Lab 2 – Odometry (Group 51)

Alex Bhandari-Young – 260520610, Neil Edelman – 110121860

2013-09-25

Abstract

An autonomous automated vehicle is programmed with a dead-reckoning system, and then enhanced with colour sensors aimed at the floor to detect grid lines. The odometer is calibrated and measured.

1 Data Collection

Repeated robot runs on the 3-by-3 are seen in without odometer correction in Table 1 and with odometer correction in Table 2.

2 Data Analysis

a) "What was the standard deviation of the results without correction (compute it for x and y separately, and provide the four (4) values in a table)? Did it decrease when correction was introduced? Explain why/why not.[1]" Standard deviation decreased when correction was used because using correction improves the accuracy of the odometer. Since standard deviation is a measure of the deviation of data from the mean, the use of correction would increase the consistency of the odometers estimated positions over a number of trials, thereby decreasing standard deviation. The completely blind reckoning odometer has a much higher chance of incurring error, which spreads data points over a larger range of values, increasing standard deviation.

b) "With correction, do you expect the error in the x position or the y position to be smaller? Explain." The error is corrected in the direction of travel. In this lab we used a left-handed co-ordinate system in which when theta=0 and the robot is pointing upward, which we define as north. When the robot moves north, x is incremented, and when the robot moves west, y is incremented. It then follows that when the robot moves east y is decremented, and when the robot moves south x in decremented. Going back to the question, this means that when traveling north or south, the robot will be correcting its x value, and when east or west it will be correcting its y value. Since the robot moves in a square starting going north, then east, south, and finally west, it corrects x, y, x, and then y. The value which is corrected more recently will incur less error because it has less time to incur error. In this case it is y. Thus the error in y is expected to be less than x.

3 Observations and Conclusion

"Is the error you observed in the odometer (without correction) tolerable for larger distances (i.e. circumnavigating the field requires a travel distance five (5) times larger than that used for this lab)? Do you expect the error to grow linearly with respect to travel distance? Explain briefly.[1]"

The error in the dead reckoning odometer (the one without correction) is not tolerable for large distances. It is expected that the error would grow linearly with dead-reckoning. This is because individual errors are independent from each other. Because future error is not effected by past error, the error will simply build up linearly over time. This is not suitable for large distances because without correction, the robot eventually goes astray. With the light sensor on the ground, it allows us to get the absolute position every tile. This corrects the odometer and limits the error. It's analogous to the heading gyroscope, which is quick-acting but neutrally-stable, being corrected by the compass, which is slow, but accurate.

4 Error Analysis

See Figure 1 and Figure 2 for values. The value of the corrected sample standard deviation is $(0.82, 0.78)$ for the uncorrected and $(0.30, 0.11)$ for the corrected. The is due to slippage, imprecision, and the inaccuracy in the angle.

actual		reported		delta	
x (cm)	y (cm)	x (cm)	y (cm)	x (cm)	y (cm)
-3.0	-0.5	-0.10	-0.10	-2.9	-0.4
-1.1	-0.6	-0.09	-0.10	-1.0	-0.5
-0.6	-0.4	-0.11	-0.14	-0.5	-0.3
-2.1	-2.1	-0.12	-0.13	-2.0	-2.0
-2.4	-1.8	-0.11	-0.10	-2.3	-1.7
-2.7	-0.7	-0.09	-0.10	-2.6	-0.6
-1.6	-2.5	-0.09	-0.09	-1.5	-2.4
-1.4	-1.6	-0.10	-0.11	-1.3	-1.5
-2.5	-1.9	-0.11	-0.10	-2.4	-1.8
-2.9	-2.0	-0.10	-0.13	-2.8	-1.9

Table 1: Reported error as read by the robot, and real error as read by a ruler and the difference between them for the [un]corrected code. The difference, as (x, y), mean is $(-1.93, -1.30)$, variance is $(0.67, 0.61)$, and the corrected sample standard deviation is $(0.82, 0.78)$.

