Design Principles and Methods - Odometry Lab Report

Harley Wiltzer (260690006)
Juliette Regimbal (260657238)

October 6, 2016

1 Objective

To determine the accuracy of the implemented odometry system, and implement a simple correction using a light sensor.

2 Method

1. In the file `Odometer.java`, implement code that performs the task of an odometer as described in the Odometry tutorial and in class. You should only need to add member variables to and modify the `run()` method of the /textttOdometer class.

2. Run the robot in a 3-by-3 tile square (where one tile is 30.48 cm in linear dimension) using the provided code and tweak the `leftRadius`, `rightRadius`, and `width` parameters passed to the `SquareDriver.drive()` method in `Lab2.java` until the robot returns (approximately) to its starting position. If your left and right wheel motors are not connected to motor ports A and B respectively, you may need to also modify those parameters. The call to `SquareDriver.drive()` is shown below.

   ```
   SquareDriver.drive(leftMotor, rightMotor, leftRadius, rightRadius, width);
   ```

3. Now, run the robot in a 3-by-3 tile square ten (10) times and measure the signed x and ydistances from the actual location of the robot on the field (recalling that its starting position isthe origin) to that reported by its odometer. Feel free to calibrate your odometer prior to thisstep. Record these results. (Note: Your odometer is reporting the position of the point directlybetween your wheel axles).

4. In the file `OdometryCorrection.java`, implement code that, using the light sensor,detects grid lines and updates/corrects the odometer's position data as needed. Your robotshould start its motion in the center of a tile, and thus grid lines will occur at x = 15 cm, 45 cm,75 cm, ... and y = 15 cm, 45 cm, 75 cm, ... You do not need to account for grid line intersectionsin this case.

5. Repeat step (3) with your odometry correction enabled. Do not recalibrate your odometer.

6. Demonstrate to a TA your code. The TA will first 'float' your motors (i.e. the motors will be shut off in such a way that they do not provide resistance to being backdriven) and push your robot on the field to confirm that your odometer works correctly. The TA will then start your robot off-center in a tile and look for it to report its correct position after running in a 3-by-3 tile square.

Figure 1: Offset from Origin - Correction Disabled

Trial No.	X (cm)	Y (cm)	θ (rad)	X Error (cm)	Y Error (cm)
1	-0.42	-0.66	0.01	0.13	0.2
2	-0.67	-0.86	0.01	-0.15	0.2
3	-0.84	-0.84	0.01	0.2	-0.05
4	-0.45	-0.46	0.01	-0.05	0.9
5	-0.64	-0.82	0.01	0.8	0.4
6	-0.82	-0.45	0.01	-0.7	-0.4
7	-0.69	-0.66	0.01	0.3	-0.3
8	-0.39	-0.44	0.01	0.1	0.1
9	-0.84	-0.66	0.01	-0.4	-0.7
10	-0.64	-0.63	0.01	0.3	0.1
Standard Deviation	0.17	0.16		0.41	0.45

Figure 2: Offset from Origin - Correction Enabled

Trial No.	X (cm)	Y (cm)	θ (rad)	X Error (cm)	Y Error (cm)
1	-0.04	-0.09	0.01	0.0	-0.1
2	-0.28	-0.18	0.01	0.1	0.1
3	-0.18	-0.19	0.01	-0.1	0.0
4	-0.28	-0.27	0.01	0.1	0.1
5	-0.31	-0.18	0.01	-0.2	0.1
6	-0.19	-0.20	0.01	0.1	0.1
7	-0.27	-0.13	0.01	0.0	0.1
8	-0.18	-0.25	0.01	0.1	-0.2
9	-0.08	-0.12	0.01	-0.1	0.1
10	-0.17	-0.11	0.01	0.0	-0.05
Standard Deviation	0.09	0.06		0.105	0.109

3 Data

4 Data Analysis

4.1 What was the standard deviation of the results without correction? Did it decrease when the correction was introduced? Explain why/why not.

$$\text{Standard Deviation} = \sqrt{\frac{\sum_{k=1}^{10}(\Delta P_k)^2}{9}} \tag{1}$$

The standard deviation was calculated by formula (1), where ΔP_k is the error in odometer measurement of the k^{th} trial in the P axis. The standard deviation of the data corresponding to the robot's distance from the origin without any odometry correction was 0.17cm on the X axis and 0.16cm on the Y axis. Furthermore, the standard deviation in the *error* in measurements on the x and y axes were 0.41cm and 0.45cm, respectively. The standard deviation did decrease significantly with the odometry correction enabled, as the standard deviation of the X position was measured to be 0.09cm, and only 0.06cm for

the Y position, and 0.105cm and 0.109cm for their respective errors. This makes sense, as the odometer reading was modified when the robot passed a grid line. The grid lines are a known, fixed distance away from each other, thus change in position can be accurately measured (in one spatial axis at a time) by detecting the grid lines. Therefore, with the odometry correction, no matter how much the wheels spin the robot could more accurately report its position as it crosses the gridline, reducing the effect of wheel spin on the overall odometer reading. This allows for more precise and accurate measurements.

