ECEN 345/ECEN 3450-001

Mobile Robotics I

Prof. A. Gilmore (Instructor)

Sajjad Alhassan (TA)

Laboratory Assignment 2

Lab 2 - Dead Reckoning: Autonomous Locomotion Using Odometry

Written by

David Perez

Department of Electrical and

Computer Engineering (ECE)

University of Nebraska

Lincoln/Omaha

Due: 2/17/2024

Introduction

The objective of this lab was to introduce some basic maneuvering tasks using the CEENBoT robot. The first task was calibrating the robot by measuring the distance it would travel by providing each motor with a number of steps. Following this, a 4ft by 4ft square path, then a maze consisting of multiple right and left turns, and finally a circular path. To complete this lab the *CEENBoT-API Programming Fundamentals by Jose Santos* was a useful resource for an intro to the servo motor control. This lab consumed roughly 12 hours of programming and was done solely by David Perez.

Background

The method of autonomous locomotion control that this lab explores is referred to as dead reckoning. Dead reckoning is defined as navigation performed by calculation. In the context of this lab the idea here is to plan the trajectory by explicitly calculating the necessary motor inputs to achieve the desired movement to navigate a predefined path.

The CEENBoT robot is equipped with two independent front stepper motors each attached directly to a plastic wheel with rubber tires. A third 360 degree free spinning wheel exists on the rear of the robot. Given this robot's architecture the locomotion (the act of making a robot move around its environment) is directly related to driving these two stepper motors. Due to the internal motor control modules that are inherent in stepper motors, the motors can make a precise step where each step is a partial rotation of the motor. For the stepper motors present on the CEENBoT, each motor rotation is achieved using 200 steps. With 200 steps per revolution, each motor turns 1.8 degrees from each step (360/200 = 1.8).

As one could expect, like any physical system, the robot's motor rotation isn’t 100% precise and there are many factors that make this the case. For instance, if the robot's motors rotate and a slippage with the surface occurs the odometry of the robot's traveled distance may be inconsistent from what is calculated. Furthermore each surface that the robot can traverse will impose a different amount of friction with the

wheels. Considering the small form factor and limited tire tread these factors can have a strong effect in the robot's ending position.

Procedure

The first part of the lab focuses on gauging the robots movements based on a provided number of steps. To achieve this there were four distances (6, 12, 18, and 24 inches) the robot was tasked to traverse where the error between the actual versus the desired distance was measured. These straight line distances were run 20 times (5 for each distance) where the error and average error were calculated for each distance. Following this another set of 20 ran with a different speed, and another set of 20 on carpet.

The second part of the lab was the task of getting the robot to travel in a 4ft by ft square. This task brought about the challenge of having the robot perform right turns. The strategy employed by the code mentioned in this report was to have one wheel rotate forward while the other wheel rotated in reverse. This design choice seemed most beneficial because, if done correctly, the robot's center position could remain the same.

The third part of the lab was a maze that contained right and left turns with straight aways of a minimum of 2 feet and a maximum of 9 feet. The left turns for this course used the same method and reasoning as did the second part of the lab. The maze that is mentioned in this report can be seen from **Figure 1** below:

