Applying PID to Botball
Youngju Yun
Polaris, 13-0535

Applying PID to Botball

1. Introduction

Our robot relies on line following for navigation. We researched different ways of line following, and found a basic line following program. A light sensor is positioned on the edge of the black line. The sensor returns a value in the range of 0 to 1000, depending on whether a white or black color is detected.

In the basic implementation, if the line is to the right of the robot, and it senses the black line, then it turns left. If it senses the white game board, then the robot turns right. The robot is not instructed to drive straight. While the program worked, it was slow and unreliable. Constant turning caused the robot to waste time moving left and right, with little forward motion. Speed is essential in Botball, because you want to reach the targets before the opposing team. Our coach introduced us to the PID controller, and explained how it could be used to create a smooth, efficient line following program.

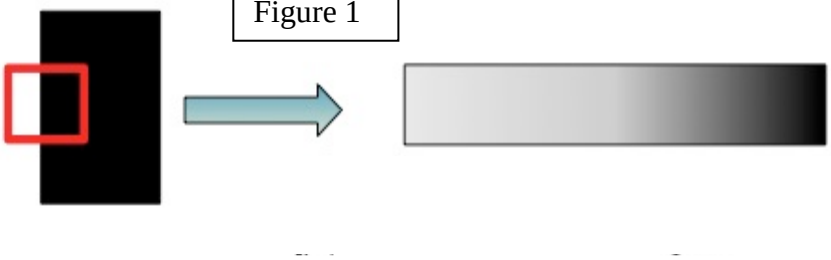
2. Fundamental Concepts

A PID controller consists of three parts, called the Proportional, Integral, and Derivative, hence the name PID **[1].** A PID controller is a feedback loop, with an input and output.

Without PID, the basic program would use “black” and “white.” Because the light sensor can only return a value from 0 to 1000, the colors “black” and “white” are artificially defined. Any value below 500 can be classified as being “white” and any value 500 or above would be considered “black.” The robot turns left when it senses “black” and it turns right when it senses “white.”

When using a PID controller, the full scale of 0 to 1000 is utilized. It recognizes the sensor values as a grayscale, not just as “black” or “white”. The shade of gray can indicate the sensor location in relation to the line, not only if it is to the left or right of it. Figure 1 shows how the light sensor takes the average of the black and white regions and interprets it as a shade of gray.

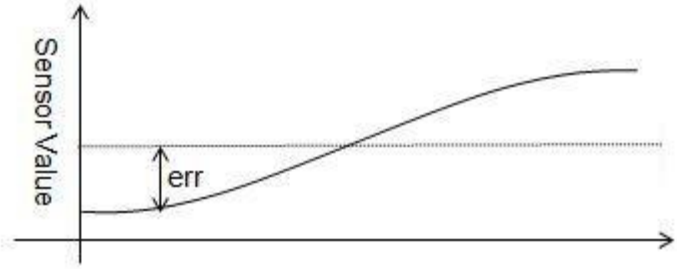
Figure 1



If the sensor (represented by the red box) is in between the black and the white border, and it will return a number between 0 and 1000 that corresponds with the shade of gray.

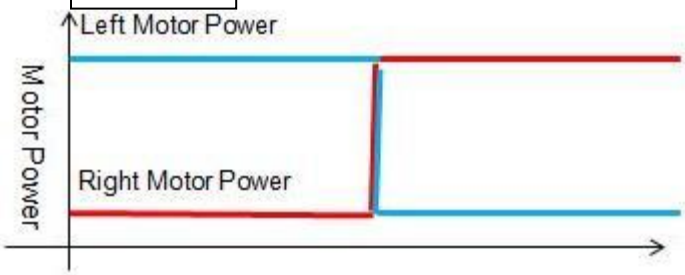
The "Proportional" control of PID uses the sensor value in order to follow the line. As shown in Figure 2c, the left motor power and the right motor power are inversely proportional to each other, and correspond to the sensor value. The amount of turn is directly proportional to the light sensor value.

Figure 2a



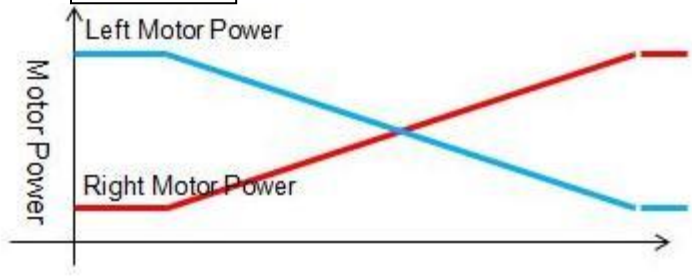
The error is the difference between the current value and the target value. In this case, the current value is the value of the light sensor, and the target value is the edge of the line(500). The further away the light sensor is from the edge of the line, the larger the error value. Left and right is represented by a positive and negative error value.

Figure 2b



This graph shows the motor powers before PID was added. The robot is constantly turning. The power of the turn is preset and cannot be changed. The direction in which it turns changes depending on whether the light sensor is over "black" or "white".

Figure 2c



This graph shows the motor powers with PID. The power of the turn is not preset, but is determined by the value of the error. The two motors are inversely proportional, and both of them react to the error value to propel the robot forward.

The "Integral" control of PID records the error value over time. If there is a consistent pattern to the errors, then the robot will compensate for it. For example, if the robot is constantly drifting to the left, the program will add a slight right turn in order to counteract the unwanted movement. Lastly, the "Derivative" portion of the program will predict the next error. It does this by subtracting the previous error and the current error.

