Brittany McGarr
CPE 470
Fall 2016

Programming Assignment #2:
Mobile Robot Path Planning Through Artificial Potential Field

Introduction

Artificial potential fields are a set of methods used in robots for determining where a target is and how to plan a path to the target. Much like the potential energy in physical sciences, the method uses a series of calculations to determine an attractive force, drawing the robot toward the target. A lower attractive force means the robot is approaching the target, and a higher potential is avoided as the robot desires to meet the target by progressing to lower potentials.

In addition to attractive forces, potential fields can also account for repulsive forces. These repulsive forces determine obstacles and more heavily weight those results so that the robot will avoid these higher potentials. In this manner, the robot can sum the two to determine a potential field wherein high values indicate an inappropriate direction or obstacle, and lower potential weights indicate a more optimal path toward the target.

This method works so well, it is used in many real world scenarios and even has some real-time applications as the robot can work quickly to derive its next heading and velocity. There are some potential pitfalls including circumstances where irregular shapes (such as a horseshoe with the dip closest to the target, impeding the robot) can create false results and “trap” the robot in its potential field, unable to decide how to proceed. However, the practicality of the method is instructive in learning static and moving target path planning.

For this assignment, the students were instructed to implement an artificial potential field simulation using a language of their choice and output the results of a linear and sine-wave path. The virtual target would move along the instructed path and the simulated robot would update its own path based on the potential calculations.

The Simulator

As with the first programming assignment, I have chosen to use Python for its easy to use NumPy library and PyPlot graphing tools. The program is divided into two parts: robotData.py and main.py. The main.py program is a simple set of data to test using the class and simulation method created in robotData.py. Three sets of data are generated: a linear path, a circular path, and a sine-wave. The sets are run through the simulator without noise first and then with noise. The results are displayed for the user for inspection and saved to jpeg files.

To run the preset data, the user should enter “python main.py” in Linux and Unix systems in the directory the project is located. The user can also load the robotData.py class definition into their Python interpreter and run the simulator with more iterations or with specific paths as they choose using the interface.

Results

The first set of data is run through without noise. A linear path and a sine-wave are indicated in the instructions, and a circular path has been added for additional data.

a) Linear and Sine Path Results:

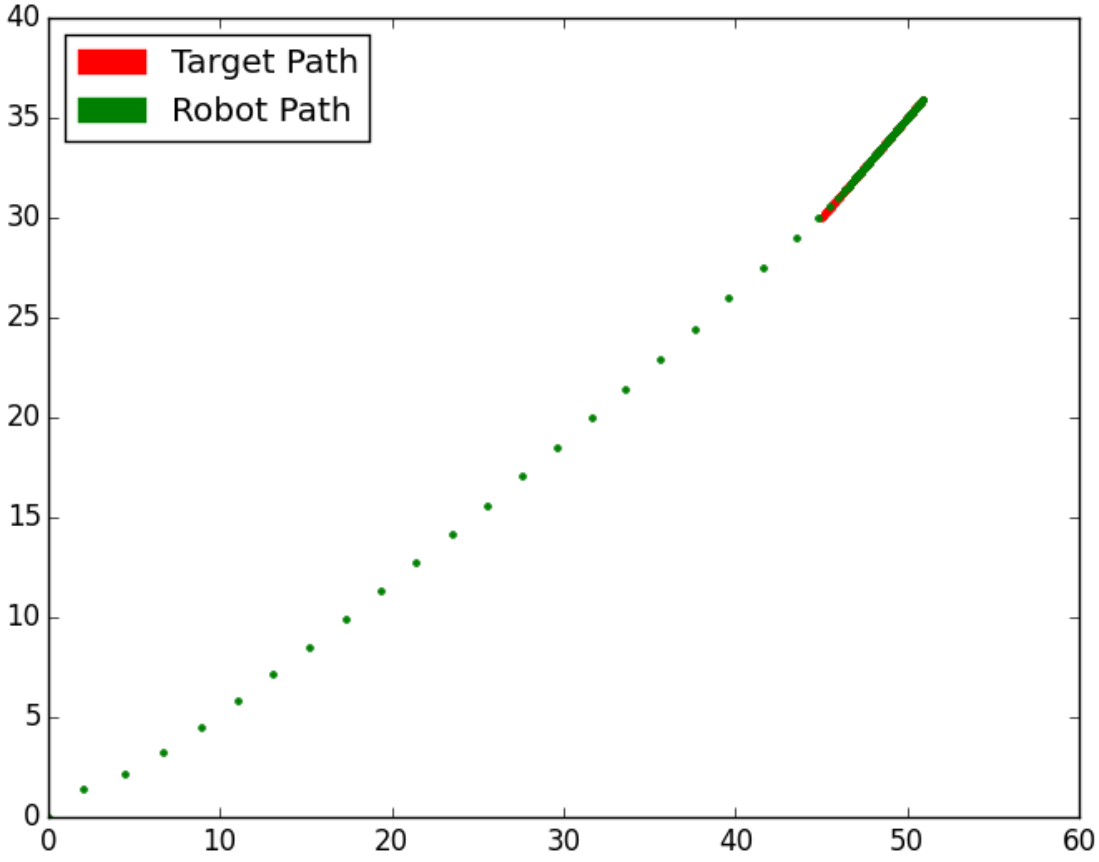

Figure 1: I) The robot and target positions graph. The target(red) follows a linear path starting a position (45, 30). The robot (green) follows rapidly at first (starting from the origin) and settles as it approaches the target.

The first run of the simulation shows the target moving toward the upper right corner of the graph at a rate of 1.2 maximum velocity. The simulation was run for 100 iterations, and the maximum velocity of the robot was capped at 50 units per 20 time steps. This is evident in that the first 21 iterations of the robot appear to go the maximum velocity with wide spaces between, but around the 22 time step, the robot starts to slow, approaching its target and leaving smaller spaces as it adjusts its speed. Eventually, the robot falls into a small distance just behind the target.

