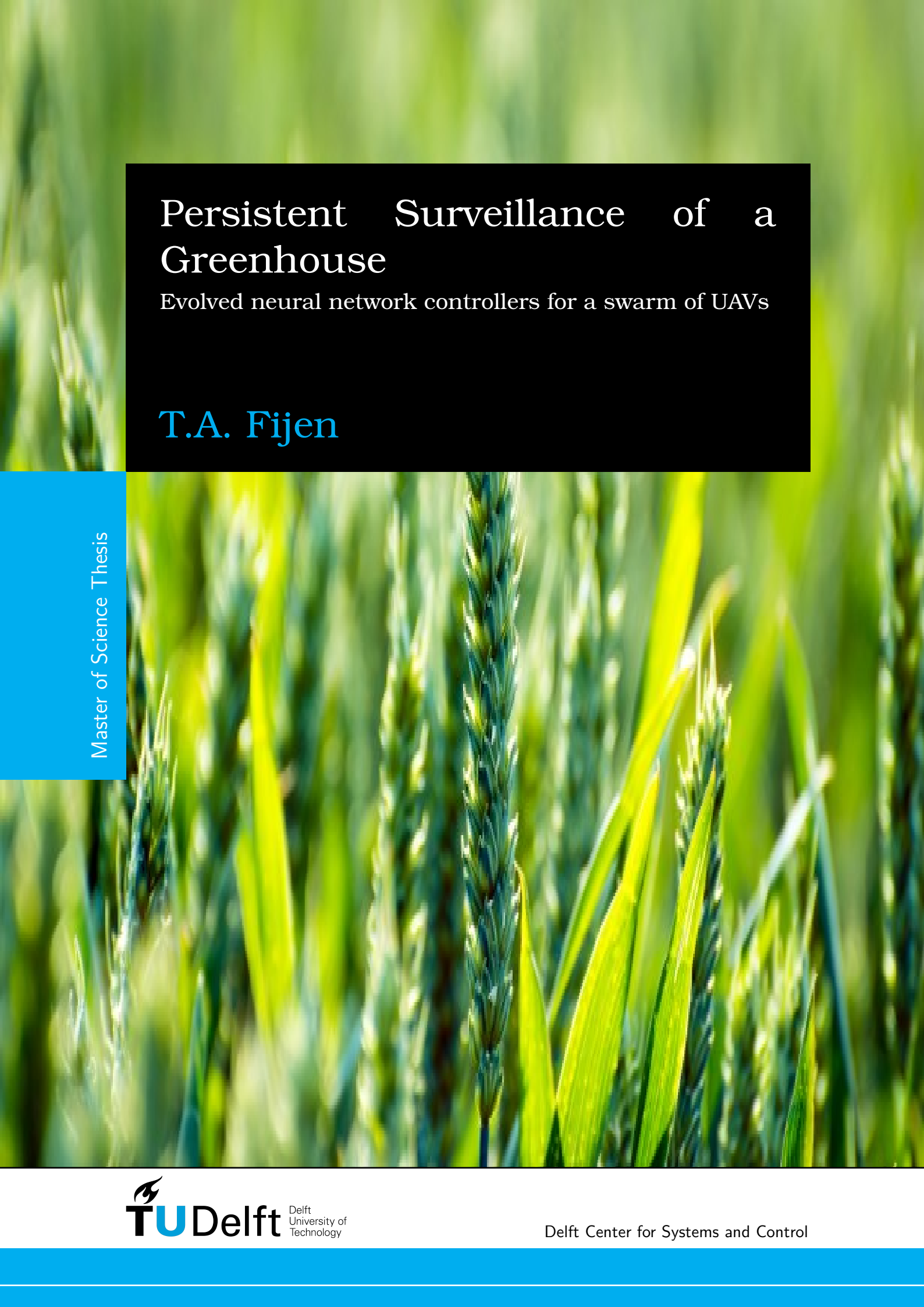


Persistent Surveillance of a Greenhouse

Evolved neural network controllers for a swarm of UAVs

T.A. Fijen

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

Persistent Surveillance of a Greenhouse

Evolved neural network controllers for a swarm of UAVs

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

T.A. Fijen

December 28, 2018

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Abstract

With the growing population the agricultural industry needs to find and implement new methods for enhancing food production. Using a Micro Aerial Vehicle (MAV) in Precision Agriculture (PA) offers a large number of benefits such as enabling the farmer to create targeted strategies to increase crop yield, reduced waste and halt the spread of diseases. Despite these advantages, the use of MAVs, particularly in greenhouses, is still very limited. To this end, this thesis seeks to combine, improve and implement existing strategies to solve the persistent surveillance task for a swarm of MAVs operating in a greenhouse environment.

Broadly speaking, the persistent surveillance task seeks to find the optimal paths for a swarm of MAVs such that every point within the Mission Space (MS) is visited and they must minimise the time between successive visits. This will ensure that the MAVs are able fly through the entire greenhouse to collect up-to-date data about all the crops and the local environment. Naturally, on a physical system one has to deal with the limited flight times of the MAVs. This factor becomes very important to the effectiveness of the solution and is critical to the continuous operation of the MAVs.

In literature, many methods have be proposed to solve this task, but the majority are still only tested in simulation. As a result, many works do not consider some physical constraints that will be applied to the system during implementation in a real-world setting. For example, in most cases the authors do not consider the limited fuel available to the agents or they do not consider a practical alternative indoor positioning system to GPS. In this work the problem has been divided into two main sub-tasks, namely; the persistent surveillance task and the refuelling task.

For the persistent surveillance task it was decided to implement a reactive controller, in the form of an evolved Neural Network (NN), which was run on-board the MAVs. The NN used positional information from the other members of the swarm along with limited environmental information to supply its MAV with a command velocity. These NN controllers could achieve coverage levels of over 95% while simultaneously avoiding collisions between 8 MAVs in a $25m \times 25m$ MS. Later, this method was shown to be robust to failures and scalable in terms of both MS and swarm size.

When dealing with the fuel constraints, a Behaviour Tree (BT) was used to determine when

the MAV should return to the depot. Surprisingly, when combined with the NN controllers the system experienced an increase in performance across all the defined metrics. No MAV failed due to low fuel levels, coverage increased to 97.41%, average cell age to 52.39s and the number of tests were no collisions were recorded more than doubled. This increase in performance was attributed to the fact that the refuelling periodically drew the MAV towards the centre of the MS. This is counter to the evolved behaviours of the NN where the MAVs would mainly focus their attention around the edges of the MS.

Table of Contents

Acknowledgements **xiii**

1 Introduction **1**
1-1 Precision Agriculture . . . 2
1-2 Persistent Surveillance . . . 2
1-3 Thesis Outline . . . 4

2 Problem Outline **5**
2-1 Problem Statement . . . 5
2-1-1 Controller Architecture . . . 5
2-1-2 Problem Formulation . . . 6
2-1-3 Reducing the Problem . . . 7
2-2 Problem Analysis . . . 9
2-3 Hardware Setup . . . 9
2-3-1 CyberZoo . . . 10
2-3-2 Parrot Bebop 2 . . . 10
2-3-3 Positioning and Communication System . . . 11
2-4 Conclusion . . . 16

3 The NEAT Algorithm **19**
3-1 NEAT Background . . . 19
3-1-1 Encoding Scheme . . . 19
3-1-2 Key Aspects of the NEAT Algorithm . . . 20
3-1-3 Mutations . . . 22
3-2 Implementation . . . 23
3-2-1 Fitness Function . . . 23
3-2-2 Neural Network Inputs and Outputs . . . 25

3-2-3 Simulated Environment for Evolution . . . 29
3-3 Simulation Results . . . 31
3-3-1 Test Procedure . . . 31
3-3-2 Input Case Results . . . 32
3-3-3 Results for Selected Input Case . . . 35
3-4 Summary . . . 50

4 Fuel Monitoring Policy **53**
4-1 Fixed Threshold Scheduling Method . . . 54
4-2 Behaviour Trees . . . 55
4-3 Implemented Behaviour Tree . . . 56
4-4 Simulation Results . . . 62
4-4-1 Avoidance Behaviour . . . 62
4-4-2 Refuelling . . . 65
4-4-3 Performance with Real World Fuel Constraints . . . 71
4-5 Summary . . . 73

5 Practical Implementation **75**
5-1 UWB Positioning Results . . . 75
5-2 High Fidelity Simulation . . . 78
5-3 NN Results . . . 81
5-4 BT Results . . . 84
5-5 Summary . . . 86

6 Conclusions **87**
6-1 Summary . . . 87
6-2 Future Work . . . 89
6-2-1 UWB Positioning . . . 89
6-2-2 NN Performance . . . 90
6-2-3 Refuelling . . . 90

A Input Case Results **91**
A-1 Input Case 1 . . . 91
A-2 Input Case 2 . . . 92
A-3 Input Case 3 . . . 93
A-4 Input Case 4 . . . 94
A-5 Input Case 5 . . . 95

B Baseline Persistent Coverage Methods **99**
B-1 Sweep Planner . . . 99
B-1-1 Boustrophedon Decomposition . . . 99
B-1-2 Extension to the Multiple UAV Case . . . 101
B-2 Random Walk . . . 102

C Collision Avoidance Proof **103**

Glossary **113**

List of Acronyms 113

List of Symbols 114

List of Figures

2-1 Example of a discretised MS 8
2-2 Parameter estimation and validation 11
2-3 Schematic description of 2-D multilateration 13
2-4 Comparison between OptiTrack and UWB positioning 14
2-5 UWB module 15
2-6 Example of the ranging messages 15

3-1 Genotype mapping example, [1] 20
3-2 Crossover example, [1] 21
3-3 Mutation examples 23
3-4 Comparison of cost functions: Cell age 24
3-5 Comparison of cost functions: Coverage 25
3-6 Sensor examples from [2] 26
3-7 Average age input example 26
3-8 Feeler input example 27
3-9 Simulation MS shapes 30
3-10 Comparison of input cases 33
3-11 Input case comparison: Average cell age 34
3-12 Input case comparison: Average coverage percentage 34
3-13 Input case comparison: Tick counters 35
3-14 Final NN controller: Coverage performance 36
3-15 Final NN controller: Crash results 37
3-16 Final NN controller: Average cell age 37
3-17 Final NN controller: Tick counter examples 38
3-18 Coverage levels for scaled up MS (8 MAVs) 39

3-19 Crash results for scaled up MS (8 MAVs) . . . 39
3-20 Coverage levels for scaled up MS (16 MAVs) . . . 40
3-21 Crash results for scaled up MS (16 MAVs) . . . 41
3-22 Average cell age results for simulated CyberZoo . . . 42
3-23 Coverage levels for simulated CyberZoo . . . 42
3-24 Crash results for simulated CyberZoo . . . 43
3-25 Average cell age results for simulated CyberZoo: 3 Agents . . . 44
3-26 Coverage levels for simulated CyberZoo: 3 Agents . . . 44
3-27 Crash results for simulated CyberZoo: 3 Agents . . . 45
3-28 Baseline cell age comparison . . . 46
3-29 Baseline coverage percentage . . . 47
3-30 Line sweep tick counter . . . 47
3-31 Robustness test 1: Cell age comparison to baseline . . . 48
3-32 Robustness test 1: Coverage percentage comparison to baseline . . . 49
3-33 Robustness test 2: Cell age comparison to baseline . . . 49
3-34 Robustness test 2: Coverage percentage comparison to baseline . . . 50

4-1 Behaviour tree example . . . 55
4-2 Node type examples . . . 56
4-3 Implemented behaviour tree . . . 58
4-4 Charging decision process . . . 59
4-5 Avoidance decision process . . . 60
4-6 Example of adjusted flight path for collision avoidance . . . 61
4-7 Example of collision avoidance behaviour . . . 61
4-8 Coverage levels with avoidance measures . . . 62
4-9 Average cell ages with avoidance measures . . . 63
4-10 Crash instances with avoidance measures . . . 63
4-11 MAV fuel levels . . . 66
4-12 Coverage levels with refuelling . . . 67
4-13 6 MAV flight path examples . . . 68
4-14 Average cell ages with refuelling . . . 68
4-15 Number of crash instances with refuelling . . . 69
4-16 Coverage levels with refuelling . . . 70
4-17 Average cell ages with refuelling . . . 71
4-18 4 MAV refuelling flight results: coverage percentage . . . 72
4-19 4 MAV refuelling flight results: average cell age . . . 72
4-20 4 MAV refuelling flight results: crash percentage . . . 73

5-1 UWB position noise example . . . 76
5-2 Kalman filter performance: Stationary MAV . . . 77

5-3 Kalman filter performance: Mobile MAV 78
5-4 Model comparison for a circular flight pattern 79
5-5 Model comparison for square flight pattern 80
5-6 Model comparison for a circular flight pattern 80
5-7 Model comparison for square flight pattern 81
5-8 Physical Flight Results: Average Age 82
5-9 Physical Flight Results: Coverage Percentage 83
5-10 3 MAV flight Results: Average Age 83
5-11 3 MAV flight Results: Coverage Percentage 84
5-12 2 MAV refuelling flight paths 85
5-13 Individual controller comparisons 85

A-1 Input case 1 results 92
A-2 Input case 2 results 93
A-3 Input case 3 results 94
A-4 Input case 4 results 95
A-5 Input case 5 results 96
A-6 Input case 5 results 97

B-1 Lawnmower search pattern 100
B-2 Boustrophedon decomposition example, [3] 100
B-3 Example of IN and OUT events given in [3] 101

List of Tables

2-1 Ranging update frequencies 16

4-1 Node types and their statuses, [4] 56
4-2 Number of tests with no collisions 64
4-3 Parrot Bebop 2 refuelling parameter values 66
4-4 Real-world refuelling parameter values 71

Acknowledgements

First, I would like to thank my both of my supervisors. To Dr Guido de Croon, my daily supervisor, your guidance and enthusiasm during our meetings were enormously helpful in keeping me focused throughout my thesis. To my departmental supervisor dr.ir. Tamás Keviczky, I found your insights into my topic and your feedback invaluable during my thesis. Together you helped me something that I am proud of and that I enjoyed working on. I truly owe a great deal to both of you. Further, I would like to thank Mario Coppola for all the advice that he offered and the hours that he set aside to help me understand the issues that I ran into while working with Paparazzi. He was always willing to make time for me no matter the problem.

