Sameen Mahtab

Chen Liu

Report Lab 1- Wall Following

1. Design Evaluation

Hardware Design

The main structure of the robot is built taking the reference from LEGO instruction. The main parts of the robot include an EV3 Brick, two motors, an ultrasonic sensor and two wheels. The components listed above are connected by LEGO sticks. Instead of installing the two motors on both sides EV3 brick, the two motors were installed under the EV3 brick. In this case, we have a thinner robot, so it's easier for our robot to turn when the space for turning is small.

For this lab the only sensor we needed to use was the ultrasonic sensor. Our robot consists of two wheels able to operate independently and a third wheel used for balance. We connected the ultrasonic sensor close to the body at the front left corner with the angle of 45 degrees with our robot as we wanted our robot to travel in an anti-clockwise direction. The sensor can see both the wall in the front and the wall on the left simultaneously so the robot knows when it meets corners.

Software Design

The idea of this lab is designing a robot that can follow a wall and keep a constant distance with the wall. The robot should be able to detect corners and determine when turning is needed. When the robot is too far or too close to the wall, it should correct the distance automatically by adjusting the speed of both wheels. The entire wall following process is achieved using two methods: Bangbang control and Proportional control.

The ideal distance from the wall to the center of the robot is set to 20cm (this is the bandCenter). An acceptable error of distance is set to 3 cm. It means that when the distance between the wall and the center of the robot is within 17cm and 23cm, the robot will simply go straight instead of turning to any direction. However, when the error occurred exceeds 3cm, the distance will be corrected. There are two ways of correction as mentioned above.

For our bangbang controller class, we implemented it in a way that out robot will turn either toward or away from the wall with a constant speed regardless of the distance from the wall. For example, when the distance detected exceeds 23cm, which means that the robot is too far from the wall, the robot needs to turn left to get closer to the wall. In this case, the left motor needs to decrease its speed, but the right motor needs to increase its speed to accomplish the turning (correcting) process. Vice versa when the robot needs to turn right. Regardless of how much the error is, the motor speed will increase or decrease by the same amount, until the error comes back to the acceptable range.

However, the proportional controller works differently. When the robot needs to turn, both motors will increase or decrease their speed by the amount that's proportional to its error. It means that when the error detected is very large, the robot will turn faster than when the error is relatively smaller. For example, in our case, the proportion ratio set is 4.5, when the error measured is 30cm, the right motor will increase its angular speed by 4.5*30=135(degree/second). However, when the error is 50cm, the right motor will increase its angular speed by only 4.5*50=225(degree/second), which turns the robot faster.

