Lab 3 – Navigation (Group 51)

Alex Bhandari-Young – 260520610, Neil Edelman – 110121860

2013-10-02

Abstract

We designed a software system that allows a robot to move to an absolute location while avoiding obstacles.

1 Data

Repeated runs of the robot path in Figure 1 gave the data in Table 1. We used more precise measurements values used in the odometer from our last lab[1]; that made the error go down.

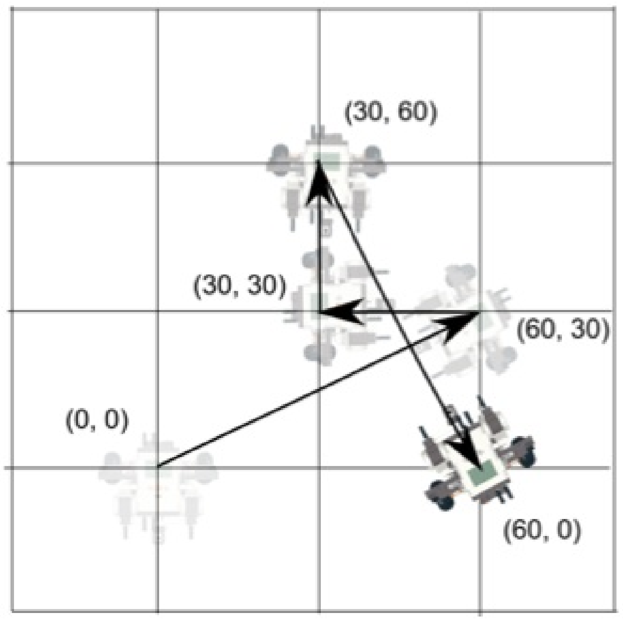

Figure 1: Robot navigation path in cm.[2]

2 Data Analysis

2.1 Discussion

The errors are part of the odometer. When we measured the odometer values more precisely, our errors dropped significantly. The navigator is just the software that reads from the odometer and decides where to go to minimise the error. It works by comparing a set minimum required error to the actual error and deciding whether that's good enough. We could reduce the error in the software, but that could create a condition where it's never good enough an it loops back incessantly.

2.2 Error Analysis

Calculate the differences in Equation 1–20.

$$\begin{aligned} d_{x,1} &= (0.0) - (-0.00) \\ &= 0.00 \quad (1) \\ d_{y,1} &= (-60.0) - (-60.11) \\ &= 0.11 \quad (2) \end{aligned}$$

$$\begin{aligned} d_{x,2} &= (0.5) - (0.18) \\ &= 0.32 \quad (3) \\ d_{y,2} &= (-59.5) - (-60.50) \\ &= 1.00 \quad (4) \end{aligned}$$

$$\begin{aligned} d_{x,3} &= (0.5) - (-0.10) \\ &= 0.60 \quad (5) \\ d_{y,3} &= (-59.5) - (-60.50) \\ &= 1.00 \quad (6) \end{aligned}$$

$$\begin{aligned} d_{x,4} &= (0.5) - (0.06) \\ &= 0.44 \quad (7) \\ d_{y,4} &= (-60.0) - (-60.31) \\ &= 0.31 \quad (8) \end{aligned}$$

actual		reported		error	
x (cm)	y (cm)	x (cm)	y (cm)	x (cm)	y (cm)
0.0	-60.0	-0.00	-60.11	0.0	0.1
0.5	-59.5	0.18	-60.50	0.3	1.0
0.5	-59.5	-0.10	-60.50	0.6	1.0
0.5	-60.0	0.06	-60.31	0.4	0.3
0.5	-60.5	0.11	-60.42	0.4	-0.1
0.0	-59.0	0.17	-60.51	-0.2	1.5
1.5	-61.0	0.06	-60.32	1.4	-0.7
1.5	-61.0	-0.04	-60.11	1.5	-0.9
1.0	-60.0	-0.09	-59.99	1.1	-0.0
0.5	-59.5	0.17	-60.50	0.3	1.0