To my hockey team, Hudito H10, our trainings and games together were the ideal way to distract me from my thesis for a while and the camaraderie you offered helped keep me sane over the years. In addition I would like to thank my friends Chris, Michen, Andre and Greg for all their support, their good humour and their willingness to listen as I described my latest round of issues.

Finally I would like to thank my parents and sisters for their love and support over the years, and in particular for inspiring me pursue my masters degree abroad. None of this would have been possible without them.

Delft, University of Technology T.A. Fijen
December 28, 2018

Chapter 1

Introduction

In recent years, the global population has increased drastically which, in turn, has led to a large increase in agricultural consumption [5]. As a result, the agricultural industry must find methods for increasing their productivity while reducing their harmful environmental impacts. One such method is Precision Agriculture (PA). PA, also known as precision farming, is a farming management strategy that focuses on utilizing site specific crop and environmental information to maximise crop yield while reducing inputs and wastage [6, 7, 8]. The quasi-real time nature of the information gathered allows the farmer to identify and quickly react to any harmful changes to their crop.

There are many methods used in practice for obtaining the necessary information but aerial imagery is one of the most commonly used techniques [7]. It has been successfully used to, amongst others, identify weeds ([9, 10]), locate infections ([5]) and for monitoring water stress ([11, 12]). Traditionally there are three methods for obtaining these aerial images, namely; satellite imagery, a commercial aircraft or a Unmanned Aerial Vehicle (UAV). UAVs are becoming the preferred method due to their lower costs, their ability to deliver high resolution images, their availability and their flexibility [13]. Most importantly, UAVs are able to operate in indoor environments such as greenhouses.

To date, most UAV solutions currently available for PA focus on the collection of aerial images by covering the area, once, using predefined back and forth sweeps ([14, 15, 16, 17]). Other solutions include [6, 13] where the UAV was controlled by a pilot, [18] where vision was used to identify target points and [9] where a random walk like approach was used. However, all these papers dealt with using UAVs in outdoor fields, while research focussing on the application of UAVs in greenhouses is far more limited. The only works found dealing with this are the papers by Roldán *et al.* ([19, 20]) which use a single UAV to collect data in a greenhouse by performing back and forth sweeps.

1-1 Precision Agriculture

While PA has received a great deal of interest in outdoor farming applications, it has only been sparsely implemented in a greenhouse environment [19]. In modern greenhouses, the farmer can create different climates and seasons allowing them to grow a variety of different crops throughout the year. However, this can be very complicated as the environmental conditions inside a greenhouse are influenced by a number of strongly coupled factors [21]. Hence, for this to be effective, farmers will be required to obtain regular climate measurements from multiple points around the greenhouse to create an objective representation of the climate gradient [22]. A poorly controlled climate gradient can severely decrease the yield and productivity of the crop and can facilitate the development of several diseases [21, 22]. A Wireless Sensor Network (WSN) offered one method for data collection in a greenhouse and has been used in many agricultural applications.

A WSN is built up of a collection of individual sensor nodes that are used to periodically record the environmental conditions in its immediate surrounding. This real-time data is then used to control the climate inside the greenhouse to increase the yield, reduce energy inputs and for Integrated Pest Management (IPM). In [23] the authors made use of temperature, humidity, illumination and CO_2 sensors to collect real-time data to automatically control the greenhouse to improve the farmers' convenience and productivity. This stored data could also then be used to create an optimal environmental plan for future harvests. Similar to this are the works [24, 25]. In these papers, the WSN were used to create decision support structures specifically for IPM through the use of extensive WSN that collected data every four minutes for [24] and every minute for [25]. In [26] environmental data was again captured in four minute intervals but this can be reduced further depending on the application. These measurements are then transmitted to a central computer which is usually located outside the greenhouse due to the high water content in the air [26].

Unfortunately there are many limitations that have prevented WSNs from being widely deployed in greenhouses. For example, creating the WSN for a $70m \times 150m$ field can require approximately 40 to 50 nodes, [27], which leads to high costs for large greenhouses. Of course this number can vary greatly depending on the goal of the WSN. In [28] further limitations of WSNs were listed as; determining the optimum deployment scheme, routing protocols, energy efficiency, communication range, scalability and fault tolerance.

Due to their high degree of mobility and small size, a Micro Aerial Vehicle (MAV) can offer an alternative to WSN but their use in greenhouses is still very limited [19]. For environmental control and IPM, swarms of MAVs can be equipped with a variety of temperature, humidity and CO_2 sensors and used to collect data throughout the greenhouse. In addition, MAVs can use digital imaging to monitor leaf temperature and water stress which traditional WSN generally cannot incorporate.

1-2 Persistent Surveillance

To replace a WSN with a MAV swarm will require the MAVs to continuously move through the greenhouse obtaining the necessary environmental readings. As mentioned in the previous section, these measurements have to be captured on a regular basis to allow for precise control

of the climate conditions. Thus, obtaining these measurements for use in PA can be described as a persistent surveillance task.

Persistent surveillance, is similar to Coverage Path Planning (CPP) in that both methods seek to find trajectories that visit every point in the given area. However, the persistent surveillance task has the added goal of attempting to minimise the time elapsed between successive visits to the different regions in the mission space. In other words, it seeks to continuously cover the mission space while CPP seeks to cover the mission space only once, [29]. In literature, this problem is also referred to as the persistent monitoring or the persistent coverage problem and has received significant interest over the last few years. One of the simplest methods for achieving complete coverage of an area is through the use of a predefined exhaustive search method such as the line sweep method implemented in [30, 16, 31, 14, 17]. These can then easily be adapted to achieve persistence by restarting the algorithm each time the area has been fully searched. The main drawback here is that the method is not able to adapt to changes in the environment. To combat this draw back, other authors relied on reactive controllers. In [32] Nigam developed a Multi-agent Reactive Policy (MRP) that selected the next point for an agent to visit based on the time since that point was last visited and the distance to the other agents. Other works that implemented a reactive control method include [9] which made use of a random-walk like approach, [33] which again used a type of MRP, a Model Predictive Control (MPC) approach used in [34, 35, 36] and an optimal control approach for 1- and 2-D given in [37, 38].

Despite the interest in the subject, there are still a number of issues that require further investigation. First, most of the proposed solutions to this problem do not consider the effect of communication constraints on the performance. For example, the works [32, 39, 33, 36, 40] all assume that each agent has full knowledge of the system. However, there are exceptions such as [34] which compares the performance of its controller for the full, limited and no communication cases and [41] which implements decentralised persistent surveillance in 1D for agents with limited local knowledge.

Next, the problem of fuel management in CPP and persistent surveillance is often not incorporated. In CPP it is usually assumed that the agents can cover the area before running out of fuel, but in [16, 17] the authors made an attempt to include fuel constraints into their problem. Here agents were forced to return to a fuel depot as soon as their fuel level dropped below a certain point. However, the authors did not schedule the refuelling task to avoid congestion at the depot. For persistent surveillance, in his final work [32], Nigam formulated the refuelling task as a Linear Programming (LP) problem which determined the optimal refuelling schedule to limit the congestion at the fuel depot. This approach required a centralised controller with full communication with all the agents.

Other papers, not dealing with persistent surveillance, where fuel management was considered include [42, 43] and the work by How *et al.* ([44, 45, 46, 47]) where the agent hands off its task to another agent when its fuel level drops too low.

From a review of the current literature it was decide to implement an evolved Neural Network (NN) controller. As this is a reactive method, it is able to adapt to changes in the environment which methods that rely on predefined flight paths struggle to do. This is important as the harsh environment inside the greenhouses could cause hardware failures, which must be accounted for. In addition, the NN approach has the added benefit of being more computationally efficient than determining a complete flight path for each MAV. This is

important as it allows the controller to be run on-board the MAV instead of on a separate, more powerful computing platform.

Of course, these advantages are shared by most other reactive methods. An evolutionary algorithm was selected as Evolutionary Robotics (ER) offers a promising approach to designing controllers for swarms of agents performing complex tasks, [48]. This approach results in scalable controllers that are computationally efficient, flexible and requires little prior knowledge about the problem, [2, 48, 49]. Evolution also offers an alternative method for determining the structure of the NN. Naturally the NNs performance is highly dependent on their chosen structure. Selecting the number of hidden layers in the system by hand can be a very difficult process that mostly relies upon trial-and-error and personal experience. Lastly, there is also very little work that focusses on using NN for the persistent surveillance task. During the literature review only the work by Miguel Duarte *et al.*, [2, 50, 51], was found that dealt with this topic. Later, during the course of this thesis, an extended abstract was published that also made use of evolved NN to control MAVs performing a persistent surveillance task. Both of these works were mainly focussed on showing that NNs, particularly those created with the NEAT algorithm, were applicable to the problem of persistent surveillance. They did not give an indication to how well their controllers compared to other methods.

1-3 Thesis Outline

The remainder of this thesis is structured as follows. In the next chapter, the mathematical formulation of the problem is given accompanied by a description of the hardware used to test the final solution. In Chapter 3, the evolutionary approach used in this work is described. The Neuro Evolution of Augmenting Topologies (NEAT) method was used to evolve a NN controller that was implemented on each of the MAVs. The fourth chapter deals with the real-world fuel level constraints that are applied to the system. Here, a Behaviour Tree (BT) is used to coordinate the refuelling task between the MAVs while relying on limited inter-MAV communication. Finally this thesis concludes with a summary of the results obtained and a description of possible future avenues of research.

Chapter 2

Problem Outline

The goal of this chapter is to provide a detailed description of the tasks that were solved during the course of this work. Further, the method of analysing the performance of the solutions is given along with a description of the hardware that was used throughout the thesis.

2-1 Problem Statement

Through a careful analysis of the current literature, the author has found a lack of information dealing with certain aspects of the persistent surveillance task. These are: decentralised control for the persistent surveillance task, inclusion of fuel management and scheduling (especially for use in a distributed/decentralised approach) and incorporation of methods for indoor localisation.

In light of the aforementioned gaps in the current literature, this thesis project seeks to find a solution to the multi-agent persistent surveillance problem that:

1. Does not rely on a centralised controller. Instead it should be implemented in a distributed or hierarchical manner.
2. Imposes real world fuel constraints onto the MAVs. In addition, the MAVs should make use of limited knowledge about the other MAVs and the refuel station (or depot) to plan their own refuelling schedule. This is done so as to avoid congestion at the depots and failure of MAVs to due low energy levels.

2-1-1 Controller Architecture

In the problem statement it is mentioned that the solution must not rely on a centralised controller. This subsection seeks to briefly motivate why this is the case.

For the centralised control structure, there is a single controller that coordinates the actions of the other components (in this case the MAVs) in the system. The main strength of a centralised approach is that it can determine globally optimum solutions. However, for this, the controller needs full knowledge of the system and must have enough processing power to control all the agents, [52, 53, 54]. Due to this, centralised control is not suited to large teams, dynamic environments and has high communication demands. Further drawbacks of this method are that it has a central point of failure and, especially for predefined paths, it cannot quickly respond to changes in the system. This could be an issue as the harsh environment of the greenhouse may lead to a number of agent failures.

When using a decentralised approach, each agent makes use of local knowledge to formulate its own decisions. This type of control is characterised by its reliability, flexibility, robustness and adaptability [54, 53]. This is especially useful as the harmful environment could lead to a number of MAVs failing during operation. Unfortunately, as a result of agents basing their decisions on local knowledge, globally optimal solutions cannot be achieved, [54]. In light of this, the distributed control structure is most suited to applications where a large number of agents are required to perform simple tasks with no strict bounds placed on their efficiency, [54].

The Hierarchical architecture lies in-between the centralised and distributed approaches and attempts to utilise the strengths of both. With this controller structure, there is no central control point for all agents. Rather, one or more agents will act as a local central controller for a small cluster of agents, [53]. This allows the structure to retain some of the flexibility, robustness and adaptability of the distributed controllers as the decision making is distributed amongst the team. This has the added benefit of giving the agents access to slightly more global information than the pure distributed case, improving the performance of the global solution. Naturally this sharing of information amongst members of the cluster gives rise to higher communication requirements.

For this particular problem, there are no strict limits placed on the data collection rate, multiple agents will be used and there is a need for robustness against failure of agents. Hence, a distributed or hierarchical controller is more suited to the problem than a centralised controller.