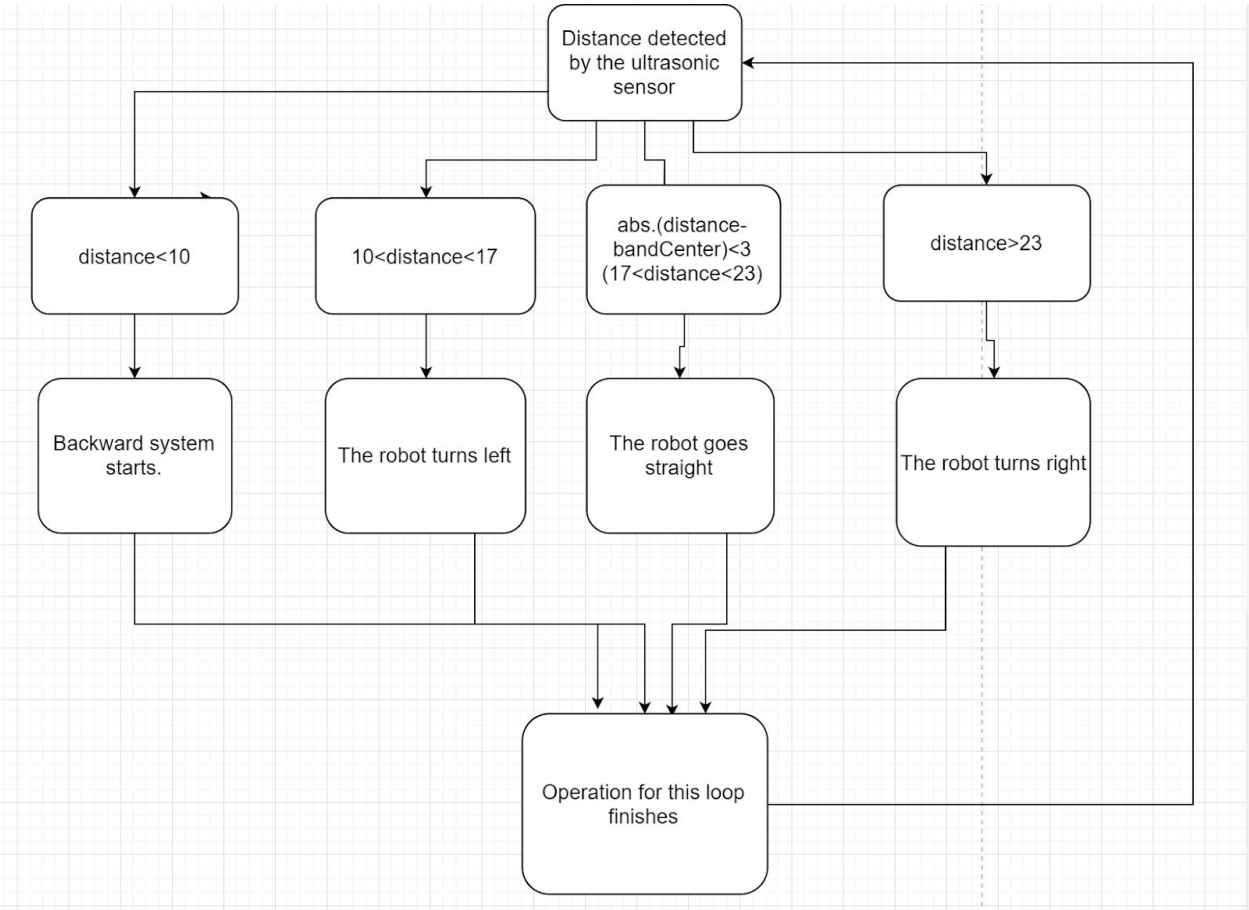
The process of finding the best proportion ratio took a few hours. In the beginning, we tried some figures that is much higher than 4.5, for example, 10. The degree of correction becomes too much. It hits the wall very often.

We decided to lower the ratio down. The robot behaved better with a lower value. We tried a few times to approach our ratio to 4. Although the robot doesn't collide with the wall, it doesn't follow the wall very closely, especially when it meets corners. For this reason, we created a method called calcTurn to make the ratio a number with a digit decimal instead of using integers. In the end, we figured out to use 4.5 as our proportion ratio.

The process of finding turning speed in Bangbang control also took time. We tried for a few hours to get to the figure that gives a perfect performance. We decided to set the speed of the right motor to 0 when the robot needs to turn right, because the robot hits the wall very often, especially when it meets corners. However, the robot still sometimes touches the wall, so we created a backward system to avoid the collision. When the distance detected is smaller than 10, which is very likely for the robot to hit the wall, the robot will simply go backwards, to have a larger space of turning. Because this method improves the performance significantly, we copied it to the proportional controller too.

As for accounting for gaps, this was done using filterControl. This was extremely useful when using the pController where the instantaneous distance between the wall and the robot was used. What the filter control does is limit the distance measured by the ultrasonic sensor to 255. This stops the robot from drastically changing wheel speed when the ultrasonic sensor for example does not detect anything with 255 cm.

Both of the methods above follow the flow chart shown below. First the distance is detected, then the robot operates differently depending on the value of the distance. It goes to a new loop after finishing the operation.


