Individual Report 5

Calum Jackson (s0812597) - SDP Group 12

I. Introduction

This report, the last in the series, will document my contributions to the SDP project. I will begin by outlining the team planned to tackle the project, and continue by explaining how I contributed to meeting our goals, finally concluding on my efforts and the teams efforts as a whole. Our team decided to split the project into the following 4 tasks:

- Robot Construction
- Building a GUI and Simulator
- Creating a Strategy
- Implementing a Vision System

I began by working in robot construction, and later moved into creating the playing strategy for the robot. I was also involved in large amounts of testing on the robot, and was a large contributor to the presentation on the final day.

II. Robot Construction

The team chose Lejos as the most suitable programming language for the NXT brick, so Marc and I began by creating a basic robot model and flashing the NXT firmware to the brick, which allowed other team members to test and get to grips with the programming task ahead of them.

The teams aim for the robot was to have a fast, lightweight robot, giving a speed advantage over other robots. I researched into gears, and discovered a 3 - 1 ratio would make our robot up to 3x faster than if the wheels were connected directly to the motors (REFERENCE Lego Mindstorms Books). Having set gears up on the robot, I found they did affect the straight line ability of the robot, shown below:

TABLE I
Distance off-line (Running for 10seconds)

Speed (units??)	Off-Centre (mm)
160	82
320	216
480	850
740	N/A - Span in Circles

As shown, increased speeds led to massive offsets in travelling in a straight line. After testing multiple different options, including changing caster wheels, using ball-bearings, and putting the motors at the front, back and centre of the robot, eventually I found that the robot would diverge from its path whilst moving forwards, but would travel in a relatively straight line whilst reversing (at the same speed). Therefore a simple solution was switching the motors around. This led to much improved results:

TABLE II
Distance off-line 2 (Running for 10seconds)

Speed (units??)	Off-Centre (mm)
160	0
320	4
480	10
740	46

These reduced discrepancies could be managed by the programming of the movement.

Further to this, I horizontally mounted the motors to lower the robots centre of gravity, repositioned the kicker motor to keep weight central whilst elevating the connection to the kicker to increase the kickers momentum whilst kicking, built large amounts of the chassis (moving through a number of different designs), and added touch sensors to the front of the robot.

Finally, numerous structural reinforcements were made to the robot to compensate for our robot being lighter than other robots and more susceptible to damage when being hit by other robots, and to compensate for the extra pressure put on the chassis by the increased amounts of torque and speed created by the use of gears. I feel the robot did have a distinct speed advantage, especially coupled with the potential fields movement algorithm, which utilised the robots speed efficiently, as well as being very structurally strong, contending with other robots without issues.

III. Strategy

This was another section to which I was a main contributor. The main focus was to create an attacking strategy which would utilise the speed of the robot. The strategy was split into two sections, of which I worked on the high level areas (where the robot will move to, when to kick, dribble etc) as opposed to the lower level areas (coding the movement of the robot). Juozas and I began by creating a simple state system strategy, which checked the current on-pitch situation (using the data sent from the camera) and decided the appropriate strategy to run.

Our strategy was based on defining a point to move towards. A point class had previously been created, holding the x,y co-ordinates of the point on the pitch. The class was extended to create the robot, ball, goal, and any other necessary points for use in the strategy.

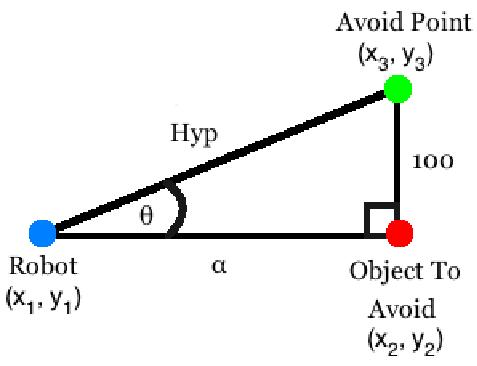
I implemented an optimum point, which would be a defined distance behind the ball at the same angle as the ball to goal angle, such that the robot would move to this point, and then to the ball, to ensure the robot was facing the goal when it reached the ball. This worked well in practice, working everytime when the robot was behind the ball, and 9/10 times when the robot was inbetween the goal and the ball (i.e. had to navigate around the ball).

