Potential Field Implementation

By: Lilian Lamb

Introduction:

Potential fields are a method of path planning and obstacle voidance used in robotics. The idea is to create a set of attractive forces to pull the robot toward its goal and repulsive forces to push it away from obstacles. In combination, the forces help the robot navigate through its environment while avoiding collisions. Being able to translate the concept to code is important when developing an autonomous system. This assignment asked students to modify the provided potential field code to properly traverse a single cave environment.

Application:

The assignment called for two methods to be modified: laser callback and the spin method. I chose to calculate the total forces in the laser call back method and translate the forces to velocities in the spin method. The laser callback first calculated the difference in distance between the current position and the goal for x and y respectively. Those distances are then used to calculate the attractive forces by multiplying the distances by gamma (the attractive force coefficient). The next step is to calculate the repulsive force for each obstacle detected by the lidar. If the obstacle is between the safe distance and beta (the high limit of detection) then the repulsive force is equal to alpha (repulsive force coefficient) divided by the square of the difference between the distance to the obstacle and the safe distance. However, if the distance to the obstacle is less than the safe distance, then the repulsive force is equivalent to alpha divided by epsilon (distance coefficient) squared. This method of calculating forces ensures that the closer to the obstacle the stronger the repulsive force. Finally, the sum of all repulsive forces and the attractive forces are calculated for x and y respectively.

Now that the total forces for x and y have been calculated in the laser callback method, the spin method can use those variables to calculate angular and linear velocity. Angular velocity is calculated by finding the desired orientation using the atan2 function and the two sums of forces and adjusting the value by 0 and 2π discontinuity. Linear velocity is found by multiplying the sum of forces by their cosine of heading and sin of heading respectively. Several checks are then made to determine what to feed the robot. If the robot is already at the desired position, then no velocity is sent. If the linear velocity is less than 0, then only the angular velocity will be sent. If the linear velocity is 0, then the robot is in a local minimum and a random force is sent in the negative direction. Finally, if the linear velocity is greater than 0, then the calculated linear and angular velocity is sent to the robot. These checks ensure that the robot will continue its path and avoid obstacles while being able to get out of local minima.

Figures:

The figures represent a part of the path the robot took to its destination. Due to the limitations of the footprint option in stage, there were some limitations on what could be shown.

