Design Principles and Methods - Localization Lab Report

Harley Wiltzer (260690006)
Juliette Regimbal (260657238)

October 20, 2016

1 Objective

To use the ultrasonic sensor and light sensor to accurately navigate the robot to a known (initial) position and orientation on the field.

2 Method

1. Put you odometer and navigation classes created in Labs 2 and 3 in the Odometer.java and Navigation.java files. Working odometer and navigation classes are also provided in case you had failed to complete either lab.

2. Fill in the code for the class called `USLocalizer`. This class actually contains two different localization routines, each of which can be implemented and tested separately. You may need to create extra functions in your Navigation in order to move as you require.

3. Test each localization routine ten times using random starting orientations (but the same starting position, notably in the corner square) and record the error in the final orientation of the robot. Compute the mean and standard deviation for each routine.

4. Based on the standard deviations from (2), determine the best ultrasonic sensor-based localization method for your robot. Use this one for the rest of the lab, but do not remove the code for the other, as you will need to submit it. Also, correct the appropriate constant in your code to make the mean error 0. You should not need to do any additional tests to confirm that your correction in fact made it 0.

5. Fill in the code for the class called `LightLocalizer`. You need not test the accuracy of this part of the localization. You'll require some trigonometric equations as outlined in the Localization tutorial. Follow the process that the tutorial demonstrates. Also when you've found (0, 0) and 0°, travel to (0, 0) and turn to 0°.

6. Demonstrate to a TA the correct operation of your robot's localization. The TA will choose the starting orientation of your robot. As can be inferred from the comments in the provided code, your robot should: a) use the ultrasonic sensor and the routine you developed and tested in (1), (2), and (3) to find and rotate to an approximatation of 0°, b) drive to the point specified in the Localization tutorial. Begin rotating and clocking angles. c) Compute the trigonometric values for the robot's heading and the (0, 0) point, and d) Travel to (0, 0, 0).

3 Data

Figure 1: Offset from 0° - Falling and Rising Edge

Trial No.	Falling Edge (°)	Rising Edge (°)
1	9.6	3.8
2	6.5	3.0
3	6.1	-1.2
4	14.7	-1.5
5	4.6	-1.9
6	5.7	0.4
7	5.0	-1.9
8	3.8	-16.7
9	2.7	1.5
10	15.5	-1.2
Mean	7.4	-1.6
Standard Deviation	4.4	5.7

4 Error Calculations

$$\text{Mean} = \frac{1}{n}\sum_{k=1}^{n}\mathcal{E}_k \tag{1}$$

$$\text{Standard Deviation} = \sqrt{\frac{\sum_{k=1}^{n}(\mathcal{E}_k - \mu_\mathcal{E})^2}{n-1}} \tag{2}$$

4.1 Falling Edge

Using formula (1), the mean error of the falling edge localization routine will be calculated.

$$\begin{aligned}\mu_F &= \frac{9.6 + 6.5 + 6.1 + 14.7 + 4.6 + 5.7 + 5.0 + 3.8 + 2.7 + 15.5}{10} \\ &= 7.4^\circ\end{aligned}$$

Next, using formula (2), the standard deviation of these measurements will be calcuated.

$$\begin{aligned}\sigma_F &= \sqrt{\frac{(9.6-7.4)^2 + (6.5-7.4)^2 + (6.1-7.4)^2 + (14.7-7.4)^2 + (4.6-7.4)^2 + (5.7-7.4)^2 + (5.0-7.4)^2 + (3.8-7.4)^2 + (3.8-7.4)^2 + (2.7-7.4)^2 + (15.5-7.4)^2}{9}} \\ &= 4.4^\circ\end{aligned}$$

4.2 Rising Edge

Using formula (1), the mean error of the falling edge localization routine will be calculated.

$$\begin{aligned}\mu_R &= \frac{3.8 + 3.0 - 1.2 - 1.5 - 1.9 + 0.4 - 1.9 - 16.7 + 1.5 - 1.2}{10} \\ &= -1.6^\circ\end{aligned}$$

Next, using formula (2), the standard deviation of these measurements will be calcuated.

$$\begin{aligned}\sigma_R &= \sqrt{\frac{(3.8+1.6)^2 + (3.0+1.6)^2 + (-1.2+1.6)^2 + (-1.5+1.6)^2 + (-1.9+1.6)^2 + (0.4+1.6)^2 + (-1.9+1.6)^2 + (-16.7+1.6)^2 + (1.5+1.6)^2 + (-1.2+1.6)^2}{9}} \\ &= 4.4^\circ\end{aligned}$$

5 Observations and Conclusion

5.1 Which of the two localization routines performed the best? Which performed the worst? What factors contributed to the performance (or lack thereof) of each method?

The falling edge localization routine, before any adjustments were made, had a higher mean error than the rising edge localization, meaning the falling edge routine was less accurate. However, the standard deviation of the falling edge routine was lower than that of the rising edge routine, meaning the falling edge localization was more precise. Because of this low standard deviation, the falling edge localization was further corrected to rotate by the mean error once the initial localization was complete, so as to reduce the mean to approximately 0 degrees. This was reliable since the falling edge routine was more consistent, so the error in each trial was closer to the mean, and it can be expected then that this extra rotation will consistently localize the robot with minimal error.

TODO: factors affecting performance

5.2 Why does the light sensor provide a more accurate means of localization than the ultrasonic sensor?

5.3 Propose a means of determining (approximately) the initial position of the robot using the ultrasonic sensor. Why is detecting minima with the ultrasonic sensor problematic?

6 Further Improvements

6.1 Propose a way to avoid small errors more accurately than a clipping filter.

To avoid small errors more acurately, there are several methods that can be used as opposed to a clipping filter. Notably, one may implement a *median filter*, which is very effective at eliminating the effects of noise for a signal from an ultrasonic sensor. It works as follows: chose some *window* size $\aleph$. Every time new data is fetched, the median of the last $\aleph$ readings is calculated. If the newest data is greater than this median, it is replaced by the median.

6.2 Propose a sensor design that would result in a more accurate and reliable reading than an ultrasonic sensor.

6.3 Propose another form of localization other than rising edge and falling edge.

Instead of detecting either rising edges or falling edges, the robot can detect one of each. For example, the robot could choose one direction to rotate in, detect a large enough change in distance (this can represent a rising edge *or* a falling edge), and continue rotating in that direction (as opposed to changing directions in the rising and falling edge methods). At this point, the robot will eventually detect another large change in distance (corresponding to a falling edge or a rising edge) and make trigonometric calculations based on the two *edges* that it detects. The benefit of this routine is that the robot never needs to switch direction, so there is no acceleration required to make that switch, effectively reducing the amount of wheel slippage.