errors, the performance is a little inproved.

- For the **PID part**, we have multiplied the error with a constant and used a global constant which would keep all the errors and also have taken into account the change in errors and returned the error.

- For Kp=0.5 and Ki=0.1 and Kd = 0.5



- This graph shows the **angle of servo B at each time step for PID.**



- This graph shows the **errors in servo B at each time step for PID.**
- As we can see, the errors are still improved than the only P part, but the performance of PI is more efficient in reducing the errors than this.