Milestone 2 Individual Performance Report

Introduction

Provided an adaptable implementation of the A* path finding algorithm, to be used when dynamically creating plans. Continued to contribute to construction, providing quick fixes and analysis of the performance of different aspects of the robot.

Path Finding

My initial approach to this problem was to look at the planning code from last year and see how other groups had implemented path-finding algorithms. The majority of the groups used A* path finding and reported considerable success from this, so I decided this would be a good basis from which to build on. I found a number of Java implementations of A* online and analysed them, finally deciding to use the one which was the best documented and easy to comprehend. I then began adapting this to our environment.

We have decided that our planning class will be continuously creating plans within a thread, and to this end the path finding algorithm should not be the bottleneck in performance. For this reason my implementation of A* allows for adaptable sized grids, allowing for a huge potential reduction in computation time when working with a smaller resolution. I started creating paths using the same resolution that the vision system was receiving from the camera, around 600 x 300 (pixels). After numerous testing with this (at least 20 trials), there seemed to be an upper bound of around 450ms to create a path. This is nearly half a second, which is too long for our purposes. I then reduced the resolution so that one grid square was equal to the width of a golf ball, this equated to a 58 x 29 grid. The upper bound after testing was now around 90ms, which is an 80% reduction in processing time. The grid size will remain flexible but it looks likely to remain at this size.

I have massively improved the ASCII printing of the paths to the console; it is now far easier to read and should prove very valuable in testing. As A* is limited by the fact that each node can only look at its nearest neighbours, meaning it can only travel in one of 8 directions, I have looked into an algorithm called Theta *. This allows for any angle planning using a greater look ahead and could prove to be a very effective solution. I will implement this sometime in the next week. There are also a number of methods to smooth the paths produced by A* into more realistic arcs, which is something David is looking into.

```
************************************************************
************************************************************
************************************************************
************************************************************
************************************************************
************************************************************
************************************************************
************************************************************
************************************************************
************************************************************
************************************************************
************************************************************
****************************B*******************************
****************************X*******************************
****************************X*******************************
****************************X*******************************
****************************X*******************************
****************************X*******************************
****************************X*******************************
****************************0000X***************************
****************************0000X***************************
****************************0000X***************************
****************************0000X***************************
****************************X*******************************
****************************X*******************************
****************************R*******************************
************************************************************
************************************************************
(25,28) (24,29) (23,30) (22,30) (21,30) (20,30) (19,29) (1
```

An example output from A* - ‘R’ represents a robot, ‘B’ the ball, ‘X’ the nodes of the outputted path and ‘O’ the nodes occupied by obstacles. The path coordinates are shown at the bottom.

Construction

The main thing I contributed in construction for this milestone was a number of quick fixes. For example, we had the same problem with wheel alignment, which I was able to fix. I also solved the problem of the wheels on the bumper sensors being pushed back over time.