CSE 460 Mobile Robotics - Lab 3

Amanda Baran[1]

***Abstract*— Lab 3 report for CSE 460 Mobile Robotic with Professor David Saldaña.**

I. Part 1: Freenove Robot with Ultrasound

The python programs I created to make the robot perform the desired tasks can be found here: https://github.com/amandabaran/robotics/blob/main/lab3/lab3.py The video recordings of the robot reaching the desired distance using different values of k can be found here: https://drive.google.com/drive/folders/1-ogbb8g2OUYtEaQSY5zNsXsCLt1K-EUE?usp=share_link

II. Part 1 Questions

A. Single K Value

Start Distance(cm)	K Value	Time to 50cm(s)
60	-100	0.760
70	-100	1.260
80	-100	2.259
90	-100	1.890
100	-100	1.793

The table above shows the elapsed time I recorded for the robot to reach the desired distance of 50cm from the wall at varying start distances. When starting at 60cm, the robot was able to move forward and then come to a stop without ever needing to move backward and readjust. On the contrary, when the robot started from further back there was more variance in how many times the robot had to go forwards and backwards to adjust its distance.

B. Initial Distance of 1m

The videos of the robot starting at 1m and reaching the desired distance of 50cm from the wall using varying k values can be found by clicking here. The table below shows the elapsed time for the three different values of K that were tried.

Start Distance(m)	K Value	Time to 50cm(s)
1	-50	0.960
1	-250	3.132
1	-1000	ERR

As the table reveals, using a smaller k value resulted in the best time to reach the desired distance. This is because using faster more aggressive k values cause the robot to move too quickly with too much adjustment occurring for slight error, which causes the back and forth continuous motion that can be observed in the video where k = 1000. This video was clipped because the robot did not stop until Ctrl-C was entered. It was observed that the sensor kept reading values surrounding 50, but always a bit too far or too close, causing this behavior.

C. Does the robot work as expected from the theory?

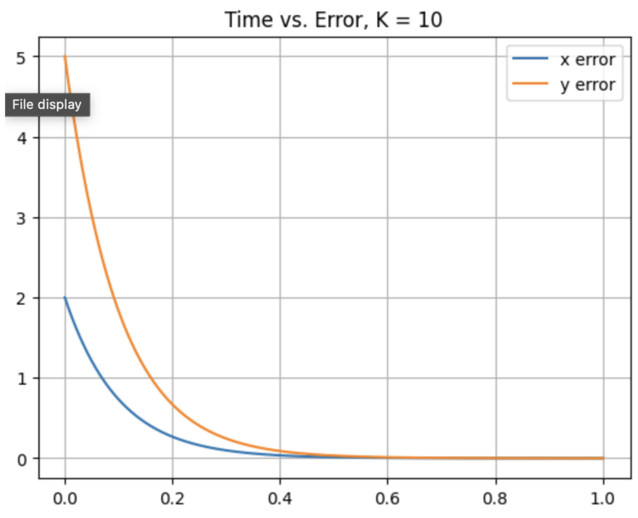
Yes, the robot works as expected from the theory, with different environment factors that come into play. As expected, a large value of k such as 1000 causes a large adjustment to be made for a small error. Furthermore, smaller values of k such as 100 cause a smaller adjustment to be made with each read of the sensor, which causes the robot to approach the wall much smoother without "over-shooting" and getting too close to the wall. However, what surprised me was the non-uniform increase in time it took to reach 50cm when using a single-k value at starting distances incrementally placed further from the wall, as shown in the first table. This is partially due to inconsistency in the sensor and when the timer actually stops due to detecting exactly 50cm. However, there were also cases where the robot did overshoot and need to back up from the wall such as at 80cm, where other runs the robot never had to readjust and move backwards, such as at 60cm.

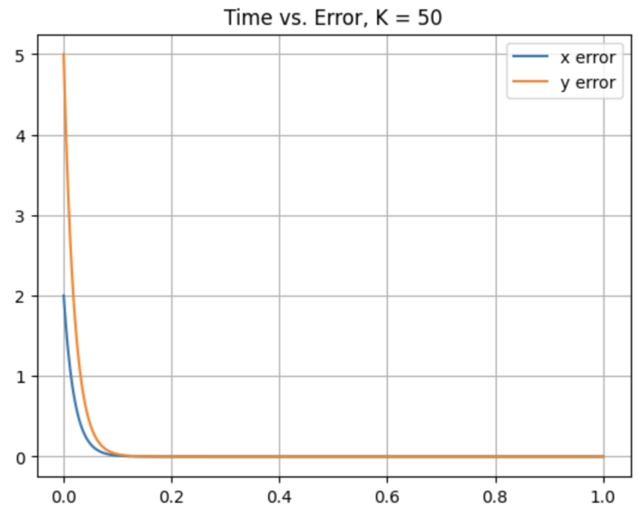
III. Part 2 Simulator

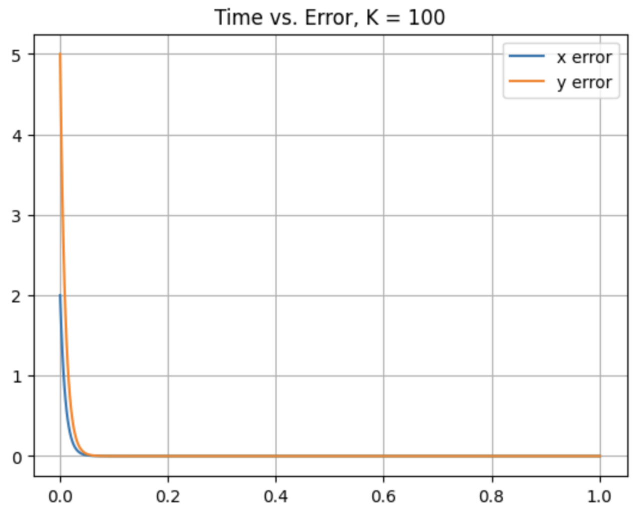
The jupyter notebook I created to make the simulator and plot the time vs. error curves can be found here: https://github.com/amandabaran/robotics/blob/main/lab3/lab3-part2.ipynb.

A. Plots

Below are 3 of the 4 plots I created of time vs. error in the x and y coordinates for k values 10, 50, and 100.







B. *Main Difference Between Robot & Simulator*

The main difference between the real robot and the simulator is that the simulator uses a "perfect" sensor to obtain the location of the robot such that y(t) = p(t). However, the real robot does not have this luxury of an exactly perfect sensor. In addition, the real robot has other factors such as the environment its in and any normal wear and tear on the robot that can cause fluctuations.

C. *Improving the Robot*

Clearly, the simulator did well with higher k values as it was able to perfectly correct itself and reach 0 error very quickly. One way to improve the robot in order to minimize error and adjust to the correct position more quickly, we could minimize the think time of the robot. More frequently occuring measurements taken can provide a more precision location of the robot, which can help it correct course faster. Also, using as controlled of an environment as possible, testing out many different k values to pinpoint the one that results in the fastest time to the desired position would help.

References

[1] Amanda Baran, Github, https://github.com/amandabaran/robotics/tree/main/lab3
[2] Amanda Baran, Google Drive, https://drive.google.com/drive/folders/1-ogbb8g2OUYtEaQSY5zNsXsCLt1K-EUE?usp=share_link