Do robot sheepdogs coral electric sheep?

Robert Kenneth James Johnson

Specification/Design Document

Submitted to

The University of Liverpool

in partial fulfilment of the requirements
for the degree of

MASTER OF SCIENCE

November 20, 2018

Student Declaration

I confirm that I have read and understood the University's Academic Integrity Policy.

I confirm that I have acted honestly, ethically and professionally in conduct leading to assessment for the programme of study.

I confirm that I have not copied material from another source nor committed plagiarism nor fabricated data when completing the attached piece of work. I confirm that I have not previously presented the work or part thereof for assessment for another University of Liverpool module. I confirm that I have not copied material from another source, nor colluded with any other student in the preparation and production of this work.

I confirm that I have not incorporated into this assignment material that has been submitted by me or any other person in support of a successful application for a degree of this or any other university or degree-awarding body.

SIGNATURE R. M. J. Johnson.

DATE June 28, 2018

Contents

Chapter 1

Abstract

The amount of prospective students wanting to study computer science has been increasing steadily, in a Higher education student statistics report it was stated that the amount of students had increased by 5% from 2015 to 2017[1]. To continue this trend it is important to create tools to inspire future students to further their education.

The aim of this project was to create a physical simulation of the predator prey pursuit problem. Such a system will help to inspire students to explore the realm of robotics and multi agent systems as well as demonstrate the possible issues as well as solutions that physical robots introduce.

The project proposes various mechanisms used to assist in capturing the prey including a method which approximates the position of the prey based on where it was seen previously and the movement available to it.

This paper has found that the time taken to capture the prey can can be decreased substantially by utilising methods to estimate the position of the prey. Making substantial decisions using this information however could result in slower capture due to incomplete information.

This project found that trying to simulate the predator prey pursuit problem physically resulted in various issues related orientation, location and sensor accuracy. Potential solutions were explored such as gyroscope usage, line alignment and multi-sampling.

Chapter 2

Introduction

The predator prey pursuit problem was proposed by Benda, Jagannathan and Dodhiawala [2] in the pursuit to study competitive co-evolution, multi agent strategies and co-ordination. Involving 4 predators who move using a specific policy and a prey that moves randomly. The predators aim is to restrict the movement of the prey with the objective to completely surround it.

There has been substantial research into the predator-prey pursuit game, however the implementation has been mainly simulator based. A physical implementation would serve as an incentive to those interested in multi agent systems, allowing them to see possible issues as well as opportunities of using communication in a multi-agent environment.

The solution produced takes an environment from a user and creates a physical simulation of the predator prey pursuit problem. Allowing the user to place predators and obstacles in the environment. The simulation uses EV3 robots which use ultrasonic sensors to calculate the distance to prey and obstacles as well as 2 color sensors and a gyroscope to assist with localisation.

Chapter 3

Aims and Objectives

3.1 Aims

To develop a physical implementation of the predator-prey pursuit problem that takes a user defined environment and displays the progress of predators. The final system will not only follow the prey but also take measures to surround it.

3.2 Objectives

3.2.1 Research Objectives

- Evaluate existing literature pertaining to the predator-pursuit problem looking at policies that involve communication between agents

3.2.2 Design Objectives

- Design multi-agent processes for solving the problem including:
  - Prey following process
  - Communication
- Implement one of the various methods in current literature.
- Ensure that the piece of software can be run on multiple EV3 robots.

3.2.3 Implementation & Testing Objectives

- Adhere to agile methodology during development, allowing incremental testing to make the final distribution robust.
- Enforce version control using Git-hub to store incremental backups.
- Discuss major problems encountered during the project as well as any interesting findings.

3.2.4 Evaluation Objectives

- Evaluate the performance of the system in terms of rotations by each agent, forward movements, failure and success rate.
- Formulate map set-ups looking at the performance and possible improvements that could be made to current search policies.

3.3 Expected outcomes

- User manual, this should contain:
  - Information relating to requirements for the overarching system including robot components and environment.
  - Information relating to how to start up the system including pre-loading each robot with map designs.
  - Information relating to the policy used in the predator-prey pursuit problem.
- Code should:
  - Be well formatted abiding by standards associated with Java.
  - Be split into manageable modules to reduce complexity.
  - Be fully commented to allow for subsequent changes or alterations to be tested.
  - Contain central machine code handling communication as well as negotiation of agents
  - Contain individual agent code to handle movement as well as autonomous exploration.

Chapter 4

Background

Recently there has been focus on the use of genetic algorithms to aid this domain [3][4]. Genetic algorithms require many repeated runs eventually converging to an optimal policy, because of the small timespan of this project such a strategy would be infeasible. Due to this, I will be focusing on static algorithms.

4.1 Previously used static policies

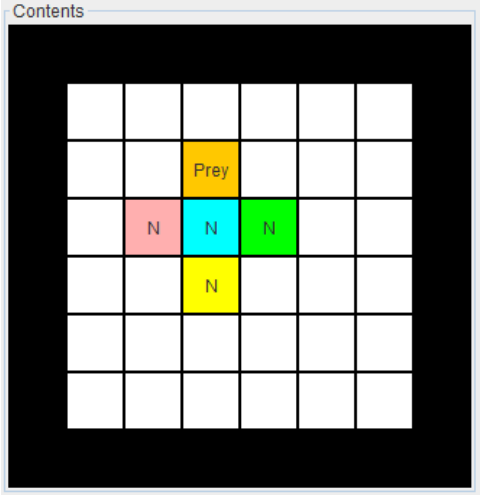
4.1.1 Shared definitions between policies

The prey is classed as being "captured" if all its adjacent cells are currently occupied by the predators or obstacles.

4.1.2 Policy 1: No communication

Stephens and Merx [5] propose a method without communication, each predator only aware of the preys and its own position.

In his policy each predator calculates the capture point that is closest and moves towards it. Suppose the state below occurs within the simulation. All predators calculate the closest capture point. Because there is no reflection on other agents locations such an action selection would cause collisions. Because of this Stephens and Merx's strategy produced poor results.



4.1.3 Policy 2: Recursive learning

Vidal and Durfee [6] suggest a method where each agent considers the location and sensory perceptions of other agent's before an action. Each agent follows the same policy recursively reflecting on these perceptions, slowly converging to a global policy. Vidal and Durfee suggest cutting this deliberation off after a set number of iterations and simply choosing the highest probability global allocation to allow for timely movement.

4.1.4 Policy 3: Coalition Payoff

A game theoretic perspective was given by Levy and Rosenschein[7]. They describe a coalition payoff function, rewarding utility based on contribution. He used a modified version of the shapely value in his implementation, a coalition consists of agents doing different "blocking" actions.

Because agents would like to maximize their payoff they would all like to choose different blocking actions incentivizing moving to diagonal positions from the prey.

Definition for blocking position

Distance to prey, x axis: dix
Distance to prey, y axis: diy

Condition

If a predator moves in the direction with the greatest value max(dix, diy) then that predator is defined as "blocking" the opposite direction.

Example

If a predator had distances (2,3) from the prey then it would be "blocking" a direction if it were to move towards the prey in the y axis.

4.1.5 Policy 4: Individual greedy payoff

Korf[8] also took a game theoretic approach requiring only predator positions. Korf focused on a greedy approach, trying to maximize personal reward. This approach tried to combine utility for maintaining distance between predators whilst reducing distance to prey. This resulted in predators approaching prey from lieb configurations (once the lieb configuration has been achieved successful capture is guaranteed). This produced good results however failed under certain conditions[9]. This method has been further explored by others[10].

4.1.6 Policy 5: Balancing communication and independent actions

Another approach discussed by Ei-Ichi Osawa [11] tries to balance communication and independent strategy, describing meta level co-ordination placing emphasis on communication cost. Suggesting slowly increasing the communication between agents depending on specific conditions. The policies in increasing order of communication are:

- Autonomous strategy - Select the nearest capture point and approach it, if prey location is unknown search using BFS.
- Communicating strategy - Communicate the location of the prey.
- Negotiating strategy - Communicate the preference to each capture position, negotiating capture points using these preferences.
- Controlling-agent - One agent is selected as the controlling agent, he computes optimal movements and communicates these to each agent, aiming to reach lieb configuration.

The algorithm is described below:

Written strategy

Each agent uses the autonomous strategy, searching for the prey and movies towards the closest capture position. If all agents have found the prey a counter is incremented expressing the number of turns agents failed to gain distance on the prey, if this happens for a set amount of turns switch to a strategy which uses a greater amount of communication. This process then repeats until the prey is successfully captured.

Algorithm

1. **Let turn $i \leftarrow 0$, counter $C \leftarrow 0$, $D_0 = \infty$, $\Delta_0 = \delta(\geq 0)$, and initial index of organization scheme $j \leftarrow 0$.**
2. **If $\mathcal{E}$ is true, then halt.**
3. **If $\mathcal{V}(R) \wedge \mathcal{V}(B_2) \wedge \mathcal{V}(B_3) \wedge \mathcal{V}(B_4)$ is false, then execute S_j, let D_i be ∞, and $i \leftarrow i+1$, and go to step 2.**
4. **If $\Delta_i > \delta$, then execute S_j, let $i \leftarrow i+1$, and $C \leftarrow 0$, and go to step 2.**
5. **Let $C \leftarrow C+1$. If $C < \tau$, then execute S_j, let $i \leftarrow i+1$, and goto step 2.**
6. **While $j < n$, let $j \leftarrow j+1$. Execute S_j, let $i \leftarrow i+1$, and $C \leftarrow 0$, and go to step 2.**

Figure 4.1

Tau: Threshold for the number of turns distance gained was less than delta.
Delta: Threshold for the distance that should be gained in a single turn by each agent.

4.2 Prey policies

4.2.1 Q-learning and Sarsa

Q-learning and Sarsa function similarly to genetic algorithms taking time to converge to a optimal policy, this wouldn't be possible to implement in such a short amount of time however it is still important to review literature to understand that such methods can be applied. In a paper by Jacob Schrum [12] he discusses the complexity of modelling the system with complete and incomplete information using partial state representations, he reviews the performance of using such strategies against Predators who use q-learning and sarsa.

Chapter 5

Design

Original Design documentation is shown at appendix K.

5.1 Problem Variations

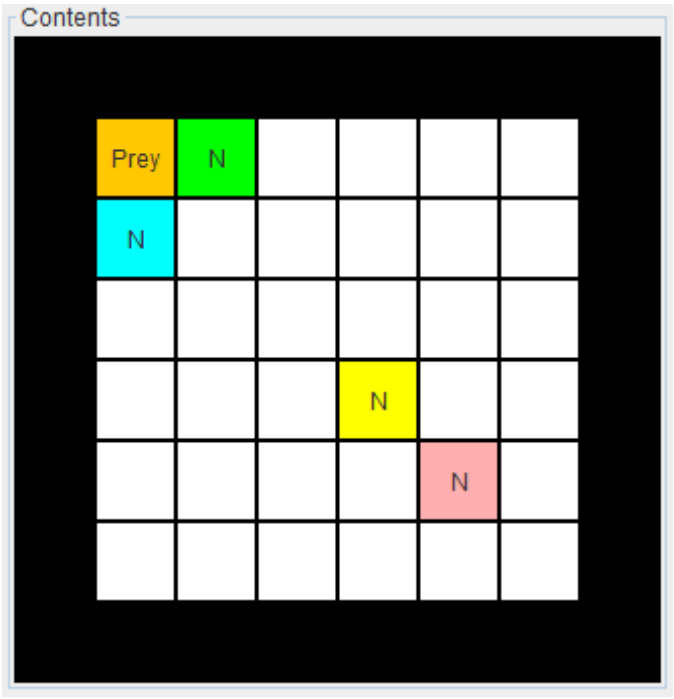
The original predator-prey pursuit problem has many different variations.

5.1.1 Number of predators and prey

The original version only comprised of 5 agents' however there have been variations consisting of many predators and prey explored by J. Avier and A. Alcazar [13] who looked at how many predators were needed to capture all prey within a specific time frame. We could also have less than 4 predators and one prey known as the Homicidal chauffeur problem[14][15][16].

