Coursework 1: Wall Following with a LEGO Robot

Simon Buist

1 Introduction

My lab partner and I have constructed a reactive robot that follows walls.

We did this to find out, experimentally, how effective reactive robots could be at achieving some basic function: wall-following in this case, *without representation*. The motivator for this project was Joanna Bryson's *Intelligent Control and Cognitive Systems* course and research papers discovered therein.

2 Approach

Ryan Hancock and I built the robot using the 9797 LEGO® Mindstorms Education Base set with 9841 NXT Intelligent Brick running leJOS 0.9.1beta-3. The physical design of the robot – the *morphology* – features the following inputs: 1 x 9846 Ultrasonic sensor, 1 x 9843 Touch sensor, and the following outputs: 2 x 9842 Interactive Servo motor. See Appendix for a photo of the robot, code of the core wall-following logic, and a finite state machine diagram. We shared the task of building the robot, co-operating on the design, and also settled on a common algorithm for wall-following. The algorithm is quite simple, and does not implement the explicit subsumption arbitrator architecture that others chose to use. It implicitly implements subsumption: `driveForward()` and `steerWhenWallSensed()` are non-blocking, so at any time when the touch sensors are pressed, they can be interrupted, and if this occurs, the blocking `backOffWhenBumpersPressed()` behaviour takes over and 'suppresses' the others until it has completed. It is sufficient to follow walls, avoid obstacles, and back off to recover from collisions.

2.1 Observations

I observed that the robot's circuit time around the LEGO® 4-box circuit, arranged in a + formation, of total perimeter length **320cm**, was **35s** at `ROBOT_SPEED=20;`.

It seemed to go faster when this was changed to be `21`. I would like to get better performance out of the robot and so would like to try varying the speed constant to find the shortest circuit time around the same obstacles.

I would like to capture what it is that is making it lap faster. Is it purely the speed or does the position of the ultrasonic sensor affect it also?

2.2 Hypothesis

1. Increasing `ROBOT_SPEED` by a factor of 2 (`ROBOT_SPEED=40;`) will result in half the circuit time.
2. The closer the ultrasonic sensor is to the center of the robot, the shorter the circuit time.

2.3 Experiment

We will run two parts for the two hypotheses. Each aim to optimise for fastest circuit speed. In order to repeat the experiment, note the following details – **NB**: The environment is as described in the obervation. Furthermore, the specific room in which the experiment is to be conducted: East Building lab 0.8 at The University of Bath, England. We will ensure no other robots are in operation during. **Before each lap**: We will position the rear of robot **4cm** away and flush with the shortest edge of the + formation, on the RHS. The rear wheel will be reset to the same orientation such that it is as far away from the body of the robot as possible. The boxes will be weighed down such that the force of the robot is not enough to move them.

See Method in the Appendix for a detailed description of the experiment.

3 Results

See Speed and Sonar tables in Appendix

4 Discussion

I learned that a great amount of the robot's intelligence is dictated by the morphology. I learned this experientially with this project, and also in reading about Shakey (Peter Hart 1975) and Polly (Horswill 1993). In the case of these robots, they were both taller than they were wide, and had limits on what could be detected due to the range finder beam reflecting away on shiny surfaces, or the bump sensor not bumping into something due to the obstacle being too

high. Shakey dealt with these limitations by having a highly pre-programmed model of the world, and the learning algorithm STRIPS. Polly dealt with it's low-powered visual regognition hardware by sampling regularly (60Hz) and using the assumption that the lower portion of it's vision is always the floor (as chosen by a benevolent human operator). I learned that in order to make practical robots, you have to do a bit of hacking. It's not all clean and easy – sensors have noise, motors slip, and bump sensors don't trigger at low speed. And sometimes that's okay – much like the simple intelligence of a fly that evolved to cope with natural environments – when you change it's environment to be a man-made kitchen, it's behaviour is good enough. It may not get out of the window, bumping into it the first time, the second time, even 100 times later, but the 101st time it will, and it's robustness, as (Brooks 1991) mentions, allows it to survive.

