Lab 4: Localization

Background

In Lab 1, it was shown how to use the ultrasonic sensor for polling that allowed for wall following. This week, it will be more convenient to deal with the ultrasonic sensor in a simpler manner. The `USLocalizer` class has a set of methods you'll need to complete to get the localization process working, one of which is a filter for the ultrasonic sensor:

```
private int getFilteredData() {
	int distance;

	// do a ping
	us.ping();

	// N.B.: There will be a delay here.
	distance = us.getDistance();

	// filter out large values
	if (distance > 50)
		distance = 50;

	return distance;
}
```

The filter shown above as an example 'clips' the signal from the ultrasonic sensor at 50 and reports '255' instead. You will probably want to create a filter that removes spurious '255' values from those reported. This should be similar to what many of you did in Lab 1.

Once the robot knows approximately where 'North' is, it should then travel forward and use the grid lines and its light sensor to get better aligned. As a footnote, you should use the design titled "**Stronger with US and Light**", posted on WebCT, for this lab. You may use whatever you wish for a caster, if you find that the standard model is problematic. The SeanBot's caster may be a good example of an alternate.

If you don't wish to use the code provided, you are free to implement your own Odometer class (as well as removing TwoWheeledRobot from the provided code), however you must adhere to the format of the USLocalizer class insofar as to the two types of localization: Falling-edge and Rising-edge.

Objective

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

Method

1. Put your navigation classes created in Lab 3 in the `Navigation.java` file. NB. You'll need to change your calls to `Motor.*.*` to `robot.setForwardSpeed(val)` and r`obot.setRotationSpeed(val)`.

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 (i.e. a goForward(double distance) method ***might*** be useful).

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. Note that the provided `Odometer` assumes that 0 degrees is along the positive y-axis, and that angles increase in a clockwise direction (like a compass).

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 two, 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 degrees, travel to (0,0) and turn to 0 degrees.

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 approximation of “0 degrees” (a.k.a. the positive y-axis).

    b. Drive to the point specified in the Localization tutorial. Begin rotating and clocking angles.

    c. Computer the trigonometric values for the robot's heading and the (0,0) point.

    d. Travel to (0,0,0).

Data

Ten results, plus the mean and standard deviation, from each of the three ‘flavors’ of ultrasonic sensor localization.

Observations and Conclusions

1. Which of the two localization routines performed the best? Which performed the worst? What factors do you think contributed to the performance (or lack thereof) of each method?

2. Why does the light sensor provide a more accurate means of orienting the robot than the ultrasonic sensor?

3. Propose a means of determining (approximately) the initial *position* of the robot using the ultrasonic sensor (Hint: Consider the minima of the ultrasonic sensor’s readings as the robot rotates). Why is detecting minima with the ultrasonic sensor problematic?

Error Calculations

1. Your error calculations should be the mean and standard calculations that were computed in the *Method* section for each of the two trials

2. All work should be shown. You may wish to use OpenOffice.org Math or Microsoft Word's Math tools for this. You may scan in the work and attach it into the report if you wish.

Future Improvements

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

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

3. Propose another form of localization than rising-edge or falling-edge.

To Sumbit

- One document in .pdf format containing the lab report (if scanned work is included, you may either attach the scan and zip the file, or put the scans into the document.)

- NOTE: Lab reports should be kept concise, and to the point. Also no cover page is needed when submitting the weekly lab reports.