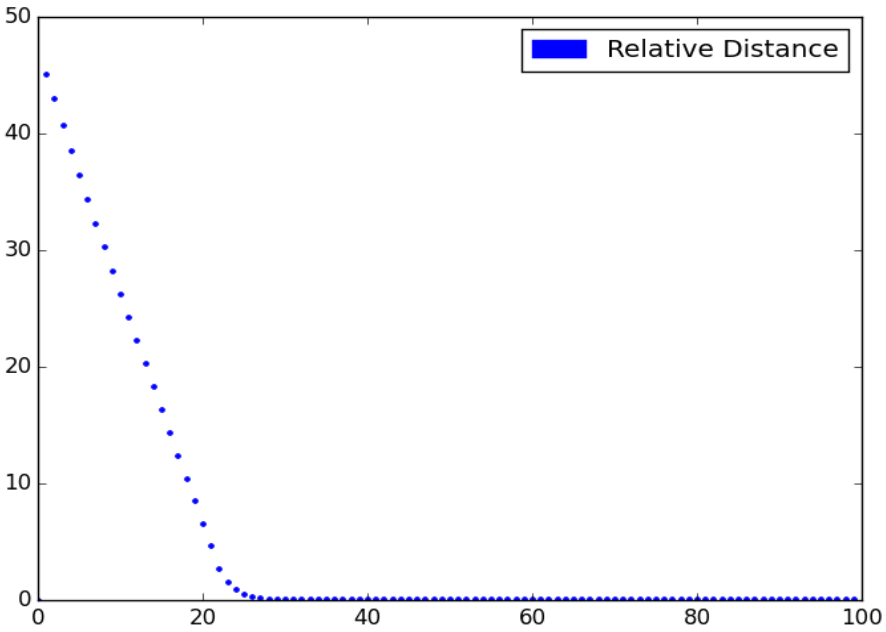

Figure 2: ii) The distance error of the linear path between the target and the robot. The distance between the robot and target started far apart (the robot starting at the origin, and the target starting at (45, 30) and move closer, resulting in an approach to zero distance.

The relative distance graph (Figure 2) shows the distance between the robot and the target over time. This simulation was run for 100 iterations, and the robot approached the target by the 22nd time step. Of particular note is the approach of the distance to zero. Because the simulation did not account for the body of the robot, the robot was able to overlap (but not surpass) the target. In the real world, this form would have to be accounted for, and a safe distance reinforcement would be added to the calculations to avoid running over the target. For the simulation, however, this overlap is fine.

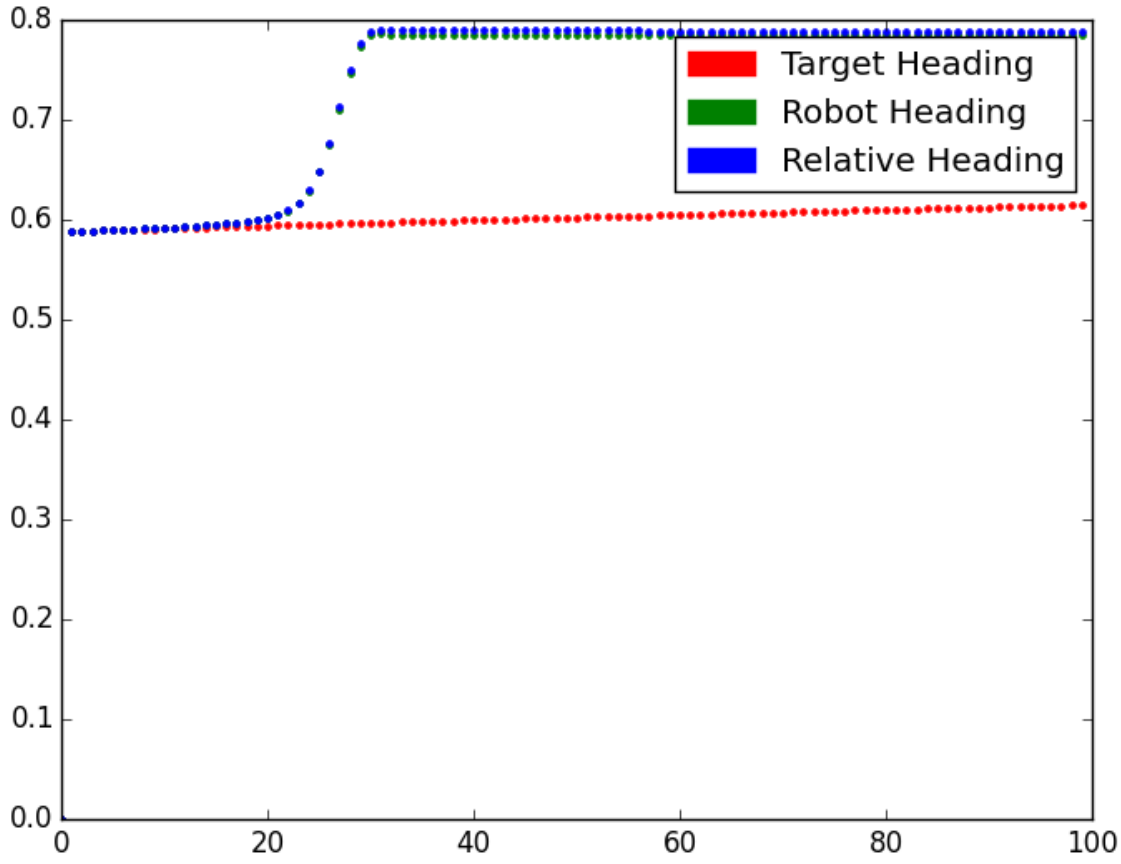

Figure 3: iii) The heading graph of the target, robot, and their relative heading in radians over time. Because this was a linear path, once a direction was determined and the robot caught up to the target, there was little change after.

Figure 3 details the heading of the robot and the target over time through the simulation. Note that the heading of the target seldom changes as it is staying on a strict, linear path. The relative headings and the robot, however, change as the robot tries to catch up and orient itself to follow the linear path. Once the robot caught up after the 20th time step, the heading because stable and did not change.