2-1-2 Problem Formulation

More specifically, the general persistent surveillance problem can be formulated as such; Given:

- A known rectangular mission space $\mathcal{G}$= [0,L_1] $\times$ [0,L_2] $\subset \mathbb{R}^2$ that can contain obstacles $\mathcal{O} \subset \mathcal{G}$ that may not be occupied by a MAV
- M MAVs (or agents), A= $\{A_1, \ldots, A_M\}$, with
  - Initial position $P_m(0)$, $\forall m \in \{1, 2, \ldots, M\}$
  - An energy level $E_m(t)$, $\forall m \in \{1, 2, \ldots, M\}$
  - A maximum energy level $E_{m,max}$, $\forall m \in \{1, 2, \ldots, M\}$
  - A velocity $v_{x,m}$ and $v_{y,m}$, $\forall m \in \{1, 2, \ldots, M\}$

  - An energy depletion rate $\dot{e}_m$, $\forall m \in \{1, 2, \ldots, M\}$

- Q depots with
  - A known position $P_{depot,q}$, $\forall q \in \{1, 2, \ldots, Q\}$
  - A recharge rate $\dot{e}_{depot}$

Determine:

- The set of flight plans, $\mathcal{FP}$ (one for each MAV), where MAVs cannot occupy the same position at the same time and each depot can refuel one MAV at any given time. This is defined as $\mathcal{FP} = \{\mathcal{FP}_1, \mathcal{FP}_2, \cdots, \mathcal{FP}_M\}$ where $\mathcal{FP}_m = \{P_m(0), \cdots, P_m(t_f)\}$, $\forall m \in \{1, 2, \cdots, M\}$.
- The age of each point, $age(x, y, \mathcal{FP}, t)$, where in $(x$,y$) \in \mathcal{G}$ are the coordinates of the points and t is the time. This age is the time since data was last collected at the given point, and is formally defined in Section 2-1-3.
- The *information age*, $\mathcal{I}_{age}$, defined in Equation (2-1) which is taken from [55]. This is used to indicate how often new data is collected by the system.

$$\mathcal{I}_{age}(\mathcal{FP}, t_s, t_f) = \int_{t_s}^{t_f} \left[\int_0^{L_2} \int_0^{L_1} age(x, y, \mathcal{FP}, t) dx \cdot dy \right] dt \tag{2-1}$$

Such that the following optimisation problem is solved:

$$\arg\min_{\mathcal{FP}} \left(\mathcal{I}_{age}(\mathcal{FP}, t_s, t_f) \right) \tag{2-2}$$

$$\text{S.T. } E_{m,max} \geq E_m(t) > 0, \forall m \in \{1, 2, \ldots, M\} \tag{2-3}$$

$$||P_i(t) - P_j(t)||_2 > 0, \forall i, j \in \{1, 2, \ldots, M\}, i \neq j \tag{2-4}$$

2-1-3 Reducing the Problem

First, to reduce the computational complexity of the problem, it is solved in discrete time and the MS is discretised into a 2-D grid, $G(N)$. Each cell, $g(n)$ $|n \in \{0, 1, \ldots, N\}$, in this grid is then assigned an age, $age(n, k)$, which represents the time since the cell was last visited by a MAV. This age is defined by Equation (2-5), where ΔT is the discrete time step, $Dist(n, m)$ is the distance between MAV m and the centre of the current cell and V is the radius of the MAVs camera footprint. This is shown visually in Figure 2-1. If the centre of a cell lies within the MAVs camera footprint, illustrated by the dotted line, then that cell is said to have been visited by a MAV and its age is reset to 100. This is indicated by the green cells in the image.

$$age(n, k) = \begin{cases} 0 & , k = 0 \\ \max(0, age(n, k-1) - \Delta T) & , Dist(n, m) > V \\ 100 & , Dist(n, m) \leq V \end{cases} \tag{2-5}$$

At first glance it may seem counter intuitive to have the age counting down from 100, when the definition of the persistent surveillance task is given as a minimisation problem. However,

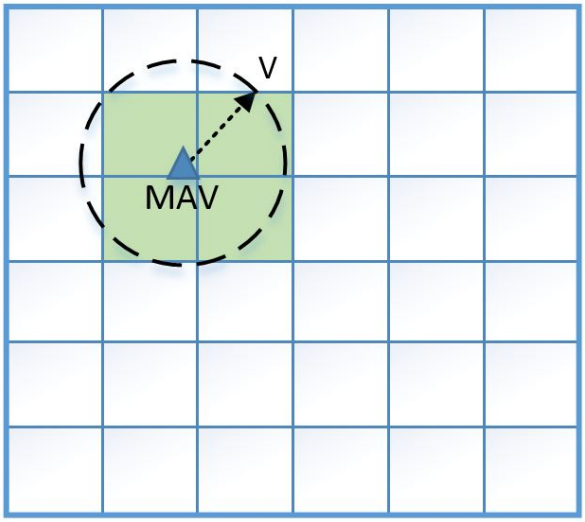


Figure 2-1: Example of a discretised MS

this is due to the method and fitness function used to evolve the NNs, which is explained in Chapter 3.

With this discretisation, the *information age* (from Equation (2-1)) can now be rewritten as:

$$I_{age} = \sum_{k=0}^{t_f} \sum_{n=0}^{N} age(n,k) \tag{2-6}$$

Combining the definition from Equation (2-5) and the above I_{age}, the optimisation problem to be solved is reformulated as:

$$\arg\max_{\mathcal{FP}} \left(\sum_{k=0}^{t_f} \sum_{n=0}^{N} age(n,k) \right) \tag{2-7}$$

$$\text{S.T. } E_{m,max} \geq E_m(k) > 0,\ \forall m \in \{1,2,\ldots,M\} \tag{2-8}$$

$$||P_i(k) - P_j(k)||_2 > 0,\ \forall i,j \in \{1,2,\ldots,M\}, i \neq j \tag{2-9}$$

In literature, this discretisation of the MS into a grid is known as approximate cellular decomposition, while the list of their associated ages is referred to as an *age map*. This gives rise to an important limitation when using the information stored in the age map. To avoid duplication of effort, each MAV must either share the same age map or they must each maintain their own version of the maps that must periodically be synchronised with the other MAVs. In this work, during the simulations, it is assumed that the MAVs have global knowledge of the system while in the practical implementation each MAV maintains their own version of the age map. As they periodically broadcast their positions to all others in range, this information can be used to update their individual age maps with regards to the effects of the other MAVs. During execution, the positional messages are broadcast at a rate of roughly $10.7Hz$ (see Section 2-3-3 for more details) and the age maps are updated at a rate of $4Hz$. This allows the age maps to be sufficiently synchronised across all the platforms.

Finally, as a reactive controller is used in this thesis instead of a global planner, it is not necessary to determine the complete flight plan for the entire duration. All that is needed is the MAVs position for the next time-step.

2-2 Problem Analysis

With the problem described above, the next step is to define performance metrics that will be used to analyse the different controllers generated in this work. Two clear choices can be found in the definition of persistent surveillance. These are: coverage percentage and the average cell age. Further, as the NN controllers are reactive rather than global planners, collision avoidance will play an important role in the success of the controllers. Therefore, the number of collisions between the MAVs is used as the third metric.

Average Cell Age:

This metric is related to the requirement from the problem definition that each of the cells must be visited as often as possible. For this, the ages of each of the cells, as defined in Equation (2-5), are averaged over the entire grid. This is recorded at each time step to show the progression over time. With a lower bound placed on the cell age, this metric can be slightly misleading as the impact of unvisited cells is limited. To combat this, a second method of visualising the persistence aspect of the controller will also be used. A counter is assigned to each cell and whenever it is visited by a MAV ($Dist(n, m) \leq V$), the counter is incremented. This is used to create a heat map of the MS which will show how often the cells are visited during the tests. In literature this is has been referred to as a tick counter.

Coverage Percentage:

Like coverage path planning, the persistent surveillance problem also has the requirement that every point/cell in the MS must be visited by an agent. This is recorded by the coverage percentage metric. It uses the tick counter mentioned previously and expresses the number of cells that has a 'count' of at least one as a percentage.

Number of Crashes:

Clearly, for the controllers to be effective in the real world, the MAVs must be able to avoid one another during operation. This is expressed as the number of crashes. During the tests, the minimum distance between any two of the MAVs is rerecorded at each time step. If this distance is less than $30cm$, the MAVs are said to have collided with one another.

2-3 Hardware Setup

This section details the hardware used during the course of this thesis. It is divided into the following section; Physical test environment, the flight platform used and the communication and positional system.

2-3-1 CyberZoo

The real-world testing was done in the CyberZoo of TU Delft. This is an enclosed, $10m \times 10m \times 7m$ research and test laboratory in the faculty of Aerospace Engineering. This area is equipped with a 12 camera OptiTrack system to provide accurate measurement of an objects position, velocity and heading through the use of reflective IR markers. These measurements are used as ground truth data.

During the validation tests, the MAVs will operate within a $7m \times 7m$ flight area at an altitude of $1m$. As it is desired that the MAVs learn to stay within the boundary of the MS during operation, there is no high level controller that will force the MAVs to remain within the area during evolution of the NNs. However, during physical testing the MAVs will be required to land if they leave this MS to prevent them from colliding with the edges of the CyberZoo.

2-3-2 Parrot Bebop 2

The flight platform used during this work is the Parrot Bebop 2, which is controlled by the Paparazzi MAV software[1] (v5.13). The Parrot Bebop 2 is a lightweight MAV equipped with an ultrasound sensor to measure altitude, a 3-axis gyroscope, magnetometer and accelerometer, a front facing (14 mega-pixel fish-eye lens) camera and a bottom camera [56, 57]. In addition it is equipped with an ARM Cortex-A9 processor, a low-tier GPU and 8 GB of flash memory.

As mentioned, the existing, pre-installed autopilot software was overwritten and replaced with Paparazzi. This is an open-source autopilot software developed for hobby and professional use, [58]. For the inner-loop controller of the MAV, an existing controller from Paparazzi was used as this was considered beyond the scope of this project. This controller used the following input variables: the commanded x and y velocities (in the North, East, Down coordinate frame), the flight altitude and the heading.

Simplified MAV Dynamical Model

A core component of this work is the evolution of the NN that will control the MAVs. Naturally, this evolution cannot be done on the actual flight platform as this would damage the MAVs. Therefore, an accurate model of the dynamics of the MAV is needed for the simulation environment. At the same time, a simplified model is desirable as this will decrease the computational complexity of the simulation, thereby reducing the time needed to perform an evolutionary run. With this in mind, the model used by Szabó in [59] was implemented as this was shown to be applicable for an Evolutionary Algorithm (EA).

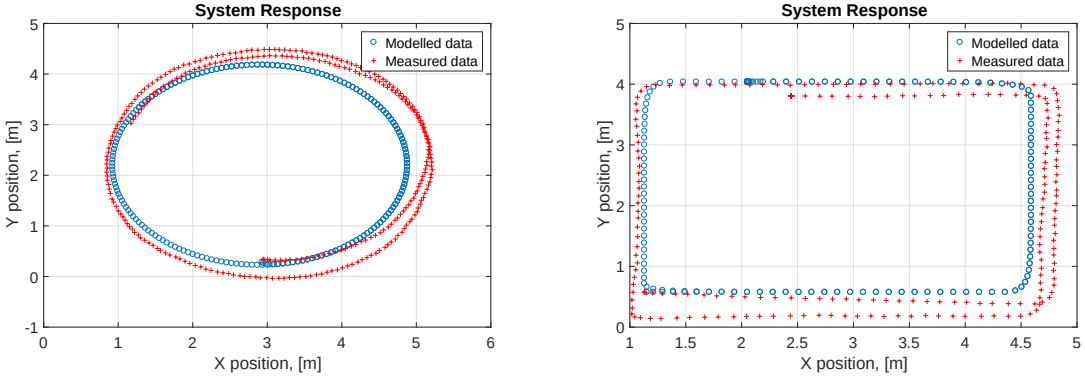
The dynamics for this platform have six degrees of freedom which relates to the translation and rotation along the three axis of the MAV. This is combined with four inputs representing the rotational speeds of the four motors which gives a nonlinear dynamic model. A decoupled linear system can provide a reasonable approximation of the platform under normal flight conditions, [59]. This models the dynamics of the MAV together with the inner-loop control. The approximation used in this work is a linear, 2^{nd} order velocity model in the X and Y

[1] Freely available from: https://github.com/paparazzi/paparazzi

directions, with no axis coupling [59]. This is given in Equation (2-10). As the MAVs will only operate in a 2-D plane, the altitude state was not included in the model.

$$\begin{bmatrix} \dddot{X} \\ \ddot{X} \\ \dddot{Y} \\ \ddot{Y} \\ \dot{X} \\ \dot{Y} \end{bmatrix} = \begin{bmatrix} -2\zeta\omega & -\omega^2 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -2\zeta\omega & -\omega^2 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} \ddot{X} \\ \dot{X} \\ \ddot{Y} \\ \dot{Y} \\ X \\ Y \end{bmatrix} + \begin{bmatrix} \omega^2 & 0 \\ 0 & 0 \\ 0 & \omega^2 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} v_x \\ v_y \end{bmatrix} \tag{2-10}$$

In this model the parameters ζ and ω are estimated through flight test data. The OptiTrack system in the CyberZoo was used to capture the positional information of the MAV during a preprogramed flight. Non-linear least squares was then used to estimate the unknown parameters to minimise the difference between the actual position and the modelled position. The results of this identification process can be seen in Figure 2-2a below. In this case the parameters were estimated as $\zeta = 58.65$ and $\omega = 14.52 rad/s$ which gave a Mean-Squared Error (MSE) of 0.0308 and 0.0408 for the x and y positions respectively.



(a) Results of parameter estimation

(b) Results of parameter validation

Figure 2-2: Parameter estimation and validation

Using these values, the results where validated on new positional data and the comparison between the modelled and the measured position can be seen in Figure 2-2b. Here, the MSE was found to equal 0.0714 for the x position and 0.0700 for the y.

2-3-3 Positioning and Communication System

In indoor areas, greenhouses for example, a reliable method for determining position that does not rely on GPS is needed. There are a number of methods to accomplish this. As in the CyberZoo, an OptiTrack or similar motion capture system could be installed. While this system can give very accurate positional information, there are a number of points that make it unsuitable for the given task. First there is the price. According to the OptiTrack website, a small motion capture system using six of their least expensive cameras will cost $5 918 and can only track eight MAVs in an area of $6m \times 6m \times 2m$, [60]. This brings us to a second issue with this method, which is the area that can be captured by the cameras. Even using 24 or

their most advanced cameras, this capture area is only $15m \times 15m \times 6m$ (and will cost you $147 849) [60]. As a result of these drawbacks, this method was not considered as a viable alternative to GPS.

Next, Simultaneous Localization and Mapping (SLAM) was considered. This would allow the MAV to use its sensor readings to create a map of the environment while at the same to localise itself within that map [61]. The SLAM problem can be implemented using a wide array of sensors, such as LiDAR, laser range finders, vision and RGBD cameras. However, many of these are not applicable for use on small MAVs due to their limited payload and power supply [62]. As a result, the authors of [62] proposed that the best sensor for MAV based SLAM is a single camera. Recently, real-time monocular SLAM has gained popularity as a result of their use in robotics, [62, 63], but there are still some drawbacks. This method is very sensitive to photometric changes, the captured image must contain varying textures and depths, and pure rotational motion causes tracking failures [62, 63].

In the end, position estimation through the use of Two Way Ranging (TWR) and multilateration was chosen. This can be achieved through the use of a system of Ultra-wideband (UWB) transceivers. These small, lightweight modules have a communication range of up to $300m$, they can achieve a positional accuracy of $10cm$ in an indoor environment and they have a low power consumption [64]. As an added benefit, these modules can be used for inter-MAV communication as well as for localisation, reducing the number of sensors needed on the flight platform. Finally, the module are also inexpensive, with a single module costing roughly $\$26$[2]. For these reasons, the UWB localisation system was selected over a monocular SLAM approach. For UWB localisation, a system of *anchors* and *tags* are used. Here, anchors (or beacons) are UWB modules that are placed at four known locations throughout the mission space while the tags are mobile nodes that are placed on the MAVs. The four distances between the *tag* and each of the *anchors* can then be used to calculate the current position of the MAV. This will be explained in more detail in the next subsection.