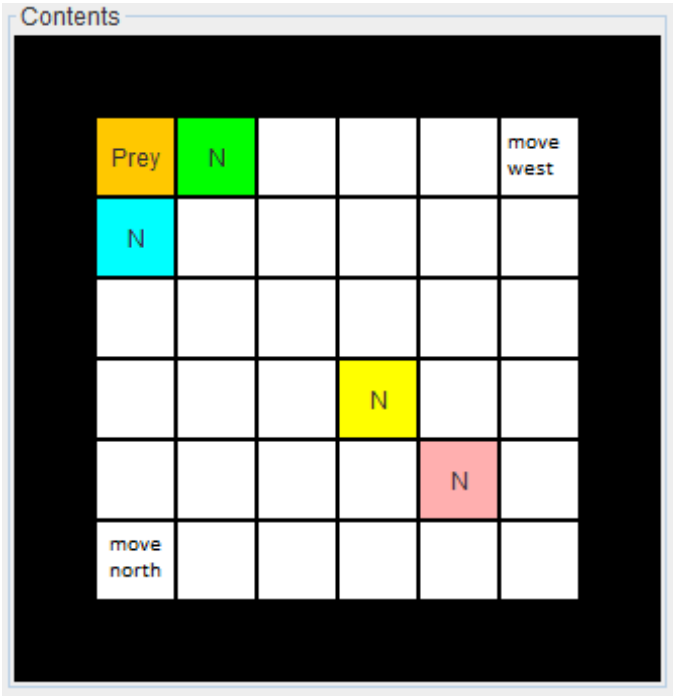
5.1.2 Environment

We could have an environment which is bounded by walls, in this case it may be possible for 2 predators to capture the prey. Such an example is shown in the figure below.



We could also model the environment as a toroidal world [3], consider the figure below, if this were a toroidal world the prey would still be able to

move north as well as west.



In a simulated environment communication may not have an overhead however in a real world setting, if the world size is large then communication may come at a cost or may not even be possible [11]. The amount of time needed to explore a larger environment may make it worthwhile introducing additional predators to speed up the search and capture process.

Typically we approximate the environment splitting up a map into smaller chunks, these chunks could be composed of different shapes each allowing more or less movement to the agents.

5.1.3 Range of movement and policies of predators and prey

It is important to look at the range of movement for each agent. Movement in a 2d world may be restricted to adjacent tiles or could be modelled as allowing movement in all surrounding tiles. In the paper by Levy and Rosenschein [7] they discuss movement into surrounding cells, giving predators 8 total actions instead of 4. They found their strategy moved agents into diagonal positions of the prey, making it easier for the agents to reach the prey as they could directly approach the prey from these positions.

Traditionally the prey moves randomly with a 10% chance to stay still, however if we would like to simulate nature it may be interesting to look into adjusting the policy to simulate running away.

Another consideration from a programmers point of view is that of ordering, if the environment is modelled as a turn-based system then what order of turns do we take, what is the frequency of each agent's turn? If the environment is modelled as a simultaneous system how do we ensure that each action is taken at the same time? If the environment is modelled as continuous environment how do we balance communication and actions?

5.1.4 Communication between agent's, prior-knowledge

Does the predator have prior knowledge of the environment, predators and prey beforehand or does this information have to be found or communicated?

5.1.5 Final variants chosen

We will be exploring the single prey, 4 predator variation in a turn based, non-deterministic, non toroidal grid world environment where only movement to adjacent cells is possible. The environment size will be 6x6. Prior knowledge about the environment is given to all predators.

5.2 Requirements of the system

The project must take an environment from a user and create a fully functional simulation of the predator prey pursuit problem. Allowing the user to choose the location to place predators and obstacles in the environment. Predators and prey will be modelled as EV3 robots that use ultrasonic sensors to calculate distance each other as well as obstacles.

5.3 Design process

5.3.1 Stage 1

During this stage it is important to create an interface allowing for easy debugging as well as visualisation of agent locations within the system, this should allow for simple message handling between agents. This stage will be completed once agents positions are shown on screen on a empty grid world environment as well as showing the movement of each agent in real time.

5.3.2 Stage 2

I intended to implement the preys movement policy during this stage, this move policy would be simple only comprising of:

- An initial scan at the start of the turn
- A movement including rotation, forward movement and empty move.

In later stages it may be possible to alter this policy to allow for a greater degree of movement.

5.3.3 Stage 3

Development on the physical aspects of the system will take place trying to ensure that movement is reliable in relation to orientation and location. The intention to move by an agent should follow a algorithm which attempts to either detect or follow the prey using ultrasonic sensors. This stage will be tested on an empty environment and be classed as completed once an agent has successfully carried out both of these tasks.

5.3.4 Stage 4

Combine the program made in stage one and three, allowing for multiple EV3's to roam the environment independently and follow the prey, avoiding each other in the process. Predators should communicate both their current position and the position of the prey if found.

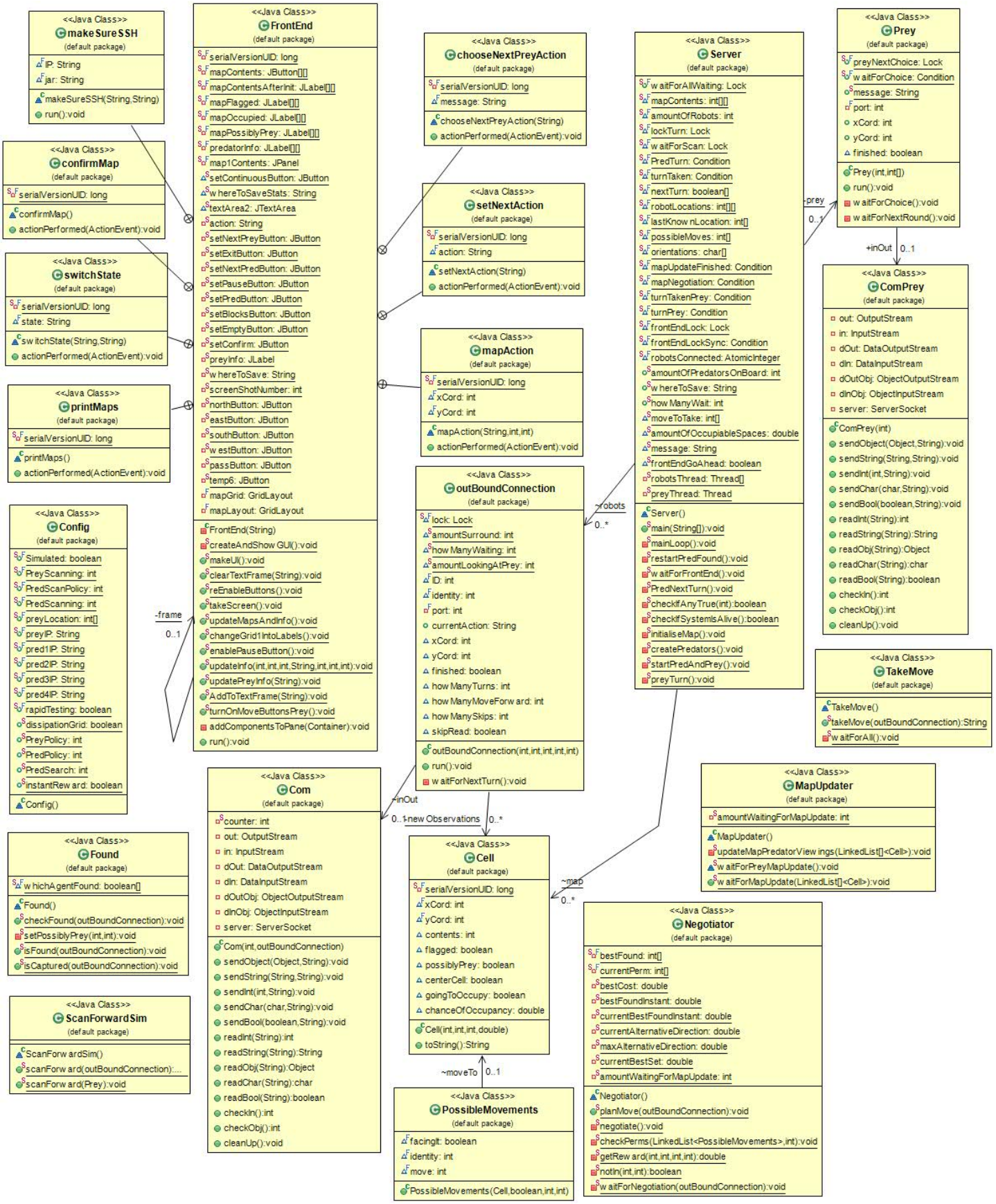
5.3.5 Stage 5

Create an overarching algorithm that is used once an agent becomes aware of the location of the prey, this algorithm should attempt to surround the prey rather than just follow it. This algorithm may be entirely new or may make use of the already developed methods described previously.

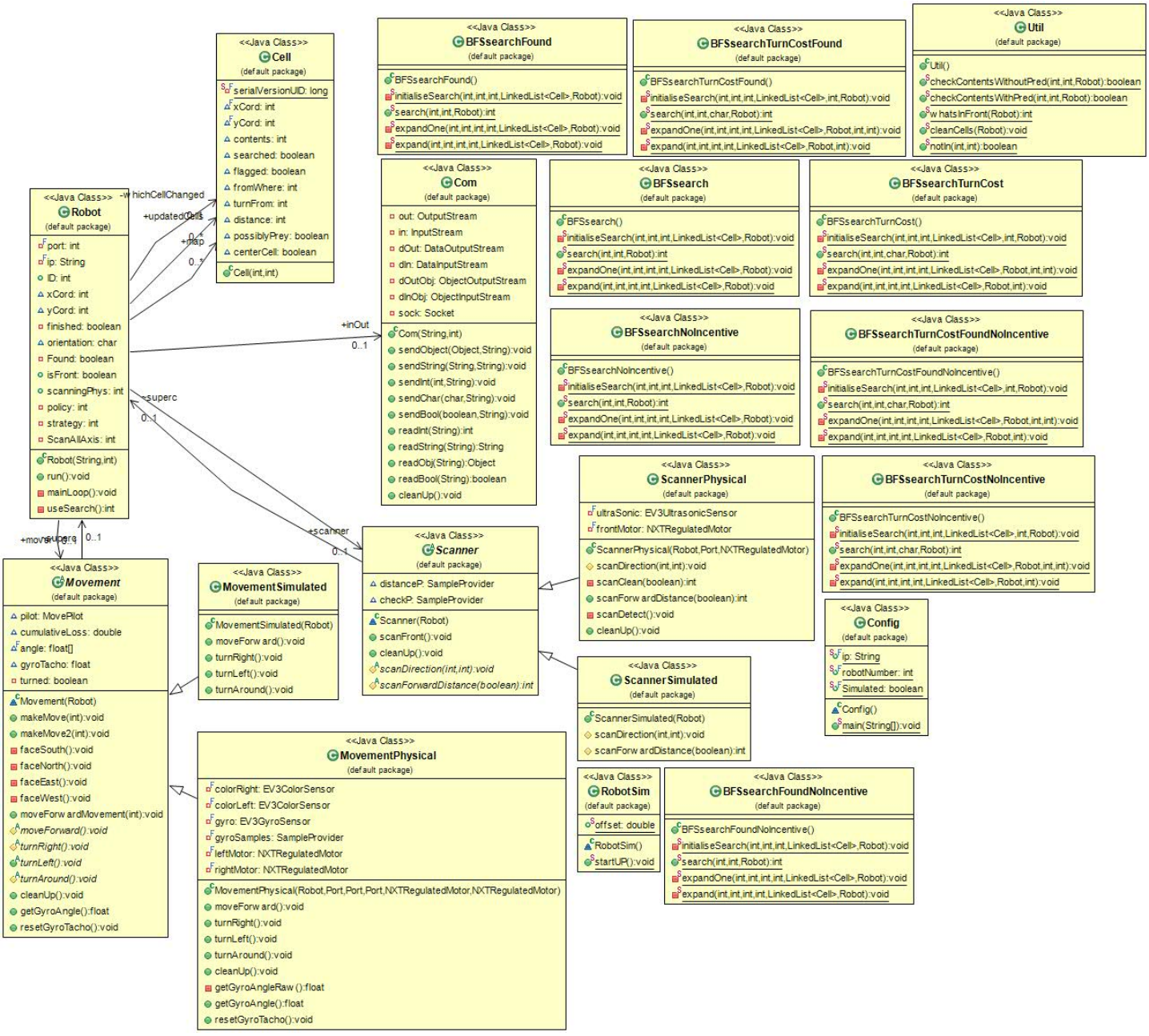
5.3.6 Stage 6

Create a system to allow different policies to be chosen for both the predator and prey, such a system would allow for easy creation and confirmation of map structure.