Distance detected by the ultrasonic sensor
distance<10
10<distance<17
abs.(distance-bandCenter)<3 (17<distance<23)
distance>23
Backward system starts.
The robot turns left
The robot goes straight
The robot turns right
Operation for this loop finishes

2. Test data

Testing the p-type controller constant

(The final ratio chosen is 4.5)

Proportional ratio	Wall touching	Band Center distance	Oscillation frequency
4	Did not touch the wall in the first lap, but touches in the second lap when turning right	Very much greater, especially after making the U-turn	Often when going straight
8	Touched the wall in the first lab when making the U-turn. Very sharp turn	Very close to the wall	Very frequent when going straight

Bang-Bang controller test

Trials	Wall touching	Band Center distance	Oscillation frequency
1	No touching	Very well kept when going straight, didn't get away too much when corner met	A bit often when going straight
2	No touching	Very well kept when going straight, didn't get	Lesser than the first trial

		away too much when corner met	
3	No touching	Very well kept when going straight, didn't get away too much when corner met	Sometimes when going straight

P-controller test

Trials	Wall touching	Band Center distance	Oscillation frequency
1	No touching	Very well kept when going straight, didn't get away too much when corner met	A bit often in the first straight line
2	Touched when turning right, but corrected itself using the backward system	A bit too far when meeting a concave corner after making a U-turn	Not very often
3	No touching	Very well kept when going straight, didn't get away too much when corner met	Not very often

3. Test analysis

What happens when your P type controller constant is different from the one used in the demo?

When the constant is greater than 4.5(the one used in the demo), the robot had hard time when following a straight wall. The turning speed is too much, so it had a very high oscillation rate when going straight. When making the U-turn, the robot overcorrected the distance, which led to a collision.

When the constant is less than 4.5, the robot performs OK when going straight, but it's very hard for it to turn right, which initiated the backward system very often.

How much does your robot oscillate around the band center?

In case of the Bang Bang controller, the robot keeps on oscillating around the band center. For the P type controller, the oscillations are not that significant. In the case when the robot is at a corner, or when it has to turn, it oscillates for both controllers.

Did it ever exceed the bandwidth? If so, by how much?

When going straight along the wall, both controllers performs well. The robot did not exceed the bandwidth. However, when making the U-turns, the p-controller sometimes have a very close distance to the wall. Compared to the P-controller, the Bang-Bang controller works better when making the U-turns. The robot would most often oscillate by around 2-3 cm. The robot would oscillate more when using the Bang Bang controller.

Describe how this occurs qualitatively for each controller?

For the BangBang controller, the robot would be further away from the band center and for a longer period of time. The P controller would quickly change the direction of the robot when the sensors sensed that the robot was exceeding the bounds.

4. Observations and Conclusions

Based on your analysis, which controller would you use and why?

In our opinion, P-controller should be the better method to use. Although in our testing, the Bang-Bang controller did perform better in making the U-turns, but the P-controller has a much higher turning speed and lower oscillation rate. The mechanism of P-controller is also more effective. The robot goes faster under the P-controller, and the turnings are also quickly made. The robot was more efficient and accurate when using the P-type controller. For instance, when using the P-type controller, in a straight line, the robot would travel straight instead of turning left and right constantly, it travels along the band center.

Does the ultrasonic sensor produce false positives(detection of non-existent objects) and/or false negatives(failure to detect objects)? How frequent were they? Were they filtered?

The ultrasonic sensor would produce errors but not so often. The most severe errors were due to interference caused by peoples feet. The false positives were caused by factors like residual noise or ground reflection. Most false responses were created when the sensor was too close to the wall, the distance sensed by the sensor would be a very large value when the robot was very close to the wall.

5. Further improvement

What software improvement could you make to address the ultrasonic sensor errors?

1) An exponential filter can be used to reduce the errors from the ultrasonic sensor. It cancels out spikes that are returned from the sensor.
2) The frequency of sampling could be increased.
3) A derivative filter can be used to take the average change of the incoming data.

What hardware improvement could you make to improve the controller performance?

1) Use two ultrasonic sensors one a little bit behind the one in front, so that even if the first sensor is too close to the wall and sending a wrong distance the other sensor can detect the wall.

2) Use wheels with greater diameter, this way the motors will not have to move as fast to create the same movement.
3) Replace the battery with one that provides stabler voltage output.
4) Replace the ultrasonic sensor with one that is more accurate, and has a higher refreshing frequency

What other controller types could be used in place of the Bang-Bange or P-type?

We could use the PID controller. The Proportional Integral Derivative Controller is a loop feedback mechanism. It is very useful in systems that require a continuously modulated control. It constantly calculates error values by finding the difference between the desired band center and a measured process variable and then applies a correction using the proportional, integral and derivative terms. This produces a more accurate result.