Table 1: Reported error as read by the robot, and real error as read by a ruler and the difference between them. The difference, as (x, y), mean is $(0.60, 0.33)$, variance is $(0.33, 0.62)$, and the corrected sample standard deviation is $(0.58, 0.79)$.

$$d_{x,5} = (0.5) - (0.11) = 0.39 \tag{9}$$

$$d_{y,5} = (-60.5) - (-60.42) = -0.08 \tag{10}$$

$$d_{x,6} = (0.0) - (0.17) = -0.17 \tag{11}$$

$$d_{y,6} = (-59.0) - (-60.51) = 1.51 \tag{12}$$

$$d_{x,7} = (1.5) - (0.06) = 1.44 \tag{13}$$

$$d_{y,7} = (-61.0) - (-60.32) = -0.68 \tag{14}$$

$$d_{x,8} = (1.5) - (-0.04) = 1.54 \tag{15}$$

$$d_{y,8} = (-61.0) - (-60.11) = -0.89 \tag{16}$$

$$d_{x,9} = (1.0) - (-0.09) = 1.09 \tag{17}$$

$$d_{y,9} = (-60.0) - (-59.99) = -0.01 \tag{18}$$

$$d_{x,10} = (0.5) - (0.17) = 0.33 \tag{19}$$

$$d_{y,10} = (-59.5) - (-60.50) = 1.00 \tag{20}$$

Calculate the sum of the differences (Equation 1–20) in Equation 21–22.

$$\begin{aligned} \text{sum}_x &= \sum_{i=1}^{10} d_{x,i} \\ &= (0.00)+ \\ &\quad (0.32)+ \\ &\quad (0.60)+ \\ &\quad (0.44)+ \\ &\quad (0.39)+ \\ &\quad (-0.17)+ \\ &\quad (1.44)+ \\ &\quad (1.54)+ \\ &\quad (1.09)+ \\ &\quad (0.33) \\ &= 5.98 \end{aligned} \tag{21}$$

$$\begin{aligned}\text{sum}_y &= \sum_{i=1}^{10} d_{y,i} \\ &= (0.11)+ \\ &\quad (1.00)+ \\ &\quad (1.00)+ \\ &\quad (0.31)+ \\ &\quad (-0.08)+ \\ &\quad (1.51)+ \\ &\quad (-0.68)+ \\ &\quad (-0.89)+ \\ &\quad (-0.01)+ \\ &\quad (1.00) \\ &= 3.27 \end{aligned} \tag{22}$$

Calculate the sum of the differences (Equation 1–20) squared in Equation 23–24.

$$\begin{aligned}\text{ssq}_x &= \sum_{i=1}^{10} {d_{x,i}}^2 \\ &= (0.00)^2+ \\ &\quad (0.32)^2+ \\ &\quad (0.60)^2+ \\ &\quad (0.44)^2+ \\ &\quad (0.39)^2+ \\ &\quad (-0.17)^2+ \\ &\quad (1.44)^2+ \\ &\quad (1.54)^2+ \\ &\quad (1.09)^2+ \\ &\quad (0.33)^2 \\ &= 6.58 \end{aligned} \tag{23}$$

$$\begin{aligned}\text{ssq}_y &= \sum_{i=1}^{10} {d_{y,i}}^2 \\ &= (0.11)^2+ \\ &\quad (1.00)^2+ \\ &\quad (1.00)^2+ \\ &\quad (0.31)^2+ \\ &\quad (-0.08)^2+ \\ &\quad (1.51)^2+ \\ &\quad (-0.68)^2+ \\ &\quad (-0.89)^2+ \\ &\quad (-0.01)^2+ \\ &\quad (1.00)^2 \\ &= 6.65 \end{aligned} \tag{24}$$

Calculate the mean from Equation 21–22 in Equation 25–26.

$$\begin{aligned}\text{mean}_x &= \frac{\text{sum}_x}{N} \\ &= \frac{5.98}{10} \\ &= 0.598000 \end{aligned} \tag{25}$$

$$\begin{aligned}\text{mean}_y &= \frac{\text{sum}_y}{N} \\ &= \frac{3.27}{10} \\ &= 0.327000 \end{aligned} \tag{26}$$

