QEA 2 Gauntlet Challenge

Caterina Cirone & Michaela Fox

Introduction

We decided to only do Level 1 (have the Neato go through a predetermined path to a predetermined location) just so we could focus on getting more of an understanding on the contents of this class. Our strategy consisted mostly of taking the LIDAR data one point at a time through for loops and utilizing the functions MATLAB has to offer to translate our math into the code. To initialize this, we took LIDAR scans of the Gauntlet challenge and converted it to points on a graph.

We proceeded on by making the potential field. In order to successfully get the path the Neato was supposed to go on, we had to make sure it wouldn't hit the walls. We used the equation

$$V(x,y) = \ln\sqrt{(x-a)^2 + (y-b)^2}$$

In order to find the sink, or where the Neato should be going towards, where a is the x-coordinate of the destination and b is the y-coordinate. To find the "source" or what the Neato should be moving away from we turned that equation into a negative value. In both situations, the equation for the gradient was

$$\nabla V = \frac{x-a}{(x-a)^2 + (y-b)^2}\hat{i} + \frac{y-b}{(x-a)^2 + (y-b)^2}\hat{j}$$

And we used this to formulate the quiver plots as well as the streamslice map. We would use this data to find the gradient descent as the Neato traveled through the Gauntlet. In this case, we would locate the local minima the Neato would have to go to, which in this case is our pre-determined sink (0.25, 1.25). The formula for gradient descent

$$w_{n+1} = w_n - \alpha\nabla_w f(w)$$

Is where w(n+1) is the new position, wn is the current position, and alpha is our step size, which needs to be between 0 and 1 if we were to reach the ball correctly and not slightly missing it or overshooting it. In our case, we made the step size 0.9.

To make the neato follow the desired path, we had to calculate the angle the neato must rotate and the velocity of its wheels. This prevents the neato from making wider or smaller turns than expected and overshooting or not reaching the highest point. To do this, we calculated the angle that the robot must rotate at each time step, or the angle of the new position subtracted from the angle of the old position. This not only gave us the angle of rotation but the direction the robot had to rotate as well, with positive angles telling the robot to rotate counterclockwise, and negative telling the robot to rotate clockwise.

To calculate rotation time, we determined a base angular velocity of 0.05 and used the equation

$$\Delta t = \frac{\Delta \theta d}{v_R - v_L}$$

which was derived by setting two equations for angular velocity equal to one another. When having the neato turn counterclockwise, we would set the right wheel velocity to our angular velocity and the left wheel velocity to the negative of our angular velocity, and have it turn for the time we calculated. This method was also used to turn counterclockwise, the only difference being that the right and left wheel velocities are swapped.

To calculate the amount of time the neato needed to drive forward, we used the equation

$$\Delta t = \Delta p / v$$

which was derived from the definition of velocity as the change in position over time. We assumed a base linear velocity of 0.1, and modified it as needed during our testing.

Code

We utilized two different codes for two different approaches. At first, we believed that just using the LIDAR data and implementing basic MATLAB functions would allow the Neato to go the correct path. We could plot everything, but we couldn't map out where the Neato was going. So, we collaborated with another team and adapted their code with our data in order to make the Neato drive - in the end, it all worked out.

Drive:

https://drive.google.com/drive/folders/1hSAneYDKNPk2BBebuNdbENZpO7lKjwkv