Design Principles and Methods

ECSE 211 (Fall 2020)

Michael Li and Cecilia Jiang

260869379, 260795889, Group #36

Lab 2: Odometer

Prof. Frank Ferrie

McGill Faculty of Engineering

Due September 26th, 2019

Section 1: Design Evaluation

Introduction

The goal of lab 2 is to design and implement an Odometry system that provides a robot's position and orientation, in order to allow a robot to autonomously navigate a field. In fact, a robot needs to keep constant track of its position (x and y) and its direction (θ). Relative positioning, using odometry is inexpensive, but inaccurate due to slipping wheels and other influences. This lab aid the robot localizes its position by combining the local navigation system (odometry) with a very basic global navigation aid (grid lines).

1.1 Software Design

At each iteration, the odometer stores the current tacho count of both motors. Hence, it is able to track how many degrees the wheels have spun from the previous position. Then, this information can be used to calculate the distance traveled by both wheels using the following formula:

$$\pi * wheel\ radian\ *\ (currentMotorTachoCount\ -\ lastMotorTachoCount)/180$$

Moreover, the vehicle's displacement is obtained by computing the average of the distances traveled by both wheels (i.e. (distance traveled by left wheel + distance traveled by right wheel)/2). We will call the latter *d*. The change in orientation is also computed by:

$$\theta_{heading} = \theta_{old\ heading} + \theta$$

where θ is *d* divided by wheelbase. Afterward, the displacement in x and y is determined by using the following formulas:

$$x = d * sin(\theta)$$

$$y = d * cos(\theta)$$

Finally, the odometer will update the current position and the current orientation by doing:

$$x_{current} = x_{old} + x$$
$$y_{current} = y_{old} + y$$
$$\theta_{heading} = \theta_{old\ heading} + \theta$$

OdometerCorrection class synchronizes coordinates whenever a black line is touched. We used the red mode of light sensor to sense black lines because this mode only measures the intensity level. Whenever the intensity level drops below the minimum non-black intensity, the Boolean variable touchedBlackLine becomes true. For the purpose of sensing black lines accurately in different environment

lighting condition, a second condition which triggers touchedBlackLine was added, complementing the first one. We compared the current intensity level with the last intensity reading and calculate last/cur intensity ratio. If this ratio is bigger than the maximum last to cur ratio, touchedBlackLine becomes true. For determining real coordinates, our implementation judges where the robot currently is based on robot's moving direction as well as recorded last vertical line or horizontal line's coordinate. Firstly, we approximated theta data to one of these four right-angle degrees, 0, 90, 180, 270. Then, based on the adjusted data, we updated real coordinates by either increasing or decreasing world-frame X coordinate and world-frame Y coordinate. The flowchart in fig 2 shows the design of odometerCorrection class in detail.