Position Estimation

The process of determining the current position of the MAV, using the TWR method with multilateration, is shown in Figure 2-3. The four known ranges are used to inscribe a circle around its corresponding beacon, indicated by the dashed lines in the figure. The position of the *tag* is then determined as the point where these circles intersect with one another.

Mathematically these four ranges can be expressed through Equation (2-11):

$$R_i = \sqrt{(x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2} \ , \forall i \in 1, 2, 3, 4 \tag{2-11}$$

where R_i is the range between a MAV and one of the four beacons, with their positions given by (x_i, y_i, z_i). This set of equations can be rewritten in the standard format of a Non-Linear Least Squares (NLLS) problem, Equation (2-12) [65], which is used to determine an estimate of the x, y and z positions of the tag.

[2]Price as of 14 Nov 2018 from: https://www.digikey.com/product-detail/en/decawave-limited/DWM1000/1479-1002-1-ND/4805335

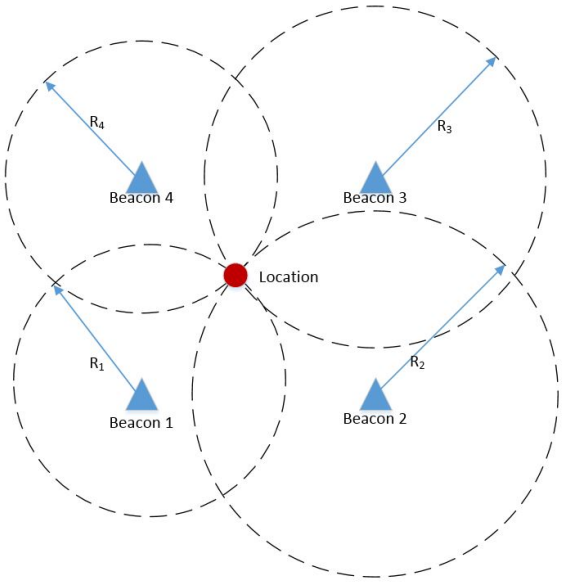


Figure 2-3: Schematic description of 2-D multilateration

$$\min_X ||e(X)||_2^2 = \min_X \sum_{i=1}^{4} e_i^2(X) = \min_{x,y,z} \sum_{i=1}^{4} \left(R_i - \sqrt{(x-x_i)^2 + (y-y_i)^2 + (z-z_i)^2} \right)^2 \quad (2\text{-}12)$$

In the above equation, $e(X)$ is the error vector and is given by

$$e(X) = \begin{bmatrix} e_1(X) & e_2(X) & \cdots & e_4(X) \end{bmatrix}^T \quad (2\text{-}13)$$

When the initial estimate of the position is near the optimum, the Hessian of the function that is minimised, $\bar{H}(X_k)$, can be approximated by Equation (2-14) [66].

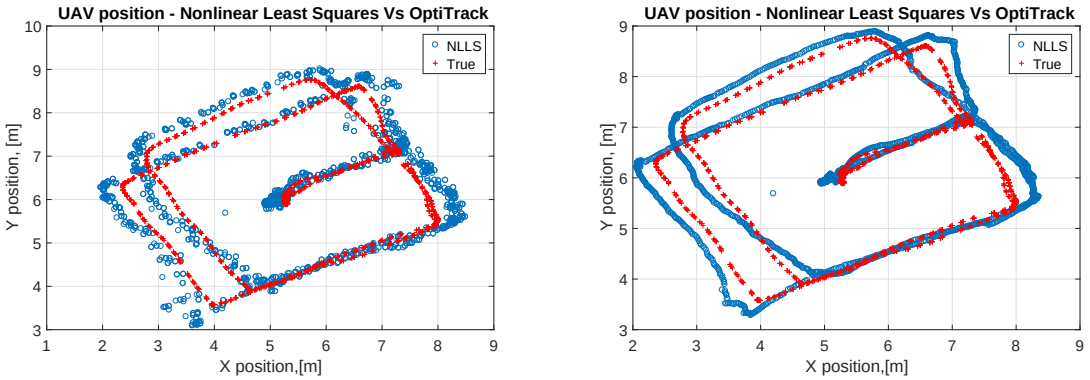
$$\bar{H}(X_k) = 2\nabla e(X_k) e(X_k) \quad (2\text{-}14)$$

where $\nabla e(X_k)$ is the Jacobian of $e(X_k)$. When updating the current position of the MAV, its previous position is used as the initial starting point for the new least squares problem. This ensures that the above approximation will hold as the update rate is fast enough and the MAV speed is slow enough that there will not be a large jump in the position between updates. Now, through the use of Equation (2-14), the NLLS problem can be solved through the *Gauss-Newton algorithm* shown in Equation (2-15).

$$X_{k+1} = X_k - \left(\nabla e(X_k) \nabla^T e(X_k) \right)^{-1} \nabla e(X_k) e(X_k) \quad (2\text{-}15)$$

As this is an iterative process that will not provide the optimal solution within a finite number of steps, relevant stopping criteria are needed. In this work the following was used: $||X_{k+1} - X_k||_2 < 0.01$ along with an upper bound placed on the number of iterations of 20 iterations.

An example of the results of the positional estimation can be seen in Figure 2-4. For this flight, the autopilot used to positional information supplied by the OptiTrack system while it recorded the ranging measurement and its estimated position. In the image on the left, the pure results from the NLLS estimation are given. It is important to note that implementation of the NLLS included a simple outlier rejection. This replaced any range measurement received that was greater than $20m$ with the previous valid ranging measurement for that anchor. As can be seen, this tracks the actual position of the MAV well but the output is rather noisy. To combat this a moving average filter was implemented where the current position was taken as the current estimate averaged with the position of the previous four positions. The results of this are shown in the figure on the right. Initially the moving average filter used the previous nine positions but during tests this made the system slow to respond to changes in the flight direction. It was found that using only the four past positional values gave the filter good noise rejection while limiting the negative effect on the systems responsiveness. With this filter the MSE for the x and y positions is 0.139 and 0.074 respectively.



(a) Without moving average filter **(b)** With moving average filter

Figure 2-4: Comparison between OptiTrack and UWB positioning

DW1000 UWB Modules

The Decawave DWM1000 UWB module[3], controlled by an Arduino Pro Mini $3.3V$ with an ATmega328 microcontroller running at $8MHz$, was used in this positioning and communication system. To connect the UWB module to the Arduino Pro Mini, a specially designed breakout board was used[4]. This is shown in Figure 2-5.

In the existing Arduino libraries for the DWM1000 module[5], there is already existing code which allows one to implement ranging between multiple anchors and a single tag. This makes use of asymmetric two-way ranging which is a TWR method for determining between two nodes that do not have synchronised system clocks. More information on this procedure can be found in [67].

As this allowed ranging to only one tag, this code was not directly applicable to the given problem. However, In a previous MSc thesis [68], S. van der Helm used these UWB modules

[3]https://www.decawave.com/products/dwm1000-module
[4]https://sites.google.com/site/wayneholder/uwb-ranging-with-the-decawave-dwm1000—part-ii
[5]https://github.com/thotro/arduino-dw1000

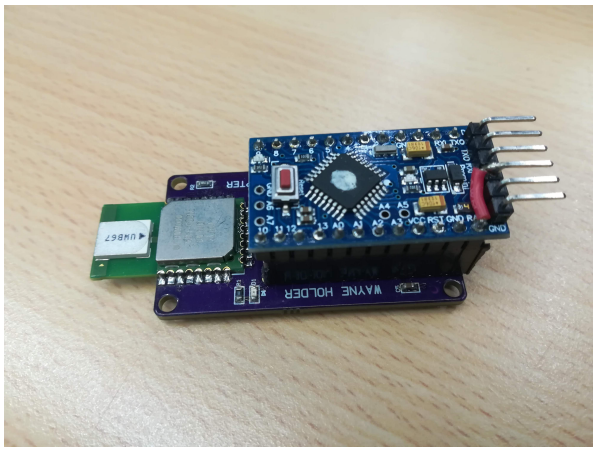


Figure 2-5: UWB module

to perform relative localisation between a swarm of MAVs. For this he adapted the existing Arduino libraries to allow for anchorless communication and ranging between mobile modules. In this implementation, a predefined sequence is used to determine which module in the network may broadcast a message to the other modules. This prevents timing issues where two or modules attempt to broadcast their messages at the same time. This message is comprised of the positional data of the current module and the separate ranging messages to each of the other modules. Modules can then reply to their received ranging messages during their own broadcast turn. Below, an example of the type of message sent is shown

$$\{[\text{ID}], [\text{Pos X}], [\text{Pos Y}], [\text{ID node 1}, \text{message}], \cdots, [\text{ID node M}, \text{message}]\}$$

There are four types of ranging messages that can be transmitted between nodes. These are used to implement the asynchronous two way ranging procedure from the original Arduino library. The messages between two nodes, A and B, are shown in Figure 2-6.

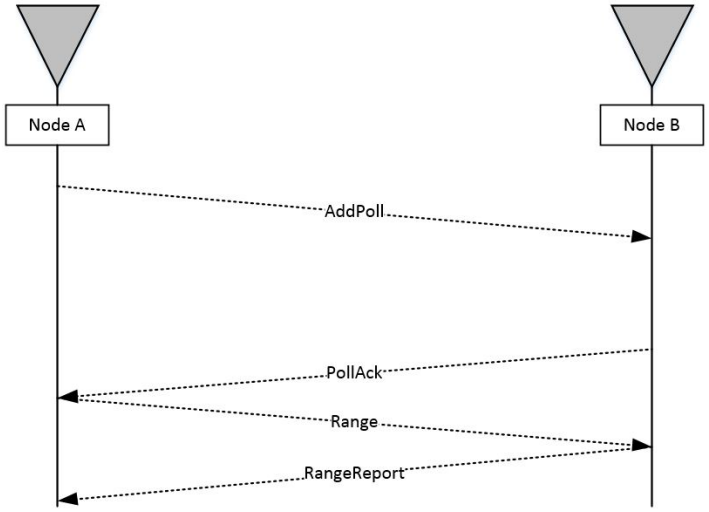


Figure 2-6: Example of the ranging messages

The procedure are as follows:

1. *AddPoll*: This message is sent by node A and it stores the time at which the message was sent. The message initiates the ranging procedure between he two nodes.
2. *PollAck*: Once node B receives the *AddPoll* message, it stores the time the message was received and replies with the PollAck message after a pre-defined delay time. Again the time at which the message was sent is stored by node B.
3. *Range*: Once node A receives the *PollAck* message, it again stores the time and replies with the *Range* message along with the two stored times and its current time.
4. *RangeReport*: After receiving the *Range* message node B can then use the received time stamps to calculate the range between the two nodes. This it then transmitted back to node A for the *RangeReport* message type. Upon receipt of this message, node A can repeat the ranging process.

However, in this work, the MAVs localise themselves with respect to stationary anchors or beacons and not to the other MAVs as was the case in [68]. As a result, the number of nodes that have to perform ranging between each other is reduced. For example, in this work it is not necessary to perform ranging between two anchors or between two tags. This allows the micro controller to reduce the memory needed to store the broadcast message as well as the amount of data to be transmitted by the UWB modules. In van der Helms original code, if five modules were used the size of the data buffer for the broadcast messages was 81 bytes while after the adaptions this was reduced to 30 bytes (Using four anchors and one tag). When using the adapted code, the rates of data transmission and the ranging updating frequency (in Hz) is shown in Table 2-1.

Table 2-1: Ranging update frequencies

Test	# Anchors	# Tags	Broadcast Freq.	Range Update Freq.
2	2	2	14.56	6.98
3	3	1	14.67	7.30
4	3	2	12.97	6.49
5	4	1	14.01	6.57
6	4	2	10.74	5.30

2-4 Conclusion

In this chapter the problem to be solved in this thesis was defined. This was split into three subsections. In the first, the possible controller architectures were introduced along with their main strengths and weaknesses. Due to their robustness properties, it was decided to use a decentralised approach. Next, the formal mathematical definition of the problem was given. This was followed by a brief description of the assumption made to reduce the complexity of the defined problem.

Next, the performance matrices that are to be used to analyse the quality of the designed controllers were introduced. They are as follows; the average cell age over the MS, the level

of coverage over the MS that is achieved and lastly the controllers ability to avoid collisions is represented by the number of crashes metric.

In the remainder of this chapter, the hardware used throughout the thesis was given. The tests will be performed in the CyberZoo using the Parrot Bebop 2 MAV. Inter-MAV communications will rely on the Decawave UWB modules. This will allow the MAVs to broadcast their current positions to all other MAVs in the MS. The UWB modules also serve another purpose, they are used to perform ranging to four stationary nodes (anchors) placed throughout the MS. This will allow the MAVs to localise themselves with respect to the anchors.

With the definitions given in this chapter, we can now move onto the method for creating the NN controllers.

Chapter 3

The NEAT Algorithm

This section describes the evolutionary approach used to create the NNs used to control the individual drones. This relied upon the Neuro Evolution of Augmenting Topologies (NEAT) algorithm first presented in [1]. This is a method for evolving not only the weights of a NN but also its structure to find a good balance between the fitness and the diversity of the solution.

Due to the nature of the persistent surveillance task, a training data set cannot be generated. As a result of this, methods that rely on optimising the connection weights of the NN, like back-propagation, are not applicable. To overcome this shortcoming, the training process must rely on finding an (near) optimal set that is defined by a fitness function. Using this view point, GAs can be used to train evolved NNs effectively [69]. Traditionally, GAs are used to evolve only the connection weighs for an existing NN structure, however, they can be used to evolve the structure as well as the weights of the connections [1].

A key reason for selecting a method that evolves the structure as well as the weight is to eliminate the expert knowledge and trial-and-error based approaches to formulating the NNs topology. In his paper, [1], Stanley showed that his NEAT method also resulted in increased performance and was able to generate an effective solution five times faster than a fixed-topology method like Enforced Sub Populations. Lastly, the NEAT algorithm is well suited to escape from local minimum which can trap fixed-topology methods as the NEAT algorithm can add additional structure to its topology [1].

The basic steps for the NEAT algorithm are outlined below, but a more detailed explanation can be found in [1, 70].