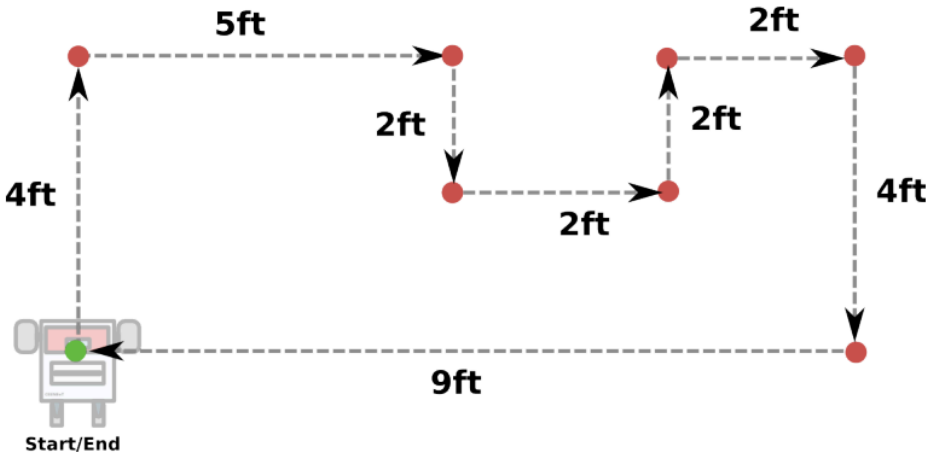

Figure 1: Lab 2 Part 3 Maze Course

Lastly, the fourth and final part of the lab was getting the robot to move in a circle. The method employed was to turn the robot a marginal amount then have the robot move forward a maringal amount and repeat these steps until the robot completed the circle and returned to its starting point.

Source Code Discussion

To complete the paths outlined in this report there exists two prominent c files (excluding main) in the code attached to this report. These two files are called **motor_control.c** and **app.c.** The motor control file is as its name implies and provides the necessary input to the stepper motors. The app file on the other hand handles the logic of when and how much to drive the motors forward or turning the robot left and right.

In **motor_control.c** there are defines that determine what the scale is of the motor rotation and that geometric relation to the size of the wheel. These defines are as seen in **Source Code 1**:

```
# define pi 3.14159265359
# define WHEEL_RADIUS 1.6250 // inches
# define CIRCUMFERENCE (2 * pi * WHEEL_RADIUS) // inches
# define SPR 200 // stepper motor steps per revolution
// Approximate Distance Per Step for each stepper motor step (in inches)
# define DPS (CIRCUMFERENCE / SPR)
```

Source Code 1: Motor Rotation to Wheel Distance Defines

With these defines in place the approximate distance the robot will travel with respect to the robot's wheel rotation per step is given by the function **steps_per_distance().** Using this, the **move_forward_in_inches(double distance)** function can use that value as a starting point for determining how much to rotate the motors. When performing the first part of the lab it was noticed that the left stepper motor rotated slightly less than the right wheel when provided the same inputs. As a result of this, the left wheel has a multiplier induced to compensate for that smaller rotation (final testing showed that **0.015** * number steps per distance was a fair compensation). Additionally, the greater the distance the robot had traveled the greater it would overshoot. To adjust for this another multiplier was performed on the number of steps per distance. This multiplier reduced the steps for each motor by a scale of **0.0185** The final compensator used adding **4** extra steps to compensate for shorter distances undershooting.

To rotate the robot 90 degrees the **turn_90_degrees(bool turn_left)** function was used. The function simply rotates the left motor in reverse **131.5** steps at the same time the right motor rotates **115** forward (if **turn_left** is true otherwise the inverse is used).

For completing each part of the lab the **app.c** file contains a function to perform each task. The function **app_lab_part1()** drives the robot forward 6, 12, 18, and 24 inches with a 6 second delay between each run. Each part of this lab follows this same naming convention e.g. the second part of the lab is completed by **app_lab_part2()**. Since these functions are simply varied ordered calls to **turn_90_degrees()**,

move_forward_in_inches(), and **turn_18_degrees(),** and their functionality is self explanatory, a deeper explanation of this code seems unnecessary.

Results

When performing part 1 of the lab the first subtask was getting the robot to drive in a straight line for specific desired distances. The results for this subtask are given in **Table 1** below where the robot was traversing on a hardwood floor.

