SDP Individual Final Report

Peter Macgregor, s1126155

April 23, 2014

1 Introduction

Throughout the SDP project my major task was to implement the vision system for our project. I wrote the majority of the code to track the objects on the pitch as well as the GUI for controlling the vision settings. Towards the end of the project, I was also heavily involved in testing and debugging the strategy code. In this report, I will discuss the vision system only since it was my main technical contribution.

2 Implementation

My motivation for most of the design choices made in relation to vision was to keep the implementation as simple as possible to begin with and to introduce complexity only where it was needed. There were 4 main objectives of the vision system:

- Detect the ball location.
- Detect the robot locations.
- Detect the robot orientations.
- Detect the pitch boundaries.

2.1 Basic Object Detection

To begin with, my idea was to use a very similar technique to solve each of the first three of the points above. In the end, however, this naive method was only kept for detecting the ball. This initial method was simply to look through each pixel in the image and compare the RGB and HSV values to the predefined conditions for being a part of the desired object. This resulted in excellent results for detecting the ball, but detecting the robot locations and orientations proved to be difficult based on the yellow, blue and grey colours alone.

2.2 Robot Detection

Detecting the orientation of the robots proved to be the most challenging task of the vision system and was constantly developed throughout the project. It requires the accurate detection of the coloured 'i' on the green plate as well as the grey dot. Following the failure of my initial naive attempt, I decided to detect the green plates in order to detect the blue, yellow, and dots. The plates proved to be a much easier colour to detect, but produced a couple of problems of their own. In order to improve the detection of the robot, I decided to only look for pixels in the green plate within a close proximity to the location of the robot in the last frame. However, this meant that there would be times when the vision system would detect another section of the pitch rather than the robot and would be unable to correct itself since it would only look at nearby pixels. In order to solve this issue, I introduced methods in the GUI which allow the user to click on the location of the robots in the image in order to set the initial location for the algorithm which gave very good results.

Once the robots' locations were detected using the plates, their orientations can be found by creating a vector from the centre of the 'i' to the grey dot. Detecting these became much easier when I limited the search to within the green plate.

2.3 GUI

In order to account for the varying lighting conditions on the pitch and to improve testing, it quickly became apparent that I'd need to provide a system to easily change the RGB and HSV thresholds for each of the colours. My implementation of this functionality was heavily based on the code from last year's group 4. I provided a separate panel for each colour allowing the values for each attribute to be dynamically altered. I also allowed the user to display the pixels detected for each colour on the screen in order to help with setting up vision and debugging. The system also allows the thresholds for each pitch to be saved to a file and loaded the next time the system is started. Another helpful function of the GUI is to select which pitch we're playing on as well as our team's colour and shooting direction. Finally, the GUI includes a panel providing information about the current state of the world according to the vision system which helped in debugging to see where the program was going wrong.

2.4 Pitch Boundaries

My initial approach to providing details of the pitch boundaries was to write a method to detect them automatically based on the colours of the pixels at the edge of the pitch. I quickly found, though, that it was more reliable to select the boundaries manually using the GUI and save the location so it wouldn't have to be done every time the system was run.

2.5 Interface to Strategy

All of the information collated by the vision system was made available to the strategy by updating a world object. This object contained all of the details described above and was accessible by both vision and strategy allowing communication between the two.

3 Analysis and Possible Improvements

The vision system was an extremely successful part of our project and provided accurate information about the locations of the objects and the orientations of the robots. It was sufficient for our purposes in every basic situation of the game. We did, however, encounter some problems - there are a few cases in which our vision system can break down. If a robot is next to a pitch boundary, sometimes the green plate is not fully detected, this leads to inaccuracies in the robots orientation and causes unpredictable behaviour. This can be fixed at the edges of the pitch by selecting a pitch size greater than the true size, however, it would be preferable to update the robot detection algorithm to allow a margin outside of the pitch boundaries in which a robot can still be detected. We also found, particularly on the side pitch, that the lighting is uneven across the pitch meaning that the threshold values will not work everywhere. This could possibly be improved by using some more sophisticated method of choosing thresholds which could include separate thresholds for each section of the pitch.