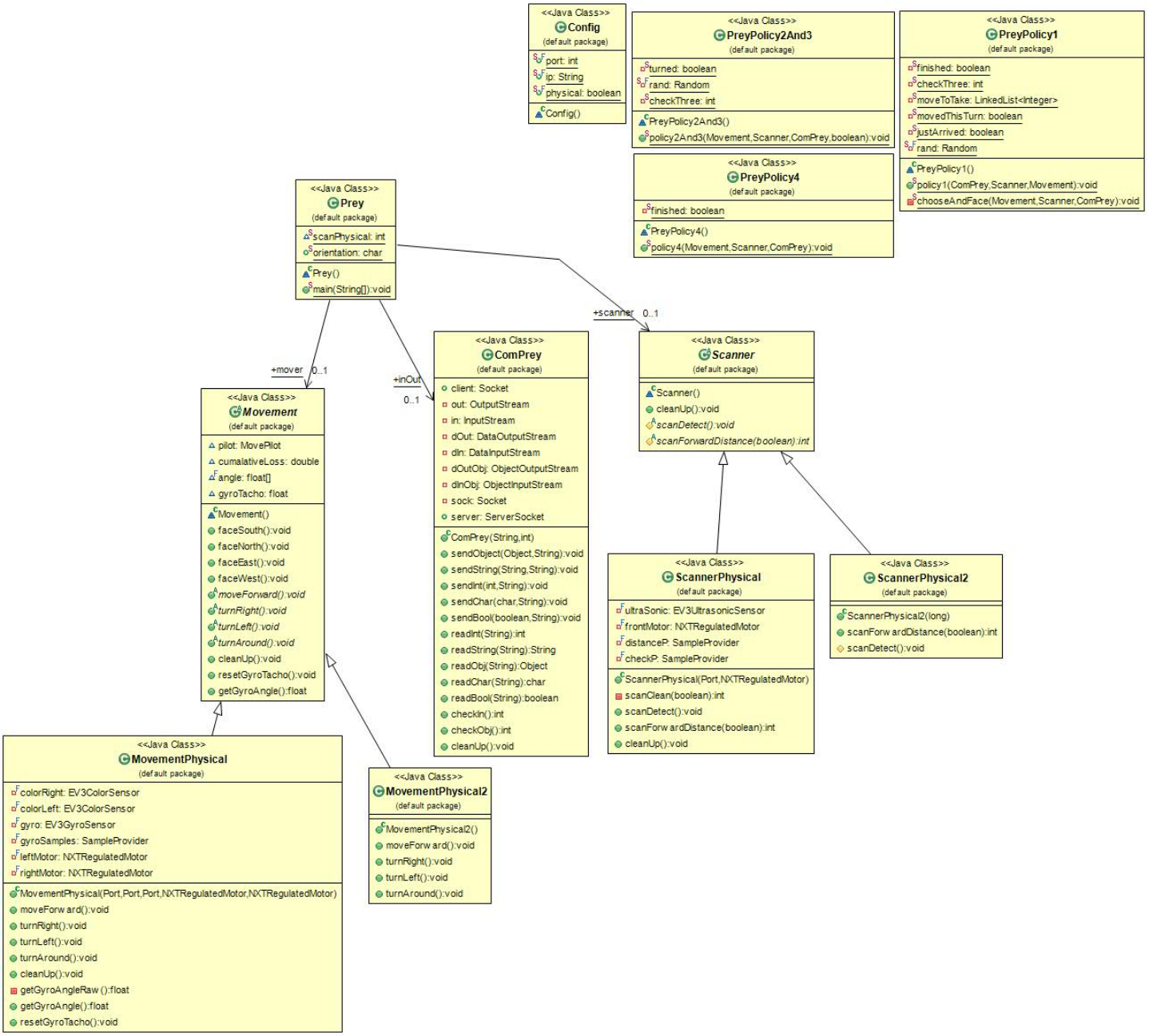
5.4 Class diagram (Blackboard)



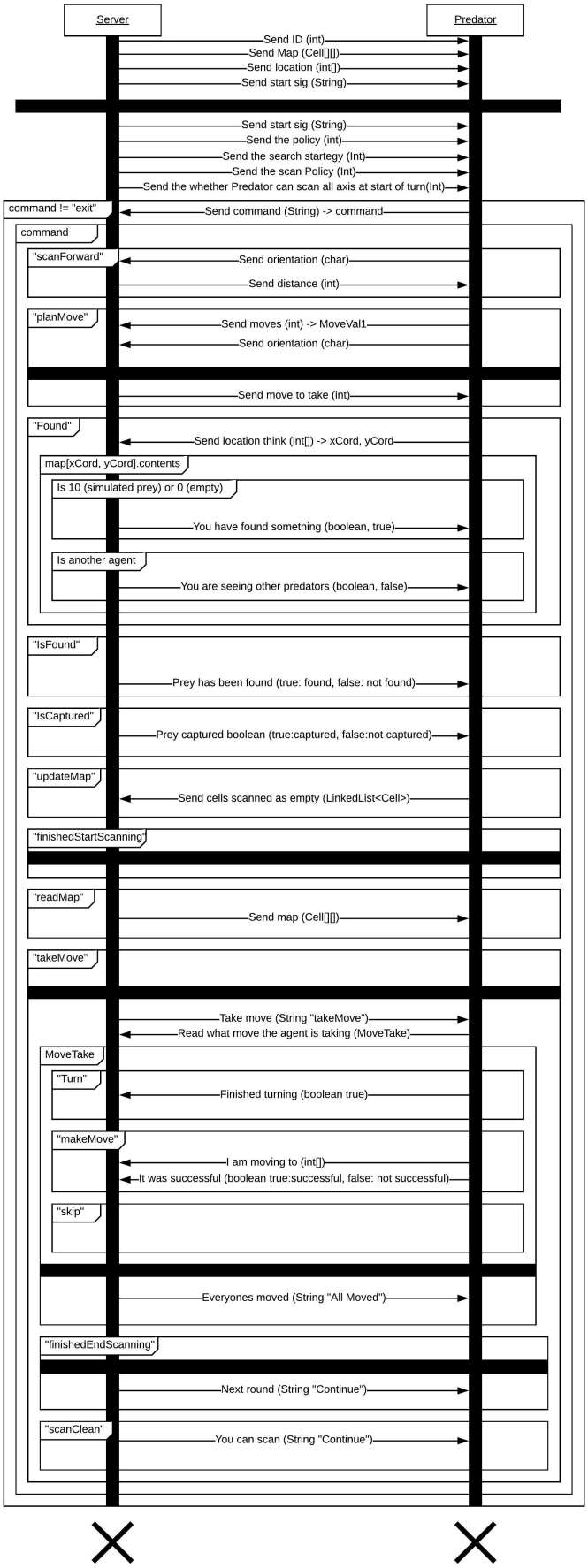
5.5 Class diagram (Predator)

