CS26410 - Assignment 3
Robotic Hide and Seek

Samuel Jackson
slj11@aber.ac.uk

April 17, 2013

1 Introduction

Assignment three for CS26410 required us to develop a collection of robotic controllers for use with the Pioneer robots in the ISL which could be used to play a game of hide and seek. This document details the solution that I have provided along with a discussion of its performance using trial runs on real robots.

2 Discussion of the Algorithms Used

In the development of this assignment I have used several different techniques and algorithms to solve the problems outlined in the assignment brief. For each of the major problems to be solved in the brief I have supplied a description of my approach to finding a solution.

2.1 Mapping Techniques

The first and most important component of the solution was to be able to create a map of the robots environment. The easiest approach to create a simple map of the environment was to create an occupancy gird of the world split into 60x60cm cells in which the robot could fit. As the robot moves through the world, it checks the readings from its sonar array and updates the likelihood that a cell is a wall using the range measured from the sensors.

In the previous assignment this mapping was done using a purely random walk. While this approach worked well, it had the major drawback of not having a way to finish mapping unless a human actively told it to stop. In this assignment I decided to take a more structured approach to mapping the environment by utilising depth first and A* search techniques.

Using the new mapping algorithm the robot moves 60cm at a time from one cell to the next while still reading input for its sensors and updating the occupancy grid. When the robot moves into a new cell it checks each of its neighbours to see if they have been discovered or visited. If neither of these conditions are true the robot will add the cell to its "frontier" to come back and look at later. The robot then attempts to keep moving forwards unless there is an obstacle in the way. If there is an obstacle in the way then it will attempt to move into cells on the left or right of it. If the robot hits a dead end it will take the next cell off the frontier and perform an A* search to find the quickest path from the current cell back to the next unexplored cell. The mapping algorithm with continue in this way until there are no more unexplored cells and hence the map is complete.

This technique proved to be fairly suitable to the problem as it ensured that, presuming the world is closed, the robot would eventually map all of it. To ensure that the same parts of the map would not be remapped I turned off the map updating function when the robot was backtracking. While I realise that the use of the A* function is not

strictly necessary as backtracking could be implemented with a simple depth-first search, it speeds up map creation considerably by not having to move back through every single previous cell, but instead taking the shortest route to the next unvisited cell.

2.2 Localization

2.3 Finding a Hiding Spot

2.4 Finding Map Differences

3 Trail Runs

4 Discussion of Trial Runs

5 Evaluation of Project