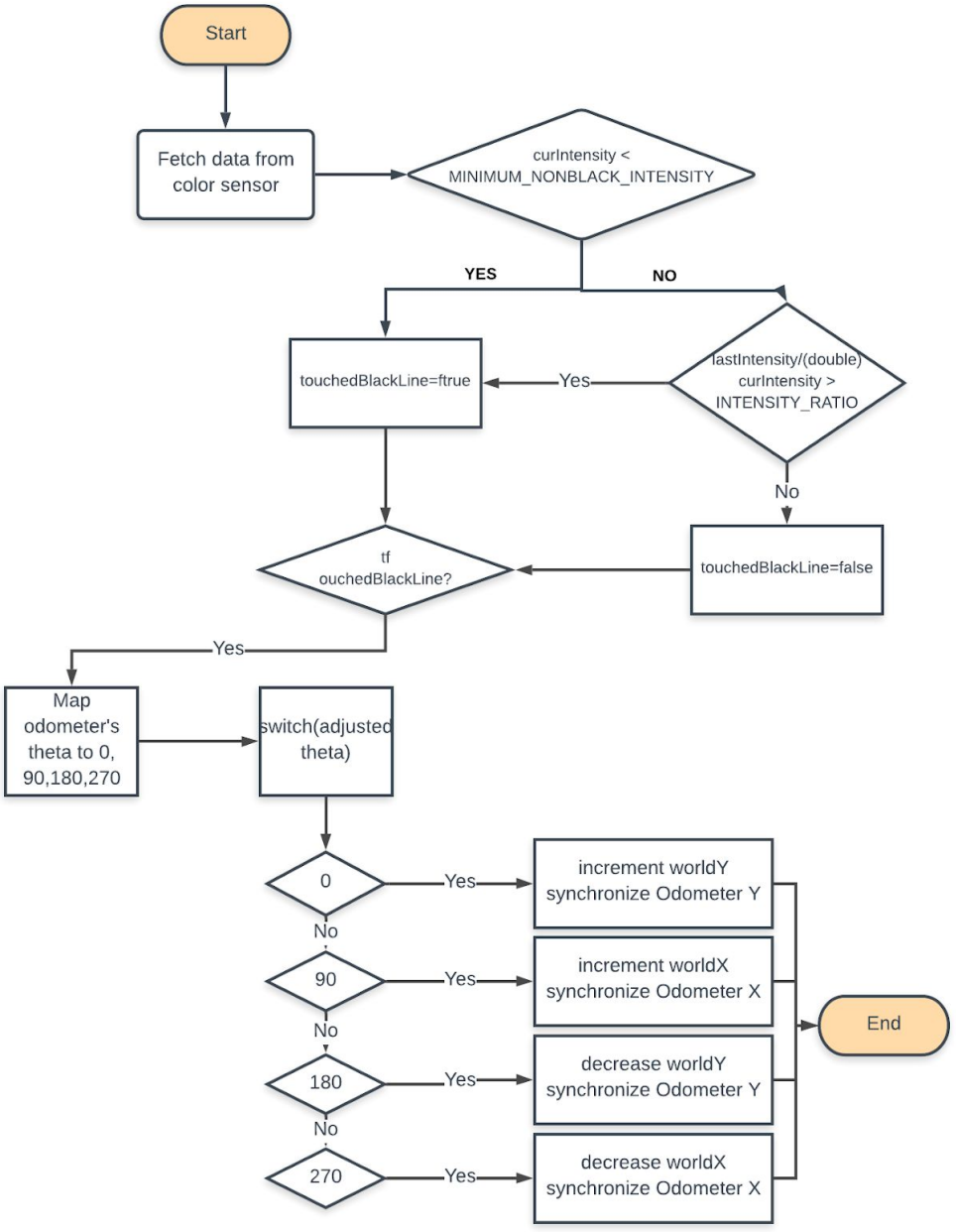

Fig. 1: Flowchart for OdometerCorrection

1.2 Hardware Design

For lab 2, we chose to stick with a wide but strong robot design shown in fig. 2 below. The main design objective is to ensure a robust robot structure. We use cross shape joints to connect the main brick to the wheels so that they won't easily fall apart or experience minor displacement. Two front wheels are assembled to two sides of the brick and we used a single rollable steel ball as the back wheel. In fact, the reason we chose the steel ball instead of the plastic wheel or a third wheel is that the steel ball can rotate in all directions, thus significantly reduce sliding and friction compared to the two other options. Besides, the height of the steel ball is perfect. After attaching it to the rear of the brick, the brick remained parallel to ground level.

