# CS 3630 Project 3 Part 2

Name: Zhen Jiang
GT username: zjiang330
GTID: 903402987
Group #: 19

Name: Fei Ding
GT username: fding33
GTID: 903444656

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?

Based on the calibration equations, $v_{right} = (g + r) * (v + \frac{1}{2}wl)$ and $v_{left} = (g - r) * (v - \frac{1}{2}wl)$, we know how the gain (g) and trim (r) will affect the robot movement. Since we move in straight line, $w = 0$, and therefore $v + \frac{1}{2}wl$ and $v - \frac{1}{2}wl$ just reduce to $v$. Hence, we only care about the left part of the equation. When the trim is positive as $g$ remains constant, $g + r > g - r$, and hence $v_{right} > v_{left}$. It makes the right wheel move faster than the left wheel, and the robot will tend to go left. In the opposite, when the trim is negative, the left wheel moves faster and thus the robot steers right. Furthermore, in both equations, $v_{right}$ and $v_{left}$ is proportional to $g$. It means that as we increase or decrease the gain, $v_{right}$ and $v_{left}$ and will also increase or decrease respectively. However, to compensate this change in $g$, we will also have to scale trim $r$ proportionally. Smaller gain requires fewer trim as the equations suggest.

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

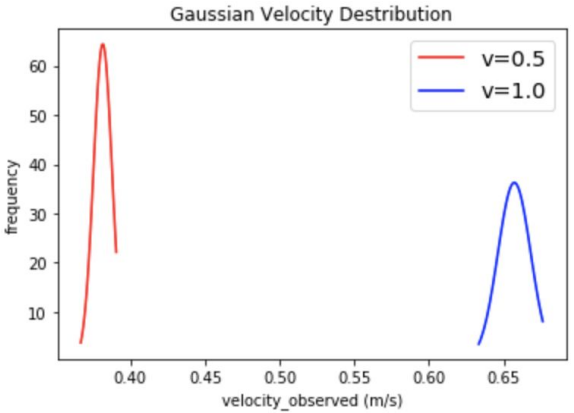
Initially, we set the trim value to 0 and ran move_in_line.py to the Duckiebot on a wooden floor. The reason that we did this is that we want to know the original movement of the Duckiebot so that we have a good idea of how we would tune the trim value. We ran the Duckiebot on a wooden floor because it provides a smooth and frictionless surface for the Duckiebot to move. Furthermore, we did this step multiple times because we wanted to make sure that we got this result not by accident. The initial result showed that the Duckiebot tends to go right, so we knew that the left wheel runs faster than the right wheel. After that, we began to apply the trim to the Duckiebot. We first started with the value 0.01 to make the right wheel run faster, and we ran this step multiple times. We found that the Duckiebot still went right, so we incremented the value of the trim again. We repeated this process a lot of times, and the final trim value we got is 0.03.

## Is there a better way of calibration? Explain your idea in detail.

When we did this project, we had a nice observation. As we set the velocity of the Duckiebot to 1.0, we had to set the trim value to 0.03 in order to make Duckiebot move in a relatively straight line. However, as we set the velocity to 0.5, we just left the trim to be 0, and the Duckiebot moved in a straight line. Unless there was an issue in the implementation of our robot, this phenomena gave us a great insight: the change in the velocity will also affect the trim value. Currently, it leaves us with a problem that, as we change the velocity for the Duckiebot every time, we need to tune the trim value again and again. However, a good solution to this problem is to calculate the ratio between the velocity and the trim. The ratio enables us to calculate the appropriate trim value to give to the Duckiebot as we alter the velocity. Therefore, this method may help to relax the burden of testing the trim value repeatedly and make the calibration process much more smoothly.

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?

Both distributions follow Gaussian distributions (has a bell curve). Since we did the experiment with enough times (10 times for half speed, 15 times for full speed) in an independent manner, according to Central Limit Theorem and Law of Large Numbers, the distribution we got for each one would be a Gaussian distribution. However, the distribution for half speed has much higher tip and also smaller standard deviation compared to that for full speed. First of all, they are different because the time it takes to accelerate to full speed is much longer. In addition, when the Duckiebot is at full speed, an unstable state, there are a lot of variations in the movement of the Duckiebot, making the value deviate from the mean more frequently.



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

We really learnt a lot in this project. Throughout the whole process of project 3, we flashed the SD card, connected our phone hotspot to the robot, utilized the secure shell to get our code running on the Duckiebot, tuned the trim value to let the robot move in a straight line, and finally sampled data to plot Gaussian curves. It was such a great experience for us to have a hands-on practice to interact with the Duckiebot. Although the project 3 was somewhat time-consuming, it was truly exciting compared to the first two projects. During this project, we had difficulties when we flashed our card. It really took a lot of time to get this done since we had never done this type of stuff before. Furthermore, it was hard to tell if the card works or not just looking at the signals from the Raspberry Pi as we inserted the card. During the process, we just remained patient and careful, going through each instruction with caution to make sure we did everything correct. Although it roughly took about five hours to flash the card, we finally did it!