3-1 NEAT Background

3-1-1 Encoding Scheme

The genetic encoding scheme used in the NEAT translates the structure of the NN into genomes, which are linear representations of the network connectivity. Each genome is com-

posed of a list of *node genes* representing the neurons and *connection genes* which lists the weights, input and output nodes, a bit expressing if the connection is active and an innovation number (*Innov*) for each of the connections. An example of this is shown in Figure 3-1. The purpose of this innovation number is detailed in the next section.

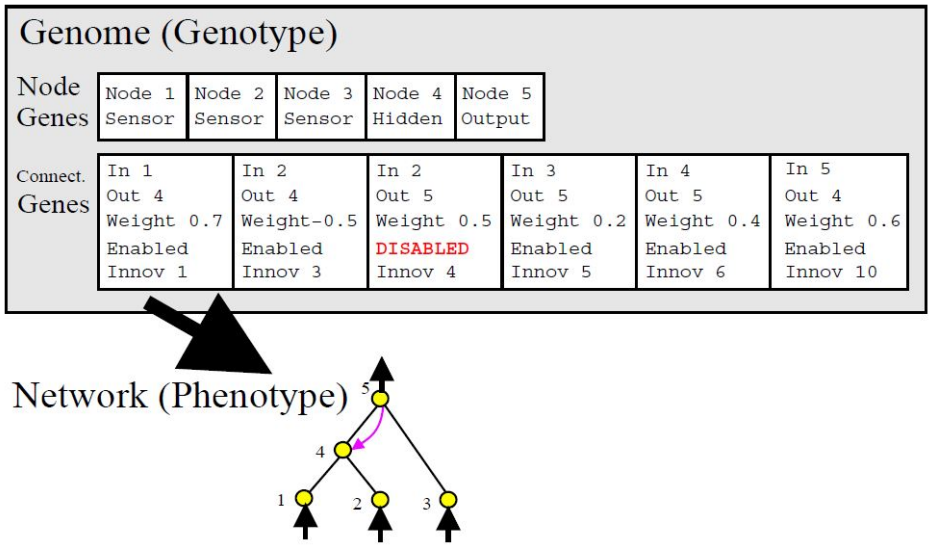


Figure 3-1: Genotype mapping example, [1]

3-1-2 Key Aspects of the NEAT Algorithm

The NEAT method applies three key techniques to evolve the NN. These techniques set the NEAT algorithm apart from other Topology and Weight Evolving Artificial Neural Networks (TWEANN) methods. They are, tracking of genes with genetic markers, protecting innovation through speciation and, lastly, minimizing dimensionality through incremental growth from a simple initial structure.

Tracking Genes through Historical Markings:

In Figure 3-1 it can be seen that there is a parameter called *Innov* in each connection gene. This is called the global innovation number and it is incrementally assigned to each new gene arising from structural mutations. This number represents the chronological order in which each gene was added to the system.

Using these innovation numbers, it is possible to determine exactly which genes match between mating pairs during the *crossover* phase of the evolutionary process. When genes from both parents have the same innovation numbers they are known as *matching* genes, while those that do not match are known as either *disjoint* or *excess* genes. This is shown in Figure 3-2 below. For the crossover process, the child networks are comprised of the matching genes between parents and all the *disjoint* and *excess* genes from the parent with the highest fitness value. The gene used, in other words its weight and enabled bit, in each of the *matching* cases are selected at random from between the parents or are assigned the average weight of

the two parents. This selection between random and average assignment is governed by the multipoint crossover probability.

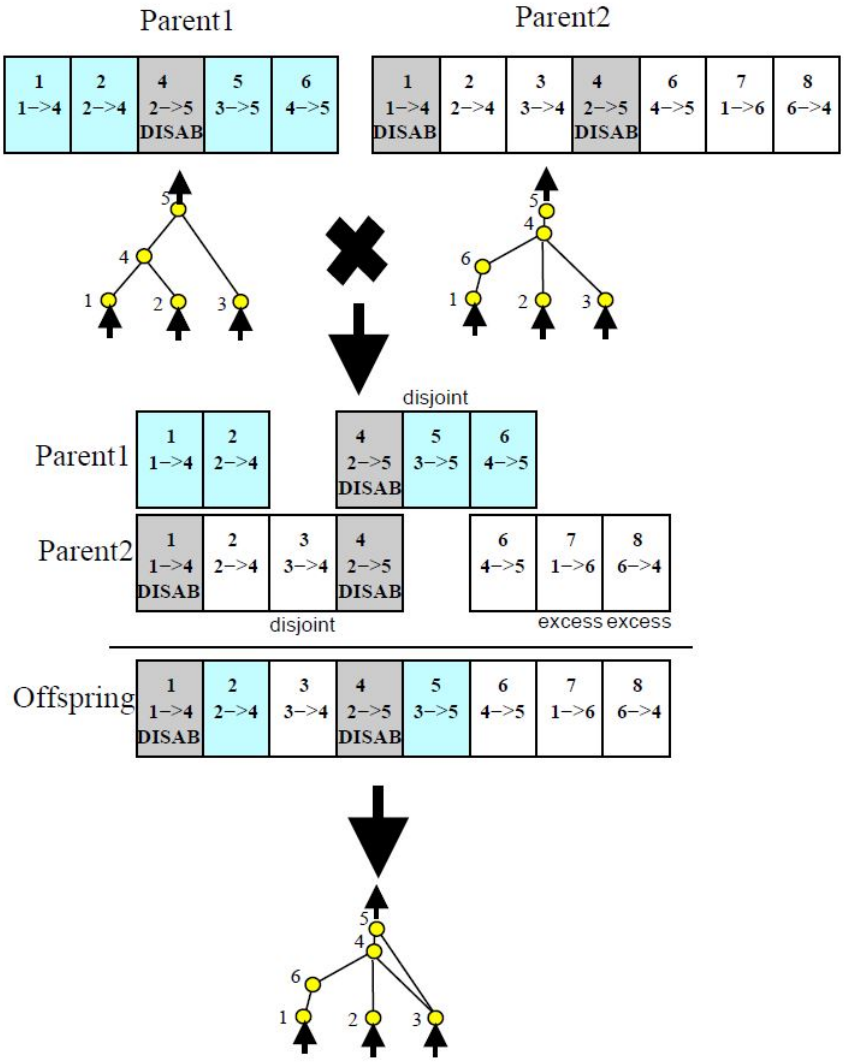


Figure 3-2: Crossover example, [1]

Protecting Innovation through Speciation:

Unfortunately, with the above mentioned process of crossover, the system struggles to maintain structural innovations created through the mutations. This is because adding nodes or connections usually initially decreases the fitness of the given network. As a result, mutations would likely not be passed on to future generations. To solve this issue and protect the innovations introduced by mutations, NEAT makes use of speciation.

With this technique, networks in the total population with similar topologies are grouped together into species. They will primarily only compete against others in their niche. As a result, new innovations are placed in their own species and, as such, they are given time to

optimise their structure before competing with others. During each generation, every network is sequentially placed into a species through the use of a compatibility distance, $\delta(i,j)$. Here, each existing species is represented by a random network from its previous generation and the $\delta(i,j)$, for the given network from the current generation, is calculated according to Equation (3-1). In this equation, E_{gene} and D_{gene} are used to represent the number of excess and disjoint genes respectively, $\bar{W}_{gene}$ is the average weight differences of matching genes and N_{gene} is the number of genes in the larger network. Lastly, the parameters c_i ($\forall i \in 1,2,3$) are weighting factors that influence the impact of the E_{gene}, D_{gene} and $\bar{W}_{gene}$ on the speciation compatibility distance. If this distance is less than a compatibility threshold, δ_t, the current network is placed in that species, otherwise it is compared to the representative for the next species. If it is not placed in any existing species, a new species is created with the current network as its representative.

$$\delta(i,j) = \frac{c_1 E_{gene}}{N_{gene}} + \frac{c_2 D_{gene}}{N_{gene}} + c_3 \cdot \bar{W}_{gene} \tag{3-1}$$

With the population divided into species, explicit fitness sharing is used as the reproduction mechanism. Here, networks in each species share the fitness value for their niche. This prevents any one species from completely taking over the entire population. The adjusted fitness, $f_i^{'}$, is calculated as:

$$f_i^{'} = \frac{f_i}{\sum_{j=1}^{N_{pop}} sh(\delta(i,j))} \tag{3-2}$$

where

$$sh(\delta(i,j)) = \begin{cases} 0 & , \delta(i,j) > \delta_t \\ 1 & , \text{otherwise} \end{cases} \tag{3-3}$$

and N_{pop} is the number of NN in the total population. Each species then produces a number of children for the next generation, which is proportional to the sum of the $f_i^{'}$ of its member networks.

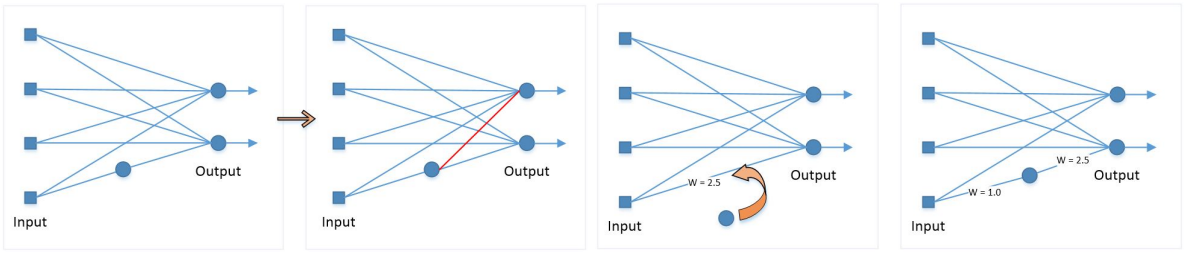
Minimizing Dimensionality through Incremental Growth from a Simple Initial Structure:

The NEAT method biases the resulting controller towards a minimal-dimensional space. This is done by staring out with a uniform network configuration with no neurons in the hidden layer. This means that each input is directly connected to each of the outputs. New structures (i.e. hidden layers) are introduced incrementally through mutations, where only the useful mutations survive.

3-1-3 Mutations

A further component of the NEAT algorithm are the mutations applied to the child networks to create the new population. There are two main types of mutations that can be applied to the networks. The first is a mutation of the connection weights where each of the connections are either perturbed or not based on a predefined probability. These perturbations can change the weight of the connection (governed by the weight mutation probability) or whether the connection is enabled or disabled (Gene re-enable probability).

The second type of mutation is a structural mutation. This can be further divided into two subtypes namely; 'add connection' or 'add node'. In 'add connection', a new connection between two previously unconnected nodes is added, with a random weight. For the 'add node' subtype, a new node is inserted into an existing connection. Here, the original connection is set to disabled and two new connections are added, one entering the new node, which is assigned a weight of 1, and the other leaves the node with a weight equal to that of the original connection. Examples of these two mutation types can be seen in Figure 3-3.



(a) Example of 'add connection' mutation **(b)** Example of 'add node' mutation

Figure 3-3: Mutation examples

3-2 Implementation

In this section, details are given for the specific implementation of the NEAT algorithm for the problem given in Section 2-1. For this, the MATLAB version of Kenneth Stanley's NEAT algorithm was used [1]. The evolutionary process was implemented using MathWorks MATLAB R2016b 64-bit running on a workstation with an Intel(R) Core i7-4700MQ CPU ($2.4GHz$), 8 GB of RAM and a Windows 8.1 64-bit operating system.

3-2-1 Fitness Function

A key component to Evolutionary Robotics (ER) is the fitness function, f. This is the means by which the solver determines which solutions are more capable of solving the given problem [71]. This is often the limiting factor in the quality of the generated solutions when evolving controllers for complex tasks, [71].

For this work, the cost function used in [2] was implemented with one small adjustment as shown in Equation (3-4). In this work, the average cell age is multiplied by the safety coefficient, S, at each time step instead of once at the end of the simulation (as was implemented in [2]). It was felt that this would give a clearer indication of the controller's ability as it only penalises the fitness gain at the instance where two MAVs moved too close to one another rather than penalising the fitness for the entire simulation. In the equation, N is the total number of cells in the MS and $age(n,t)$ is the current age of the given cell as defined in Equation (2-5). You will note that this fitness function is very similar to the definition of the information age given previously in Equation (2-6), with the exception that here the ages

[1] http://eplex.cs.ucf.edu/neat_software/

are averaged over the MS and duration of the simulation. This is known in literature as an aggregate cost function.

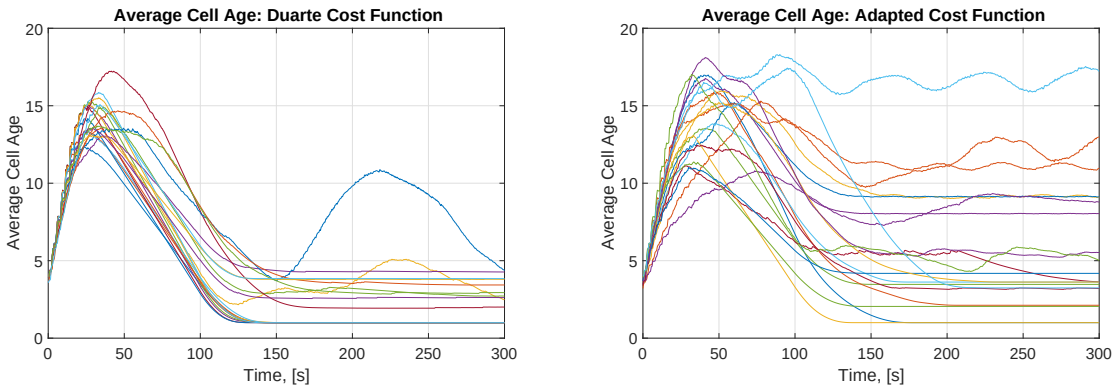
$$f = \frac{1}{t_f} \sum_{t=1}^{t_f} \left[S \left(\frac{1}{N} \sum_{n=1}^{N} age(n,t) \right) \right] \tag{3-4}$$

where the safety coefficient is taken directly from [2] and is given as:

$$S = 0.1 + 0.9 \frac{\max(0, \min(3, minDist))}{3} \tag{3-5}$$

In the above equation for the safety coefficient, $minDist$ is the minimum distance between any two MAVs at the current time step. This safety coefficient is related to the positional constraint placed on the system in Section 2-1. It is used to penalise any solution that allows the MAVs to come within $3m$ of one another.