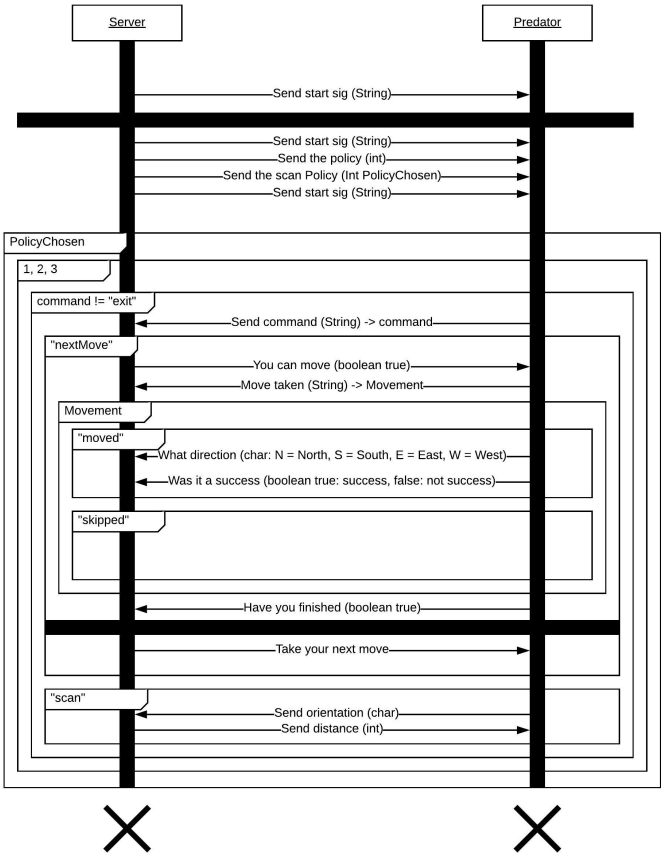


5.6 Class diagram (Prey)

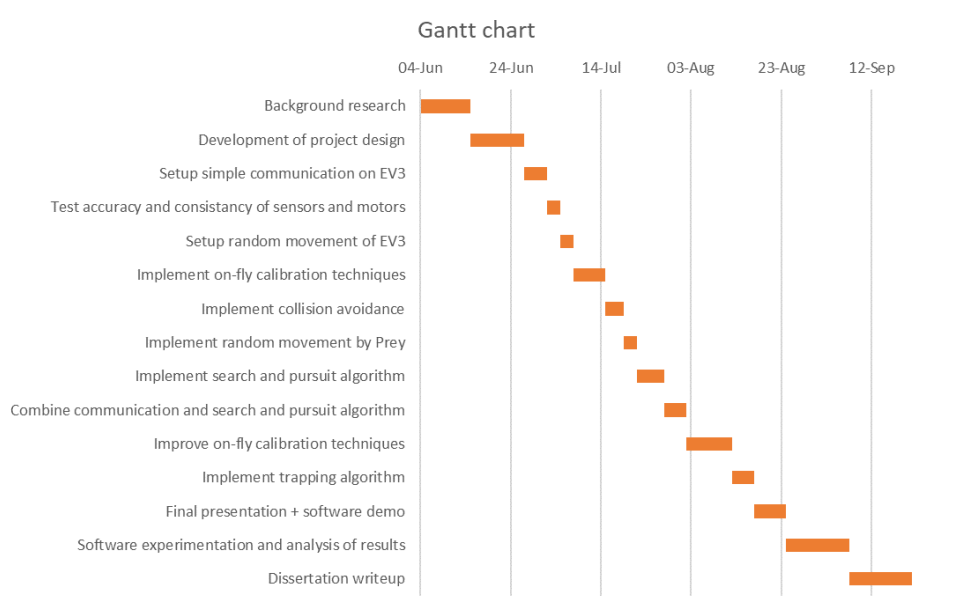


5.7 Communication diagram



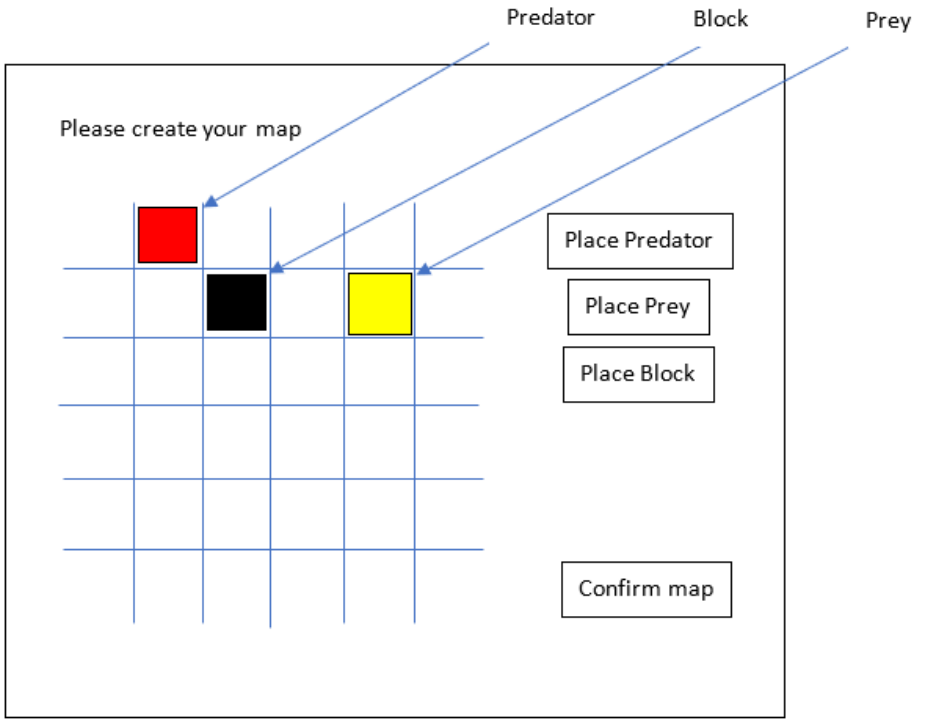


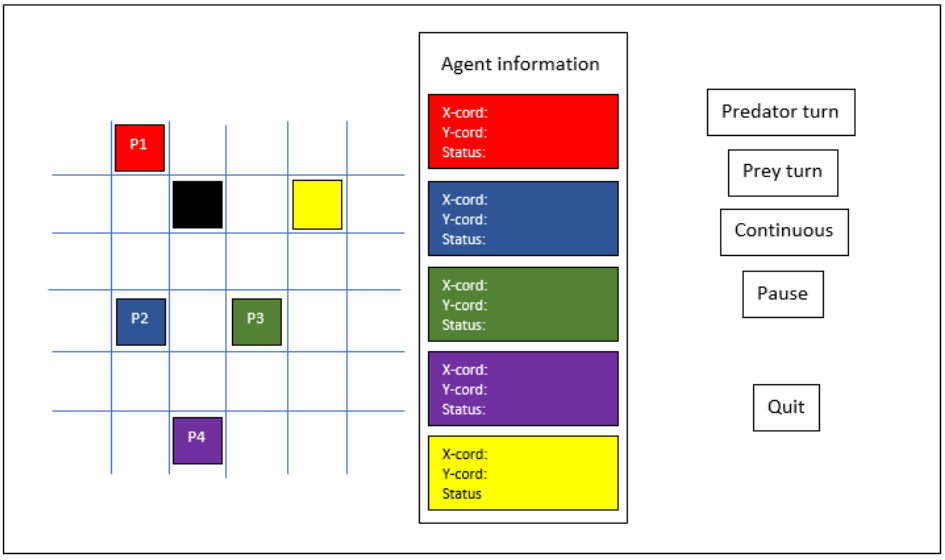
5.8 Gantt chart



Task	Start Date	Duration	End date
Background research	04-Jun	11	15-Jun
Development of project design	15-Jun	12	27-Jun
Setup simple communication on EV3	27-Jun	5	02-Jul
Test accuracy and consistancy of sensors and motors	02-Jul	3	05-Jul
Setup random movement of EV3	05-Jul	3	08-Jul
Implement on-fly calibration techniques	08-Jul	7	15-Jul
Implement collision avoidance	15-Jul	4	19-Jul
Implement random movement by Prey	19-Jul	3	22-Jul
Implement search and pursuit algorithm	22-Jul	6	28-Jul
Combine communication and search and pursuit algorithm	28-Jul	5	02-Aug
Improve on-fly calibration techniques	02-Aug	10	12-Aug
Implement trapping algorithm	12-Aug	5	17-Aug
Final presentation + software demo	17-Aug	7	24-Aug
Software experimentation and analysis of results	24-Aug	14	07-Sep
Dissertation writeup	07-Sep	14	21-Sep

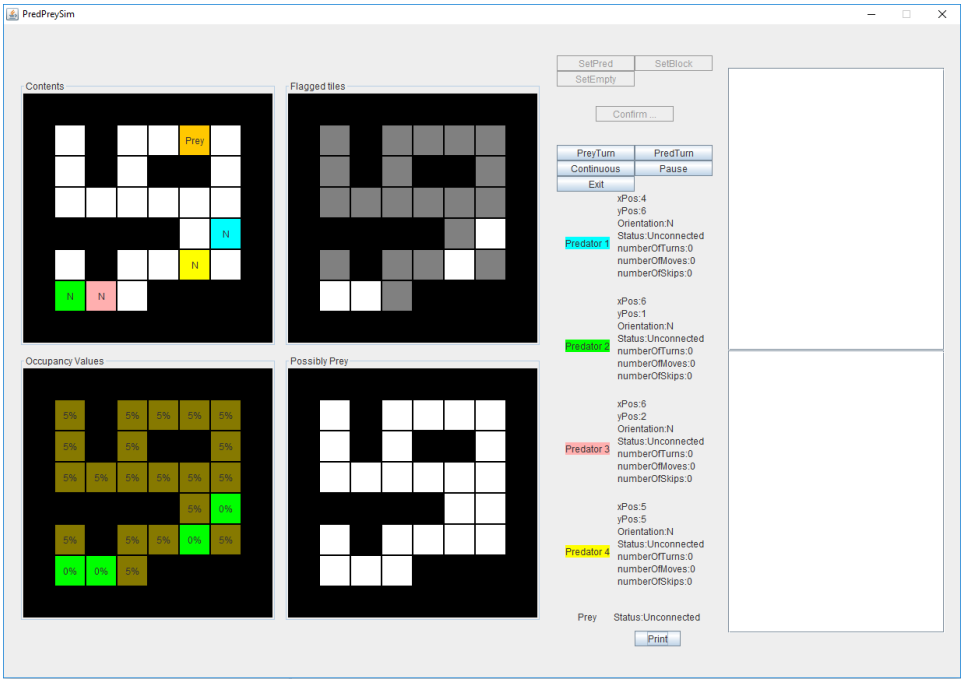
5.9 User interface design (Mock)





Chapter 6

Final piece of software



The user will be welcomed by an interface allowing them to place predators, obstacles or free space, this can be done by toggling "setPred" and clicking on the map, a similar process can be followed to place obstacles or free space. When the user has created a map they can confirm it using the "Confirm" button, which will SSH into each EV3 robot and launch their individual programs.

Once all robots have started and connected to the blackboard it will output a beep indicating that the user may start controlling the flow of the episode. The user interface will also show 4 separate grids displaying information about current locations of each agent, locations that still need to be searched, the chance that a cell is occupied and finally, cells that predators believe the prey to be located within.

To control the order of operations in the current episode the user has three options, "PredatorTurn", "PreyTurn" and "Continuous" which starts the Predator turn followed by the Prey turn until the "Pause" button is clicked or until the prey has been captured.

The user interface will show the moves that the prey has made, potential moves that each agent deems favourable in the current state and the

moves that the negotiation policy decides are globally beneficial (moves which agents are permitted to take). This will occur twice during each predator round however this process will be further detailed within the "Inner workings" section of this report.

During each round in the episode the user interface will update the amount of skipped turns, moves, and rotations that each agent takes. At the end of the episode the interface will automatically close and the episode will be pushed to folder.

6.1 Inner workings

6.1.1 Preys perspective

The prey's policy involves turning to a random orientation prioritizing both front and sides, scanning after each rotation and, if the location is empty moving into it. The prey can neither pinpoint their current location nor the differences between predators and obstacles making more strategic thinking difficult. The Prey stops once it has reached a state in which it has turned to every orientation and verified that each side is blocked, it does this twice to take into account erroneous scanning.

6.1.2 Predator Perspective

The predators scan in front of themselves, informing the blackboard of any readings that return a distance of 0 and should be empty implying that the prey has been seen. After confirmation or rejection about whether this scan is the Prey the blackboard then updates its internal belief state to represent this. Agents then request up to date information regarding the beliefs including information relating to the status of the prey (Found or Captured) and locations of other predators. Depending on whether the Prey has been found at the time of request agents will then perform an altered BFS search, which either calculates the actions needed to travel to the nearest un-searched cell, or to the preys possible movement positions (the locations that the Prey could make if it were to take a turn).

Once the agent has decided upon the moves that it deems beneficial these are then sent to the blackboard which tries to maximize the amount of movement and "Global instantaneous reward" from negotiation, this process is detailed within 6.1.4.

After calculating the global policy each move is sent back to the Predators, if the agent needs to rotate it may do so else it sits idle whilst other agents rotate. Each agent that rotated now re samples the environment, informing the blackboard of beliefs gathered. The blackboards internal beliefs are updated and each agent then requests up to date state information, recalculating their favourable moves using the BFS search method described in 6.1.3, If the move selected does not require rotating then this move is sent to the blackboard who calculates the best possible move set using the same method described in 6.1.4 (due to the state of the environment changing between the first negotiation and the second, the choices made by the agents as well as the negotiation strategy may change). The blackboard then outputs the viable move set to all agents. The agents who were allowed to move now rescan the environment one final time, update the blackboards state and finally retrieve an up to date map at the end of the round.

6.1.3 The BFS search used for exploration

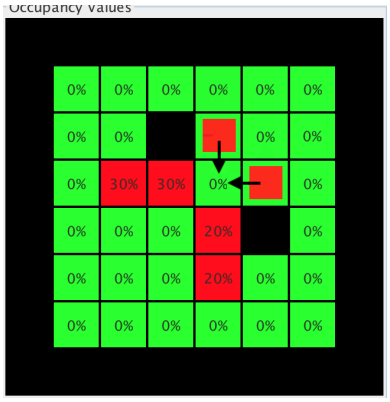
The final search technique used was an altered BFS that worked out the possible actions that would reach the nearest unexplored tiles taking into account cost for moving through other agents, this technique did not work out the single path that reached the destination but the multiple paths that could reach it. The method used for exploration as well as capture is located at appendix B.

6.1.4 Negotiation strategy

Upon receiving the move set from each agent the blackboard attempts to look at all possible permutations of action selection policies available to it, pruning joint actions which are conflicting i.e would result in collisions. The blackboard reduces the list of permutations by trying to maximise the amount of non-null moves, if there are still permutations which exist at this stage in negotiation process then permutations will be selected using either instantaneous reward which is discussed below or at random.

6.1.5 Instantaneous reward

Instantaneous reward (The reward if agents were to move to the chosen locations and scan forward with a range of 2 cells) becomes relevant when we have map states like the one shown below, consider that there are two agents one at (2,4) and another at (3,5) facing South and West respectively. There are two maximal strategies, the first being that agent 1 moves south to occupy (3,4), the second which allows agent 2 to move to (3,4), in this case the decision chosen should be the one that will have the highest reward which in this case would be agent 2 gaining a reward of 60% compared to the possible 40% if agent 1 were to have moved.



The fact that agents can rotate and remain still should be taken into account by the negotiation strategy as it may be beneficial. Consider the example above however the two agents are now facing North and East respectively, if each were to turn South and West and scan forward it is possible that one, or both scans would yield useful information. Under this circumstance the negotiation manager should output the strategy that turns both agents. Using the new information would allow for a more informed decision to be made by the negotiation manager.

6.1.6 Map updating (Dissipation grid)

It was found that agents were able to track the prey more efficiently using a dissipation grid which decreased the time and variance taken to catch

the prey. This method was inspired by by Alberto Quattrini Li, Fancesco Amigoni, Raffaele Fioratto and Volkan Isler [17] who describe decomposing a map into viewable space. Using this they were able to decide important decisions about the movement of predators that would maximize the information gained, and "approximate" possible locations of the prey given previous perceptions as well as the amount of movement that the prey has available. The method used is described below.

