Matthew Lai

05/17/2015

CS 521 Robotics

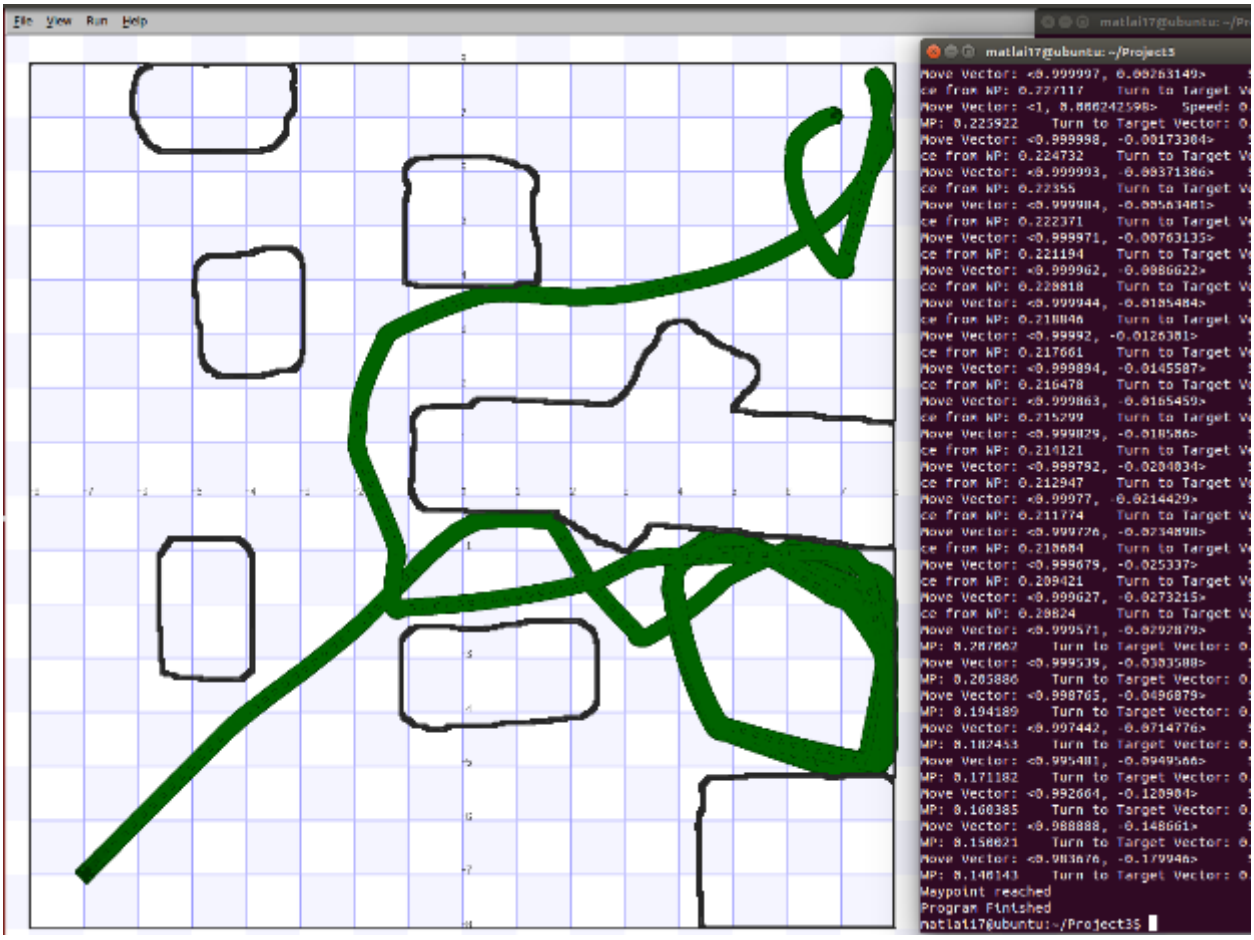
Project 3

My implementation of the third Player-Stage programming assignment essentially uses potential fields. It calculates the vector towards the goal point and uses this as the starting unit “move” vector. It then scans the ranger data and, for each discrete object that it detects, it calculates some tangential and repulsing fields, whose vectors are determined by its proximity to the robot and the direction the robot is facing, and adds the resultant vectors to the move vector. In addition to the tangential and repulsing fields, an attractive field acts on the robot as it approaches the goal point. Once the move vector is determined and normalized, it is then used to determine the speed and turn of the robot, each of these two are calculated by a combination of the move vector, the robot’s heading vector, and the distance to the goal point.