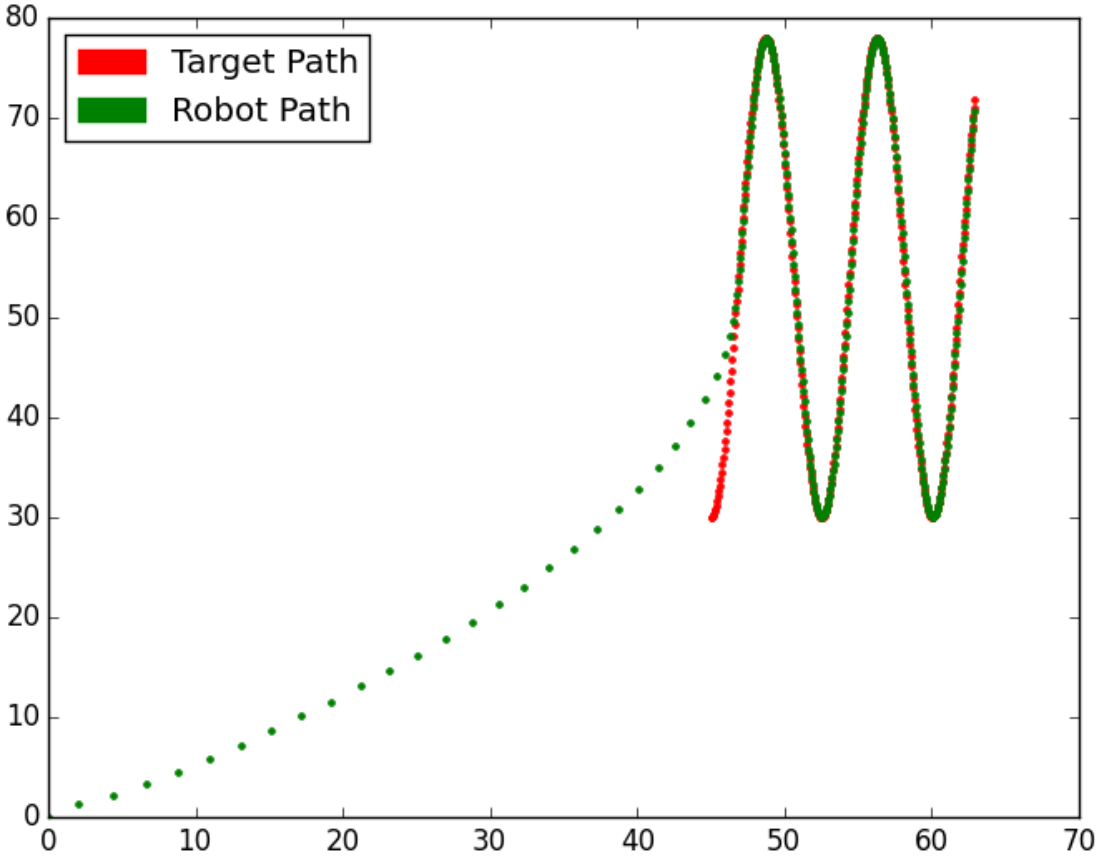

Figure 4: i) The sine wave path was run through 300 iterations to more emphasize the curves and give a broad sample. Again, the target was set farther away from the robot with the robot adjusting its speed and path to intercept the target.

As with the linear path, the robot was set at the origin and set to chase the distant target. This target moved in a sine-wave. Like before, the robot had to move at its maximum speed in order to close the distance between the robot and target, and when it did approach the target, their paths were overlapping. Of particular note in this run of the simulation, the sine path put more distance along the Y-axis of the target, and the robot moved more dramatically along the Y-axis toward the 20th time step.

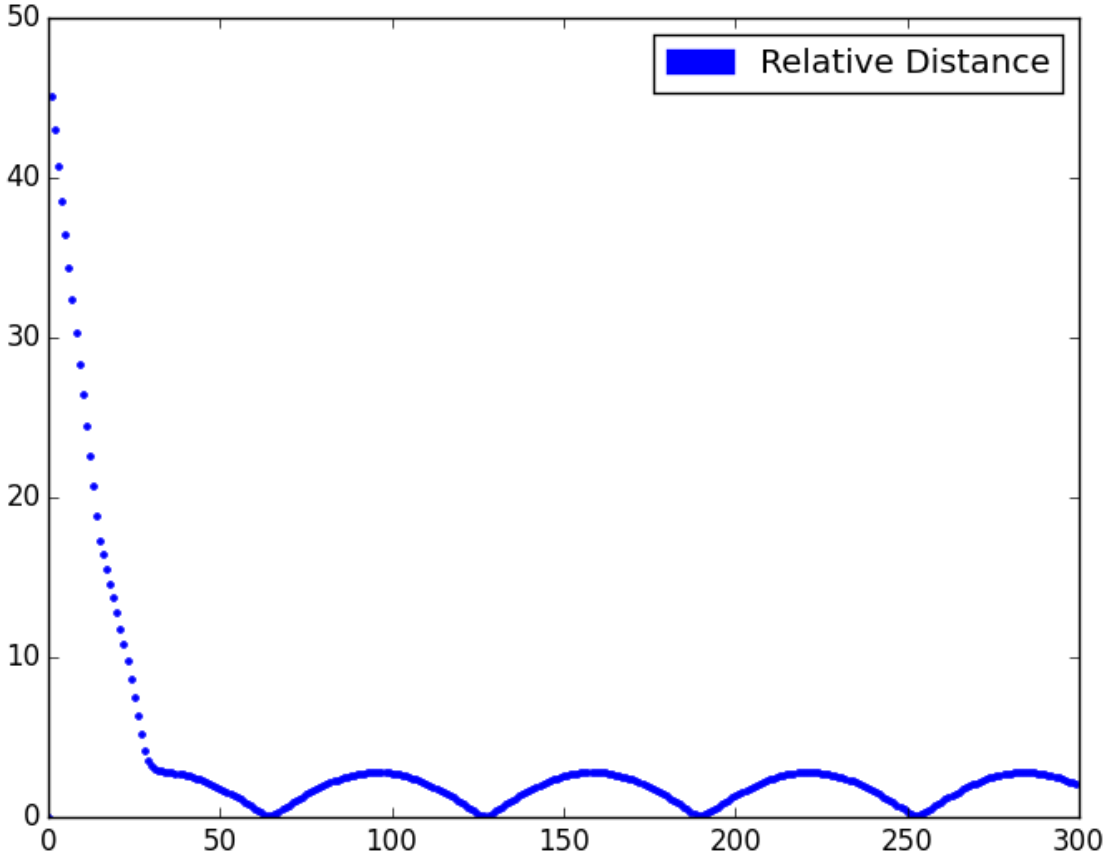

Figure 5: ii) The relative distance between the vehicles in the sine-wave path. The appearance of four minor “humps” correlate with the four changes in direction along the Y-axis in Figure 4.

Figure 5 shows a much different result than the linear path of Figure 2. In this path, the robot would suddenly be too far from the target as the target sharply changed direction. Like the real world, when cars drift around corners, the speed has to be adjusted to return to the proper trajectory. In Figure 4, the green dots of the robot's path were thicker around the corners, indicating more time steps were used to get the next position, and so, the robot was going slower to find the target around the corners. The dips in Figure 5 indicate when the robot accelerated and caught up to the target again, closing that distance.