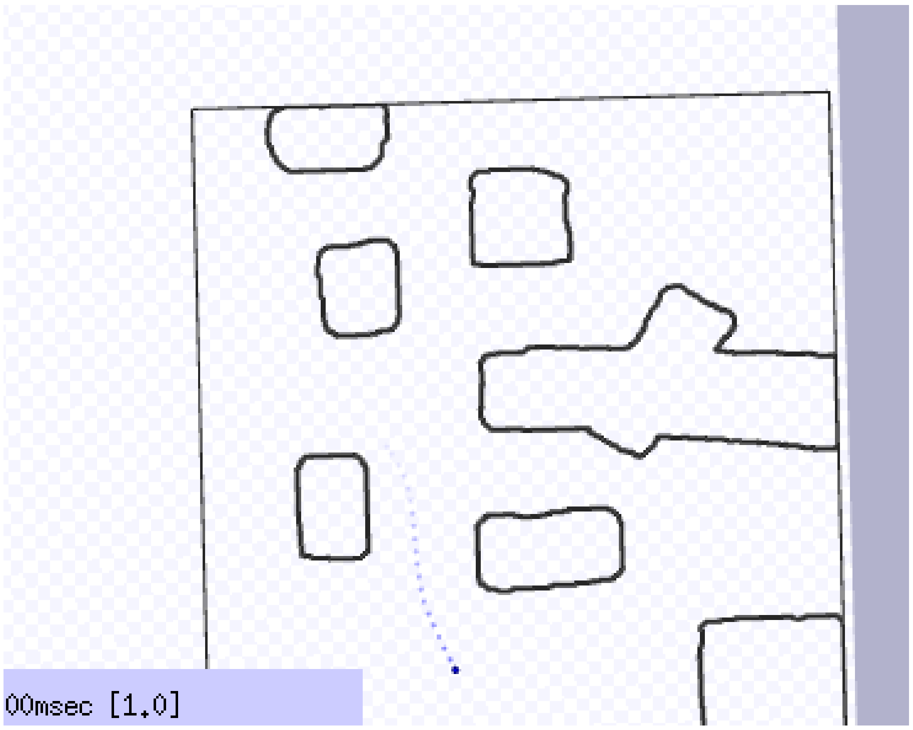

Figure 1: The robot avoiding obstacles while reaching the goal.

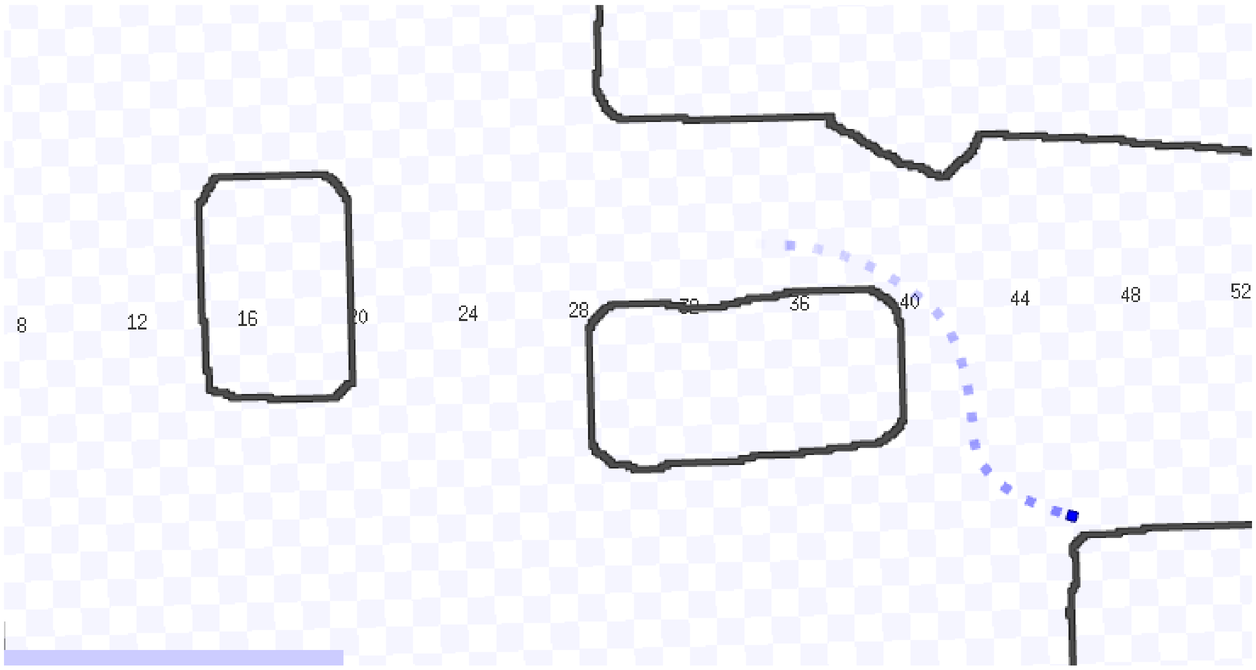

Figure 2: Robot avoiding obstacles while heading to goal that is far in the positive x (right) direction.

Conclusion:

The program was far from perfect and tended to get close or accidentally hit some objects. These miscalculations were rare but often happened when approaching at awkward angles. However, there were several times where the robot was unable to reach its goal due to the nature of the obstacles. I believe that this is due to the heading as there were times where the robot would rotate back and forth but fail to move forward. Through the project I gained a deeper understanding of potential fields and how the attractive and repulsive forces work together to create an optimal system that has an easily avoidable flaw.