CS 3630 Project 3 Part 2

Name: Andrew Wang
GT username: awang350
GTID: 903378837
Group #: 13

Name: Matthew Yang
GT username: myang349
GTID: 903447357

Given the calibration equations, how do you think increasing and decreasing the trim will affect the robot movement? We have set gain (g) to 1, how will your calibration be affected if you increase or decrease the gain?

According to the calibration equations, if we increase trim (r), the velocity of our right wheel increases and velocity of our left wheel decreases. If we decrease trim, then the velocity of our right wheel decreases and the velocity of our left wheel increases

$$v_{right} = (g + r) \cdot (v + \tfrac{1}{2}\omega l)$$
$$v_{left} = (g - r) \cdot (v - \tfrac{1}{2}\omega l)$$

This means that we can adjust the trim in order to counteract a curve in the robot's trajectory, which is caused by one motor rotating faster than the other one.

Gain acts as a multiplier on velocity. This means that with respect to calibrating the curve in the robot's trajectory, increasing gain will LESSEN the effects of adjusting trim. On the other hand, decreasing gain will INCREASE the effects of adjusting trim. (e.g. if the value of gain is something small such as 0.1, then a small change in trim of, say, 0.05, would drastically affect the ratios of the wheel velocities and the robot would be thrown completely off course, as Vr ~ .15, and vl ~ .05).

How did you tune the trim value to calibrate the motors? What is the final trim value you got?

We tuned the trim value via experimentation. First, we set trim = 0 and observed the direction that the robot would veer off, and by how much. Then, we used the calibration equations to "eye-ball" estimate the amount of trim required

For instance, if the robot's trajectory curved right initially, then vl > vr and we need to increase trim. On the other hand, if the robot's trajectory curves left, then vr > vl and we need to decrease trim.

We basically did trial-and-error until our robot followed a generally straight trajectory. Additionally, we had to re-calibrate the robot when we changed velocities / fiddled with the cords inside.

In the end, our final trim value was 0.03 for the v = 0.5 m/s trials, and our trim value for v = 1.0 m/s was -0.01.

Is there a better way of calibration? Explain your idea in detail. (1)

There are certainly better ways of calibration. Unfortunately, it may not be worth it. At least for us, in practice, it seems that the trim value required for a straight trajectory changes everytime we restart the robot / adjust any of the wires. Additionally, the velocity of the wheels continue to change even during the course of one run, as sometimes the robot will curve left then curve right (TA’s said this was fine). Nevertheless, here are some ideas:

1. Attach a piece of string (on a roll) to each of the left and right wheels. Then, set vl = vr in the code and let the wheels spin, pulling on the string. We can measure the amount of string used and treat it as distance “traveled” for each wheel. Then, we can compute the velocity of each wheel by dividing the string lengths by the runtime. Thus, we can solve for the exact amount of trim needed using the calibration equations. Here is a *rough* derivation, where vleft and vright are from the string measurement:

   The benefit here is that we are measuring purely the motor rotation, and environmental factors that vary from run to run are ignored.

$$(g+r)*\left(v_{string-measured\ right}+\frac{1}{2}\omega l\right)=(g-r)*\left(v_{string-measured\ left}+\frac{1}{2}\omega l\right)$$

$$g*\left(v_{right}+\frac{1}{2}\omega l\right)+r*\left(v_{right}+\frac{1}{2}\omega l\right)=g*\left(v_{left}+\frac{1}{2}\omega l\right)-r*\left(v_{left}+\frac{1}{2}\omega l\right)$$

$$v_{right}g+\frac{1}{2}\omega lg+v_{right}r+\frac{1}{2}\omega lr=v_{left}g+\frac{1}{2}\omega lg-v_{left}r-\frac{1}{2}\omega lr$$

$$r(vright+vleft)=g(vleft-vright)-\omega lr$$

$$r=\frac{g(vleft-vright)-\omega lr}{vright+vleft}$$

Is there a better way of calibration? Explain your idea in detail. (2)

I already explained the previous method in detail, but some here are some other ideas are:

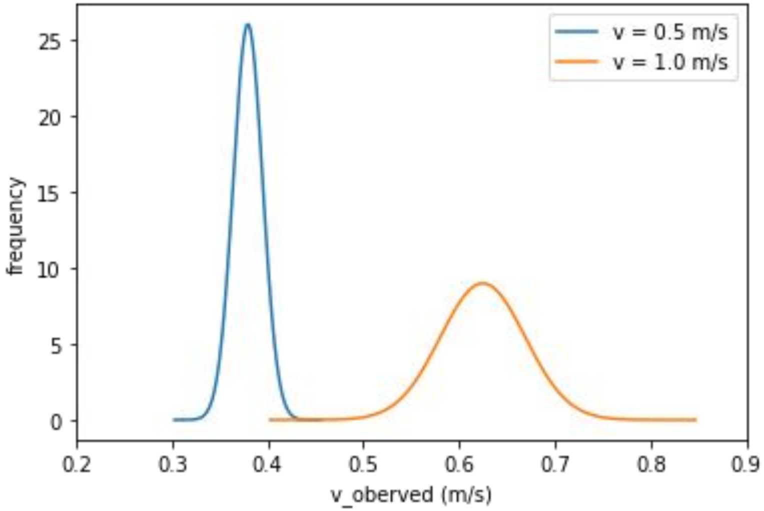
1. Running many trials, and calculating the average radius of the trajectory, then solving for the trim.
2. Running many trials with varying amounts of trim, and plotting the average offset vs. trim values. Then, we can pick the trim that minimizes the offset.

Compare the two Gaussian distributions, one for v at full speed and one for half speed. How are the two distributions different? Why are they similar and why are they different?

Firstly, I’d like to point out the average v_fullspeed < 2*v_half. The reason is due to inconsistency in the duckie bots, as it is more likely the duckie bot will make random turns when traveling a longer distance. However, for consistency, the distance measured is from start to finish, disregarding the path the robot took.

The two distributions are similar in that they are both unimodal at the mean velocity and have small enough standard deviations such that the v_observed has a reasonable range for both (i.e. the bots velocities are decently close to the expected).

In addition to the aforementioned speed difference, the distribution for v = 1.0 m/s clearly has a higher standard deviation (.044), resulting in a flatter shape. On the other hand, the distribution for v = 0.5 m/s has a sharper peak and lower standard deviation (0.015)



What did you learn in this project? What are the difficulties during this project, and how did you solve it?

I learned a decent amount in this project. In terms of logistics, I learned that it's best to start early (and we did!) in case you run into technical difficulties (sadly, we did). In addition, this was our first time gaining hands-on experience with hardware and not just coding software, which was pretty cool.

In terms of course content, I learned how to calculate differential-drive robot's trajectory, how to optimize the duckie bot's movement for real-world conditions, how to use matplotlib, and I also improved my critical thinking skills / lab skills.

The main difficulties in this project were related to setting up the hardware, getting the SD drive to flash properly, and getting the robot to move. We resolved these difficulties by spending many hours with the TA's down in the COC lab.