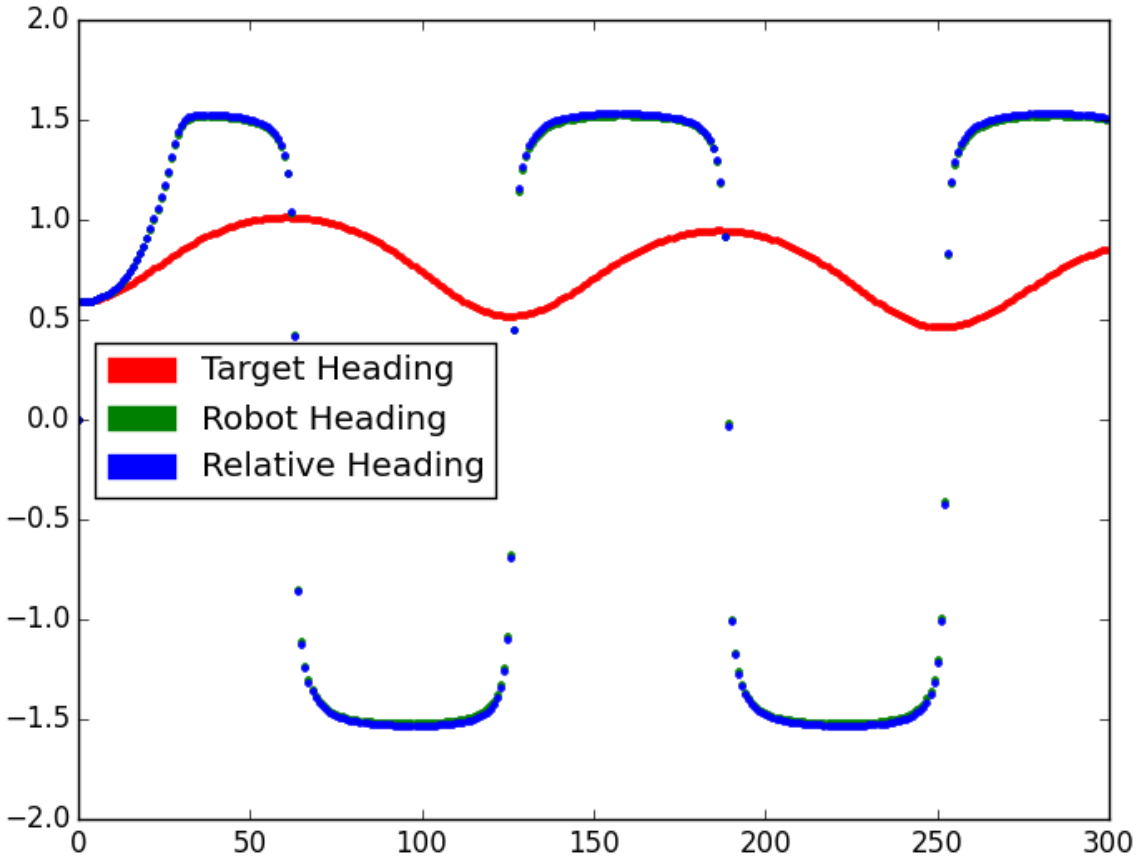

Figure 6: iii) The headings of the target, robot, and their relative graphed. The robot's heading followed the relative closely, oscillating between 1.5 and -1.5 radians through the 300 iterations.

Figure 6 shows the sinusoid pattern of the headings. The robot's heading started at 0.5 radians and quickly caught up to the target's orientation, oscillating between 1.5 and -1.5 radians. Long periods of the extremes indicated the corners detailed above when the trajectory of the target sharply changed and the robot was forced to adjust and keep pace.

b) Linear and Sine Path With Noise Results:

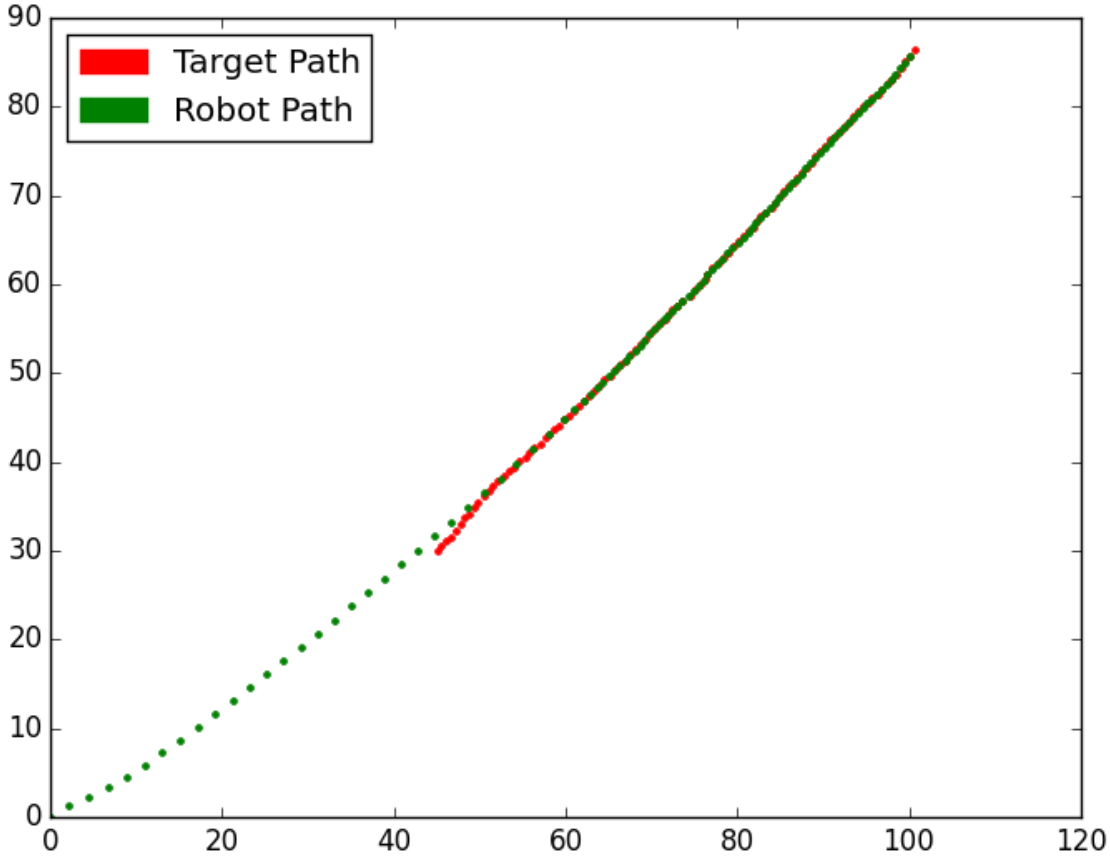

Figure 7: i) The linear path of the target is sensed with noise, resulting in a jittery, but still determinable, path.

