ME 5751 Module 2: Brush Fire Algorithm

Cal Poly Pomona

Pomona, CA

Jonathan Jenkins

[jjenkins@cpp.edu](mailto:jjenkins@cpp.edu)

***Abstract— describe what you have achieved This project looks at creating a proportional controller for a 2 wheeled “Roomba” style robot to track a point. The methods used to test the controller involve a gui (graphic user interface) created by Dr. Chang which simulates the robot. The mathematics involved were sourced from “Introduction to autonomous mobile robots”, 2nd Ed. by Siegwart, Nourbakhsh, and Scaramuzza. The findings in this project were each k value had a slightly different effect on the robot trajectory as a whole. Kp effects the speed of the robot, Ka effects the angle of attack from the current position to the next, and lastly Kb effects how aggressively the robot oriantes itself to the desired position.***

***Keywords—Proportional Control, Trajectory, Motion Planing, Error, Proportional gain,***

I. INTRODUCTION

Survey on grid-based map representation and grid-based panning methods

Briefly discuss the procedure of Brushfire algorithm, describe how you initialize the queue and how you define distance to obstacles at 4 connection or 8 connection

OBJECTIVE – IMPLEMENT BRUSTFIRE ALGORITHM AND UNDERSTAND HOW POTENTIAL FIELD MAP IS GENERATED

What does autonomous cars, Boston Dynamic robots, and your rumba have in common? They all are robots capable of navigating in 3D environments. Navigation is a major focus for modern robotic. A wide variety of different sensors and algorithms are used to measure, analysis, and control a robot’s path through space.

Proportional control is a type of linear feedback control system in which a correction is applied to the controlled variable, and the size of the correction is proportional to the difference between the desired value and the measured value

In this experiment proportion control is used to control a robot navigation route in a virtual environment. As shown in figure 1, a robot will independently navigate to 4 different waypoints. The robot will use a custom proportional controller to influence the linear and angular velocity until it reaches the desired waypoint.

This paper will go into depth on different elements of the experiment including control theory, robot kinematics, programing script, controller improvements and evaluation of the final robot motion.

II. BACKGROUND

Just like how maps are used to assign human navigation maps are also used to assist robotic navigation. In robotics a map is a symbolic representation of selected characteristics of an environment. A wide variety of different maps are used to accumulate statistical evidence about the occupancy of a 3-D environment. As shown in figure 1 a map can be used to represent a areas open space.

Diagram, schematic

Description automatically generated

Figure 1

These volumetric maps are often referred to as grid maps because they are created from a series of pixels or grids. A map’s grid size directly represents the map’s resolution. With a larger pixel density, the map will contain a higher resolution. A general map consists of two types of pixels. A pixel that represents an occupied space often shown in black. And a pixel that represents a free space often represented in white, as shown in figure 2.

Chart

Description automatically generated

Figure Map Space

As mentioned, a map can be used to represent a variety of information. In this project a brushfire and cost map were used. A brushfire map uses the algorithm to approximate distance of empty spaces to occupied spaces. This algorithm produces a map of distance values to the nearest obstacle as shown in figure 3.

A picture containing text, crossword puzzle

Description automatically generated

Figure

The distance assigned to each occupied space is based on the Manhattan distance formula as shown in equation 2.1

|  |  |
| --- | --- |
|  | (2.1) |

Another type of map often used to guide a robot is the costmap or also known as a potential map. These maps assign a risk value to all empty spaces. Spaces at a closer distance to an occupied space will be given a higher risk or cost and spaces that are farther away are assigned a lower risk. An example of a cost map is shown in figure 4.

Table

Description automatically generated with low confidence

Figure

The cost map can determine the “risk” of a cell by a variety of methods. An exponential equation such coulomb’s law, as shown in equation 2.2, or a linear equation as shown in 2.3 can be used.

|  |  |
| --- | --- |
|  | (2.2)  (2.3) |

III CONTROL DESIGN

You must generate the "distance map" (i.e. the cell value is the distance to the nearest obstacle). From the distance map, you can generate the cost map (potential map) with inflation. The way to define potential function is of your choice. Report how you generate the "distance map", and what is your potential function. You can use different potential functions and discuss the pros and cons of each.