Method

1. Assign each tile which is occupiable by the prey an equal distribution called its occupancy chance.
2. During the predators turn if a cell is scanned distribute its occupancy chance to all cells which could possibly hold the prey.
3. If a cells occupancy chance is above a threshold value mark it to be searched in the future.
4. At the preys turn take each cells occupancy chance and distribute it to its neighbouring cells and itself given the distribution of the choice the prey has available.
5. If a cells occupancy chance is above a threshold value mark it to be searched in the future.

6.1.7 Map (Capture definition)

The predators are classed as capturing the prey if all adjacent cells are occupied and all predators are facing the prey to prevent false capture due to erroneous readings.

6.1.8 Calibration methods

Throughout the project I have been developing ways to improve the process of maintaining location and orientation within the environment. I ended up using both color sensors and a gyroscope which complemented each other, each having benefits and drawbacks which are stated below some of which were solved by using an ensemble of the two calibration techniques:

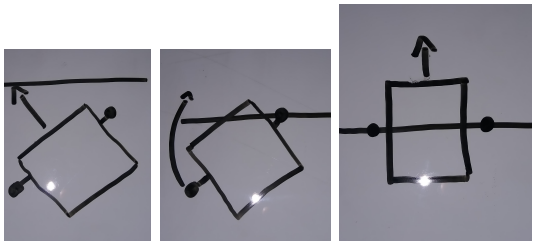
Gyroscope	
Benefits	Drawbacks
Fixes geometry with a 1° error range	Subject to cumulative errors
Quick to perform, minimal overhead	Doesn't maintain location within environment only orientation.
Assists with collisions with the environment	

Line alignment	
Benefits	Drawbacks
Fixes orientation as long as it can be successfully carried out.	Expensive to perform
Fixes location.	If orientation is not maintained then distance as well as orientation can be made worse.
Not subject to cumulative errors	Can cause issues with hanging if line alignment fails.

There are various benefits to be gained from line alignment if it can be carried out successfully, however this technique requires that the agent be at the approximate orientation which is where the gyroscope excels. Because line alignment is not subject to cumulative errors it is possible to use it as a recalibration method for gyroscope usage.

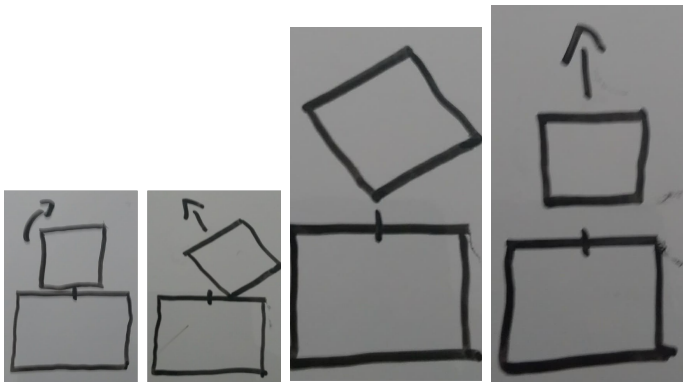
Line alignment

Given a map with black grid lines a robot that has 2 color sensors is able to detect which color sensor hit the line first and adjust accordingly, code located in appendix A.1.



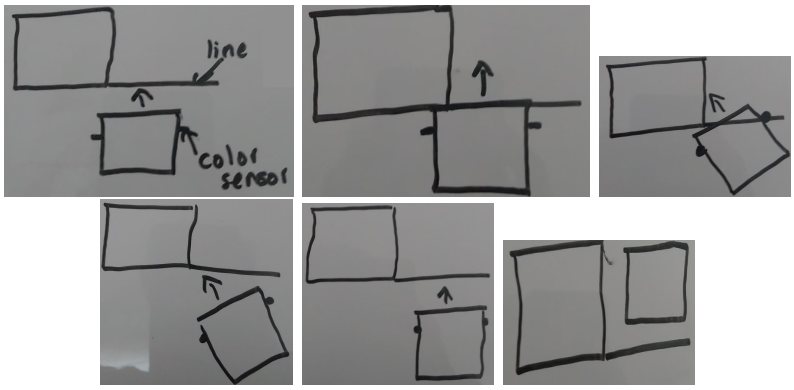
Gyroscope

The gyroscope is able to measure the speed of turning to derive the actual angle rotated, due to this its results are prone to errors often being a degree off. When rotating the gyroscope stores the angle it believes the agent has turned, in the case that the agent has collided with an object this angle would be much lower, using this information we can correct for this, multiple times if necessary. In the case where multiple attempts are required, often the agents location within the environment would also be affected. I tried to compensate for this by moving forward for every extra gyroscope correction needed, code is located in appendix A.2.



Ensemble (Gyroscope and Line Alignment)

When using line alignment it is possible that agents hit a wall on one side of the robot preventing full line alignment from occuring. In such a case, motors would stop and then the motor on the opposite side of the agent would continue to turn. This would continue indefinately without user input using pure line alignment. To fix this I decided to make a check which would stop attempting line alignment after a finite amount of time, reverse and fix orientation using the gyroscope. It would then re-attempt line alignment, this whole process had the benefit of correcting position in the environment as well as maintaining orientation, code located in appendix A.1. This mechanism is shown below.



6.1.9 Ultrasonic sensor

The piece of software uses EV3 ultrasonic sensors however there were a few limitations that were discovered when using these which will be discussed later. First I will discuss the various different ways the EV3 sensor can be used.

- Distance mode: The sensor continuously polls trying to retrieve up to date distance information.
- Listen mode: The sensor does not poll the environment at all instead it "listens" for other agent's ultrasonic sensors.

Issues discovered

Because distance mode constantly samples when not being called it means that inherent noise is produced, this noise can be removed by turning off the sensor using the enable and disable methods. However alternating the use of these methods quickly results in the sensor becoming unresponsive

preventing any further scanning. Instead of switching the sensor off and on we can instead switch between the two modes stated above, this still runs into the issue of the agent "Getting stuck" in a mode however does not hold up the system. For some reason making small changes to the agents orientation allows for further switching after "Getting stuck" additionally the agent can still check which mode it is in by retrieving samples. Because of this we can check that the sensor is in the correct mode after switching and if not rotate a small amount and try again.

There were issues relating to angled surfaces and other robots within the system. This was aided using multi-sampling which was conducted by taking the mode distance after multiple scans. To try to account for angled surfaces the agent turns its front motor to $-15°$, $0°$ and $15°$ allowing for obstacles to be detected even if they were slightly angled, code located in appendix A.3.

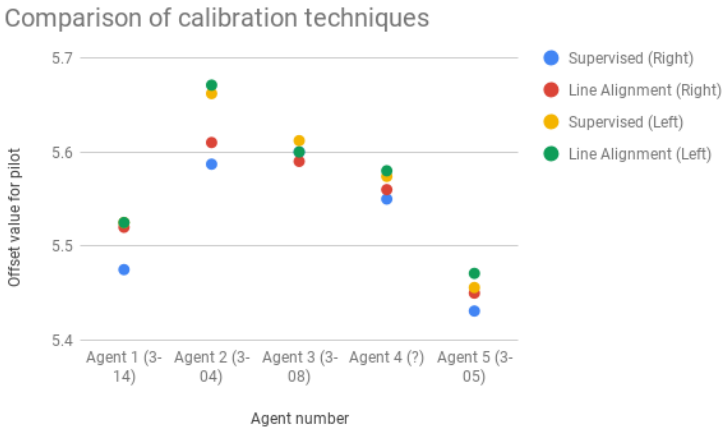
Chapter 7

Realisation - Physical implementation and analysis

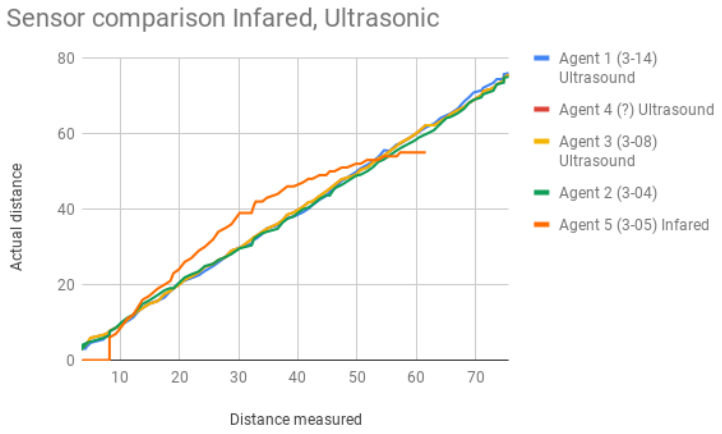
Within this section I will be describing the process of creating the piece of software mainly focusing on issues related to calibration of robots, including where possible details about issues descovered as well as solutions with performance metrics.

7.1 Pre calibration

I compared two different methods for calibrating motors. Supervised learning, carried out by rotating the agent 360° and then labelling the turn as either overturn or under-turn, in comparison the unsupervised method labelled using line-alignment. Both methods performed similarly, looking at the calibration values we notice that the variance between agents is substantial (If we were to use the highest value on the lowest agent it would result in a 12° difference when turning 360°). Because of this I decided that before every simulation it would be essential to assign each agent its own calibration values to ensure that movement was as deterministic as possible.



Next I moved onto comparison of sensors: Infra-red and Ultrasonic. In terms of accuracy I found that infra-red was difficult to retrieve reliable readings from due to the measuring cone being so narrow, causing it to hit the floor, because of this I only managed to gather readings from one agent rather than multiple. Looking at the results we see the infra-red sensor was inaccurate during mid regions, Ultrasonic sensors measuring distance accurately and consistantly between all agents.



Because ultrasonic sensors use sound waves to measure distance it makes them prone to issues relating to angled surfaces. When measuring an object at a distance of 5 and 15cm any angle above 40° had a significant impact on the distance measured, moving onto 25cm the angle of the object became much less impactful.

7.2 Initial set-up

7.2.1 Search technique

The search technique used was a BFS which didn't take into account the cost of moving through other agents, this technique returned multiple beneficial moves where possibleB.1.

7.2.2 Negotiation strategy

The moves provided from each agent were then used by the blackboard which tried to maximize the amount of allocated moves and sends these back to each agent.

7.2.3 Map updating

If we enter a state where all cells have been searched, the blackboard then sets all cells to be re searched.

7.2.4 Classification of a turn

Due to the system being turn based it was important to have a clear understanding of what a single turn for an agent consists of.

Predators:

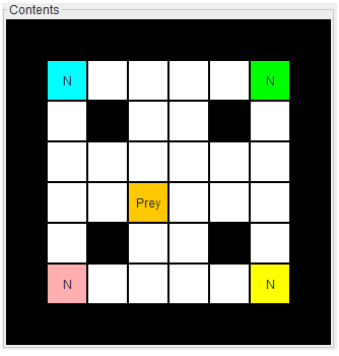
- An initial scan at the start of the turn
- A movement including rotation, forward movement and empty moves.
- Another scan at end of turn.

Prey:

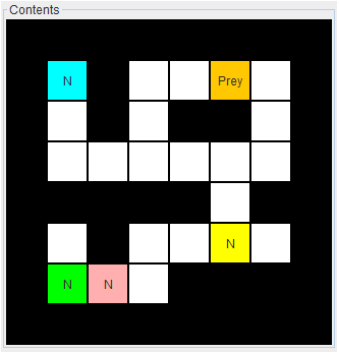
- An initial scan at the start of the turn
- A movement including rotation, forward movement and empty moves.

7.3 On-fly calibration

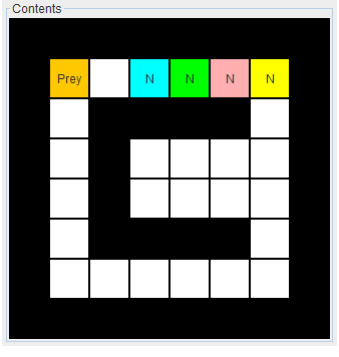
I decided to test the performance of each calibration method on 4 separate maps which are displayed below each containing a different characteristic.