Also see Lessons Learned in Appendix.

5 Conclusion

The robot we designed is capable of robustly navigating a natural environment using a threshold of ultrasound distance to guide the robot motion towards/away from walls. The results show that increasing the `ROBOT_SPEED` does make for a quicker circuit time, and the quickest circuit time with a fixed `ROBOT_SPEED=20;` is with the ultrasonic sensor positioned 2 blocks from the back of the robot body.

6 Appendix

6.1 PHOTO: LEGO Robot

6.2 Finite State Machine diagram

6.3 CODE: Core wall-following logic

```
private static void mainLoop() {
    initialiseMotorsAndSensors();
    while(true) {
        driveForward();
        steerWhenWallSensed();
        backOffWhenBumpersPressed();
    }
}

// (NON-BLOCKING)
```

Figure 1: Front and Side-view of Lego Mindstorms NXJ robot showing ultrasonic sensor and bump sensors

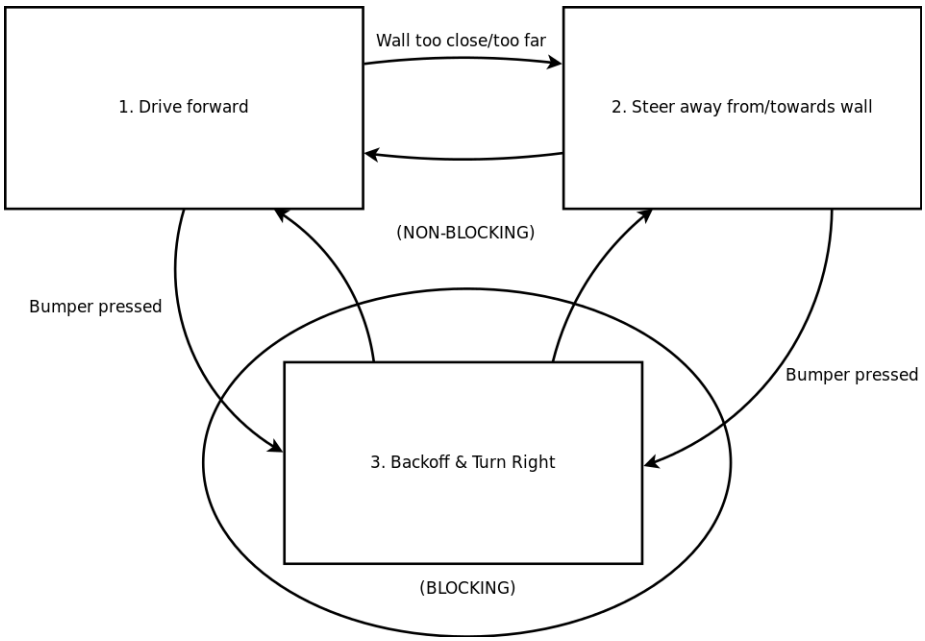

Figure 2: Flow diagram of robot finite state machine

```
private static void steerWhenWallSensed() {
    int sonarDistance = sonar.getDistance();
    if (sonarDistance < TOO_CLOSE) {
        steer(LARGE_ROTATION);
    } else if (sonarDistance < CLOSE) {
        steer(SMALL_ROTATION);
    } else if (sonarDistance > TOO_FAR) {
        steer(-LARGE_ROTATION);
    } else if (sonarDistance > FAR) {
        steer(-SMALL_ROTATION);
    }
}

// (BLOCKING)
private static void backOffWhenBumpersPressed() {
    if (leftBumpSensor.isPressed() ||
            rightBumpSensor.isPressed()) {
        backOff();
        turnRight();
    }
}
```

6.4 Speed Table

ROBOT_SPEED	Total time (s)	Avg. time (s)
20	1:45	0.35
23	1:33	0.31
27	1:26	0.29
32	1:24	0.28

