Steven Cangul (260744412) and Noah Levine (260684940)

Lab 1: Wall Follower

Data Analysis

After the building and testing phase of the wall follower, we noticed that the bang bang controller isn’t actually capable of efficiently keeping the robot at a user specified distance BandCentre. The reason for this is partly due to the fact that sensor wasn’t as accurate as we had hoped. It often got confused and read values that were significantly different than what the robot was actually experiencing.

In the testing phase we noticed something particular about the trajectory and motion of the robot. The robot was oscillating from one side of the band and the other repeatedly. The reason for this oscillatory motion is due to the fact that the sensors are hard mounted to the robot’s frame. As the bang bang or P-controllers try to implement a correction due to the robot wandering off too far from the wall or getting dangerously close to the wall, as expected, the robot’s frame turns and consequently drags the sensor along. For example, if the robot tries to move away from the wall, the sensor would be pulled away from the wall. Because of this, the sensor now reads a false reading. As our deadband is relatively unforgiving given the accuracy of the sensor, the robot tries to then get closer to the wall. This happens back and forth and the robot appears to oscillate. It is important to point out though that this isn’t controller specific. In fact, due to the fact that bang bang functions by turning the motors on and off in short and direct pulses encourages the variations in the sensor readings. P-controller on the other hand implements more gradual and smooth corrections which explains why the oscillations are much less frequent.

Observations and Conclusions

As stated previously, the ultrasonic sensors weren’t as accurate as we had hoped. Many times, the sensor would read false readings, whether it be seeing an object that doesn’t actually exist or failing to see an object. This was mostly noticeable when the robot was either too far or too close to the object in question. In fact, when the sensor is too close to the wall, the wave it sends out isn’t capable of coming back to the receptor of the sensor in time for the computer to make the optimal correction to said situation. There are many factors that may cause these minor or sometimes large fluctuations. The first of which is the charge status of the robot’s power bank. In fact, a weak battery could in fact alter the readings the ultrasonic sensor would normally read, as the weaker battery won’t be able to provide enough voltage for the sensor to function properly. Luckily however, these errors are, for the most part, filterable.

The first way to filter the readings from the sensor was to actually code the robot’s program to reject any reading that was above 255cm, as this is the maximum distance the sensors are rated to be able to output with a certain degree of certainty. Any value above this is considered garbage and should be discarded as it would confuse the robot.

Another way of filtering the sensor’s output is to actually count the number of consecutive readings of the same magnitude. In fact, if the sensor reads an X distance a certain number of times, this would strongly suggest that the reading is in fact accurate. However, if reading a value X only a few times in a short period of time would strongly suggest that the reading is in fact an anomaly and shouldn’t be taken into consideration by the robot when the decision taking phase occurs.

Further Improvements

Since no project is ever finished nor perfect, there are certainly many improvements that could have been made to our robot. The first pertains to the way the robot was built. We noticed that our robot would often get way too close to the walls and often make extremely tight turns. To remedy this, we could have put our motors closer to each other. This way, our wheels would be closer to each other and would in turn reduce our turning radius drastically as well as package our robot in a tighter and more compact frame.

Another area that could have been improved is the number of ultrasonic sensors used. Ideally, the best solution would be to place one camera facing the front of the robot and one the side of the robot. That way we could constantly monitor both the side and front distances at the same time. We could have done this using a single sensor mounted to a motor that cycles through multiple different angles to take measurements. However, using this method would require us to limit our sampling rate which would cause the robot to hesitate under sudden changes.

Since no project is ever finished nor perfect, there are certainly many improvements that could have been made to our robot. The first pertains to the way the robot was built. We noticed that our robot would often get way too close to the walls and often make extremely tight turns. To remedy this, we could have put our motors closer to each other. This would in turn reduce our turning radius drastically as well as package our robot in a tighter and more compact frame.

Another area that could have been improved is the number of ultrasonic sensors used. Ideally, the best solution would be to place one camera facing the front of the robot and one the side of the robot. That way we could constantly monitor both the side and front distances at the same

time. We could have done this using a single sensor mounted to a motor that cycles through multiple different angles to take measurements. However, using this method would require us to limit our sampling rate which would cause the robot to hesitate under sudden changes.

Finally, a better filtering approach could have been implemented. Our robot only used basic filtering techniques that were described previously. However, using multi-threaded programming, a better filter could have been implemented. There could have been a main thread and a timer thread. The main thread would read the distance continuously. However, the timer thread would be sleeping most of the time and only activated at certain clock pulses. During these pulses, the timer thread would read the distance measured and update the robot's motors appropriately. Thus, the robot would oscillate less and read in less false values, as most measured values would be ignored since the timer is asleep.

Yes, there are other controller types that could have performed better than the bang-bang and p-type. One such example is the PID controller. The PID controller works very similarly to the p-type controller. In addition to calculating the adjustment needed based on a proportional constant, the PID controller also takes into account two other factors. These factors are the integral and the derivative of the error. The proportional portion accounts for the current measured error, the integral portion accounts for the trends of past errors and the derivate portion accounts for potential future trends in error. Thus, a better correction would be applied than the p-type controller. Of course, the correction would be better than the one returned by the bang-bang controller as well.