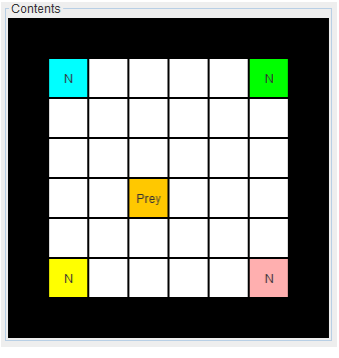
(Map 1) Restricted vision



(Map 2) Heavily obstructed



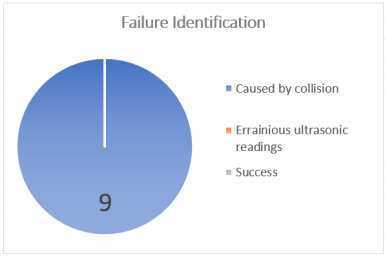
(Map 3) Heavy negotiation

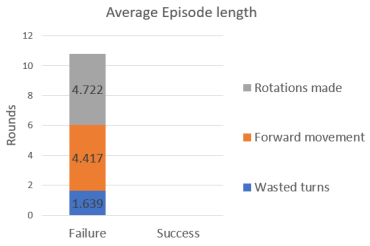


(Map 4) Heavy planning

To test the impact of on-fly calibration techniques (Line alignment, gyroscope) I carried out the simulation on map 1 for all combinations of calibration techniques.

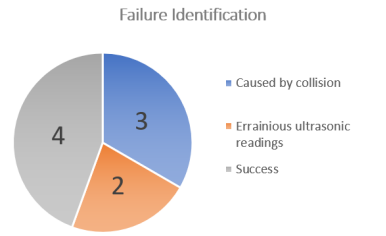
7.3.1 Control (No calibration)

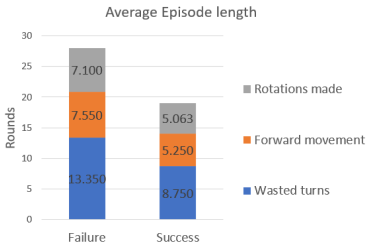




All episodes failed, all agents loosing not only their orientation due to wall collisions but also location within due to rotations also moving the agent in the environment.

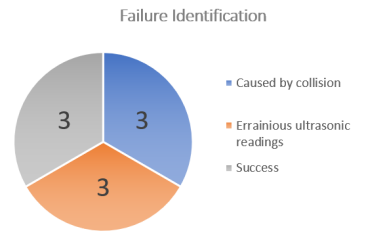
7.3.2 Gyroscope

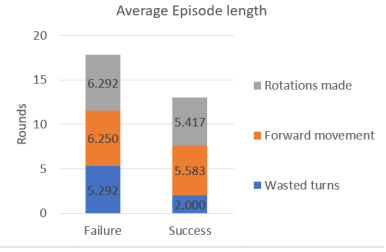




44% of episodes succeeded, failed runs were due to loss of position in the environment caused by the agent's center of gravity being slighty to the front of the robot, rotations also changing its location in the environment.

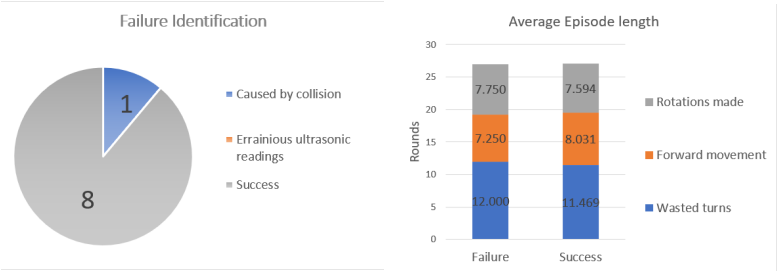
7.3.3 Line alignment





33% of episodes succeeded, failed runs were due to loss of orientation within the environment due to collisions with walls.

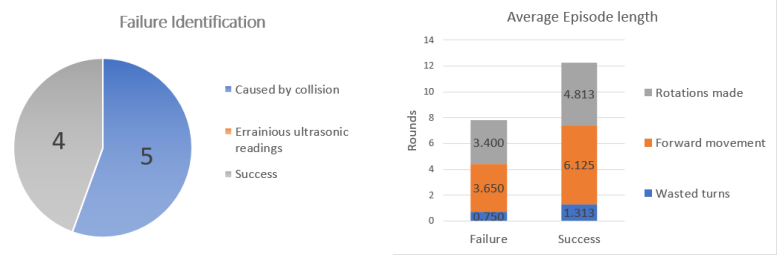
7.3.4 Ensemble (Line alignment and Gyroscope)



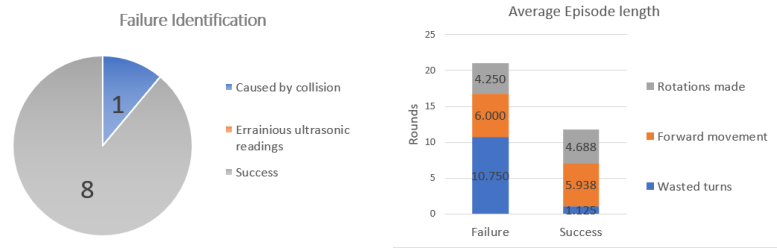
88% of episodes succeeded, both gyroscope and line alignment complemented each other, failed runs resulted from sensor errors which led to collisions. Failed and successful episodes were of similar length, indicating that further improvement on this map would be difficult without addressing sensor issues.

7.4 Continuous improvement of calibration techniques

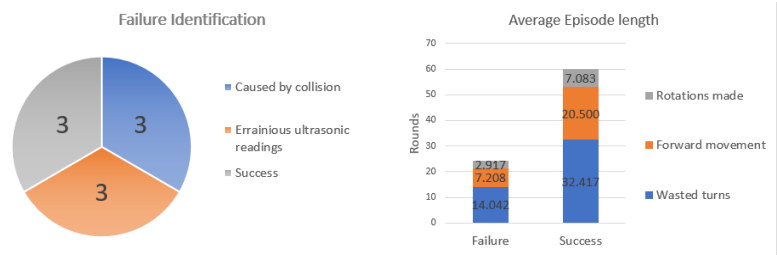
7.4.1 Map 2



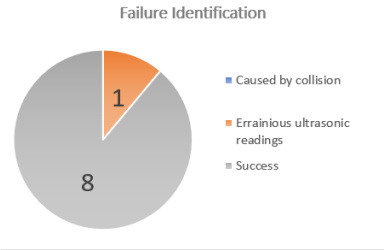
Failures resulted from collisions within the environment this was caused by heavy impacts with walls, the current mechanism used to fix these collisions used a single gyroscope correction when multiple were necessary.

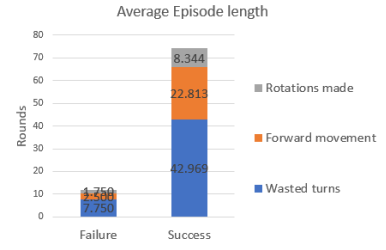


7.4.2 Map 3

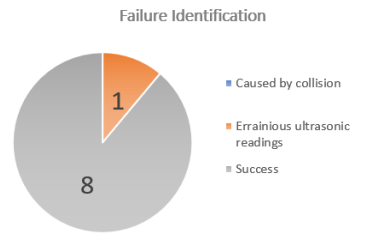


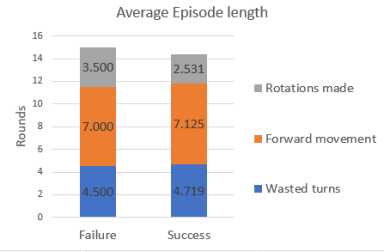
Similarly to map 2, failures resulted from collisions, gyroscope corrections maintaining orientation within the environment however friction with walls caused agents position to be altered leading to future issues when attempting line alignment, to help allevate these issues I used the methods described in the final product 6.1.8 6.1.8.



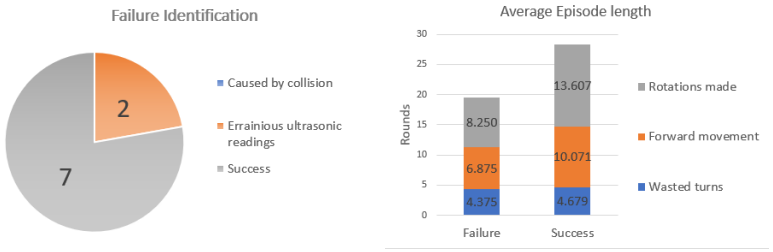


Making these changes increased the success rate however episodes lasted an exorbitant amount of time. This was caused by the predators search technique not taking into account the cost waiting for others. To help incentivize moving rather than waiting I decided to add a cost when planning journeys passing through others B.3. This substantially reduced capture time as shown below.



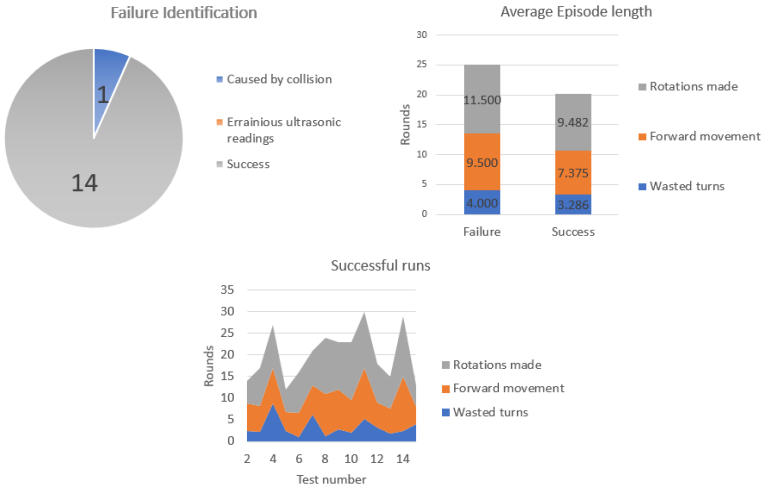


7.4.3 Map 4



Failures resulted from ultrasonic sensor readings which were effected by the noise produced by other agents due to them polling the environment continuously. This was the point that I implemented the changes mentioned within the final product 6.1.9 of switching between sensor modes as well as adding multi-sampling, which increased performance.

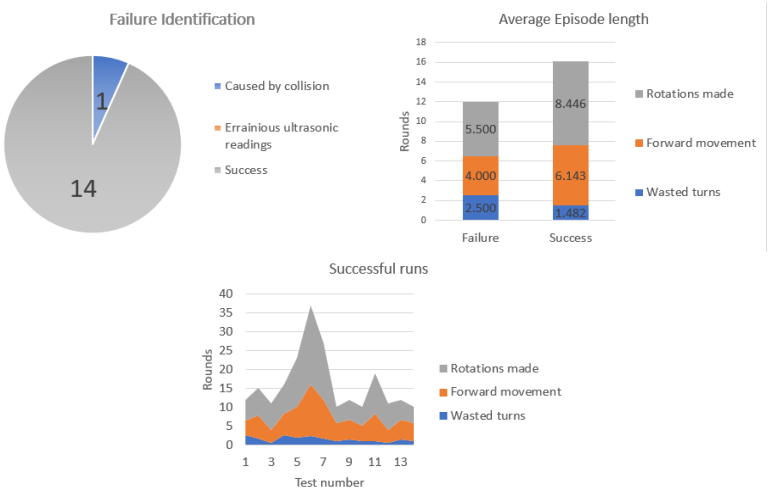
7.5 Dissipation update



At this point I decided to use the dissipation grid 6.1.6. We can see that on average it took 2 turns more to capture the prey and reduced the max by 5 when using this method. Because these differences were so small it was impossible to state with certainty that the method is better or worse than its counterpart. The dissipation map was developed to assist with tracking of the prey during evasion, because of this I tried to change the constraints put upon the prey to improve this aspect:

- Make any number of rotations.
- Make any number of scans.
- Move forward once during its turn if it has not rotated.

