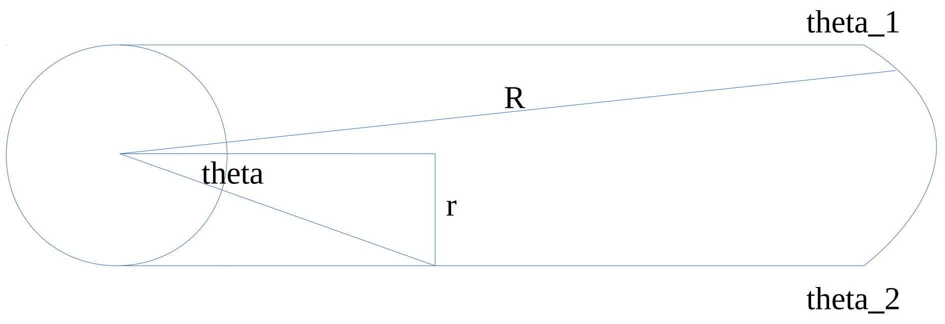
EECS 376 PS2

The goal of this assignment is to drive a turlebot equipped with a lidar around like a roomba. To do so, the lidar scans must be analyzed to determine if there is an object that the robot will run into if it moves forwards.

This is achieved by constructing a “corridor” of vision around the robot. If the measurements taken by a laser scan are within this “corridor,” then there is an object in the robots path, and the robot must turn to avoid it. Visually, this is shown as:



This “expected distance (e_dist)” is now a piecewise function. Along the corridor, we need to solve for the hypotenuse of the triange, giving:

$$e\_dist = Sqrt(r^2 + dist\_along\_corridor^2)$$

We can solve for the dist_along_corridor by noticing that:

$$\tan(theta) = r / dist\_along\_corridor$$
$$dist\_along\_corridor = r*\cot(theta)$$

This gives us:

$$e\_dist = Sqrt(r^2 + r^2 * \cot^2(theta))$$
$$e\_dist = r*\csc(theta)$$

This formula is unbounded: for small theta values, the value of e_dist grows. This physically makes sense, as the theta value approaches 0, we are examining laser scans closer to being directly in front of the robot, so if the robot is going to run into an object directly in front of the robot then the robot would just have to see an object in front of it, no matter the distance. However, this is not sufficient for the purpose of avoiding obstacles, we want to avoid obstacles if they get close enough, not if they exist. To account for this, some maximum radius (R) is specified, such that if a measurement is larger than R, then it is ignored. Therefore, e_dist becomes:

$$e\_dist = \begin{cases} R, & \text{if } theta\_1 <= theta <= theta\_2 \\ r * \csc(theta), & \text{otherwise} \end{cases}$$

Given a set of measurements (r_i, theta_i) pairs, if r_i <= e_dist, then there is an obstacle in the robot’s path, and the robot should rotate to avoid it.

When developing this algorithm, I paid attention to several details. The first is the stability of my e_dist function. e_dist is defined only for theta values between -Pi/2 and Pi/2 (since theta = 0 is the axis directly in front of the robot), then I need to restrict the laser measurements that are analyzed. For instance, the minimum and maximum theta values for a laser scan from stdr_simulator are larger than -Pi/2 and Pi/2, so part of the laser scan must be ignored. To do this, I developed an algorithm that, given a laser scan, a minimum theta and a maximum theta value returns the index pairs in the laser scan where the given theta range exists, and a theta pair representing the actual theta range to iterate over in the scan (assuming the given theta values may be close but not actually represented by discrete increments in the laser scan).

Using this algorithm and the above e_dist function, I was able to develop a method of parsing a laser scan to determine if an obstacle exists within the “corridor” of the robot and is close enough that the robot should rotate to avoid it.

Finally, I adapted code from Professor Newman that controls the robot given a “lidar alarm” to rotate in a pseudo-random direction (positive about the z axis or negative about the z axis) to prevent the robot from getting “stuck” in a loop as a result of always rotating in the same direction. To do this, I chose a predetermined yaw rate, and upon each lidar alarm (detection of an obstacle), I generated a random number, modded it with 3 to guarantee the number was in the set of {0, 1, 2}, subtracted 1 so the set is now {-1, 0, 1}, and multiplied the yaw rate by -1 if the random number was < 0. This strategy does not guarantee that the robot will not get stuck in loop, but does guarantee that the robot is not stuck in a loop forever, given enough running time, the robot will eventually turn in the opposite direction of the loop and break the loop.