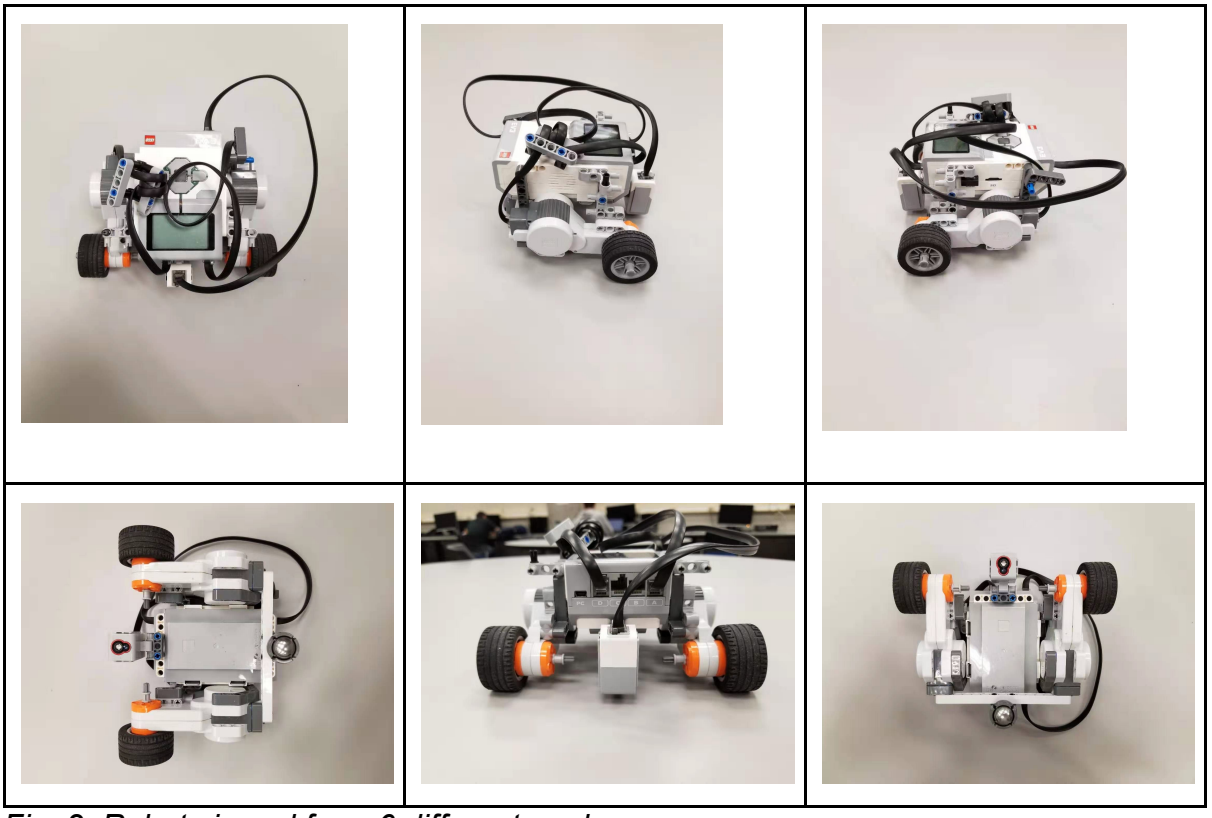

Fig. 2: Robot viewed from 6 different angles

The color sensor is mounted at 0,3 cm in front of the wheelbase. We wanted to fix it as close as possible to the wheelbase in order to reduce the error caused by this distance. In fact, the difference between the moment where the sensor detects the black line, and the moment where the wheels are actually on the black line, can influence the distance calculated by the odometer. Hence, we tried to align the color sensor to the wheelbase as much as possible. In addition, the color sensor is 0.3 cm from the ground (see the second image on the second row in fig. 2). This design decision was made in order to optimize the color sensor's performance. We noticed when the sensor is too far from the ground, the collected data by the color sensor was not consistent and the robot occasionally missed the black lines. When it is too

close from the ground, it reads a constant intensity of 0.1 since the color sensor does not detect any light. Therefore, we decided to fix the color sensor at 0.3 cm from the ground in order to get an optimal difference in intensity when detecting a tile and a black line.