actual		reported		delta	
x (cm)	y (cm)	x (cm)	y (cm)	x (cm)	y (cm)
-2.9	1.0	-3.06	1.14	0.2	-0.1
-2.4	1.3	-2.48	1.39	0.1	-0.1
-2.1	1.4	-2.56	1.41	0.5	-0.0
-3.1	1.1	-3.12	1.09	0.0	0.0
-1.9	1.5	-2.01	1.49	0.1	0.0
-2.8	1.1	-3.01	1.10	0.2	0.0
-2.7	1.0	-2.81	1.25	0.1	-0.2
-2.2	1.5	-2.46	1.38	0.3	0.1
-3.5	1.1	-3.48	1.01	-0.0	0.1
-1.1	1.3	-2.10	1.29	1.0	0.0

Table 2: Reported error as read by the robot, and real error as read by a ruler and the difference between them for the [un]corrected code. The difference, as (x, y), mean is $(0.24, -0.02)$, variance is $(0.09, 0.01)$, and the corrected sample standard deviation is $(0.30, 0.11)$.

5 Further Improvements

a) "Propose a means of, in software, reducing the slip of the robot's wheels (do not provide code).[1]" The easiest way to reduce wheel slippage is to limit the robot's speed. The simplest way to implement this would be do set any motor speed values above a certain threshold to that threshold value. However, in the extreme this is impractical because the robot would be too slow to get anything done. The bigger the "field" or area the robot is allowed to move in, the more important going fast becomes because the robot has to cover move ground. But the farther the robot travels, the more error is accumulates. In software, a balance needs to be reached that weighs speed with accuracy. Additionally, if the robot has access to another sensor such as a light sensor and grid lines, it would go fast between grid lines and correct the accumulated error as it crosses a line. This is a method for reducing the error caused by slip, but not the slip itself. Other than determining the distance it has to travel, and using an algorithm to determine an optimal speed that minimizes slip and balances speed with accuracy, any other software method would require another sensor to be able to detect slip and correct for it. We demonstrated this in this lab using the light sensor to correct for error using the grid lines.

b) "Propose a means of, in software, correcting the angle reported by the odometer, when (do not provide code):[1]"

i) “The robot has two light sensors.[1]” If the robot had two light sensors placed side by side on the front or back of the robot, it would be able to determine the angle it was facing every time it crossed a line, and correct theta accordingly. If the robot is aligned perpendicularly to the line, then the sensors will cross and thus detect the line at the same time. If it is not there will be a delay in the second sensor detecting the line. This time delay can be converted into a distance by getting the tachometer value of the motors when the first light sensor detects and finding its difference from the tachometer reading when the second sensor detects the same line (assuming the robot is traveling straight, a more complex algorithm could be used with measures both tachometers), converting this angle difference to radians, and multiplying by the radius of the wheel to get the distance traveled between the light sensors detecting the line. This creates a right triangle with the robots skew and the line it is crossing. Determining the corrected theta will depend on which direction the robot is traveling in but the error in theta will be the inverse cosine of this calculated distance divided by the distance between the light sensors.

ii) “The robot has only one light sensor.[1]” If the robot only has one light sensor, there is not way to determine the angle it is facing as it crosses a line. However, if the robot remembers when it crosses a line and then measures the distance to the next line, assuming it knows beforehand the distance between the lines, which we did in the case of this lab, it can calculate its error in theta based on the error in the expected versus the measured distance traveled between lines detected by the light sensor. For example, if the robot has a theta of 0, 90, 180, or 360 degrees and it is correct in this measurement, then the measured distance traveled should equal the actual distance between the lines. If the robot travels a distance larger than the actual distance between lines, and it is traveling in the simple square used in this lab, then a right triangle could be created with the actual distance the robot expects to travel between the lines and the distance it measured. The error in theta will be the inverse cosine of the actual distance between the lines divided by the measured distance traveled between light sensor detections. The robot has to be traveling straight for this to work. This could also be used it the robot was traveling in a more complex pattern than the square in this lab, but the algorithm would be more complex. The expected distance traveled based on theta would have to also be determined using the cosine of theta and compared with the measured distance.

References

[1] McGill, 304-211, Odometry Lab Instructions.

[2] J. A. Simpson, E. S. Weiner, et al., The Oxford english dictionary, vol. 2. Clarendon Press Oxford, 1989.