Calculate the variance from Equation 21–22 and 23–24 in Equation 27–28.

$$\begin{aligned}{\sigma_x}^2 &= \frac{\text{ssq}_x - \frac{{\text{sum}_x}^2}{N}}{N-1} \\ &= \frac{6.58 - \frac{5.98^2}{10}}{10-1} \\ &= 0.333684 \end{aligned} \tag{27}$$

$$\begin{aligned}{\sigma_y}^2 &= \frac{\text{ssq}_y - \frac{{\text{sum}_y}^2}{N}}{N-1} \\ &= \frac{6.65 - \frac{3.27^2}{10}}{10-1} \\ &= 0.620001 \end{aligned} \tag{28}$$

Calculate the corrected sample standard deviation from the variance (Equation 27–28) in Equation 29–30.

$$\sigma_x = \sqrt{{\sigma_x}^2} = \sqrt{0.333684} = 0.577654 \tag{29}$$

$$\sigma_y = \sqrt{{\sigma_y}^2} = \sqrt{0.620001} = 0.787401 \tag{30}$$

3 Observations and Conclusion

We noticed experimentally that one actuator is not working, so we placed it on the top of our robot to hold the sensor. As such, the sensor can not move. This is an additional design constraint.

We tried `rotate` and `rotateTo` methods as seen in the last lab[1], but these inherently blocked output. The `setSpeed` method was more appropriate. `setSpeed` sets it to the absolute value of the speed; you then call `forward` or `backward` depending on the sign. This is not documented in the API.

The errors resulting from the navigator are due to the threshold distance from the target calculated, which was set to 3 cm during this lab; because theta is corrected every ten cycles of the while loop, the robot consistently stops within 1 cm of the target point. This can be corrected by decreasing this tolerated distance, but choosing a value too low will waste time because the robot will have to oscillate more to achieve it.

The controller works by pointing the robot towards its target coordinate. It moves toward it, running in a while loop that checks the ultrasonic sensor for obstacles every cycle, and incrementing a count which allows it to check theta and run `turnTo` to correct its angular error every ten cycles of the while loop. This loop runs while distance is greater than the threshold distance from the point, which is set to three. This results in no noticeable robot oscillation. When `turnTo` is called the robot stops to turn, this happens about once every 5-10 seconds; further improvements were implemented to adjust theta while the robot is in motion, but were discarded because the time constraint.

The controller consistently moves to within 1 cm of the target point as read by the odometer. However, the odometer incurs an additional error of approximately 1 cm from wheel slippage, making the overall accuracy bound 2 cm.

Increasing the acceleration of the robot will result in greater wheel slippage, and increase the error in the odometer. The main sources of error using navigation and odometry is from slippage error and from mis-calibrations. The battery voltage gives an unknown variable for which the constants of calibration do not account. Wheel slippage can be reduced by decreasing the acceleration of the robot. Excluding the odometer, error in the navigation is also caused by the distance threshold, but this can be reduced by reducing the threshold.

4 Further Improvements

A software solution to reducing error in wheel slippage is to reduce the acceleration at which the robot moves; this becomes an issue when time is important. We haven't used the limit acceleration function of lejos. Use of the an additional sensor, such as the light sensor, to aid in determining the robots position is a hardware solution.

Reducing the acceleration of the robot will reduce the error in odometry because wheel slippage causes the odometer indicate erroneous values. Wheel slippage can cause linear and angular error. Linear error can be corrected. Angular error, where there is slippage in one wheel that introduces error in the direction the robot thinks it is facing, can be more serious; as the robot moves forward, the error is linear in the distance traveled. The use of the light sensor to correct angular would be a very effective solution. The robot is able to correct θ with a single light sensor by remembering when it crosses a line, determining based on its knowledge of the distance between lines when it should detect the next line, and using trigonometry to determine the error in θ. This and the method using two light sensors was described in detail in the previous lab report[1].

References

[1] A. Bhandari-Young and N. Edelman, "Lab 2 odometry (group 51)," *McGill*, 2013.

[2] McGill, 304-211, *Lab 3: Navigation and Obstacle Avoidance.*