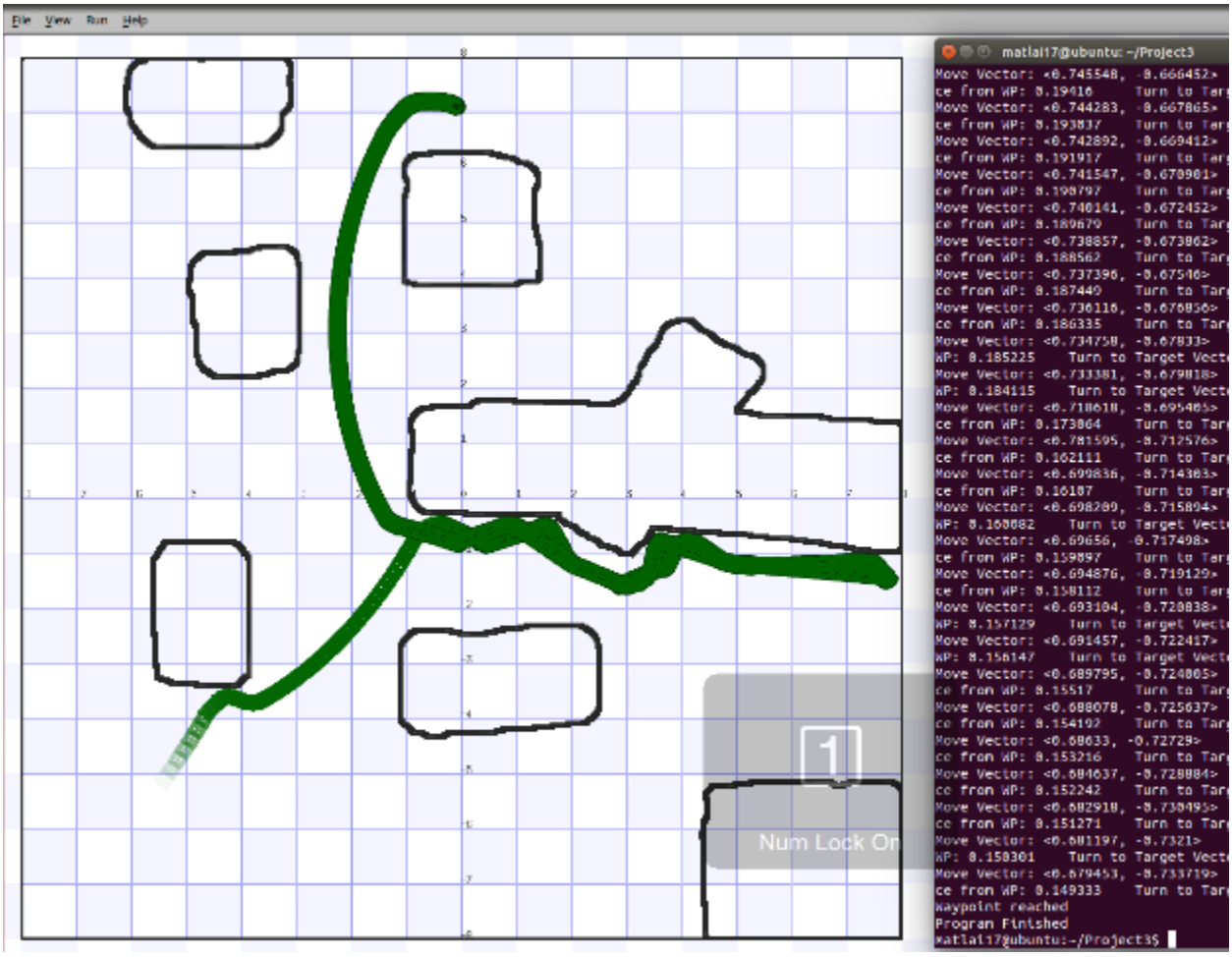
A potential problem occurs in the way the tangential fields attempt to force the robot to move in one direction along an obstacle in the hopes of getting around the obstacle. Because of the way it is implemented, most of the corners in a stage are especially dangerous for the robot as the robot will approach the corner and, as the robot’s closest distance to an object – and also therefore the tangential vector – is calculated, the tangential vector will oscillate between left and right walls of the corner, forcing the robot to move further and further into the corner. This was solved by introducing an escape protocol that will force the robot to randomly choose a direction to rotate 135^{o} in an attempt to escape the corner. The robot will then continue on attempting to reach the goal point.

To run the program, simply call the proj3 executable with two numbers as the executable parameters, i.e. `./proj3 x y` where x and y are the coordinates to the goal waypoint you wish the robot to traverse to.