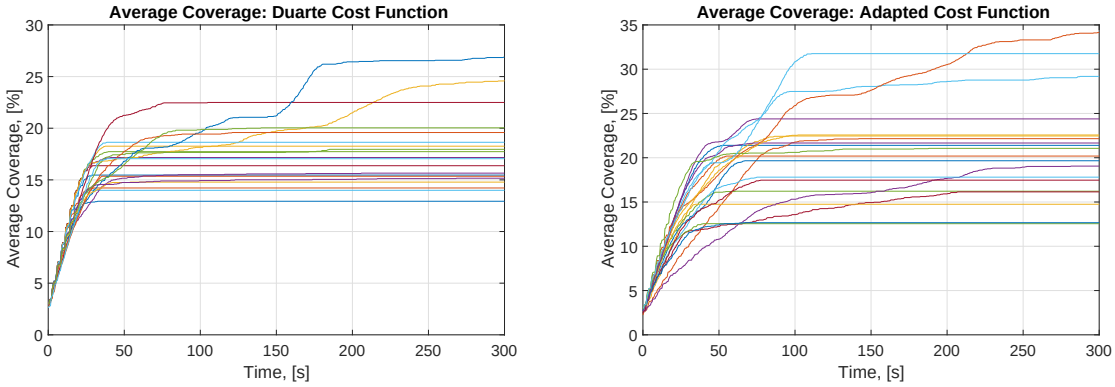
In Figure 3-4 a comparison between the performance of controllers generated with the pure Duarte cost function and the adjusted one used in this work is given. For this test, the experiments performed by Duarte were recreated in a $25m \times 25m$ MS with a discretisation resolution of $1m$. Four separate evolutionary runs were then performed, using two MAVs, for each of the two cost functions. From each of these runs, five controllers with the highest fitness scores after 100 generations were then post evaluated and the average cell age and coverage level at each time step was recorded. The average cell ages for the 20 controllers can be seen in Figure 3-4 while the coverage results are presented in Figure 3-5. From this it can be seen that the controllers generated with the pure Duarte cost function produce a more consistent result than that of the adjusted cost function. However, the controllers that used the adjusted cost function can often outperform their counterparts in terms of average cell age. In terms of the best performing controllers for these two cases, there was a $9.35s$ improvement when using the adapted cost function. When considering the coverage levels achieved, depicted in Figure 3-5, it can be seen that, again, the adapted cost function slightly outperforms the original. For this metric the increase in performance between the two best controllers from each set is 7.26% while the average difference between the sets is 3.14%. As such, the adapted cost function will be used in the remainder of this work.



(a) Duarte cost function age results

(b) Adapted cost function age results

Figure 3-4: Comparison of cost functions: Cell age



(a) Example of add connection mutation

(b) Example of add node mutation

Figure 3-5: Comparison of cost functions: Coverage

3-2-2 Neural Network Inputs and Outputs

The NNs output layer consisted of two nodes. These represented the x and y velocities in the global reference frame. As the outputs are bounded on the interval $[-1, 1]$, these values must be adjusted so that they correspond to the maximum velocity of MAV m, $\forall m \in \{1, 2, \cdots, M\}$. Hence, the x and y velocities are given as:

$$v_{x,m} = out_1 \cdot v_{max} \cdot \cos(\theta) \tag{3-6}$$

$$v_{y,m} = out_2 \cdot v_{max} \cdot \sin(\theta) \tag{3-7}$$

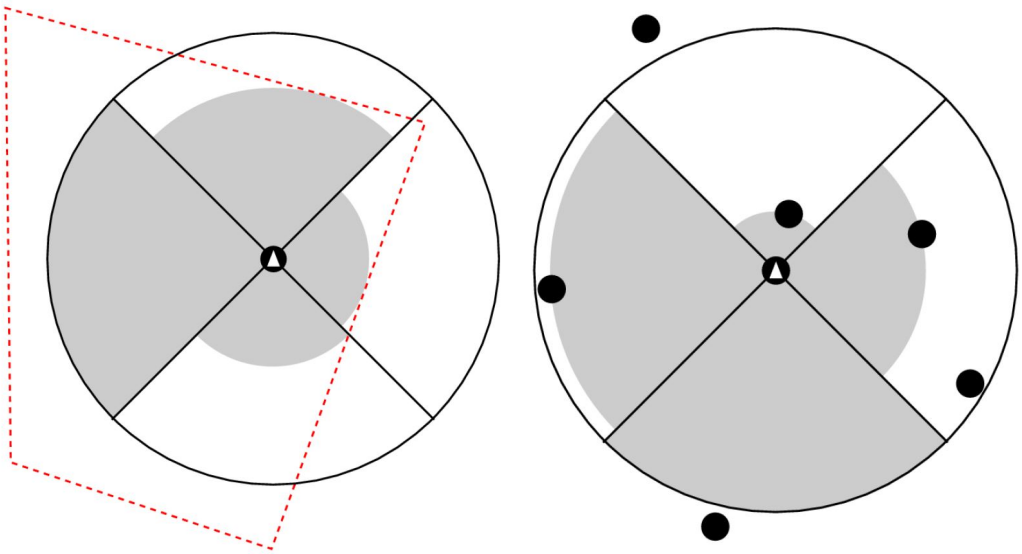
$$\theta = \begin{cases} 0 & , out_1 = 0 \\ \tan^{-1}\left(\frac{out_2}{out_1}\right) & , \text{otherwise} \end{cases} \tag{3-8}$$

where out_i is the value of each of the output nodes and v_{max} is the maximum velocity of the MAV. For the majority of the simulations, this maximum velocity was defined as $v_{max} = 0.5m/s$. The exception was with the simulated and physical tests in the CyberZoo where the velocity was limited to between $[-0.3, 0.3]$ (i.e $v_{max} = 0.3m/s$).

For the inputs into the NN, five different cases where tested where each of these cases requires a different level of knowledge from the environment. These are described in detail below.

Input case 1:

The first input case was an attempt to recreate the results presented in [2] and apply the controllers to MAVs instead of aquatic robots as used in the paper. This consisted of ten inputs into the NN which are; four normalised geo-fence sensor readings, four normalised robot sensor readings, one boolean input showing whether the robot was inside the MS and, finally, a bias node. Here, the geo-fence sensor is used to determine the range to the border of the MS in the four sections surrounding the robot as seen in Figure 3-6a. The robot sensors are very similar to the geofence-sensors. Instead of returning the range to the border, they return the range to the nearest other robot in each of the four quadrants. This can be seen in Figure 3-6b.



(a) Geo-fence sensor example **(b)** Robot sensor example

Figure 3-6: Sensor examples from [2]

Input case 2:

This input case was the first attempt at including knowledge about the MS into the controller. This was done by dividing the entire MS into four quadrants, as seen in Figure 3-7, based on the current position of the MAV. The average cell age for each of these four quadrants was calculated, normalised and used as the first four inputs into the NN. The remaining ten inputs used were identical to those presented in [2] and described above.

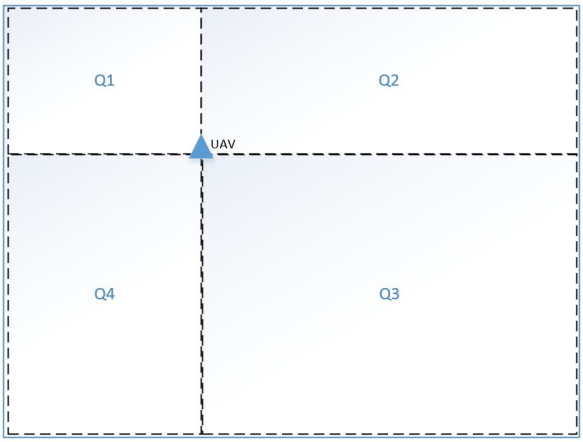


Figure 3-7: Average age input example

The new inputs were implemented according to the pseudocode presented in Algorithm 1.

Algorithm 1 Pseudocode: Input case 2

1: $X_{idx,current}$ = x index of MAVs current cell
2: $Y_{idx,current}$ = y index of MAVs current cell
3: $TotalAveAge$ = average cell age over the MS
4: $Q_1 = \text{sum}(age(1 : X_{idx,current}, Y_{idx,current} : end))/TotalAveAge$
5: $Q_2 = \text{sum}(age(X_{idx,current} : end, Y_{idx,current} : end))/TotalAveAge$
6: $Q_3 = \text{sum}(age(X_{idx,current} : end, 1 : Y_{idx,current}))/TotalAveAge$
7: $Q_4 = \text{sum}(age(1 : X_{idx,current}, 1 : Y_{idx,current}))/TotalAveAge$

Input case 3:

The next attempt at including knowledge of the MS involved eight virtual sensors that were modelled as line segments that radiated outwards from the MAV. These sensors, termed 'feelers', then reach out from the MAV until they either encounter an obstacle or they reach their maximum range. They then return the age of the furthest cell that they can reach. This is shown in Figure 3-8. In this image, the dark blue cells represent an obstacle, the grey cells are the areas that can be 'seen' by the agents virtual sensors and the black lines show these eight feelers. As in the previous two cases, the remaining ten inputs are identical to those

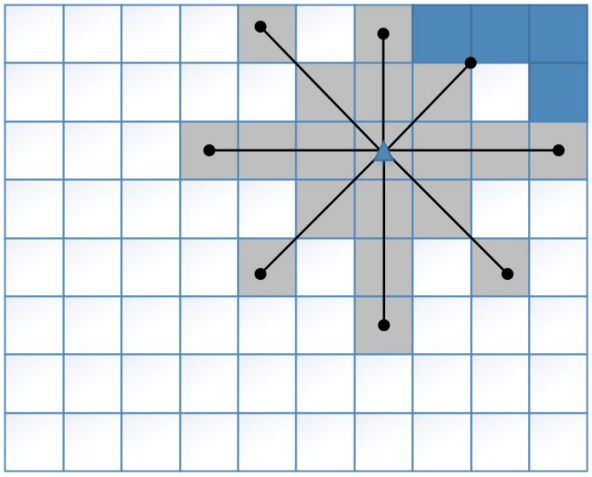


Figure 3-8: Feeler input example

used in [2]. Namely; four geo-fence sensors, four robot sensors, a boolean showing if the MAV is inside the MS and a bias input set to 1.

In Algorithm 2 an example of the pseudocode used to implement one of the feelers is given. This feeler extends outwards from the right hand side of the MAV, running parallel to the x axis and returns the maximum distance that it can extend. This distance is given as the parameter *range* in the example. Further, the parameters $X_{idx,current}$ and $Y_{idx,current}$ represents the x and y indexes of the cell in which the MAV is located, $cellRes$ is the resolution of the cells, $range_{max}$ is the maximum range that a feeler can extend and M_{MAVs} is the number of other MAVs in the system. Finally, $X_{idx,i}$ is the x cell index for MAV i. A similar method is used to implement the remaining seven feelers.

Algorithm 2 Pseudocode: Input case 3

1: X_{maxIdx} = furthest cell that could be reached along +x axis
2: $range = X_{idx,current} \cdot cellRes - posX$
3: $i = X_{idx,current}$
4: **while** $i < X_{maxIdx}$ **and** $range \leq rangemax$ **do**
5: $i = i + 1$
6: **if** $cell(i, Y_{idx,current})$ **not** *obstacle* **then**
7: $range = range + cellRes$
8: **else**
9: break
10: **end if**
11: **if** $range \geq range_{max}$ **then**
12: $range = range_{max}$
13: **end if**
14: **end while**
15: **for** $i = 0$ **to** M_{MAVs} **do**
16: **if** $X_{idx,i} == X_{idx,current}$ **and** $X_{idx,i} \geq X_{idx,current}$ **then**
17: $dist$ = distance to MAV i
18: $range = \min(range, dist)$
19: **end if**
20: **end for**

Input case 4:

In this case, the sensor inputs from the work by M. Duarte ([2], [50]) were not used at all. Instead, this case made use of the feelers introduced in the previous subsection. Now, the feelers are used to not only return the cell age, but also the ranges that the feelers can extended. When using the feelers are used in this manner, the other MAVs are treated as obstacles. This gives 16 inputs from the feelers which are combined with a boolean showing if the MAV is inside the MS and a bias node.

As can be seen from Algorithm 3, the implementation of the virtual feelers is nearly identical to that of the previous method. The exception is in lines 3 and 9 where the age of the cell, given as $age(X, Y)$, is stored.

Algorithm 3 Pseudocode: Input case 4

```
X_maxIdx = furthest cell that could be reached along +x axis
range = X_idx,current · cellRes − posX
feeler_age = age(X_idx,current, Y_idx,current)
i = X_idx,current
while i < X_maxIdx and range ≤ rangemax do
    i = i + 1
    if cell(i, Y_idx,current) not obstacle then
        range = range + cellRes
        feeler_age = age(i, Y_idx,current)
    else
        break
    end if
    if range ≥ range_max then
        range = range_max
    end if
end while
for i = 0 to M_MAVs do
    if X_idx,i == X_idx,current and X_idx,i ≥ X_idx,current then
        dist = distance to MAV i
        range = min(range, dist)
    end if
end for
```

Input case 5:

The last set of inputs into the NN that was tested was an attempt to recreate the results of a recently published extended abstract, [72]. In this work, the authors also attempted to produce monitoring behaviour in MAVs using the NEAT method. While they also used the cost function given in [2] and they used UWB for inter-drone communications, they relied on a motion capture system to provide the positional data and they limited their tests to a $5m \times 5m$ area. For the actual tests this was reduced to $4.5m \times 3m$. A further key difference was that they did not consider the effects of real world fuel constraints in their work.

In this work, the authors only had three inputs into their system. These were; a bias input that is always set to 1, the range to the nearest other MAV and the final input was the range to the nearest wall.

