SDP Individual Performance Report 2

Tomas Tauber (s0943263)
Group 7

I. Introduction

For the second milestone, I wrote simple methods for getting the robot's orientation without the exact Vision feedback and for navigating the robot as a backup solution, wrote unit tests for some helper methods I reused from the planning code of the last year's Group 5, discovered a few bugs in the Vision sub-system, debugged the initial A* path search algorithm, and partially set up a continuous integration system.

II. Simple navigation & Vision bugs

As the Vision sub-system was incapable of capturing the robot's orientation and Strategy/Planning design was still being discussed, I implemented a "quick and dirty" backup solution, reusing some methods from the planning code of the last year's Group 5 (for which I wrote 9 unit tests to verify potential changes in them): it takes the robot's position, sends a move command to the robot, reads the new position, and computes the angle w.r.t. the origin. After that, it calculates how much the robot should turn to face the ball and issues a corresponding rotate command to the robot. Finally, the robot moves, eventually continuously corrects its angle if it does not face the ball (with a 5° tolerance), until it reaches the ball.

When I tested it, I found a few bugs in the Vision sub-system, most notably it was not providing correct positions of the robot. The Vision was still not able to provide the robot's orientation 3 days before the milestone, so we decided to use and refine the above mentioned backup solution. It worked with the stated 5° tolerance (it captures the kicker's range when near, 5-10 cm, the ball) – the potential problems would be only if the robot's initial position was near a wall, facing it, or if the Vision lost the ball's position.

III. Debugging A*

Laurie asked me to help him to fix the first version of the A* path finding code he had written, because it had a strange behaviour in some situations (e.g. it went off one more diagonal than needed when avoiding an obstacle). I examined the code and found out it was inefficiently storing neighbours of each node in the grid, plus it added wrong neighbours for some nodes. I rewrote this part to retrieve neighbours on-the-fly and most problems disappeared.

IV. Continuous integration software

I configured Jenkins to build our development branch every 15 minutes. I managed to execute it remotely on a different machine. Because refining the navigation had a higher priority, I stopped at this stage. The problem I face here is that it has to run stand-alone on DICE, i.e. it cannot run 24/7 as a deamon or be deployed as a Java servlet.

V. Conclusion

Even though I had less time for the second milestone (2 interviews outside Edinburgh and one major coursework deadline), my early contribution with a simple backup solution for navigating the robot turned out to be fairly important. As the Vision did not achieve their goal to provide information about the robot's orientation on the pitch, it was better to have this simple method for retrieving the robot's angle than nothing. The other contributions were rather aimed at the overall process and future milestones.

For the first friendly matches and the third milestone, I will help to integrate the real Strategy/Planning sub-system, implement any missing parts of the system if necessary (e.g. the plan translation to low level commands), and try to get Jenkins starting up automatically with a shared configuration.