The next part of the assignment instructs the students to add random, Gaussian noise to the position of the target. In the real world, sensors can have estimation errors and can cause deviations from the true path of the target. Here, the simulator added noise to the linear path and the robot then determined where the actual position was. As seen in Figure 7, the robot was able to use the attractive potential field to determine the next most likely position of the actual target and follow in a linear trajectory. The time to reach the target and slow down seemed to take a couple of iterations longer, and variations to the path were recorded even after catching up. However, the adjustments were negligible and could be further refined by adding a Kalman Filter to account for the reading noise.

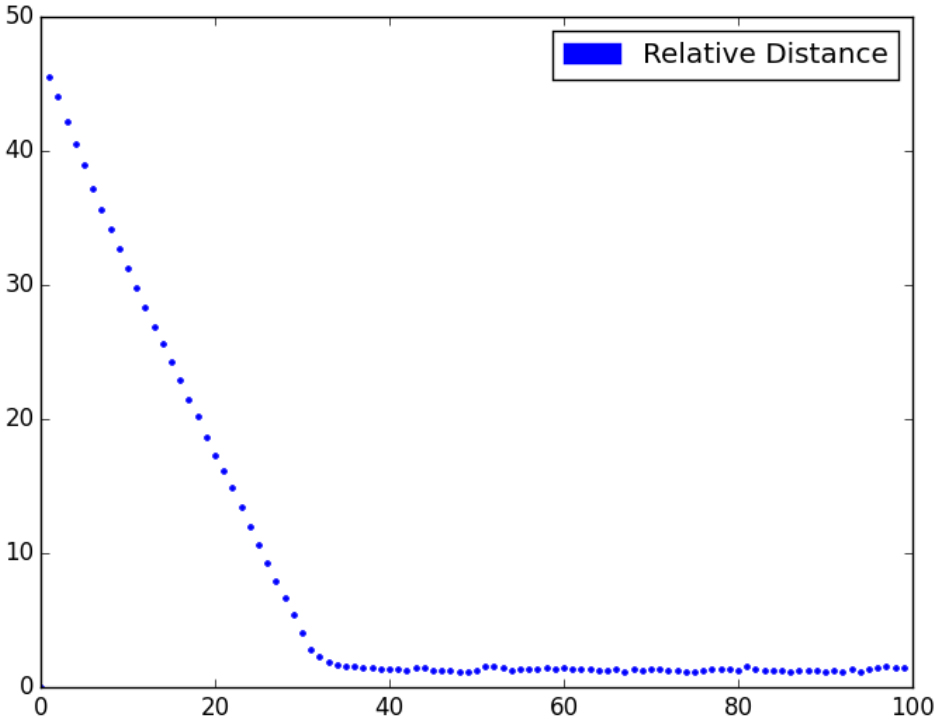

Figure 8: ii) The relative distance of the linear path with noise added. Again, the robot closes the distance, but with the noise added, random changes occur.

The relative distance with the noise follows a similar track to the linear path without the noise, but small spikes in distance indicate instances when the robot had to speed up to catch its target's seemingly sudden change in movement. The overall path, however, stays close to 1.0, the mean of the measurements.

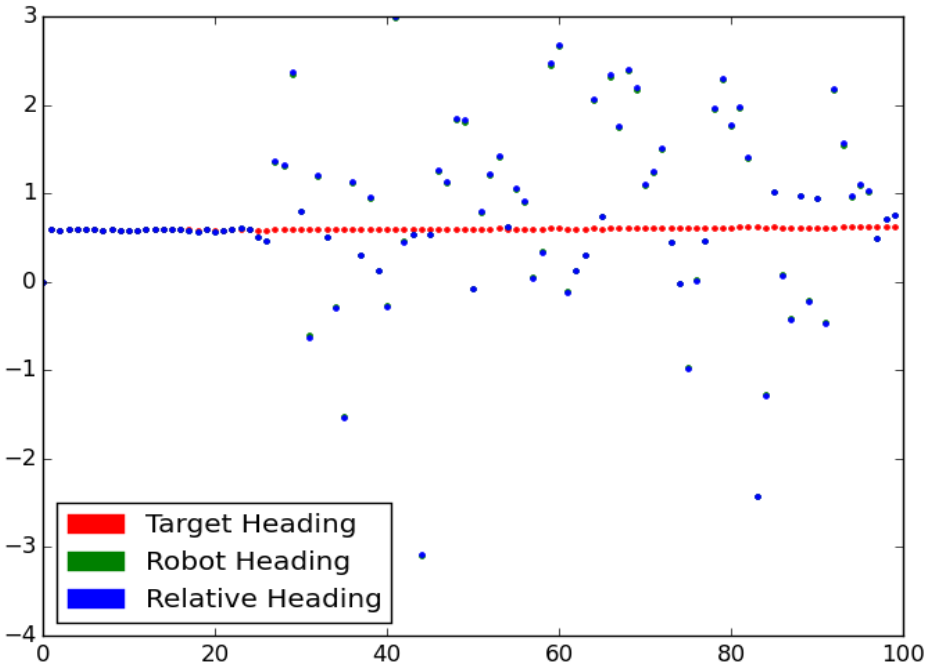

Figure 9: iii) The heading of the linear path with noise added produces a much more chaotic graph.

The random additive noise did produce a significant change to the heading plots. Instead of plateauing when the robot approached the target, the heading became sporadic. This is because the position of the target appeared to wildly jump around, and as a result, the potential field weighted the direction indicated to attract the robot to the new position. While this is seemingly incorrect and dangerous, it is actually an indication that the relative heading tried to adjust for anomalies to orient the robot toward the target.