To work out the angle between the ball and the goal, I created a function which used the following formula:

$$angle = atan2((y_2 - y_1), (x_2 - x_1))$$

where the $ball = (x_1, y_1)$ and the $goalCentre = (x_2, y_2)$. Later I moved this to a higher level, allowing the function to take any two points and return the angle between them.

The next aim was to be able to navigate around objects. Points were used again, as I created a function which would calculate an avoidance point at a $90°$ angle and a defined distance from the object to avoid. The function would return the avoidance point, and the robot would move to it.
This function was dynamic, and the point would move as the robot moved, allowing the robot to navigate the object more smoothly.
The function calculated the avoid point using trigonometry, as shown in the following diagram and explained after:



The hypotenuse of the implied triangle was calculated using the distance between the robot and the object, θ, and the distance between the object and the avoid point, 100.

$$hyp = \sqrt{\alpha^2 + 100^2}$$

Next, the angle between the robot and the avoid point was calculated.

$$\theta = \sin(\frac{100}{hyp})$$

After this, the offset for the (x,y) co-ordinates were calculated.

$$xOffset = hyp(\cos\theta)$$

$$yOffset = hyp(\sin\theta)$$

Finally, these offsets were added to the robot co-ordinates, giving the avoid point co-ordinates which were returned by the function.

$$(x_3, y_3) = (x_1 + xOffset, y_1 + yOffset)$$

Again, this function was abstracted to a higher level to calculate the avoidance for any points, as the robot may need to avoid a ball or a robot, which was useful for the function I created to avoid the ball when moving the robot to get to the correct side of the ball (so the ball would be between the robot and the goal we were attacking).

Alongside these I implemented functions to decide if the ball was in more difficult positions, such as being close to the walls or in the corners, so that different strategies could be brought in to deal with these situations (added through the state system).

I feel a more precise shooting system (i.e. being able to shoot directly into the corners of the goal) and a defensive strategy would have improved our performance in the final competition, as a number of shots were saved in the centre of the goal, and we had no real strategy to deal with the opponent having possession, and whilst usually the attacking strategy would cover most defensive situations, we were found out once or twice in the final game.

IV. Testing

The team's target for testing was to find bugs in the robots performance and to check the code was performing as expected. The simulator was useful for large amounts of testing the strategy, but I also tested each new section on the robot to ensure it would work correctly in real situations. I helped with the extensive testing done on the potential fields movement algorithm, as this was difficult to implement and did not perform as expected on the pitch compared to simulations (i.e. the robot would spin in circles on the pitch).

V. Friendly Tournaments

During the friendly tournaments I aimed to get a good view of our robot in competition and find any bugs which had not shown up in testing. During these matches

VI. Presentation

VII. Final Tournament

The final matches came with disappointment, as myself and the rest of the team had high hopes for our robot, which were unfortunately brought crashing down in a 3-1 loss in the quarter-finals. We encountered unforeseen problems in that our vision could not reproduce the high frame rates we consistently had throughout the semester, and as our robot reacted to the frames being processed, our strategy ran at a much slower rate. Despite still playing fairly well, our robot failed to convert numerous chances, missing 2 penalties, and lost a match I feel could easily have been won.

VIII. Teamwork Contributions

Throughout the project the goal for teamwork and communication was to keep all team members up to date with work being done on any part of the robot, and make sure members knew what their tasks were. I contributed to this by adding updates to the team website every-time I worked on the robot and code, mentioning areas which weren't finished and areas which could be improved with possible additions. I also commented the code thoroughly to help other team-mates understand it, and constantly added TODO's to make sure tasks were not forgotten and allow other team-mates to see how they could contribute. Additionally, during team meetings I regularly contributed ideas for future improvements across a range of areas including how to improve the robots structure and additional areas to be added to the strategy. I feel these contributions were all very useful in the project, and worked well having looked at all other team-mates contributions, as this allowed me to see how the whole project was progressing and notify other team members on how to improve their sections, and vice versa.

IX. Conclusion