Dean Fletcher

Report – Lab 5

Video: https://youtu.be/PBFeSPdHjXA

Task 1: Localization from Motion Estimation

For this task we are to use IMU and Encoders to determine where the robot is in the global space. By knowing where to start and by using the IMU to tell us the direction the robot is heading we can keep track of where we are. My controller is design to put the wall of the map on the right side of the robot. Once this is done all that needs to happen is wall follow using encoders. Once the robot has found that the outer wall has been navigated it determines if the middle cells have been visited. If not orientate and travel those next. Once the robot has done so it will find itself surrounded by visited cells and terminates.

Kinematics:

Known: Velocity, Distance

Turn velocity = w(R + axle/2)
Wheel Velocity = velocity / wheel radius
Drive Time = distance / speed

Task 2: Localization from Measurement Estimation to Colored Landmarks

Like task one our goal is to navigate all unvisited cells. The added complexity here is that we don't always know our heading (orientation) or starting cell. We need to determine all values based on the trilateration using the camera and colored cylinders. By using trilateration, we can find out (x, y), which can then be used to determine the current cell. To find orientation without the IMU, the controller uses timed turns from the cylinders. With this logic we can navigate the maze as in task one.

Trilateration:

$$A = (-2 * x1) + (2 * x2)$$
$$B = (-2 * y1) + (2 * y2)$$
$$C = (r1^2 - r2^2 - x1^2) + (x2^2 - y1^2 + y2^2)$$
$$D = (-2 * x2) + (2 * x3)$$
$$E = (-2 * y2) + (2 * y3)$$
$$F = (r2^2 - r3^2 - x2^2) + (x3^2 - y2^2 + y3^2)$$

$$RobotX = \frac{C * E - F * B}{E * A - B * D}$$
$$RobotY = \frac{C * D - A * F}{B * D - E * A}$$

** Note: if ($E * A == B * D$) then divide by zero error
** To fix pick different cylinder and recalculate D, E and F

PID:

Kp (0.1, 0.5, 1.0, 2.0, 2.5, 5.0)
$error = desired - measured$
$control = Kp * error$
$saturation = f_{sat}(control)$

$$f_{sat}(control)\begin{cases} if\ control > max & control = max \\ if\ min \leq control \leq max & control = control \\ if\ contorl < min & control = min \end{cases}$$

Implementation:

$Error = sensor_{value} - Goal_{distance}$
$Wheel\ PHI = Error * Kp$
max velocity: $-6.28 \leq max \leq 6.28$

Conclusion:

This lab was surprisingly difficult to get working correctly. I had some trouble figuring out an algorithm that wouldn’t leave any cells unvisited. Initially I designed it to perform a snake like pattern to leave no sells unvisited. In some circumstances this works perfectly, but on edge cases it was a nightmare to correct. This ended in me tearing it down and instead to prioritize traveling outer edge first. The change wasn’t too drastic but did cause all kinds of bugs that I had to suss out. Once that was working that logic closely resembled what needs to be done in task two. The added difficulty was no IMU or encoder to figure out where the robot is. I overcame this by first using trilateration to determine (x, y), which enables me to get the cell (n). Now that I knew my pose, I had to use the cylinders and timed turns to determine my heading. The combination of these tasks took a lot of manipulation to get working. In result both work as expected, though the time spent to get to this point did not leave much time to slowly increase reliability. This results in little error adding up as it traverses, which makes it increasingly difficult to remain on path.