Robot Code Documentation for Project 2
March 31, 2003
Team 4 – Justin Fuller, Rahul Kotamaraju, Matthew Lawrence

1.0 Overview

The software used for the second project was designed to complement the robot's sensor array and accomplish the tasks necessary for the assignment. While the architecture used does not exactly fit any of those discussed in class, it most closely resembles a reactive paradigm with subsumption. The software consists of four distinct behaviours arranged in a hierarchy such that any single behaviour can override any lower-level behaviours. These behaviours were designed to search for the target light bulb, advance toward the target light bulb, avoid large buckets, and recover from contact with obstacles. Additionally, the software includes an arbitration routine to determine which behaviour will be allowed to control the robot and a motor control routine to execute the appropriate actions.

2.0 Behaviours

The software for the robot implements four behaviours, with each behaviour intended to solve a unique problem encountered by the robot. The first problem encountered by the robot is how to behave in the absence of sensory input; this is solved by the random walk. Next the robot must be able to follow the light once it has been located – a problem solved by the goal seeking behaviour. Another problem faced by the robot is how to navigate around immovable obstacles; this can be assuaged by slip recovery. Finally, the robot must never come in contact with any of the buckets; this can be guaranteed by the bucket avoidance behaviour. These four behaviours are threaded and thus are evaluated in parallel, so an arbitration scheme is included to determine which behaviour will directly control the robot, and which will be overridden.

2.1 Random Walk

The random walk behaviour is designed to allow the robot to initially locate the target light bulb. Under this behaviour, the robot cruises around the arena by driving forward for a random amount of time, then spinning in place for a random time. This activity allows the robot to effectively cover the entire arena in search of the currently lit bulb. Once the target has been acquired, the goal seeking behaviour will override the random walk.

2.2 Goal Seeking

The goal seeking behaviour allows the robot to hone in on the target light bulb. This behaviour receives input by continually polling the four light sensors on the front of the robot. Two conditions must be met before the robot can begin to seek the goal. First, at least one of the sensors must yield a reading that indicates the presence of light greater than ambient light. Second, there must be sufficient difference among the four sensors to indicate that the light being detected is not ambient. Anytime these two conditions are simultaneously met, the goal seeking behaviour can become active. Under these conditions, the behaviour determines which light sensor perceives the greatest amount of light. It then suggests that the robot move in the corresponding direction - either a left arc, a right arc, or straight ahead. This behaviour will remain active as long as the robot senses the light, unless it is subsumed by one of the higher-level actions.

2.3 Slip Recovery

The slip recovery behaviour is designed to detect when the robot is stuck and to respond accordingly. The robot is said to be stuck when its powered wheels are driving in a forward direction and yet the passive wheels are not turning. This condition is determined by a global variable indicating the action (or inaction) or the powered wheels along with sensor input from an encoder associated with a passive wheel. At regular intervals, the robot polls the encoder to determine the amount of movement by the passive wheel. If this value is below a given threshold, the robot is clearly not moving forward. Then, if the global variable indicates that the powered wheels are driving, the slip recovery routine becomes active. In this case, the behaviour checks to see whether the robot has already become stuck in recent history. If it has not, the behaviour suggests that the robot spin in place for a fixed amount of time. If the robot has recently been stuck, the behaviour reasons that the previous recovery was insufficient and suggests that the robot arc backwards away from the obstacle. This behaviour is sufficient for maneuvering around some obstacles (lamps, rocks, and boundaries) but is insufficient for avoiding buckets since it requires physical contact with the obstacle in question.

2.4 Bucket Avoidance

The bucket avoidance behaviour prevents the robot from ever coming into contact with the buckets scattered throughout the arena. Because avoiding the buckets is the top priority for the robot, this is the highest-level behaviour. This behaviour receives sensory input from the two optical rangefinders mounted high on the front of the robot. Readings from each sensor are thresholded to determine whether the sensor is sufficiently close to a bucket. The behaviour will become active if either one of the sensors indicates the nearness of a bucket obstacle. The behaviour responds by suggesting that the robot spin in place, thus orienting it away from the obstacle.

2.5 Arbitration

Arbitration among the four behaviours is very straightforward. The robot evaluates the behaviours from high to low, seeking out the highest-level behaviour which is currently enabled. The robot then takes the suggested response from that behaviour and assigns it to the motor control command. The arbitration routine runs continually, so the motor command is always dictated by the appropriate behaviour and its suggested action.

3.0 Motor Control

Motor control for the robot is implemented in a manner consistent with the hardware's differential drive mechanism. The robot's left and right motors are independently powered according to the dictates of the motor command as assigned by the arbitration routine. By driving both motors forward at the same power, the robot can be made to drive straight ahead. If the motors are given differential powers, the robot will move forward in an arcing pattern. By driving one motor forward and the other backward, the motor control routine can direct the robot to spin in place. Finally, motor command may disable one motor and drive the other in reverse, causing the robot to move backwards in an arcing fashion. The motor command routine executes independently of any sensor information or behaviours – with one exception. When the goal seeking behaviour is dominant it is necessary to drive the robot more slowly so that it does not overshoot the target. For this reason, the motor command will disable both motors on every third iteration when the goal seeking behaviour is dominant.