Table 1: Table showing `ROBOT_SPEED` set in the code, and the resulting circuit times (total and average) over 3 laps

Multiplying the speed by 1.6 (20 x 1.6 = 32) did not result in a time smaller by a factor of 1.6 (this would be a time of 0.21). Instead the time at `ROBOT_SPEED=32;` resulted in a time of 0.28. (Using: Distance = Speed x Time)

6.5 Sonar Table

Sonar position	Avg. time (s)
2	31.6666
4	33.6666
6	39.0
8	39.6666

Table 2: Table showing the sonar position (in increments of 2 LEGO squares at a time) and the resulting average circuit time over 3 laps

Contrary to our hypothesis, the further from the rear we got, the *greater* the circuit time.

6.6 Method

1. **Time**: 2:45pm. **Lighting**: Strong daylight from windows, artificial halogen lighting inside the lab. **Method**: We positioned the robot as described above, and I coodinated with my lab partner to press `Run Default` when I said "GO" to start the robot after boot up. I started my stopwatch on "GO", and observed the robot navigate the LEGO® boxes. Speeds above `32` caused the robot to drive so fast it fell over when it clipped the edge of the boxes, so we stopped measuring after confirming this with 2 more trials.

2. **Time**: 5:24pm. **Lighting**: Weak daylight from windows, same artificial halogen lighting inside the lab. **Method**: Same orientation, but we began by positioning the ultrasonic sensor at the very rear of the robot and winched it forward in increments of **2** blocks. The furthest measurement (effectively the ultrasonic sensor was positioned to be in parallel with the bump sensors) was very slow as it clipped the corner of every edge in the circuit.

6.7 Lessons Learned

We had some problems with the bumper design – it would not trigger when the robot approached a wall at angles less than approximately 45 degrees. Designs we tried included: 1) A loose bumper per touch sensor, spring-loaded using rubber bands to provide tension 2) A fixed bumper fit across both left and right touch sensors, that attached directly to the bumpers via the slot in the touch trigger 3) An over-hanging cross-bar attached to the body of the robot, with the cross-bar (that spanned both sensors) resting on the touch triggers. We considered increasing the speed of the robot so that it would drive into the walls

with enough force to trigger the sensors, but the third configuration provided a much more sensitive trigger that activated at our original `ROBOT_SPEED=20;`, so we opted for that. In addition to this, the design of the rear wheel meant that the robot often veered off in unpredictable directions unless we were very precise about it's starting orientation. For this study, I believe we were not precise enough and/or we should have used a different rear wheel design that could be controlled more reliably. Perhaps a friction ball design like the old computer mouse balls.

In exploring potential parameters to change, I coded a Monte Carlo localization algorithm which uses a 1-dimensional representation of the world where the robot drives in a straight line, in order to measure the accuracy of the ultrasonic sensor, and motors, testing the error incurred in robot motion. My motivation for exploring this behaviour was in reading (Thrun 2014). I have programmed it to go forwards for 4 wheel rotations, then backwards 4, then forwards 4, etc, an EVEN number of times. A kind of robot waltz. If the robot were perfectly accurate, it would end up where it started, but due to the way the robot is built and error incurred in sensor readings, it ends up quite far away. I probably need to use an accelerometer to have another measure of how much the robot is moving through space. So in order to actually update an internal representation, the uncertainty of the robot motion and sensing must be determined experimentally and factored in to the calculations.

7 References

Brooks, Rodney. 1991. "Intelligence Without Representation." *Artificial Intelligence* 47: 139–159.

Horswill, Ian. 1993. "Polly: a Vision-Based Artificial Agent." In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, 824–829. AAAI Press.

Peter Hart, Nils Nilsson. 1975. "SHAKEY: an Experiment in Robot Planning and Learning." USA.

Thrun, Sebastian. 2014. "Artificial Intelligence for Robotics." *CS373 Unit 1 - The Problem of Localization* (February). https://www.udacity.com/wiki/cs373/unit_1#robot-movement.