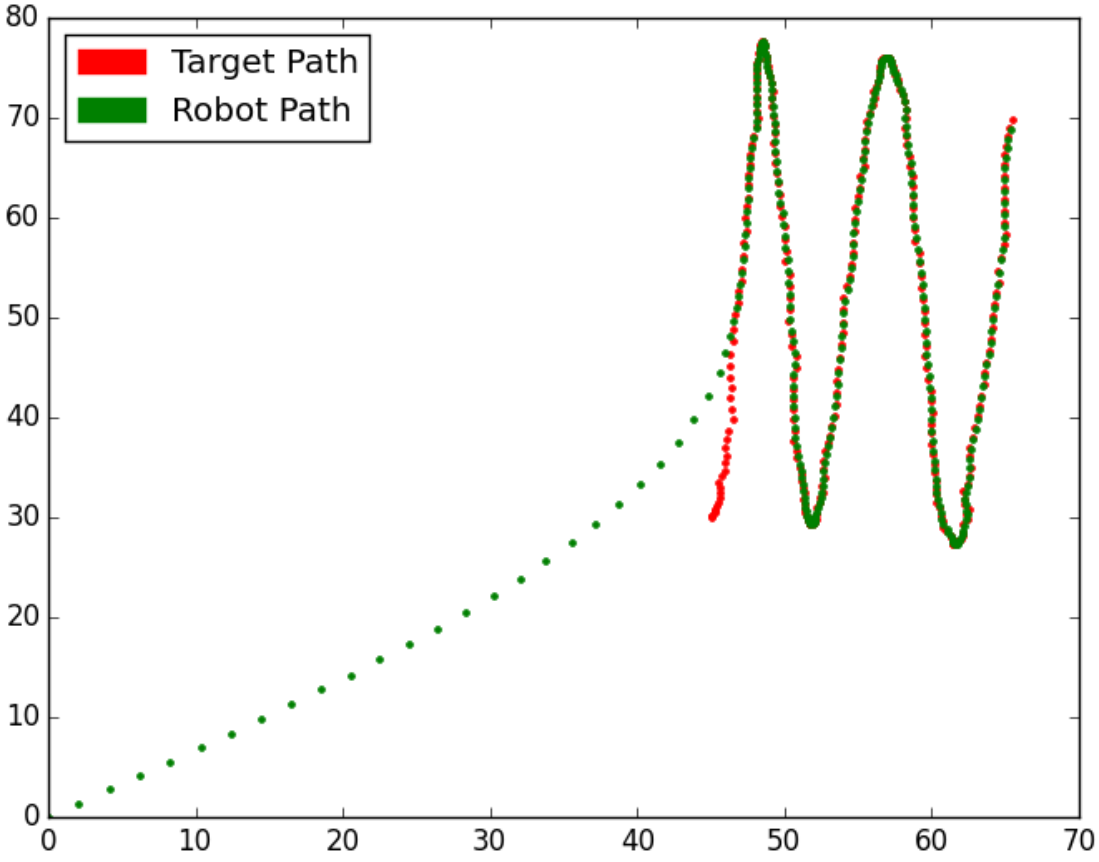

Figure 10: i) The sine path with random, additive noise, and the pursuing robot.

As with the linear path, the additive noise produced a jerky set of data to throw off the robot. This time, the robot managed to stay within a similar speed with small variances when the target appeared to move. Again, the potential field helped indicate where the target was most likely to be and helped the robot to intersect the target through the noise.

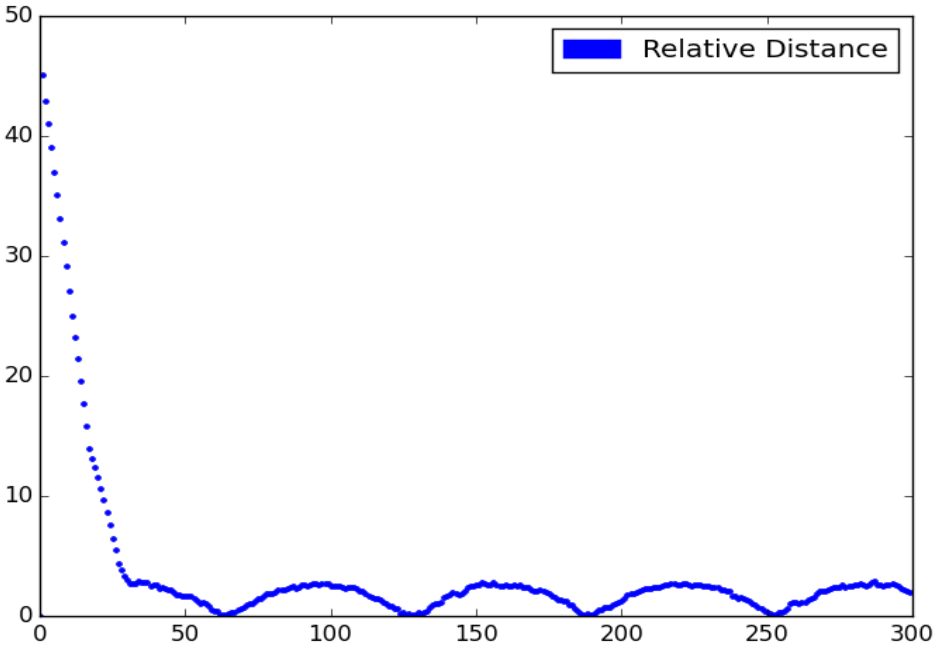

Figure 11: ii) The random noise for the sine-wave path created small spikes in the distance calculations, but overall, the robot stayed close to the target.

Again, the random noise caused small spikes in the distance graph, but the robot was able to intercept and adjust. This time, the “dips” in the graph were less pronounced due to the errors in measurement as the robot struggled to catch up when the target appeared to change rapidly. However, the path is very similar to its preceding with distance widening in the “corners” and lowering in straightaways.