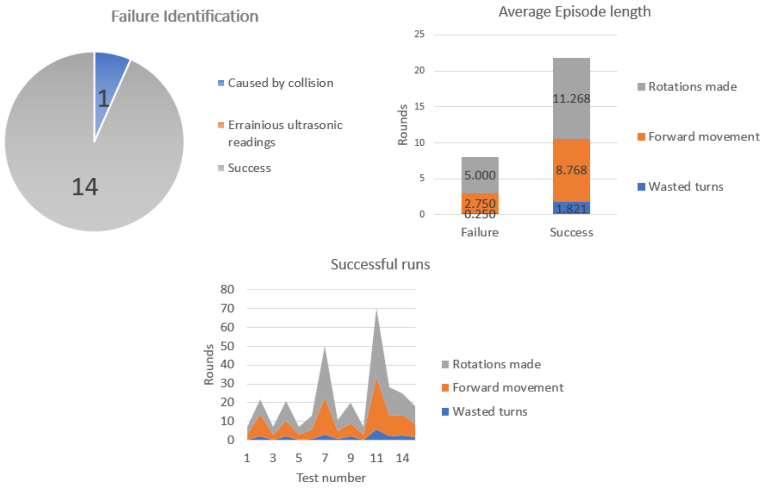
7.5.1 Map 4 without dissipation grid



The maximal time taken to capture the prey increased and the average and minimum time decreased to further explore why this may be the case I have decided to further increase the movement of the prey.

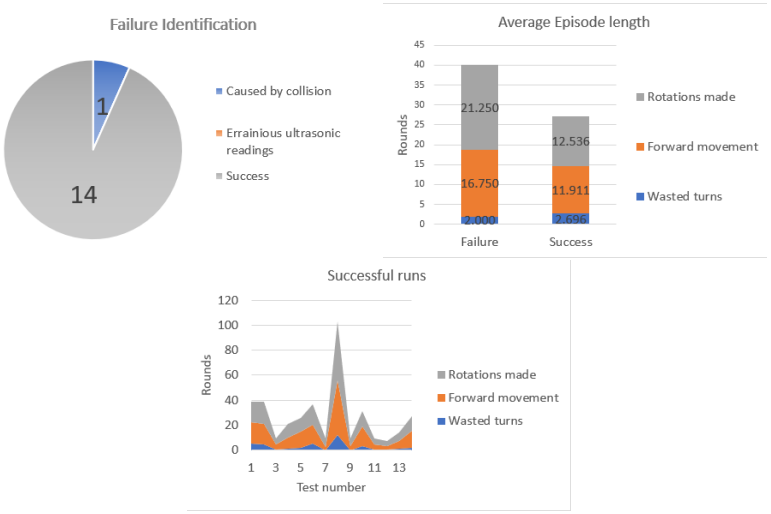
- Make any number of rotations.
- Make any number of scans.
- Move forward once.

7.5.2 Map 4 without dissipation grid



The maximum time increased and the minimum time decreased looking at episode 3 (Appendix C.1.1) I noticed that the prey actually moved in a way that aided predators with capture, moving into the corner. In comparison episode 11 (Appendix C.1.2) which was the longest episode consisted of the prey evading the predators, repeatedly being lost during the process of capture.

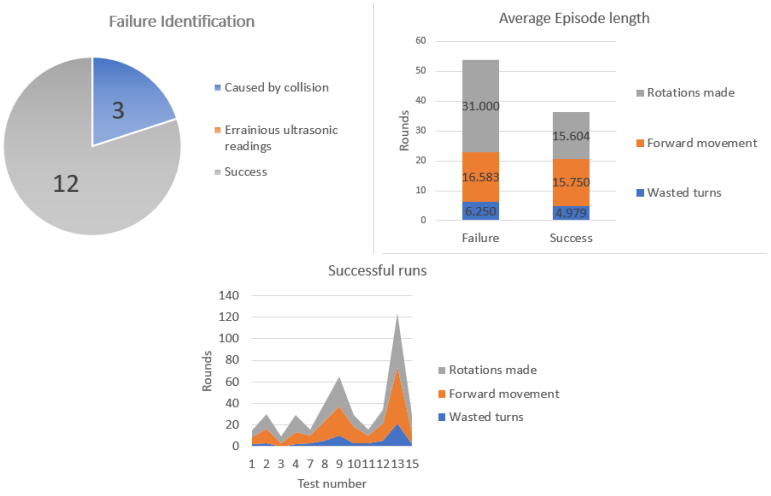
7.5.3 Map 4 with dissipation grid



As you can see the method had a low minimum and a extremely high max value. Looking at episode 8 (Appendix C.2.1) I noticed that the reason for this was different than without the dissipation grid. The prey was able to outrun predators however they were also able approximate its position much more accurately creating an evolved following behaviour this actually hindered the predators. When using the current search method the agent rotates 50% of the time, because of this I decided to implement a search technique which also took rotation cost into account.

7.5.4 Map 4 with dissipation grid and altered search method to take rotation cost into account

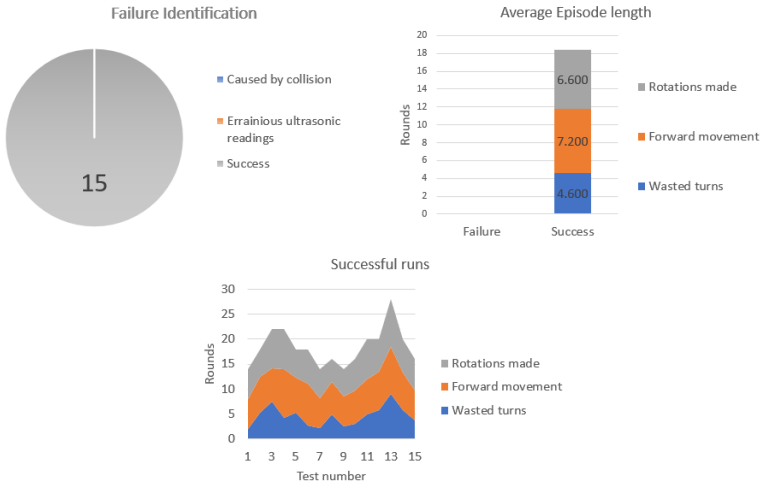
The altered search method attempted to put an emphesis on the cost associated with turning if that turn would consume an action, this method is described in the appendix B.4



Using this new policy did not seem to help, this was due to a reduced number of valid policy combinations submitted to the blackboard, this meant that negotiation was almost non existent. No further improvements were possible at this stage so I moved onto increasing the amount of movement available to the predators to see how this effected capture times.

- Make one rotation followed by a single movement

7.5.5 Map 4 with dissipation grid

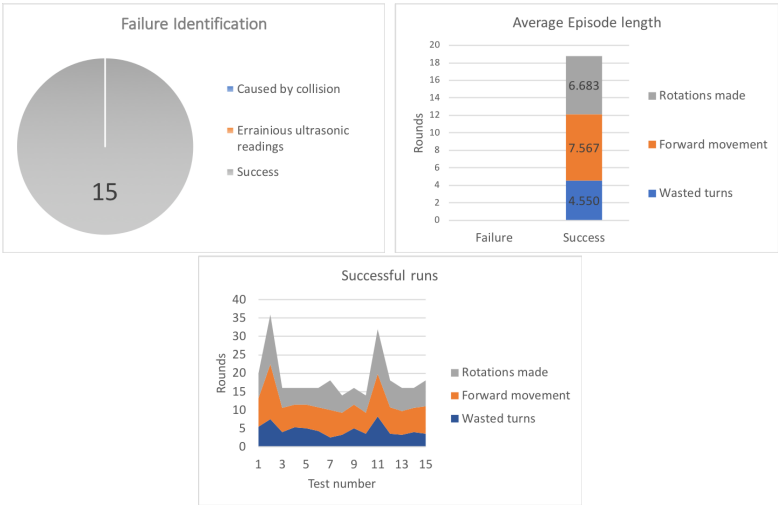


Using the new movement policy I noticed that the amount of turns taken to capture the prey lowered substantially in terms of max and average value. Looking at the longest episode I noticed that most of the time taken consisted of locating the prey rather than capturing it, due to this product being demonstrated it is invaluable to show such evasion is still possible. Once the prey had been located it was successfully captured within 7 turns.

After still feeling that performance was lacking I decided to try to improve the system by utilising data derived from the dissipation grid.

7.6 Instant Reward

I decided to implement the method discussed previously 6.1.5 however it seemed as though it negatively effected results, this was due to the dissipation grid not fully capturing the policy of the prey. The prey being biased towards forward and side movements.



7.7 Testing

7.7.1 Search techniques

To verify each search technique works as intended I will test each in a simulated environment before gathering synthetic data to make conclusive decisions about the performance of all methods discussed in this project. Proof of testing is located at appendix E.

Test	Predator Search policy	Outcome	Changes required
1	Pure BFS	As Expected	N/A
2	Pure BFS Incentive to move	As Expected	N/A
3	Pure BFS Incentive to move rotations cost	Agent had a more reduced move-set than it should have	Found that the search method had major issues requiring it to be remade from scratch. After making changed worked as intended.
4	Pure BFS rotations cost	As Expected	N/A

7.7.2 Predators mechanisms

Test	Description	Expected outcome	Outcome	Changes required
1	Line alignment at 45° angle.	Robot should align with line and continue	As expected	
2	Left wheel hit wall right wheel hit line.	Robot should stall on wall for a few seconds then should reverse and turn and move forward to retry line alignment	As expected	
3	Erroneous sensor value causing prey sighting	Robot should wait for other agents to explore erroneous value, after erroneous value has been clarified by other agents agent should treat value as being obstacle and re-plan their move-set	As expected	
6	Agent scans length that does not agree with map	Agent should rescan, agent should then reduce the length to the maximum length and return to server.	As expected	
7	Agent hits wall whilst turning causing it to move in environment	Agent should attempt to fix its location by accounting for this wall friction by moving forward	As expected	
8	Agent moves into another agent due to erroneous sensor information	Agent should continue to try to move into cell for a couple of seconds then reverse, fix orientation and then rescan.	Robot moved forward as expected however when reversing moved too little.	Had to adjust the robot to move backwards 3cm further on scanning obstacle. Rerun, ran as expected.
8	Agents orientation is hijacked	Agent should fix its own orientation using gyroscope before any future scanning / movement	Agent did not fix orientation before scanning front when orientation was hijacked	Added orientation correction before scanning. Rerun ran as expected.

7.7.3 Prey mechanisms

Test	Description	Expected outcome	Outcome	Changes required
1	Line alignment at 45° angle.	Robot should align with line and continue	As expected	
2	Left wheel hit wall right wheel hit line.	Robot should stall on wall for a few seconds then should reverse and turn and move forward to retry line alignment	As expected	
7	Agent hits wall whilst turning causing it to move in environment	Agent should attempt to fix its location by accounting for this wall friction by moving forward	As expected	
8	Agent moves into another agent due to erroneous sensor information	Agent should continue to try to move into cell for a couple of seconds then reverse, fix orientation and then sit still.	Robot did not move far enough backwards.	Increased reverse distance by 3cm. Rerun ran as expected.
8	Agents orientation is hijacked	Agent should fix its own orientation using gyroscope before any future scanning / movement	Agent did not fix orientation before scanning producing erroneous results	Added gyroscope correction before robot scans. Rerun ran as expected.

Chapter 8

Evaluation

8.1 Software

The first aim of this project was to create a physical implementation of the predator prey pursuit problem that allows users to define a map and track the progress of the predators. This has been successfully implemented the user interface showing all aspects of the program including internal beliefs and agent choices. Additionally two posters as well as a user guide have been made to help provide easy use of this piece of software, these have been included as an appendix F, G, G.

The initial system used a simplistic move policy for predator and prey using the method discussed by Korf [8] which uses a greedy policy that penalises moving through others which was one of the research aims. Collisions as well as negotiation were dealt with by the blackboard. This system Investigated the possible improvements to policys, as well as looking at move policies which work well together (Result in capture in a reliable amount of time and produce interesting use cases). This paper explored the use of noval techniques to track the prey such as the dissipation grid.

The system managed to capture the prey 94.8% of the time over 134 different runs compared to 0% over 10 runs when using no on-fly calibration methods at all.
The different combinations of move policies and "on" and off-line calibration techniques throughout the project have been taken note of to provide a thorough exploration of the benefits.
In addition to the program described above I have also created a method to turn this demonstration into a practical activity, where predators follow a set strategy and the prey moves using decisions made by the player, if the player manages to survive a certain number of rounds they "win" else they loose and can try again.