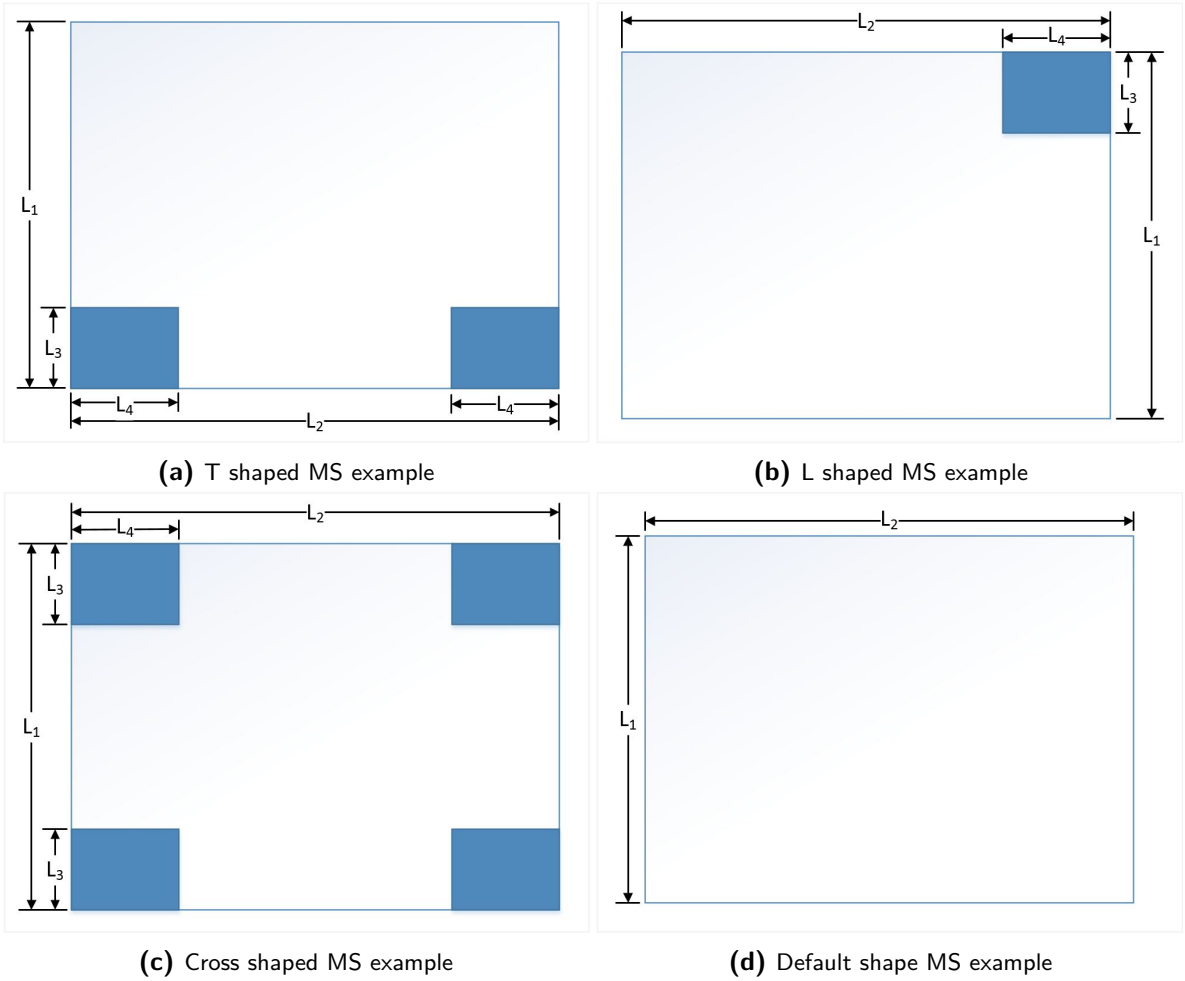
3-2-3 Simulated Environment for Evolution

The last consideration taken into account during the implementation of the NEAT algorithm is the simulated environment that is used to test the NNs. This requires a number of parameters to be defined at the start of each evolutionary run, which are:

- The (maximum) length and breadth of the MS (L_1 and L_2 respectively from Figure 3-9)
- The size (L_3 and L_4) of any obstacles in the area

- The resolution of the MS discretisation
- The total duration and time-step used during the simulation and,
- The number of MAVs used in the evolutionary run

To ensure that the behaviours evolved do not depend on the specific MS used in each evolutionary run, the overall size and shape of the MS was varied for each generation of controllers. This was done by first randomly varying the length and breadth of the MS between their specified maximum distance and 75% of their maximum distance. It is important to note that the lengths chosen are multiples of the discretisation resolution. From there, the shape of the MS could be randomly altered between the four shapes shown in Figure 3-9. For these different shapes, the size of the obstacle blocks are equal to the size specified at the beginning of the evolutionary run. For the L shaped MS, the corner that is covered by the obstacle is randomly selected, while for the T shape the placement of the obstacles is also randomly chosen between either the top or the bottom of the MS.



(a) T shaped MS example

(b) L shaped MS example

(c) Cross shaped MS example

(d) Default shape MS example

Figure 3-9: Simulation MS shapes

Next, to ensure that the controllers do not depend on the starting positions of the MAVs, a

set of 3 initial positions for each of the MAVs were randomly generated at the beginning of each generation of the evolutionary run. Every controller in the current population was then applied and tested in 3 simulations were the starting positions for each of the iterations was given by one of the sets of initial positions. The controllers final fitness given as the average fitness generated by Equation (3-4) averaged over all the iterations.

Lastly, to reduce the time taken to perform each evolutionary run, a simulation could be stopped if the MAVs left the MS and did not attempt to return. This was accomplished by tracking the time each MAV spent outside of the MS and if this was greater than $0.2 \times t_f$ then the simulation was terminated. Each time a MAV re-entered the MS its tracking time was reset to 0. To ensure that this did not positively influence the fitness of the given controller, the cost function still averaged the cell ages over the entire simulation time, t_f, and not over just the reduced simulation time.

3-3 Simulation Results

In this section, the performance of the evolved controllers for each of the input cases are given.

3-3-1 Test Procedure

For each of the input cases mentioned previously, four evolutionary runs were performed with 3 MAVs used for the first run, followed by 4, 6, and 8 for the remaining runs. The parameters used during the evolution are as follows, for the simulation environment:

- Length and width of the MS were $L_1 = 25m$ and $L_2 = 25m$ respectively.
- Size of obstacles: $L_3 = 3m$ and $L_4 = 3m$.
- MS was discretised into $1m \times 1m$ cells.
- A simulation time of $300s$ with a $0.5s$ time-step was used.

The NEAT parameters are as follows:

- Number of generations = 100
- Initial population size = 150
- Recurrency was disabled
- Add node probability = 3%
- Add connection probability = 5%
- Weight mutation probability = 25%
- Multipoint crossover probability = 60%
- Probability of gene becoming re-enabled = 25%

- For the other parameters not mentioned, the default values (given in [1]) were used.

After completing all the evolutionary runs, the five controllers from each run with the highest fitness score were post evaluated to give an indication of the effectiveness for each input case. For the post evaluation, the MAVs were tested in a simulated $25m \times 25m$ MS over a period of $300s$. 50 iterations of the test were performed for each of the controllers, with the starting positions of the MAVs assigned randomly at eh beginning of each iteration. The final estimate of the controllers quality was determined as the mean coverage levels and average cell age achieved over all the iterations. The results of this post evaluation are given in the next section.

3-3-2 Input Case Results

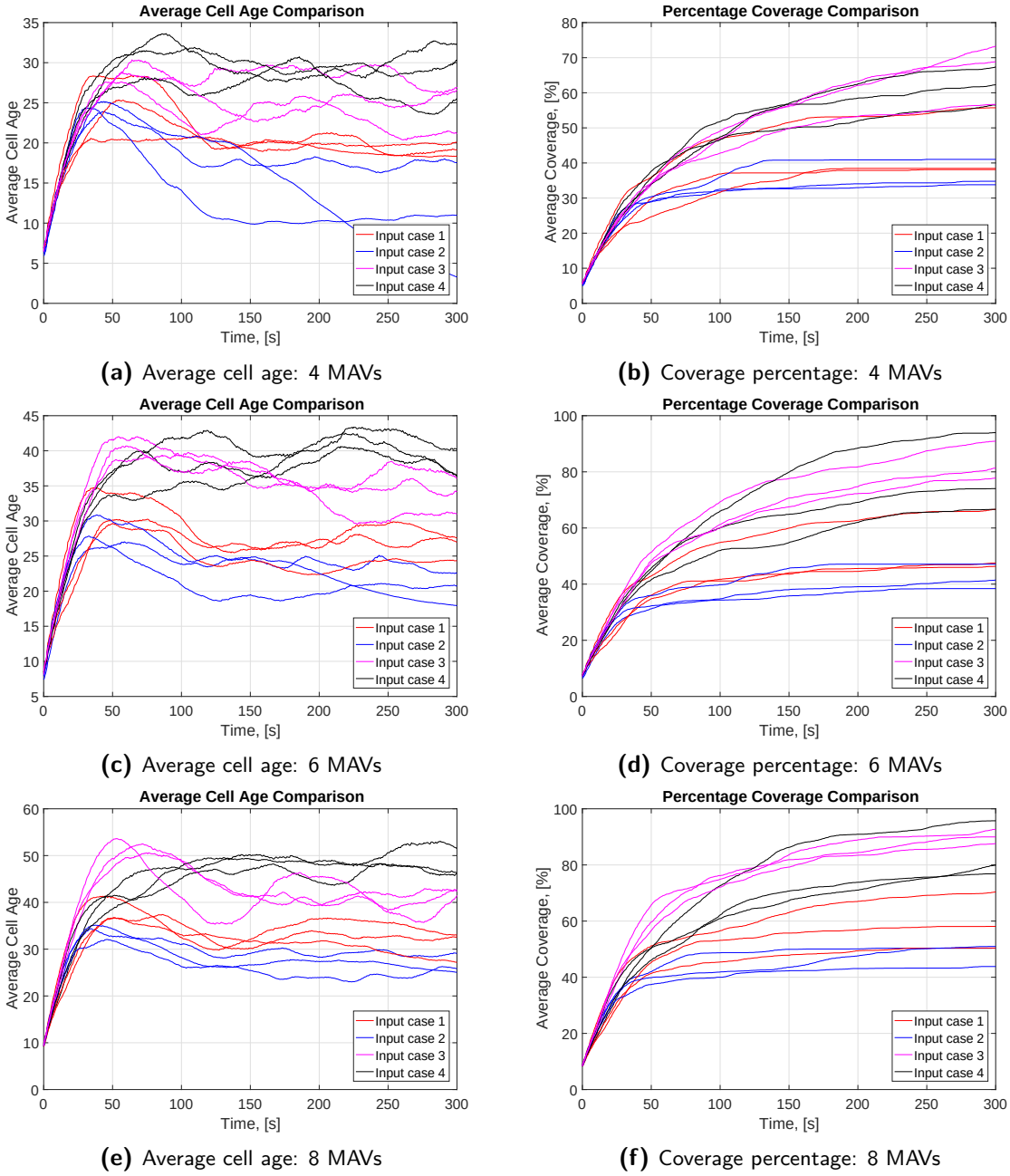
In Figure 3-10 the average cell age over time and the coverage levels achieved by the controllers is shown. In each graph, the performance of the top three controllers from each of the input cases is shown for 4, 6 and 8 MAVs. The results for all 20 controllers for each input case can be seen in Appendix A. As a brief reminder, the first case is a reproduction of the test performed in [2], case 2 has environmental data added in the form of the average ages from four quadrants around the MAV, case 3 uses adds the virtual feelers to return environmental data and case 4 uses just the feelers to return both range and age data.

While input case 1, used in [2], does not produce the worst performing controllers, its coverage levels are still well below what is needed. This is not entirely unexpected as this input case uses the least amount of environmental data and relies solely on the MAVs repelling one another into previously unexplored areas. Consequently, it was decided that this input case is not suitable for the given problem.

From the figures, it can also be seen that input case 2 actually decreases the performance of the system, compared to that of input case 1, despite its greater awareness of its surroundings. As such, this was also considered to be a poor choice for this problem. With cases 3 and 4, high levels of coverage, $> 80\%$, were finally achieved. It would appear like input case 4 has the best performance overall. In terms of the average cell age, these controllers consistently outperform the others. For the coverage levels this is not as clear. While a controller from case 4 does have the highest coverage for 6 and 8 MAVs, at first glance it might seem like case 3's controllers perform better on average. However, by considering Figure 3-12, which shows average coverage and the standard deviation for the 20 controllers, it can be seen that case 4 does achieve the best coverage levels out of the input cases investigated. Hence, input case 4 will be used in all further experiments.

Remark 1 (Input Case 5)**.** *You will note that input case 5 was not included in the above comparison. While the authors of [72] were able to obtain reasonable results using these inputs, their results could not be replicated in this work. For more details and an overview of the results obtained, please see Appendix A-5.*

Figure 3-13 shows a heat map of the coverage achieved by the top controller for each case. For this, a grid, which is referred to as a tick counter, was created to record the number of times each cell in the MS was visited by one of the MAVs (recall Section 2-2 which discussed the performance metrics). This again just highlights the poor performance of the first and second



(a) Average cell age: 4 MAVs

(b) Coverage percentage: 4 MAVs

(c) Average cell age: 6 MAVs

(d) Coverage percentage: 6 MAVs

(e) Average cell age: 8 MAVs

(f) Coverage percentage: 8 MAVs

Figure 3-10: Comparison of input cases

input cases and the higher coverage levels of the remaining cases. In the third and fourth cases, the shading across the heat maps is far more consistent than in the first two cases. This shows that the cells are visited more frequently and fairly in these cases. Therefore, in terms of the persistence, these two methods are superior which reinforces the results of the average cell age.