As mentioned the type of control implemented in this project is a Proportional control system. To implement our P controller onto our virtual robot we developed custom python code to take the current robot kinematics and derive the error between the current robot position and desired robot position and apply the proportional gain value to correct the robot motion every 0.1 seconds till the robot reached the desired waypoints.

Our code is separated into 6 steps, each done in a specific order to derive the system error and apply the proportional gain. Step 1 is to store the current kinematic data from both the desired goal position and the current robot position. As shown in figure 7. The python script runs a series of functions and stores the new kinematic data in a series of variables.

Text

Description automatically generated

Figure

Step 2 takes the positional data from the robot in the global coordinate system and transforms them to positional data relative to the goal frame. Position transformation is used to allow for easier calculation of rho, alpha, and beta. Using homogeneous transformation equation 4.1- 4.3 we are able to obtain the robot's position and angle relative to the “goal” frame and vice versa. The python script used to perform the transformation is shown in figure 8.

Text

Description automatically generated

Figure

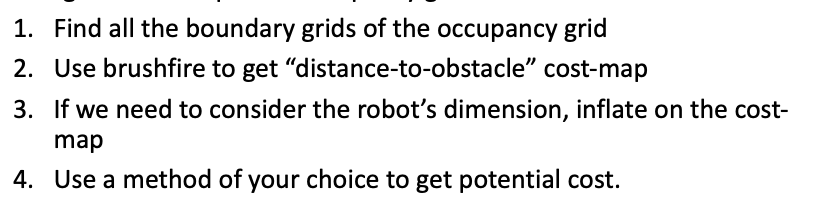
Step 3 of the python program derives the alpha, beta, and rho values from the known positions points. The equation for rho (⍴), Alpha (𝛼), and beta (β) was highlighted in the equations 2.1, 2.2, and 2.3 respectfully. Figure 9 shows the custom python script for step 3.

Text

Description automatically generated

Figure

Step 3 of the python program derives the alpha, beta, and rho values from the known positions points. The equation for rho (⍴), Alpha (𝛼), and beta (β) was highlighted in the equations 2.1, 2.2, and 2.3 respectfully. Figure 9 shows the custom python script for step 3.

A picture containing diagram

Description automatically generated

Text

Description automatically generated

Figure

Step 4 is the regulation of alpha and beta. Both of these units are in the polar coordinate system. And to be used by the proportional gain equation we need to have the values within the + pi and - pi range. A simple function is used to regulate the variables with our desired range, as shown in figure 10.

Graphical user interface, text, application

Description automatically generated

Figure

The final step is used to identify when the robot has successfully reached the desired goal point. As mentioned before each desired goal point specifies a position (x,y) and an orientation (theta) relative to the global frame. During each loop the difference between position and orientation of the robot is calculated. The robot is considered at the desired point when the difference between the goal point’s and the robot’s position and orientation is zero (or close to zero).

Once the robot has reached the desired location the next waypoint location data is loaded and the robot proceeds to the new goal point, running the same proportional control algorithm. Only when all goal points have been reached does the robot stop. The python script shown in figure 12 identifies when the robot reaches the desired goal point.

IV. METHOD

Make your own occupancy grid map( 3 maps with your own narrative of why theses maps are worth testing

After the completed implementation of the P control algorithm in python we then conducted a series of testing to evaluate the effect changing the proportional gain values Kp, Ka, and Kb had on the system. For each test we had the robot move to the exact same goal point positions. As shown in figure 17 the robot begins to travel to the right of the map and then proceeds clockwise around the entire map.

Chart

Description automatically generated with medium confidenceChart

Description automatically generated

In order to determine the effect of the proportional k values we began with a control group and during each trial we modified one k value out of the set of 3. We then describe t

VI. EXPERIMENT RESULTS AND DISCUSSION

Make your own occupancy grid map( 3 maps with your own narrative of why theses maps are worth testing

As mentioned in order to determine the effect of the proportional k values we began with a control group. When choosing the k values we ensured they meet the general k value criteria shown in equation 5.1. Our control k values were:

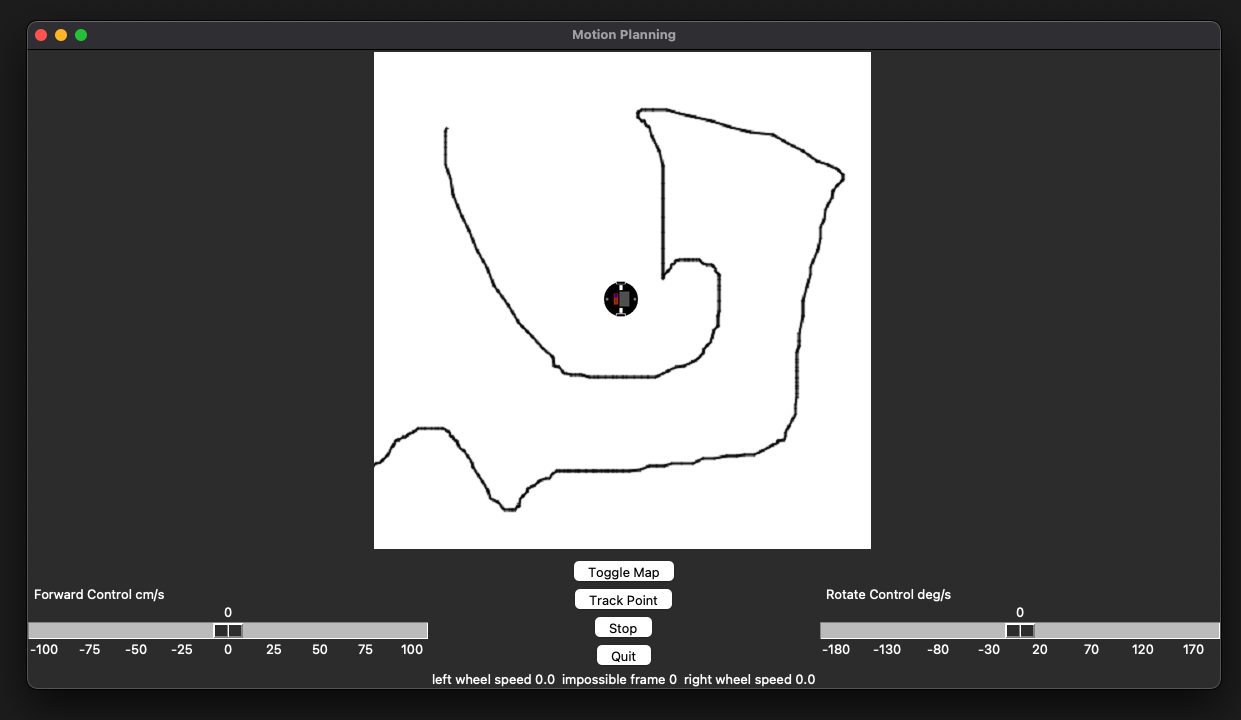
Text

Description automatically generated

Text

Description automatically generated

Now with the control completed we conducted another test with a slightly different Kp value. The values used in this test are listed below. The robot trajectory in this test is shown in figure 19. Reducing Kp would reduce the overspeed of the robot; it also seemed to cause the over compensation of alpha and beta after point 2. This trajectory also had 3 impossible frames.

 Graphical user interface

Description automatically generated

Figure 19. Kp Modified Trajectory

Next we conducted a test with a slightly different Ka value. The values used in this test are listed below. The robot trajectory in this test is shown in figure 20. Reducing alpha widens the arch of the trajectory of the robot between waypoints. This path also had no impossible frames.