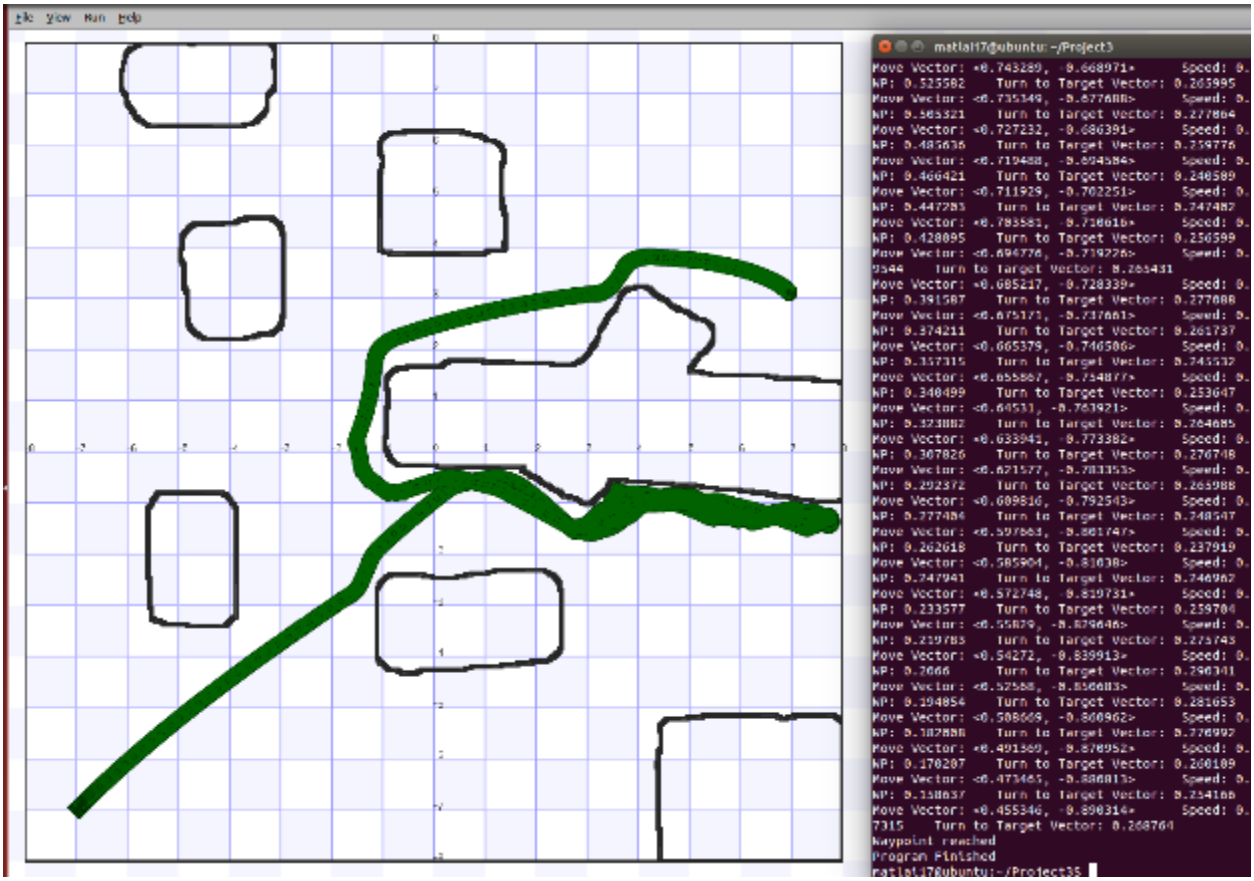
Traveling from the default point to goal point (7, 7). Because the program does not include any planning, it circled around in the bottom left corner until, by chance, it hit the corner escape protocol and was able to break free of the cycle. Near the end, the tangential fields on the walls extended too far out and caused the robot to hug the wall instead of moving straight towards the goal. The robot eventually broke free of the vector field and was able to arrive at the goal point. Many of these deficiencies in this first run were later fixed.