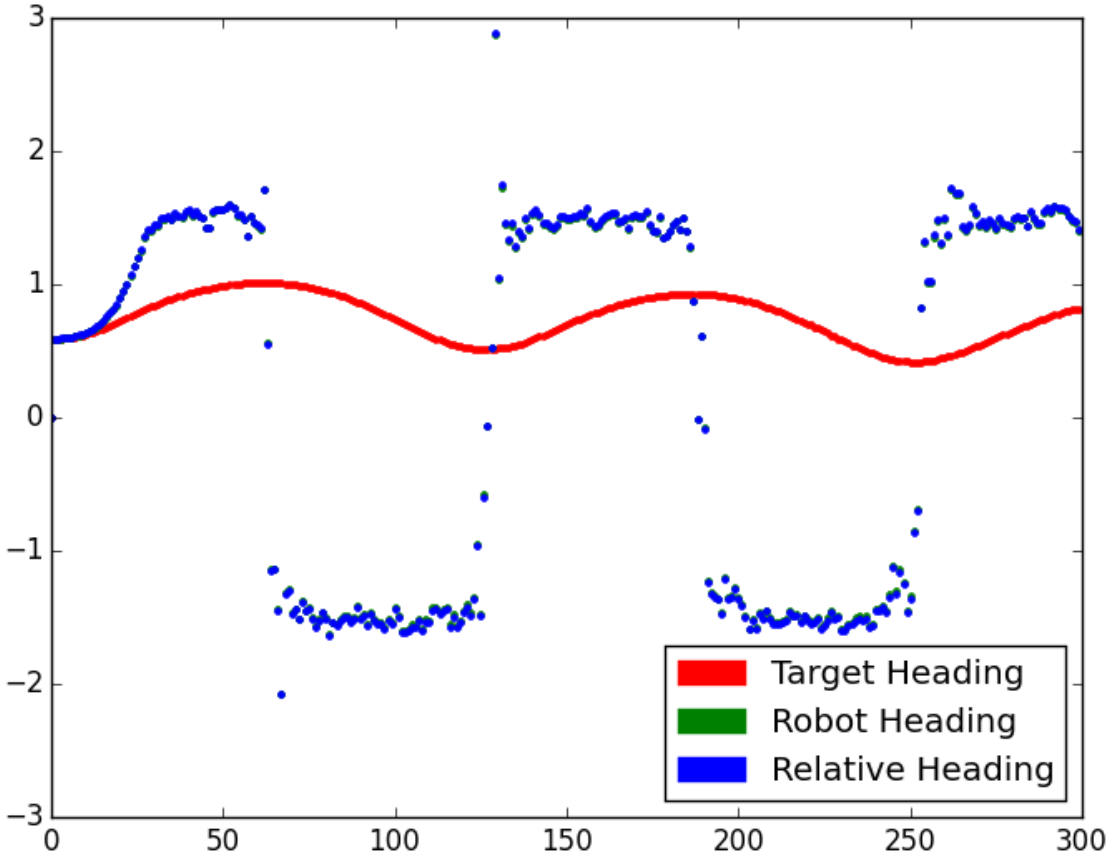

Figure 12: iii) The headings graph when noise is added to the sine path. Note the scattering in the apexes.

The new shape of the sine path did have some interesting incidentals when it came to heading. In the areas where sharp turns had to be made, the graph still plateaus with a period of adjustment, but the noise adds to the confusion, making the heading spike around. The opposite happens in the straightaways where fewer measurements incidentally led to less variance. Here, the potential weighted the heading change very little, and the robot was fast to pursue the target.

Conclusion

While no perfect method exists for target tracking, Artificial Potential Fields provide a robust method for path planning. Calculations to uncover the velocity and heading relative to the target's perceived parameters are able to make rapid adjustments even on moving paths. The introduction of random, Gaussian noise even has only narrow margins of error, indicating refinement with a minor introduction of a Kalman Filter can give practical results.

Further Data

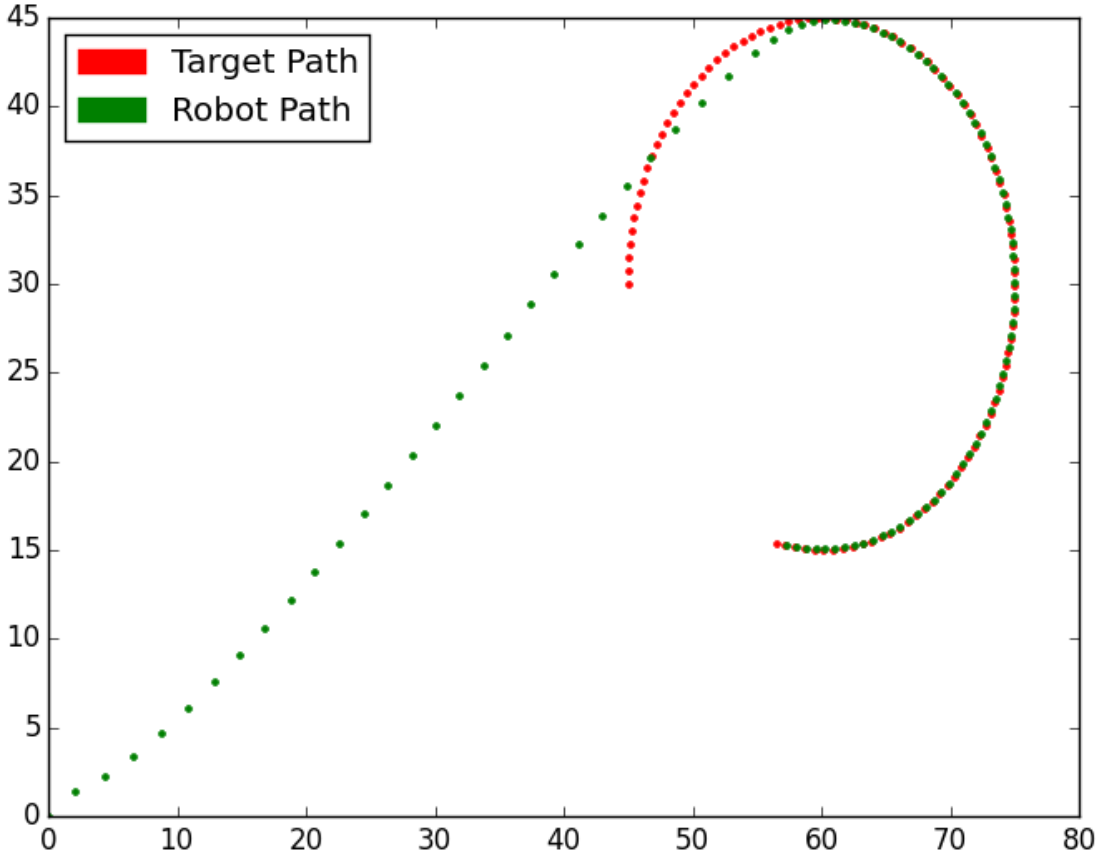

Figure 13: A circular target path and the pursuing robot.