Desired Distance: 6 inches		**Desired Distance: 12 inches**		**Desired Distance: 18 inches**		**Desired Distance: 24 inches**	
Actual	**Error**	**Actual**	**Error**	**Actual**	**Error**	**Actual**	**Error**
4	-2.00	11.75	-1/4	17.125	-3.5/4	23.75	-1/4
5.25	-3/4	11.875	-1/8	17.9375	-1/16	22.5	-1.5
5.9375	-1/16	11.25	-3/4	16	-2	28	+4
5	-1	11.75	-1/4	18.03125	+1/32	24.5	+1/2
5.75	-1/4	11.875	-1/8	18.125	+1/8	24.5	+1/2
Average Distance	**Average Error**	**Average Distance**	**Average Error**	**Average Distance**	**Average Error**	**Average Distance**	**Average Error**
5.1875	-0.8125	11.7	-0.3	17.444	-0.55625	24.65	0.91

Table 1: Lab 2 Part 1 Sub Part 1 Straight Line Tests

When conducting experiments part 1 sub part 1 the robot was operating at a speed of 200 steps/second with an acceleration of 200 steps/second^2. When conducting these tests the robot's shortest distance traveled was 4 inches and the longest distance being 28 inches.

The second sub part of part 1 consisted of testing the robots straight line travel for distances of 12 and 24 inches. For this test the speed of the robot was increased from 200 to 300 steps/second. **Table 2** shows the results of these tests.

Desired Distance: 6 inches		**Desired Distance: 24 inches**	
Actual	**Error**	**Actual**	**Error**
5.5	-1/2	24.25	+1/4
5.75	-1/4	23.875	-1/8
5.75	-1/4	23.75	-1/4
6	0	24.0625	+0.5/8
5.875	-0.5/8	24.0625	+0.5/8
Average Distance	**Average Error**	**Average Distance**	**Average Error**
5.775	-0.2125	24	0

Table 2: Lab 2 Part 2 Sub Part 2 Straight Line 300 steps/second

When conducting straight line motion at 300 steps/second the average error for the 24 inch distance was zero and the average error for the 6 inch distance was -0.2125. It would appear that the robot had better accuracy at higher speeds which seems counter intuitive. Something to keep in mind is that the robot's battery is continuously being drained and with a lower battery the robot's wheels have less torque and also become more inaccurate. Additionally, since the robot was overshooting the 24 inch distance in the previous sub part it would make sense to see that overshoot correcting when the battery drops. Furthermore, the 6 inch distance loses a bit of its validity considering there was an outlier for the first run of the 6 inch test that swayed the average error.

The final sub part of part 1 of this lab was performing a straight line distance test of 6, 12, 18, and 24 inches where each distance was driven 5 times while on carpet. The results of these tests are presented in **Table 3**.

Desired Distance: 6 inches		Desired Distance: 12 inches		Desired Distance: 18 inches		Desired Distance: 24 inches	
Actual	**Error**	**Actual**	**Error**	**Actual**	**Error**	**Actual**	**Error**
6.25	+1/4	12	+0	18.5	+.5	24.5	+.5
6.25	+1/4	11.75	-1/4	18.5	+.5	24.6	+.6
5.875	-1/8	12	0	18.6	+6/10	24.65	+.65
5.875	-1/8	11.875	-1/8	18.6	+0.6	24.5	+.5
5.9375	+.5	12.5	+.5	18.5	+.5	24.5	+0.5
Average Distance	**Average Error**	**Average Distance**	**Average Error**	**Average Distance**	**Average Error**	**Average Distance**	**Average Error**
6.0375	0.05	11.825	1/40	18.54	0.54	24.54	0.55

Table 3: Lab 2 Part 1 Sub Part 3 Straight Line Carpet Tests

When performing the straight line test on carpet the largest distance traveled was 24.65 inches during the 24 inch distance test and the shortest distance being 5.875 inches during the 6 inch distance test. The average error increased with distance during these tests as expected.

As previously stated the second part of this lab was getting the robot to traverse a 4ft by 4ft square. For this test the error was calculated by measuring the distance from each expected corner from the robots actual distance (measuring from the front of the robot). The desired path is shown in **Figure 2** with the results of this task shown in **Table 4**.

Distance Error (1st corner)	Distance Error (2nd corner)	Distance Error (3rd corner)	Distance Error (4th corner)
1.5	4.5	4	18.5
Average Error: 7.125 inches			

Table 4: Part 2 Subpart 1 Square Path

When performing the square path for the first time the average error was 7.125 inches. The 90 degree turns were achieved by rotating the left wheel 125 steps forward and the right wheel in reverse by 125 steps. The speed that these motors operated was 300 steps/second.