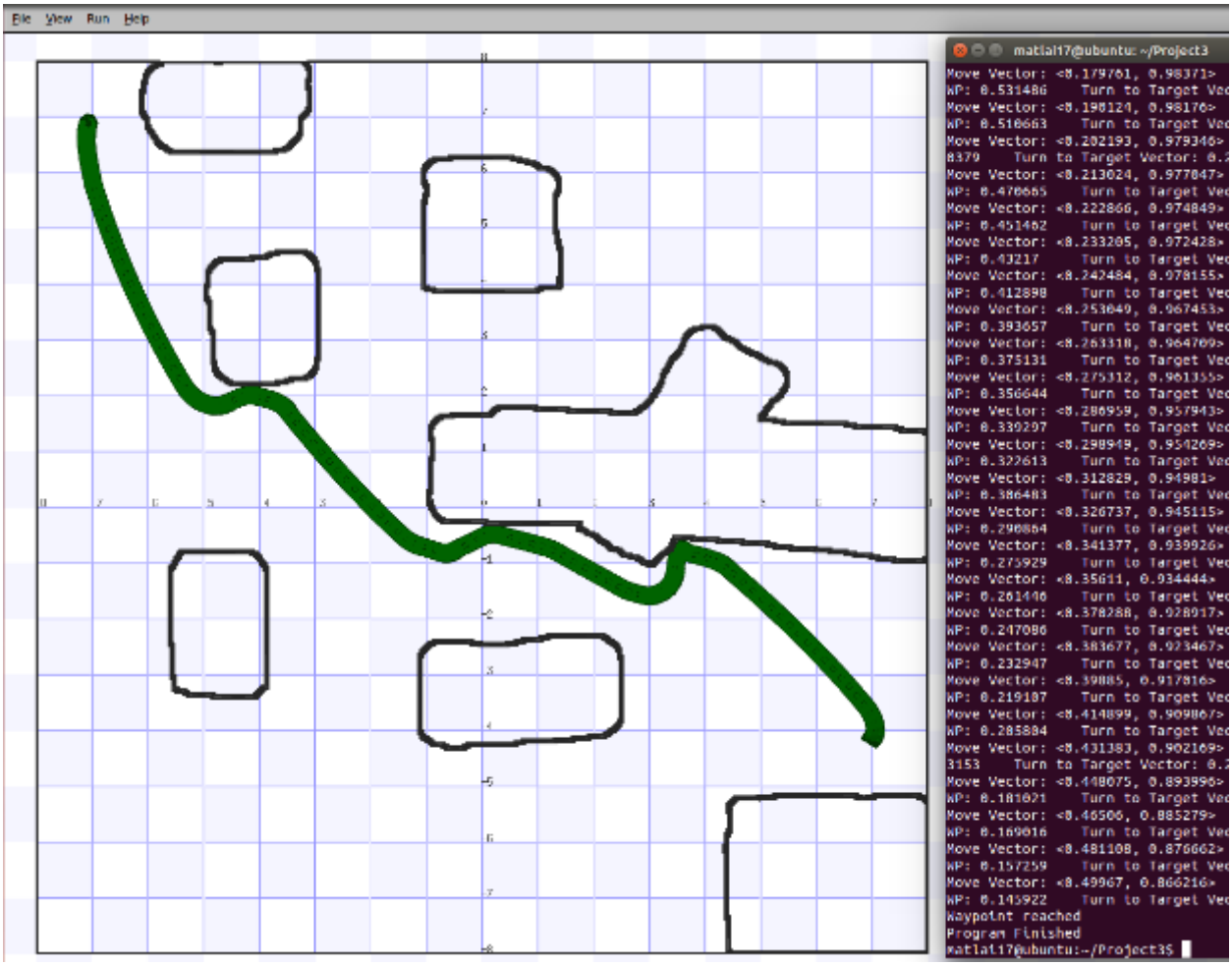
The second run started again at the default location with a goal point of (0, 7). This time, the robot also took a detour to the bottom left corner but its stay in the area was shorter. It quickly turned around and found its way around the large obstacle and made its way to the goal point.



In this test run, the robot moved from the default position to the waypoint (7, 5). I was able to increase the responsiveness of the robot's turning capabilities for this test. I noticed in the previous tests that the robot was turning sluggishly and was able to find the cause. As with all of the other goal waypoints that reside in the right half of the map, the robot traversed the lower right quadrant before making its way to the goal point in good time.



For this fourth test, I moved the robot to a starting point of (7, -4) and set the goal waypoint to (-7, 7). The robot's responsiveness is much improved from the first two tests. The potential fields, vector addition implementation appears to be working out very well for this programming assignment.



For this fifth and final test (third test with the final version of the program) I once again started the robot at coordinate (7, -4) and set the goal waypoint to (2, 7). The robot was very responsive in reaching the goal waypoint and did not penetrate any walls. The wall collision could be slightly reworked to perhaps increase the robustness of the robot obstacle avoidance but I am already quite pleased with how this program has turned out.