Aims and objectives

At the beginning of this project I reviewed previous policies implementing one inspired by Korf ensuring that this policy could be ran on multiple EV3's in a turn based system. I created this piece of software using an agile methodology using git-hub to backup working versions of the program incrementally and have discussed the issues as well as benefits that both on-fly calibration and off-line calibration can introduce. I have reviewed the performance of different polices on different maps finding that there is no "best" search policy, as performance not only relies on the amount of

movement available to the predator and prey but also the map structure.

Shortfalls

The initial product was intended to be ran on a sphero as the prey, however due to the following issues this was unable to be done.

- The Sphero's light was too dim in comparison to the environment to locate using saturation.
- The Sphero's body is round preventing the use of ultrasonic sensors.
- The Sphero is made of clear plastic preventing the use of infra-red sensors.
- The Sphero's programming allowed the detection of a collision however the force needed to trigger this event makes the sphero very quick preventing the predators from ever finding it.

The initial product was meant to be ran in a continuous system, however due to difficulty with interference between robots this was unable to be completed in time. Additionally a continuous system may reduce the negotiation required, instead I opted to focus on the agents in a turn based system.

Due to all aims being for-filled I would consider this project to be a success. It would be helpful to continuously improve this system over time adding new move policies such as those that utilise machine learning. It would be useful to gather feedback from students on open days to find out how they would improve the program or expand upon it to make it more understandable.

8.2 Discussion of findings

Below is a discussion of performance when predators use certain attributes and policys on specific maps. These attributes are described below as well as previously in this report, the statistics that these findings are based upon are located in the appendix with sample sizes of 100 for each. D.

- Incentive to move around: The agent takes into account the time wasted waiting for other agents to act incentivising moving around each other B.3.
- Turning adds distance: The agent takes into account the cost of turning it if were to only be able to rotate or move during a single round. i.e if the agent was facing North then it would only take a single action to move to the North cell however it would take 2 actions to move to the East, South or West tiles B.2.
- Instant reward: The blackboard tries to approximate the knowlege that would be rewarded by taking a group action policy and then scanning6.1.5.
- Dissipation Grid: The map is updated after every round to describe potential movements that the prey could have performed6.1.6.

Map 1

Incentivizing movement around other predators and the use of a dissipation map proved useful. Under some circumstances the addition of adding a cost to turning actually improved results slightly (Predator: Turn or move, Prey Turn or move). When using instant reward without using dissipation updates performance increased slightly, indicating that the reason why instant reward under performed previously was not that the concept was poor but the way that knowledge is represented within the system was too insubstantial or inaccurate to use.

Map 2

The composition of the map is heavily restricted, this not only had an impact on predators but also the prey as it had less room to escape. Because of this the standard deviation of runs is very low in comparison to other maps making trends difficult if not impossible to be seen (Predator: Turn and Move, Prey: Turn or Move & Predator: Turn or Move, Prey: Turn and Move). When the prey is allowed more movement within this map it seems as though predators are able to catch it more reliably and quickly we can also see that the incentive to move still plays a heavy impact on this map.

Map 3

We see that all policy combinations are greatly impacted by the use of incentivizing agents to move around each other, we can also see that in certain circumstances the use of a dissipation grid outperforms the full refresh method (Predator: Turn and Move, Prey: Turn and Move).

Map 4

Some runs were uneffected by not having an incentive to move around others (Predator: Turn and Move, Prey: Turn or Move) we also note that the major

feature controlling the length of the episode was the addition of adding a cost to turning which seems to effect all runs negatively apart from when the predator and prey were only able to turn and move, in which case this attribute became beneficial.

Comparison to physical results

Finally when looking at the similarities between physical and non physical capture rates I noticed a high degree of variance, this may either be due to sensor inaccuracy when searching for the prey, or due to small sample sizes, to verify which of these reasons is the major contributor it is necessary to increase our sample size, however this would take longer than this project's length.

Chapter 9

Conclusion

9.1 Localisation

This paper has investigated the various issues and opportunities that line alignment and gyroscope usage within a grid world can offer. Finding that the combination of these two methods produces incredible results. We have also explored a potential issue that using multiple agents within one environment can introduce such as ultrasonic interferance, and have investigated ways to deal with this issue.

9.2 Results

We have seen that all features can be beneficial given unique circumstances not only relating to the policy of both the predator and prey but also the composition of the map. Heavily restricted maps favouring movement around other predators whereas open environments such as map 4 can be negatively impacted by the same technique.

9.3 Further research

It would be interesting to look at what sort of impact adding a learnt strategy would have for the prey such as q-learning or sarsa in a obstacle orientated map when predators are only using a basic search strategy. It would also be interesting to look into the continuous predator prey pursuit problem as well as real time negotiation to look how these aspects affect performance such as approximating prey position and communication costs. It is also important to further explore the methods of calibration that could be used in the future such as using camera's as well as convolutional neural networks to predict position as well as orientation within the environment. Finally it would be interesting to explore how much of an effect sensor inaccuracy within the current system effects the overall time taken to capture the prey however this would require substantial physical testing which is outside of the scope of this project.

Chapter 10

Learning points

10.1 Skills and knowledge

Creating a piece of software that distributes work to agents and combines this work into a single solution on a centralised server was something I had never explored but found gratifying. I also worked with Swing and have come up with a product which shows all information within the system in a timely and clear way without delays something which I have had trouble with doing previously.

I have utilised Java's conditional objects to provide a way to deal with race conditions and synchronisation whilst tasks were being processed such as negotiation, until now I had only ever worked with threads to deal with halting actions such as waiting for user input.

Issues relating to accuracy became extremely important during this project adding an aspect to programming that I had not accounted for in the past. I had to not only look at the functionality of the piece of software but also deal with these issues such as reading erroneous sensor values.

10.2 Crucial actions

Because the project focused heavily on robotics the facilities that I needed were only available at certain times during the day, during these times it was vital for me to be on site. It required me to carry out the physical tests during this time and analyse results at home, doing this allowed me to exploit these periods to the fullest. I also worked on producing a secondary piece of software that simulated the movement and scanning of agents within the system, allowing me to test new methods when facilities were not available, these methods could then be directly translated into the physical version of the program.

10.3 Things that would be done differently

Work could have been optimized earlier in the process allowing the results from these runs to be used for analysis purposes. Some of the issues with the system were not dealt with as quickly as they could have, often overlooking these issues to carry out multiple tests even though errors were evident such as those of gyroscope collisions. If I had taken more time at the beginning of the project analysing the different components and their respective API's

I would have changed the sensor used for an NXT ultrasonic sensor which allows for single sampling.

Chapter 11

Professional skills

It has been critical to follow the BCS code of conduct making sure all requirements have been met. Below are a subset of these that are closely related to my project.

11.1 Professional competence and integrity

During milestone reviews I revisited the plan discussing any requirements I felt were too complex to complete within the remaining time such as real time implementation [18](2a). It was necessary to review previous literature relating to the predator prey pursuit problem such as issues and solutions. I have made substantial use of the Lejos API [19] and have followed the standards put forward by Oracle within their Code Conventions booklet[20] [18](2c). Because my Project is going to be used in demonstrations it was necessary to seek criticism of my work and act upon the advice offered by others, one such criticism related to the look of each agent within the system, being advised to add something to both predator and prey to help distinguish between them [18](2e).

11.2 Duty to Relevant Authority

Towards the end of the project I suggested that the product produced within this project could be used outside of open day activities, requesting permission to get in touch with external groups to prevent conflicting interest between myself and my supervisors [18](3b). A user manual and statistics relating to the performance of the various methods available within the system has been included, as well as the raw run data to ensure that information is not misrepresented or withhold [18](3e).

11.3 Duty to the Profession

Due to the room that I was working in being closed it was necessary for me to communicate with members of The University of Liverpool computer services support to get access to these facilities whenever possible when doing so I acted with integrity and respect [18](4a). Finally I have discussed and given advice to other members of the BCS as well as other disciplines encouraging the development of the software being built suggesting improvements where possible [18](4f).

[18]

11.4 Ethical use of data

This project has not used any human data or human participants and has only used data that is freely available in the public domain.

Bibliography

[1] R. Mantle, "Higher education student statistics: Uk, 2016/17 - subjects studied," Jan 2018.

[2] M. Benda, V. Jagannathan, and R. Dodhiawala, "On optimal cooperation of knowledge sources - an empirical investigation," Tech. Rep. BCS–G2010–28, Boeing Advanced Technology Center, Boeing Computing Services, Seattle, WA, USA, July 1986.

[3] T. Haynes and S. Sen, "Evolving behavioral strategies in predators and prey," in *ADAPTATION AND LEARNING IN MULTIAGENT SYSTEMS*, pp. 113–126, Springer Verlag, 1996.

[4] K.-C. Jim and C. Lee Giles, "Talking helps: Evolving communicating agents for the predator-prey pursuit problem," vol. 6, pp. 237–54, 02 2000.

[5] L. M. Stephens and M. B. Merx, "The effect of agent control strategy on the performance of a dai pursuit problem," in *Proceedings of the 10th International Workshop on Distributed Artificial Intelligence*, 1990.

[6] J. M. Vidal and E. H. Durfee, "Recursive agent modeling using limited rationality," in *ICMAS*, 1995.

[7] R. Levy and J. Rosenschein, "A game theoretic approach to distributed artificial intelligence and the pursuit problem (abstract)," vol. 13, 12 1992.

[8] R. E. Korf, "A simple solution to pursuit games," in *11th International Workshop on Distributed Artificial Intelligence*, pp. 183–194, 02 1992.

[9] T. E. Haynes, K. F. Lau, and S. Sen, "Learning cases to compliment rules for conflict resolution in multiagent systems," 2002.

[10] J. Reverte, F. Gallego, and F. Llorens, "Extending korf's ideas on the pursuit problem," in *International Symposium on Distributed Computing and Artificial Intelligence 2008 (DCAI 2008)* (J. M. Corchado, S. Rodríguez, J. Llinas, and J. M. Molina, eds.), (Berlin, Heidelberg), pp. 245–249, Springer Berlin Heidelberg, 2009.

[11] E.-I. Osawa, "A metalevel coordination strategy for reactive cooperative planning," in *1st International Conference on Multiagent Systems*, pp. 297–303, 1995.

[12] J. Schrum, "Competition between reinforcement learning methods in a predator-prey grid world," Tech. Rep. AI08-9, The University of Texas at Austin, Department of Computer Sciences, November 2008.

[13] J. A Alcazar, "A simple approach to the multi-predator multi-prey pursuit domain," 01 2011.

[14] S. D. Bopardikar, F. Bullo, and J. Hespanha, "A cooperative homicidal chauffeur game," in *2007 46th IEEE Conference on Decision and Control*, pp. 4857–4862, Dec 2007.

[15] V. S. PATSKO and V. L. TUROVA, "Level sets of the value function in differential games with the homicidal chauffeur dynamics," *International Game Theory Review*, vol. 03, no. 01, pp. 67–112, 2001.

[16] A. W. MERZ, "The homicidal chauffeur," *AIAA Journal*, vol. 12, pp. 259–260, Mar 1974.

[17] A. Q. Li, F. Amigoni, R. Fioratto, and V. Isler, *Session 47: Robotics: Planning*, p. 1693–1701. AAMAS, 2018.

[18] "Bcs, the chartered institute for it trustee board ...." `https://www.bcs.org/upload/pdf/conduct.pdf`.

[19] "Lejos api." `http://www.lejos.org/ev3/docs/`.

[20] "Code conventions for the java programming language: Contents." `http://www.oracle.com/technetwork/java/javase/documentation/codeconvtoc-136057.html`, Apr 1999.

[21] C. Stachniss and W. Burgard, "Particle filters for robot navigation," *Foundations and Trends® in Robotics*, vol. 3, no. 4, pp. 211–282, 2014.

[22] F. Abrate, B. Bona, M. Indri, S. Rosa, and F. Tibaldi, "Multirobot localization in highly symmetrical environments," *Journal of Intelligent & Robotic Systems*, vol. 71, pp. 403–421, Sep 2013.