The second subpart of this lab was attempts at improving the robots accuracy. To achieve this the robot's left wheel steps were decreased from 125 to 115 and the right wheel steps from 125 to 122 These results can be seen in **Table 5** below.

Distance Error (1st corner)	Distance Error (2nd corner)	Distance Error (3rd corner)	Distance Error (4th corner)
4.5	5	7	6.5
Average Error: 15.875 inches			

Table 5: Part 2 Subpart 2 Square Path

Adjusting the amount of wheel rotation from the left and right motors did increase the error by almost 2 fold. That said, this isn't well indicative of the robots performance. During the previous subpart the robot had a final error of 18.5 inches while subpart 2 was closer to the starting point although its average error had increased.

Adjustments in wheel speed were also investigated in this part of the lab. Namely, the wheel speed of each motor was increased from 300 to 400 steps/second. See **Table 6.**

Distance Error (1st corner)	Distance Error (2nd corner)	Distance Error (3rd corner)	Distance Error (4th corner)
4	2	7	6.375
Average Error: 4.844inches			

Table 6: Part 2 Subpart 3 Square Path

When the speed of each motor was increased surprisingly the robot's performance improved. Compared to previous tests it had an average error of 3 inches lower than the initial run and about 11.5 lower error than subpart 2.

This part of this lab was the maze that is shown in **Figure 1**. The error for each distance is calculated in the same fashion as part 3 of this lab. For the sake of redundancy every test run was not recorded while the one with the best performance is shown in **Table 7**.

Distance Error (1)	Distance Error (2)	Distance Error (3)	Distance Error (4)	Distance Error (5)	Distance Error (6)	Distance Error (7)	Distance Error (8)
4+1/8	3.5	4.75	3	6.75	6	5.25	4.75
Average Error: 5.703							

Table 6: Part 3 Maze Path

When performing the maze path the robot was successfully able to complete the maze and end up near the starting position with an error of 4.75 inches. When testing different parameters the robot's precision started to struggle with the series of shorter 2ft turns. These errors would accumulate and on previous runs would misalign the robot's turns and eventually place the robot's final position far off from its starting position. Turning the motors simultaneously still appears to be the optimal choice of rotating the vehicle in order to maintain its position. If there was differential control the performance of this robot could be drastically improved. The reason being is that the wheels rotate at different speeds making it very difficult to get the robot to consistently drive in a straight

line. Further issues that arise when being completely reliant on motor rotations is that any slippage in the wheels completely throws the robot's sense of position off and error can accumulate very rapidly. Furthermore, battery life has a drastic effect on the robots' autonomy. When conducting some of these tests, the robot would not perform the same if it had a lower battery life in comparison to a fully charged battery.

The final part of this lab was getting the robot to drive in a 2 ft diameter circle. The strategy employed was to marginally rotate the right wheel in reverse and the left wheel forward. This was followed by moving the robot a few steps forward. This process was completed 24 times until the robot had completed the circle. In hindsight the performance could substantially be improved if the robot has instead continuously rotated the left wheel more than the right wheel since the robots reaction time for repetitive motor calls has a large delay.

Conclusion

The objective of this lab was to introduce a fundamental approach to robot locomotion. The locomotion strategy employed in this lab is referred to as dead reckoning which is where the predetermined path was traversed by calculating the robots travel distance when provided motor inputs. The paths discussed and completed were 4 straight line paths of 6, 12, 18 and 24 inches, followed by a 4 ft by 4ft square, then a maze of left and right turns and lastly a circle of 2 ft diameter. Through these various tests a mapping of motor inputs to real world distance was calibrated to provide the robot with the ability to traverse a predetermined path with fairly accurate results. As seen from the results section, there are many factors such as battery life, tire slips, and terrain that affect the accuracy of the robots motions and limitations and challenges that robot locomotion by direct calculation can bring about.