Section 2: Test Data

Odometer test

Table 1: Test results of Odometer

Xf	Yf	X	Y	Euclidean error distance ϵ
1.28	-1.44	0.6	1.9	3.4
0.36	0.14	2.1	1.9	2.5
0.02	1.08	2.6	1.2	2.6
-0.38	0.22	5.0	1.5	5.5
0.12	0.31	3.8	1.8	4.0
0.08	-0.93	0.1	1.8	2.7
0.07	0.26	4.4	2.6	4.9
0.35	0.2	3.5	3.0	4.2
0.09	-1.39	2.4	1.2	3.5
1.07	0.11	2.4	1.1	1.7

Table 2: Mean and standard deviation for X, Y and ε of Odometer test

X Mean	X Standard Deviation	Y Mean	Y Standard Deviation	Error Mean	Error Standard Deviation
0.306	0.477	-0.144	0.782	3.5	1.13

Odometer Correction

Table 3: Test results of Odometer correction

X	**Y**	**X_F**	**Y_F**	**Euclidean error distance ϵ**
12.88	16.5	12.1	17.9	1.60262
11.31	18.85	11.7	21.1	2.28355
12.05	19.85	12.5	19.5	0.57009
12.88	18.33	13.3	18.5	0.4531
11.74	17.71	11.8	19.8	2.09086
11.56	19.33	11.75	20.4	1.08674
10.22	15.31	10.3	19.5	4.19076
13.41	18.67	13.1	19.2	0.614
10.16	18.94	10.1	17.6	1.34134
10.75	17.04	10.6	15.4	1.64685

Table 4: Mean and standard deviation for X, Y and ε of Odometer Correction test

X Mean	X Standard Deviation	Y Mean	Y Standard Deviation	Error Mean	Error Standard Deviation
11.696	1.1289	18.053	1.4071	1.588	1.1105

Formula for the Euclidean distance:

$$E = \sqrt{((X - XF)^2 + (Y - YF)^2)}$$

Section 3: Test Analysis

1. How do the mean and standard deviation change between the design with and without correction? What causes this variation and what does it mean for the designs?

Without correction, both the reprted X and Y mean values, 0.306 cm and -0.144 cm are close to zero since we consider the starting point to be (0,0) in world coordinate system. The reported X and Y data are always close to zero even though the robot's final position is a bit off by the starting point. It indicates that the robot is insentive to oversteer and understeer occured at turns as well as slippery. The error mean of the odometer is relatively larger compared to the error mean in odometer correction. It suggests that odometer with correction gives a more accurate data.

For the odometer design without correction, the standard deviation of the x values is relatively small at 0.477 cm where the standard deviation of the y values is at 0.782 cm. These values are smaller than the standard deviation values from the odometer with correction which are respectively 1,1289 cm and 1,4071. However, the standard variation for both the odometer with correction and without correction is 1,13 cm and 1,1105 which is very close. This means that the distribution of the final positions (x,y) predicted by the odometer without correction is more compact and less dispersed. This means that their values are more similar to each other than the x and y values from the odometer without correction. On the other side, the standard deviation of the final positions (x,y) predicted by the odometer with correction is bigger. This means that the values predicted are more diverse and their distribution is more dispersed.

The fact that the final positions (x,y) predicted by the odometer with corrections have a higher standard deviation means that it predicts a bigger range of values based on the path that the robot travels on every test. In fact, the odometer with corrections reported more accurate values, therefore they are not similar to each other. This explains the fact that they have a higher standard deviation. The fact that the odometer without corrections reported similar values for each iteration means that the odometer is biased (i.e. no matter which path the robots takes, it will always report the same final positions). This demonstrates that the odometer with correction is more accurate than the odometer without correction.

2. Given the design which uses correction, do you expect the error in the X direction or in the Y direction to be smaller?

The error will be smaller in the X direction than in the Y direction, because on the last edge of the square, the correction is made on the x-axis (i.e. each time the light sensor detects a black line, a correction is being made on the x-axis).