A screenshot of a computer

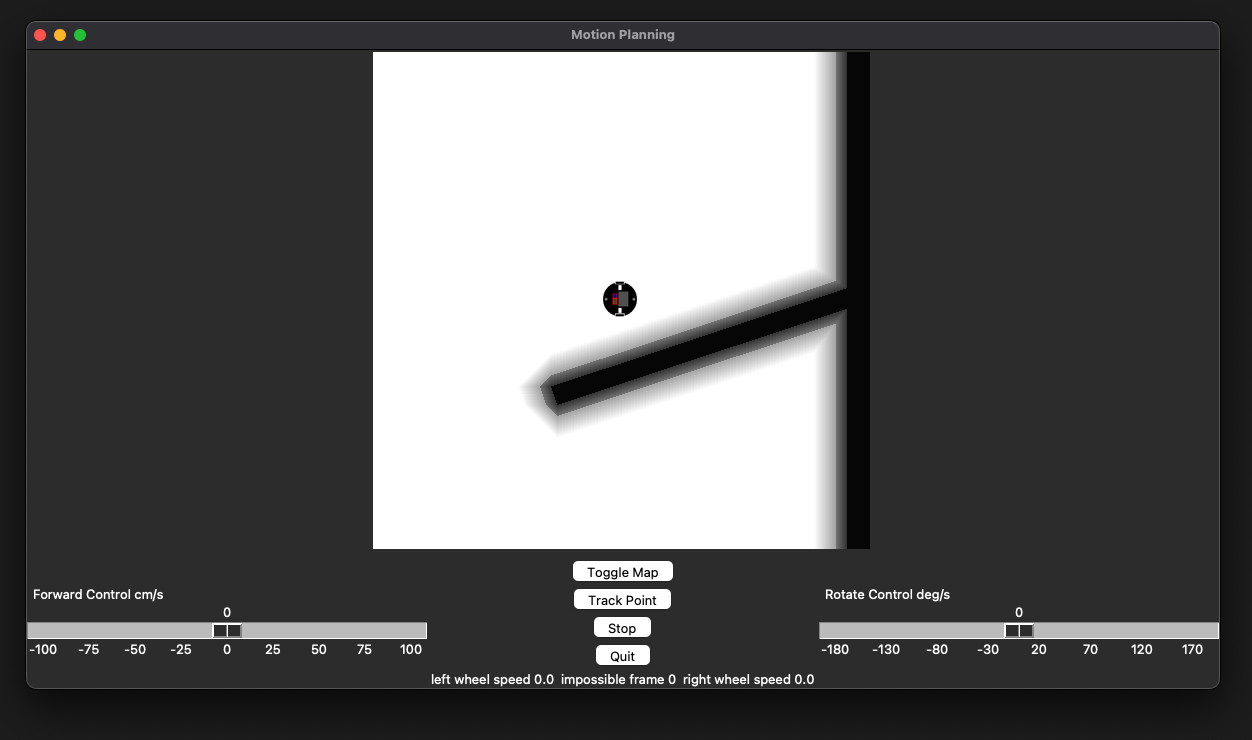
Description automatically generated with medium confidence 

Figure 20. Ka Modified Trajectory

Our last test we slightly changed the Kb value. The values used in this test are listed below. The robot trajectory in this test is shown in figure 21. Reducing beta caused a significant alteration to the robot trajectory. Expecily between the 3rd and 4th waypoint. This new k value also caused 7 impossible frames to be formed.

A picture containing graphical user interface

Description automatically generated Text

Description automatically generated with medium confidence

Figure 21. Kb Modified Trajectory

VII. CONCLUSION

In modern robotics control theory is a critical area of focus, it is only with the use of these control algorithms that robots can function in real world environments. The goal of this lab is to explore proportional control to control a robot’s route in a virtual environment. Our robot independently navigates to 4 different waypoints using a custom proportional controller algorithm we developed in python. This proportional controller uses the linear and angular velocity of the robot until it reaches the desired waypoint.

After completing a series of tests using different values we were able to achieve a trajectory that worked on a wide variety of motion pathways without any impossible frame errors.

In our experimentation we determine the effects different k values for alpha, beta, and rho had on the system. Reducing Kp would reduce the overspeed of the robot but it could cause imbalance between linear and rotational velocity causing oscillations. Reducing Ka affects how direct the robot will take to the next goal frame. A lower value allows the robot to take a larger arch. Lastly Kb would also cause the path to widen its arch but it could cause oscillations as the angular velocity becomes over compensated.

Overall we are satisfied with the effectiveness and the level of versatility shown by our custom Pcontrol algorithm.

VIII. ACKNOWLEDGEMENTS

None

IX. REFERENCES

Jenkins, Jonathan. “ME 5751 Module 1: Proportional Control for Point Tracking.” *ME5751\_Module1*, GitHub, 2022, <https://github.com/jonjenkins31/ME5751_Module1>.

Siegwart, Roland, et al. *Introduction to Autonomous Mobile Robots*. Second Edition ed., Massachusetts Institute of Technology, 2011.