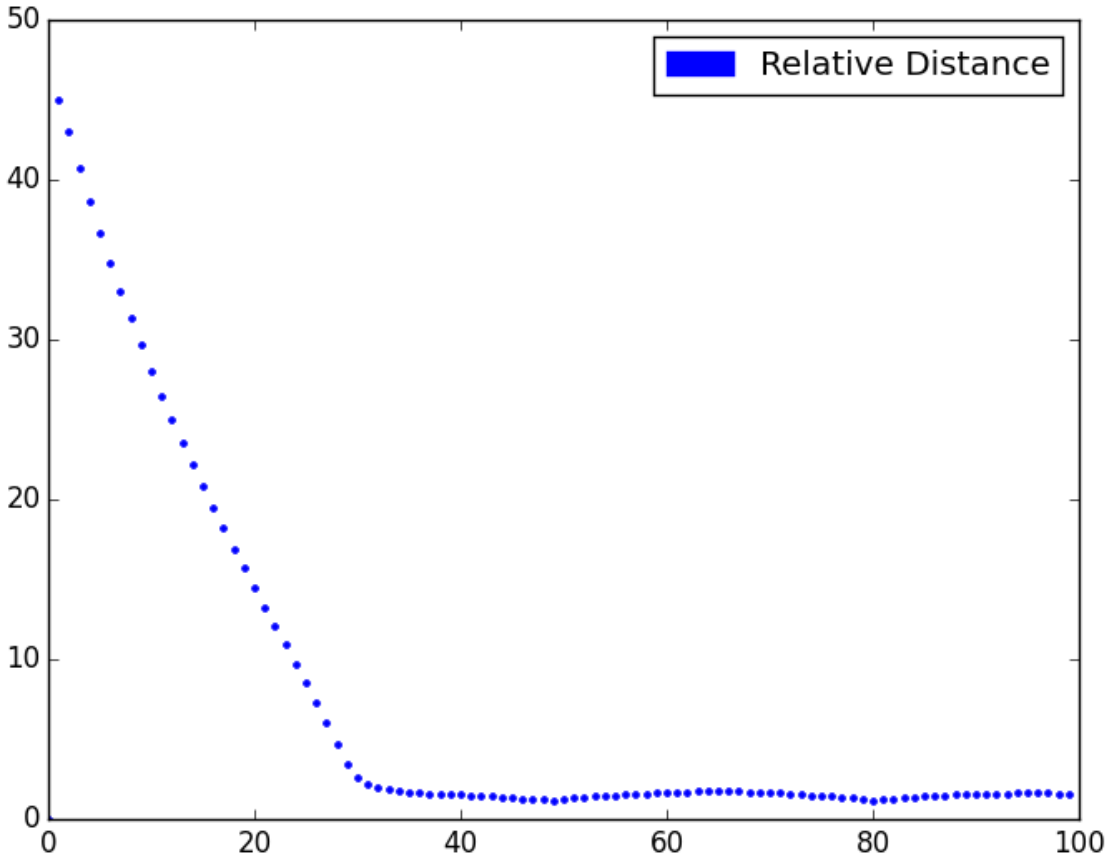

Figure 14: The circular path distance between the robot and the target. Like the sine-wave, the areas of curves are represented by gradual “humps.”

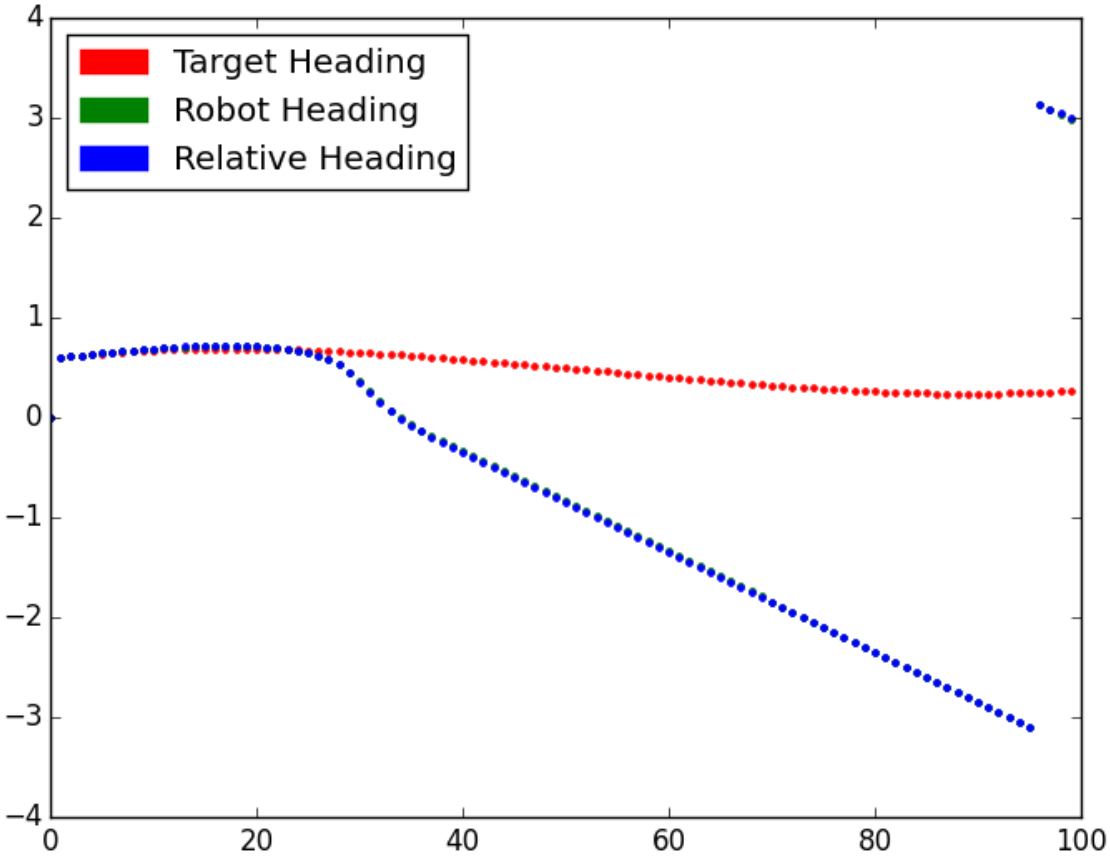

Figure 15: The headings given by the circular path. Note that the last four measurements are the zero-point of the circle, when the path has reached the starting point of the rotation angle.