Therefore, the error on the x-axis will most likely be smaller than the error in the y-axis. In addition, if on the last turn, the robot did not turn by exactly 90 degrees, its displacement won't be straight and therefore its y-axis will contain error that is not corrected.

Section 4: Observations and Conclusions

Is the error you observed in the odometer, when there is no correction, tolerable for larger distances? What happens if the robot travels 5 times the 3 - by - 3 grid 's distance?

Generally, when there is no correction, the error in odometer increases quickly, and reaches around 3.5cm when it completes a 3 by 3 grid square, it is not a big difference and tolerable for a single lap but for larger distance, the error becomes significant. If the robot travels 5 times the 3 by 3 grid's distance, the error reaches around 16.2cm. The error on every lap is accumulated.

Do you expect the odometer's error to grow linearly with respect to travel Distance? Why?

If the robot is driving in a straight line, then the linear relationship between error and travel distance exists, since slippery leads to an underestimation of the odometer. If the robot is driving in a square, it depends on whether the robot drives in a perfect square. If the robot drives in a perfect square, error caused by slippery canceled out at the opposite sides of the square. If the robot does not drive in perfect square and it oversteers or understeers, then the robot assumes it turns 90 degrees clockwise but in fact, it is not exactly 90 degrees and the errors add up at every corner. Hence, the error grows linearly with respect to the distance traveled.

Section 5: Further Improvements

Propose a means of reducing the slip of the robot's wheels using software.

Reducing the motors' acceleration as well as the motors' speed will help to reduce the slip of the robot's wheels. In fact, the worst slippage period occurs when the robot is making a 90 degrees sharp turn at each one of the 4 corners. When the acceleration is too high, the wheels turn too fast and consequently cause it to slip on the board. In fact, reducing the acceleration will make the wheels spin slower which will make the turn more smooth and therefore reduce significantly the amount of slip on the robot's wheels. In addition, Lejos provides a smoothAcceleration() method which could be implemented instead to slowly accelerate the motors' speed, therefore reducing the risk of slippage on sharp turns.

Propose a means of correcting the angle reported by the odometer using software when:

1. The robot has two light sensors.

When the robot has two light sensors, it is possible to correct the angle by putting one on each side. Therefore, whenever the robot traverses a black line, if both sensors detect the black line at the same time, it means that the robot is travelling straight on one of the edges of the square, hence its orientation is either 0, 90, 180 or 270. Note that for this lab, our theta is calculated with respect to the y-axis, therefore on the first edge, its orientation should be 0, then 90, 180 and 270. In fact, the difference in time that each sensor detects the black line can be used to find the current orientation. If the left sensor detects the black line first, then the sensor on the right, it means that the robot is slightly tilted to the right of the desired angle (i.e. 0, 90, 180 and 270). On the other side, if the sensor on the right detects the black line first, then it means that the robot is tilted counter-clockwise from the desired angle. This tilted angle can be calculated using the following formula:

$$\theta \approx d / WB$$

where WB represents the robot's wheelbase and d is equal to the difference between the distance travelled by the left wheel - the distance travelled by the right wheel when the last sensor (left or right) detects the black line. This angle is then added to the desired angle to determine its current orientation.

2. The robot has only one light sensor.

When the robot has only one sensor, it is more difficult to correct the angle. We know for a fact that the tile size is 30.48 cm. Therefore, the distance between the two black lines is 30.48 cm. We can find the angle of the robot with respect to the desired angle (i.e. 0, 90, 180 and 270) by using the "SOH" formula:

$$\sin\theta = (tile\ size/hypotenuse)$$

Hence, the angle is given by:

$$\Theta = \sin^{-1}(tile\ size/hypotenuse)$$

where the hypotenuse represents the actual distance travelled by the robot from the first black line to the second black line. However, this method doesn't tell if the robot is tilted clockwise or counterclockwise from the desired angle. In fact, it is possible to have a general idea of the robot's orientation from the odometer which is pretty representative.