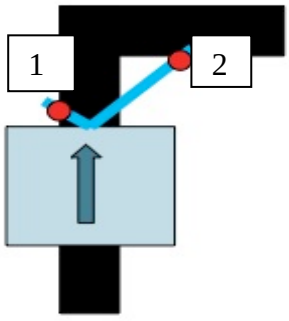
If the error is getting smaller, then the program will do nothing, but if the error suddenly increases, the derivative will be detected and the motor power will be adjusted in an attempt to minimize the error.

The PID controller manipulates the robot so that the sensor input changes the desirable outcome. We have observed that PID control makes our line following faster and more accurate. It utilizes proportional control in order to correct any mistakes that the robot makes. Not only does the PID controller immediately rectify errors, but it can also compensate for past miscalculations. It keeps a record of all the errors it made previously, and it makes a prediction as to what the next error will be. It tries to prevent the robot from making the next error and minimizes each subsequent error.

3. Our Innovation

After we created a line following program with a PID controller, we continued to improve it. PID still has difficulty following a line through a 90 degree turn. When the robot moves too quickly, the program cannot react in time to make a sharp turn. Reducing the robot's speed would solve this problem, but the robot moved too slowly to be useful. Our solution was to modify the program and the robot. We attached an arm with a second light sensor to sense the ground at the front-right of the robot. In this year's game, the black line loops around the entire game board. The robot only has to follow it going clockwise, and only makes right turns. The new light sensor in the front-right will sense the black line when there is a sharp turn. Now, the robot slows down when it senses the 90 degree turn. This allows the robot to maintain a rapid pace, while still being able to traverse the sharp turn.

Figure 3



The blue box represents the robot moving towards the right angle turn. The two red dots labeled "1" and "2" represent light sensors.

Sensor 1 is used to follow the line.

Sensor 2 is used to detect the right angle turn ahead of it, giving the robot time to react appropriately.

We also explored the use of PID to control all of our mechanical functions. We used a PID controller on the camera to locate and pick up objects. The program follows the control mechanism of three PID components. In this program, the input is the position of the colored blobs on the camera screen. The proportional driver will change the position of the robot until the blob is centered on the screen. This means that if the blob is left or

right of the screen's center, the robot would turn right or left respectively, to center the blobs on the y axis. If the blob is above or below the center, then the robot moves forward or backwards, until the blob is on the x axis. The PID controller counts the pixels between the center and the blob, and uses the pixel value as the error signal as shown in Figure 2a. This process is repeated until the object is centered. The integral and derivative sections remain largely unchanged.

4. Data

The data tables below show comparisons between two programs, a line following program with PID, and a line following program without PID as the control. Both programs were downloaded into a test robot, and timed as they followed the line. The time is in seconds, and the experiments were conducted ten times each.

Figure 4

PID	8.43	7.97	8.23	8.73	8.45	8.65	8.12	8.03	8.47	8.38
Control	15.41	15.67	15.08	15.16	15.09	15.07	15.59	14.65	14.75	15.01

Figure 4 is a data table that shows the performance of both programs following a straight line. The data clearly shows that the program using PID was faster than the control program. While the experiments were being conducted, it was easy to see that the program with PID ran smoothly, while the control program was constantly turning and changing directions. The length of the track was 5.5 feet.

Figure 5

PID	2.78	2.71	2.69	2.94	2.67	2.82	2.86	3.29	3.14	2.44
Control	4.71	N/A	5.17	5.92	N/A	5.22	N/A	5.29	4.93	N/A

Figure 5 is a data table that shows the performance of both programs while following a 45 degree turn. The data shows that the program with PID was consistently faster than the control program, and could manage the 45 degree turn all of the time. However, the control program failed the 45 degree turn and ran off the course about half the time. These trials are represented by a N/A in the data table. The track was 2 feet long, with the 45 degree turn in the center of the track.

Figure 6

PID	3.92	3.61	4.59	4.45	3.23	4.55	4.61	3.82	N/A	4.34
Control	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A

Figure 6 is a data table that shows the performance of both programs while following a 90 degree turn. The program with PID also used the additional light sensor in the front-

right of the robot. We observed that the robot sensed the upcoming 90 degree turn and it slowed down in preparation. The program succeeded 9 times out of 10, only missing the turn once. The control program was not able to follow the line throughout the sharp turn at all. The track itself was similar to that of the 45 degree turn, being 2 feet long and having the turn in the middle of the track.

5. Benefits of PID

The PID controller has many advantages over other types of controllers. The robot runs more smoothly compared to non-PID black and white control. Before PID, we instruct the robot turn or move a preset distance, and the robot often undershot or overshot the target. The proportional driver eliminates that problem, and calculates what is needed to get to the target. Also, if the robot runs into an obstacle, or has some other difficulty, then the Integral driver can compensate for it. Lastly, the robot will be able to predict the future path, and it can prevent itself from making errors. With PID control the robot will maintain its course and travel much more efficiently.

6. Conclusion

PID is a more efficient, reliable way of programming. By using proportions instead of set intervals, the program becomes for accurate and runs smoothly. In addition, the program compensates for its past mistakes, and it learns from them to prevent mistakes in the future. In our tests, programs with PID were significantly faster than their non-PID counterparts. PID can be applied to almost any program, and it is a valuable tool that all programmers should know how to use.

7. Sources

[1] Sluka, J. "A PID Controller For Lego Mindstorms Robots." *PID Controller For Lego Mindstorms Robots*. N.p., n.d. Web. 2 June 2013. <http://www.inpharmix.com/jps/PID_Controller_For_Lego_Mindstorms_Robots.html>.