System Design Project 2012
Individual report for Milestone 1

s0908031

1. INTRODUCTION

The aim of this report is to present my individual contribution towards Milestone 1, within group 8 and to state the aims for the next milestone.

2. DECISIONS

At our first meeting, the team split into smaller groups, in order for everyone to contribute to some aspect of the project. The teams were Robot Construction, Controller, Vision and Simulator. After reading through the source code and the reports from 2011, we chose to develop on top of the project of team 5, which had a clearly structured architecture.

3. RESEARCH

Aim: Understanding the vision techniques implemented in the previous year

I have chosen to work on the vision system, as I have some knowledge of image processing from the Introduction to Vision and Robotics course taught at the University.

The first task of the vision team was to become familiar with the vision techniques used by last year's team. This involved reading their source code, understanding how the vision system communicates with the rest of the systems and understanding the techniques employed for robot and ball recognition.

The image processor from team 5 iterates through the pixels of the image, isolating only those belonging to the pitch. If a pixel's RGB channels are within certain values, they are classified as red, yellow or blue.

My team agreed that their technique for detecting the ball was not very reliable, since they were searching for a pixel whose colour was the closest to 'perfect red'(*) and assumed that it was the centre of the ball.

At the suggestion of Gediminas Liktaras, Andrei Manolache and I started doing research into JavaCV, a java wrapper of the OpenCV library. Since the JavaCV documentation does not exist, we looked at the OpenCV libraries[1] for image processing and object detection.

4. IMAGE PROCESSING

Aim: Detecting the robots and the ball, using live images from the camera

In order to detects the objects, we used a mixture of techniques learnt in Introduction to Vision and Robotics(**) and techniques used by last year's team 9(***).

The processor uses a background image representing an empty pitch. This will be used to perform background extraction on every frame.

The resulting image is normalised, to compensate for shadows and then is split into three channels, for Red, Green and Blue. A contour detection function is then performed on each of the channels. The contour whose bounding box has the largest area should correspond to the yellow robot in the red channel, and to the blue robot in the blue channel.

In order to detect the ball, the original image is thresholded and a filter is applied to the resulting image. Another function is then used to get the spatial moments[2] of the detected shape(only the ball in this case).

5. CONCLUSION

In conclusion, my contribution consisted in doing research on JavaCV and OpenCV and becoming familiar with the image processing libraries. Andrei Manolache and I have created an ImageProcessor class that detects the robots and the ball. This class was then refactored and integration code was added to it by Gediminas Liktaras. I have also created a basic GUI class that should display the live images from the camera, but this was quickly replaced by a class created by another member of the team.

Having fulfilled our first aim and a part of the second aim(****), we now plan on improving our image processor and testing it with images from the camera.

APPENDIX A - REFERENCES

[1]. The JavaCV webpage
http://code.google.com/p/javacv/

The OpenCV documentation
http://opencv.itseez.com/

[2]. The function detecting the ball was adapted from the following code:
http://ganeshtiwaridotcomdotnp.blogspot.com/2011/12/object-tracking-in-java-detect-position.html

APPENDIX B

(*) - 'Perfect red' was defined as {255, 0, 0} for the red, green and blue channels

(**) - We have used ackground extraction and thresholding on different channels

(***) - We have used their approach on detecting contours from an image

(****) - We are currently using test images, not live images