4.2 With correction, do you expect that the error in the x position or the y position will be smaller? Explain.

With the odometry correction, the errors in the x position and y position should be smaller, as described in **4.1**. Considering the design implemented in this lab, there is no reason why one direction would see more error than the other, as the motion is perfectly symmetric. To avoid direction bias and compromised corrections to the odometer readings, the `OdometryCorrection` implementation does not tamper with odometer data at the first line crossing after each corner. Unmeasurable slipping occurs when turning corners, thus there is no guarantee in displacement when the robot turns a corner and crosses a line. Therefore, `OdometryCorrection` simply saves the current odometer readings at the first line crossing after a corner, and offsets those readings at subsequent line crossings. This way, there is no reason why either the x position or y position should have smaller error.

5 Observations and Conclusion

The error observed would not be tolerable should the robot travel greater distances. Just observing the robot making a few laps around the track as is, it is clear that the error in the odometry readings increases with each successive lap. The uncorrected odometer readings get their main source of error from wheel slippage. As the path the robot must travel increases in distance, opportunities for wheel slippage continue to come and increase the error in odometry. However if the nature of the path and robot's motion remains the same - that is to say the robot isn't made to quickly accelerate and the path the robot drives on keeps the same physical properties - the frequency of slippage f remains the same. The longer path simply increases the amount of time necessary to complete the task, resulting in a larger number of slips proportional to time. So error in odometry under these conditions should grow linearly with respect to distance. Therefore, if the robot were to travel at a distance of five times the length that was tested, the error of the odometer will be approximately five times greater, which would not be tolerable.

6 Further Improvements

6.1 Propose a means of, in software, reducing the slip of the robot's wheels (do not provide code).

To reduce the slip of the robot's wheels in software, the acceleration of the motors can be decreased. When the angular acceleration of a wheel is too high, slipping occurs. The motors of the robot accelerate when the robot starts moving and when it turns corners. Thus, by reducing the acceleration of the motors, the amount of slipping can be drastically reduced.

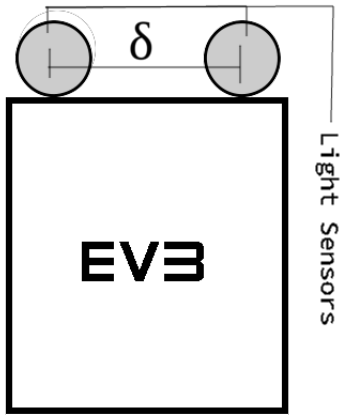
6.2 Propose a means of, in software, correcting the angle reported by the odometer, when (do not provide code):

6.2.1 The robot has two light sensors.

Say the robot has two light sensors at its front, one on each side. When one sensor detects a grid line, the current odometer reading of position is stored. Then, when the other sensor detects a grid line, the

new position readings are compared to the saved ones to determine the change in tachometer readings of the left and right motors. With this data, the angle can be calculated. For the sake of simplicity in the

Figure 3: Configuration of light sensors on the robot



following example, assume the robot is travelling approximately in the positive y direction. To correct the heading, the robot can start by storing its ideal heading $\theta = k\frac{\pi}{2},\ k \in \mathbb{N}$ where k depends on the amount of lines already crossed. In this example, since the robot travels in the positive y direction, $\theta = \frac{\pi}{2}$ radians. When one sensor crosses a grid line, the robot's position (x_0, y_0) is stored. Next, when the other light sensor crosses a grid line, the robot's position (x_1, y_1) is stored. Now, the points $(x_1, y_1), (x_0, y_0)$ depict the linear trajectory of the robot as it crosses the grid line. With this data, the corrected heading Θ can be calculated as $\Theta = \theta - \arctan(\frac{x_1 - x_0}{y_1 - y_0})$, again assuming the robot travels along the y axis. Similar logic follows for other orientations of the robot.

6.2.2 The robot has only one light sensor.

With one light sensor, the angle of the robot can be corrected as well. At each line crossing, the odometer reading of the robot can be stored. For the purpose of explanation, assume the robot is moving in the positive y direction. Now, between two line crossings, it is guaranteed that the y position of the robot has increased by 30.48cm. The difference in x positions can be measured then, and the angle Θ can be calculated as $\Theta = \theta - \arctan\left(\frac{\Delta x}{30.48}\right)$, where θ represents the same ideal heading as described in **6.2.1**, and Δx represents the change in x position in this example. Once again, similar logic follows for other orientations of the robot. Note that in this correction scheme, the heading is corrected using data acquired from two separate line crossings, whereas the scheme proposed in **6.2.1** makes a correction using data acquired throughout one line crossing. The robot travels a smaller distance between measurements in the scheme presented in **6.2.1**, therefore there is less error introduced into the algorithm, so it will likely produce a more accurate correction than the algorithm described here in **6.2.2**.