Dale Myers 0942590

MILESTONE 2 REPORT

This week my contribution was to test the efficiency of different methods of calculating the robots orientation and to implement the communication method between different components in our program.

In order to test any methods of efficiency, we first needed a testing system. We knew from the very first day that testing the vision system would be a difficult task. The method I devised was to get the vision system to use a single frame as an example, save this image and record where it things the robots are, their orientations and the balls position. This data would be stored as XML data. We could then have a separate program which would load in this data and the image and then proceed to ask the user where the robots and ball is. It would then compare the user generated data with the data produced by the vision system. The vision system would pass if the generated data was within certain tolerances of the user generated data.

This system was implemented and was rather successful. The difficulties were found with getting the vision system to output the data. I needed to restructure the program a small amount in order to implement this in a sensible manner. From this point it was a matter of trying different methods of calculating the orientation. The method deemed most likely to succeed, based on the use by previous years, was implemented by Rado. This method was to use the centroid of the T shape and then look for the grey circle nearby and use that to calculate the orientation. The methods I was to test were using a projected line, rotating until the shape is upright and using the centroid of the plate. A proposed method was to use a projected plate in a similar manner to the projected line method, but was never implemented.

Each of these methods was implemented in turn and tested. For each one the results were consistent. They were simply not as accurate as the grey circle method. They would jump around and not know if they were facing forwards or backwards. Even with tweaking and calibration they simply were not showing the same promise as the method Rado was working on. Each method was then abandoned. A final method showed some success for accuracy. This method involved finding the vertices of the T shape, finding the two closest points and calculating the orientation as perpendicular to the line between these two points. This method was rather accurate, but had two large disadvantages. The first was that it required an average of around 5-10 previous orientations in order to be steady enough to be useable. This created a delay of around half a second, a delay which was completely unacceptable. The second was that due to this averaging procedure it encountered a lot of difficulty if the orientation passed 0 or 360 degrees. These flaws rendered the method useless and it was also abandoned.

Efforts for orientation finding from here are to use Rado's method and improve it as much as possible. I do intend on trying out alternative methods if they occur to me, as well as trying out a few ideas which I already have but have not been able to implement due to time constraints. I aim to have these completed by the end of the week. From here on we have suggested cutting the size of the vision team from three down to two or one so that we have more coders for strategy. It is unknown which I will be doing at this point but due to this it is unwise to plan too far ahead.

The other system which I worked on with Chris was to implement a method of passing information between the different components of our system. We decided that we would like to have a single static class which would hold the information and could be accessed by anyone. This class was to be as simple as possible with all the calculations done elsewhere if possible. The data to be stored was relatively simple: the pitch, the blue robot, the yellow robot and the ball. It turned out to be slightly more complicated than first imagined however. It was not a difficult problem to solve as such. The complexity arose from the sheer number of getters and setters in the class. Depending on what was being calculated we needed to assign data in multiple different ways. An example might be setting the balls position. This was easily done calculation to calculation. Setting the velocity however was more difficult. Velocity is a direction and a magnitude so we had to provide getters and setters for changing each component individually, both together, or even just replacing it with a reference to an entirely new object.