Lab 1: Wall Following

Background: Events

For the purpose of making a wall following robot, it is necessary to develop a good means of reading the ultrasonic sensor. The LEGO Mindstorms ultrasonic sensor is capable of getting data at around 25 times per second, however due to the fact that we won't be using timers, we will poll as fast as the ultrasonic will allow in a thread. After each polling, the `UltrasonicPoller` will update the value of the distance integer in the controller for which is being processed at that time.

In Lab1.java we create all the objects that we will need, which includes the two controller types to be implemented. Based on whether the left or right button on the NXT is pressed, it will choose the controller to be run accordingly.

The `UltrasonicPoller` object (called `usPoller`) is what gets the data from the ultrasonic sensor, and gives it to any object of a class that implements the `UltrasonicController` interface. Its constructor thus takes as arguments an `UltrasonicSensor` object (which is part of the leJOS API), and any object implementing the `UltrasonicController` interface. In the `UltrasonicPoller.java` file provided, we see on line 16 how it gets the ultrasonic data, then passes it off to the `UltrasonicController, cont`:

```
public void run() {
      while (true) {
            //process collected data
            cont.processUSData(us.getDistance());
      }
}
```

This is where any filters for the ultrasonic sensor should be implemented, such as the removal of spurious 255 values.

The final element is the `processUSData(int distance)` method in `Pcontroller` and `BangBangController`.

```
public void processUSData(int distance) {
	this.distance = distance;
	// TODO: process a movement based on the us distance passed in (BANG-
	BANG style)
	// wall on the right
}
```

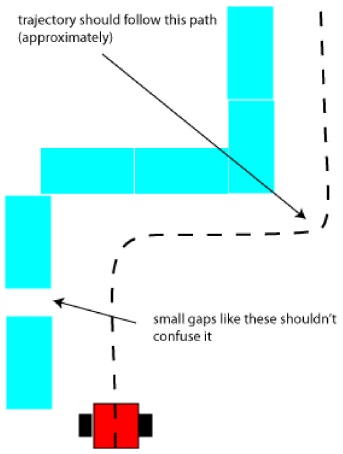
Objective

To navigate around a sequence of cinderblocks, making up a ‘wall’ containing gaps and both concave and convex corners, without touching it or deviating too far from it.

Method

See the attached code for a working example of how to do wall following. Modify it to:

- Avoid getting confused by gaps
- Turn concave corners (i.e. corners which the robot would run into were it to travel in a straight line)
- Turn convex corners sharper
- You need to implement both the bang-bang and P-type controllers, each is contained in their own .java files.



Build the “Stronger with Rotating US” robot, whose LXF file is provided on WebCT. You may modify the ultrasonic sensor mount as you please.

Data

All data for this lab is qualitative.

Data Analysis

a) Did the bang-bang controller keep the robot at a distance `bandCentre` from the wall? Why is it expected that the robot will repeatedly oscillate from one side of the band to the other with the bang-bang and p-type controllers?

Observations and Conclusion

What errors did the ultrasonic experience? Were these errors filterable? Does the ultrasonic sensor produce false positives (i.e. the detection of non-existent object), false negatives (i.e. the failure to detect objects), or both?

Further Improvements

What improvements could you make to both the physical or software designs to improve performance of the wall follower? (At least 3 are needed) Are there any other controller types that may have better outcomes than the bang-bang and p-type?

To Sumbit

- One document in .pdf format containing the lab report