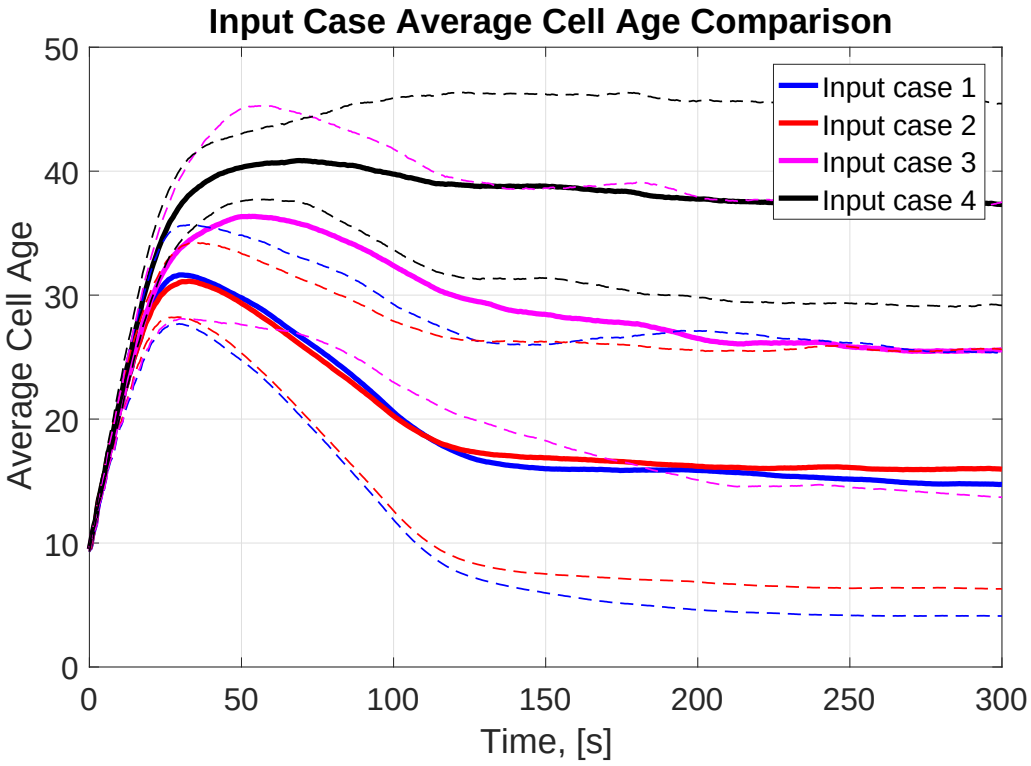


Figure 3-11: Input case comparison: Average cell age

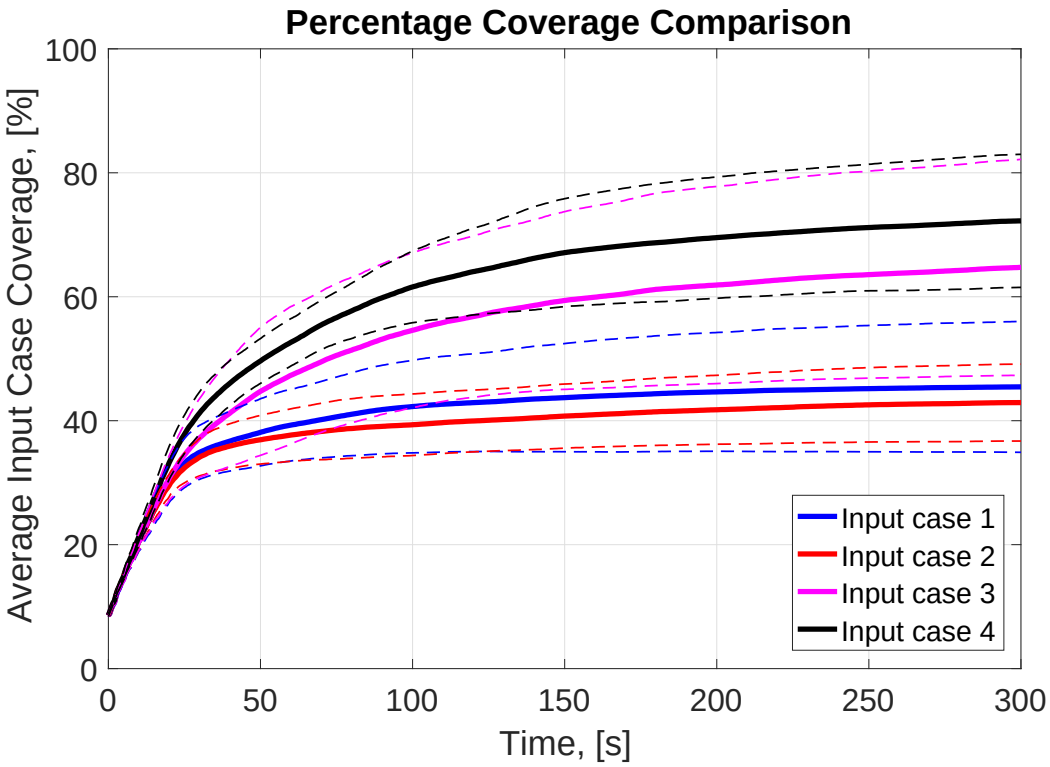
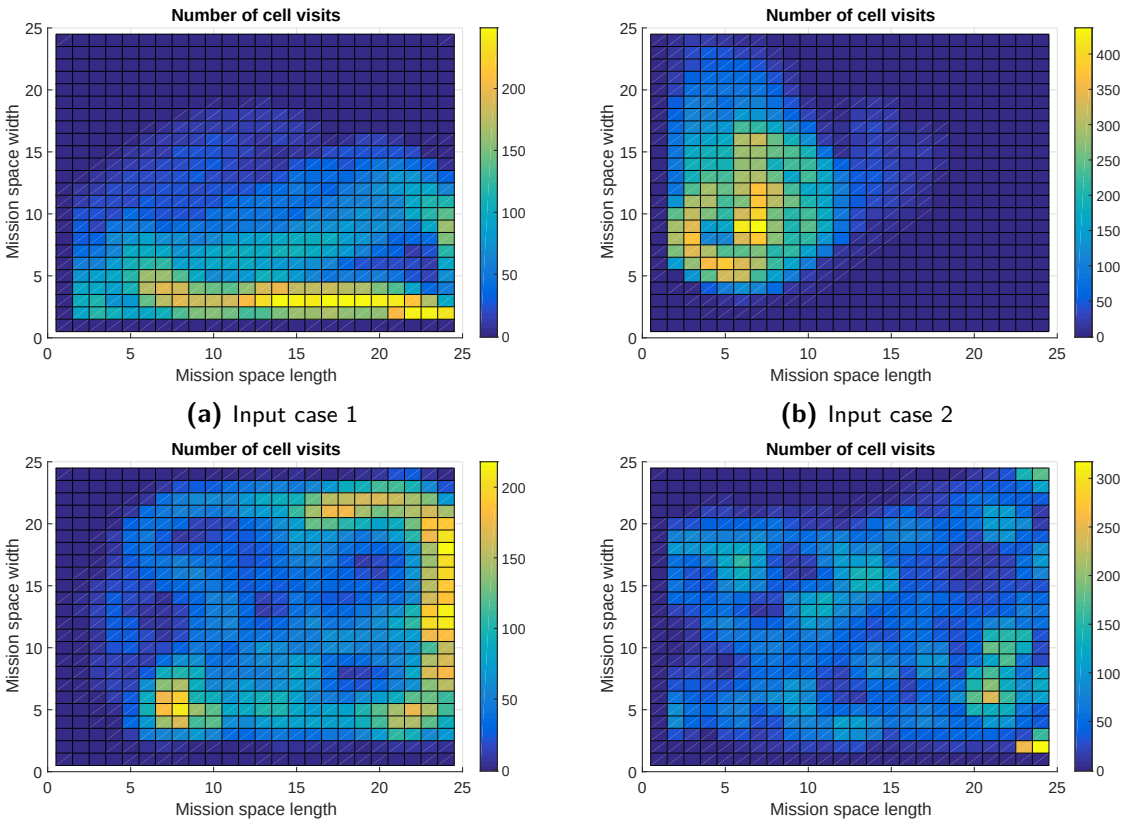


Figure 3-12: Input case comparison: Average coverage percentage



(a) Input case 1 **(b)** Input case 2

(c) Input case 3 **(d)** Input case 4

Figure 3-13: Input case comparison: Tick counters

3-3-3 Results for Selected Input Case

With the best set of input parameters chosen, a large number of evolutionary runs was performed with a variety of MS sizes. The five controllers with the highest fitness values from all these runs were then post evaluated in 50 simulations. During the post evaluation, as in the evolution of the controllers, the initial positions and the size and shape of the MS are randomly varied at the start of each simulation. A summary of the coverage results obtained by the controllers can be seen in Figure 3-14.

This figure shows the boxplots of the coverage levels achieved by 20 of the controllers tested. In the plots, the average for the controller is indicated by the blue circle while the red crosses are used to shown the outliers. As can be seen from the image, the controllers can produce very good results, with coverage levels of over 95% being achieved by a number of controllers. On average, theses 20 controllers have a final coverage percentage of 94.69% (σ= 3.239), with six of the controllers able to achieve full coverage of the MS. The best performance was achieved by controller 8 which produced an average coverage level of 99.10% ($\sigma = 0.86$).

At the same time, it is also necessary to compare the avoidance behaviour because there is often a trade-off between the coverage and the ability to avoid the other MAVs. For the avoidance analysis, the minimum distance between any two of the drones was recorded at each time step during their post evaluation simulations. If this distance was less than $30cm$ the

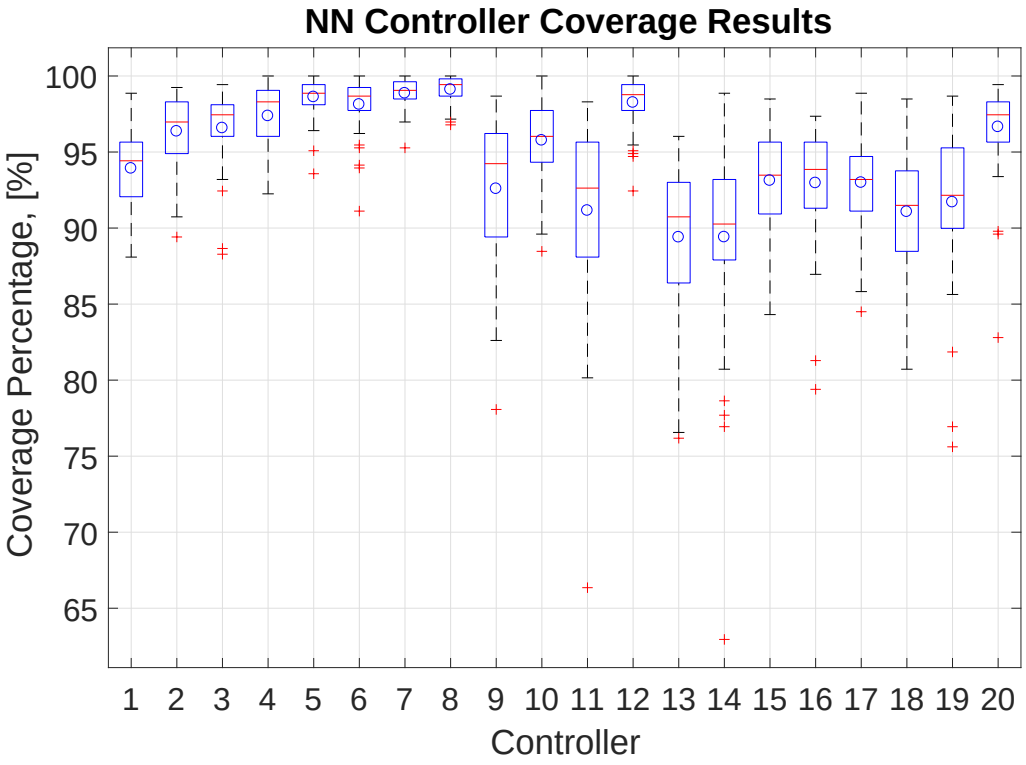


Figure 3-14: Final NN controller: Coverage performance

MAVs were said to have crashed. The total number of crashes during each of the iterations of the different controllers was counted and the results are shown in Figure 3-15. The results seen in this figure seem promising as most of the controllers only allowed two MAVs to come within $30cm$ of one another less than 5% of the time. Each of the controllers were also able to produce at least one case that gave no collisions during the course of a test and, on average, would result in a crash only 1.78% ($\sigma = 1.593$) of the time during operation.

Of course, just covering the area once is not enough for this problem of persistent surveillance. Each of the cells should also be visited as often as possible. Two metrics are used to analyse this, first, is the average cell age and the other is the tick counters mentioned in Section 3-3-2. Four examples from different NN controllers are shown in Figure 3-17 and, as can be seen, the shading across the heat map is fairly consistent over the whole area. This means that the cells are visited by a MAV for a similar number of times during the simulations. As expected from the coverage results, these figure also show that there are a few cells that have not been visited. Similarly, as can be seen in Figure 3-16, the average cell age if fairly consistent amongst the different controllers with an average cell age of $47.03s$ ($\sigma = 4.831$) being achieved.

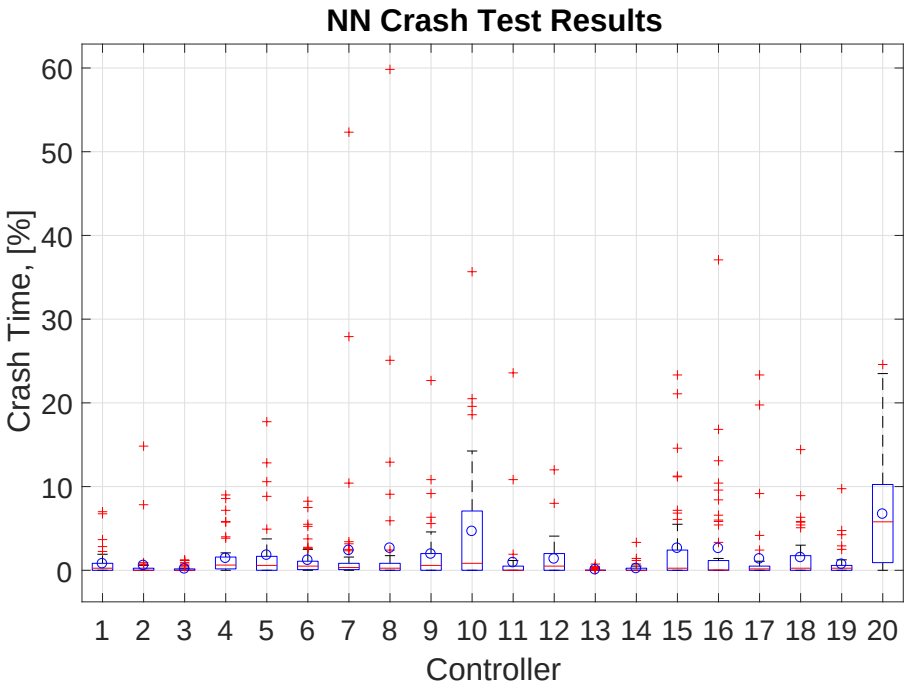


Figure 3-15: Final NN controller: Crash results

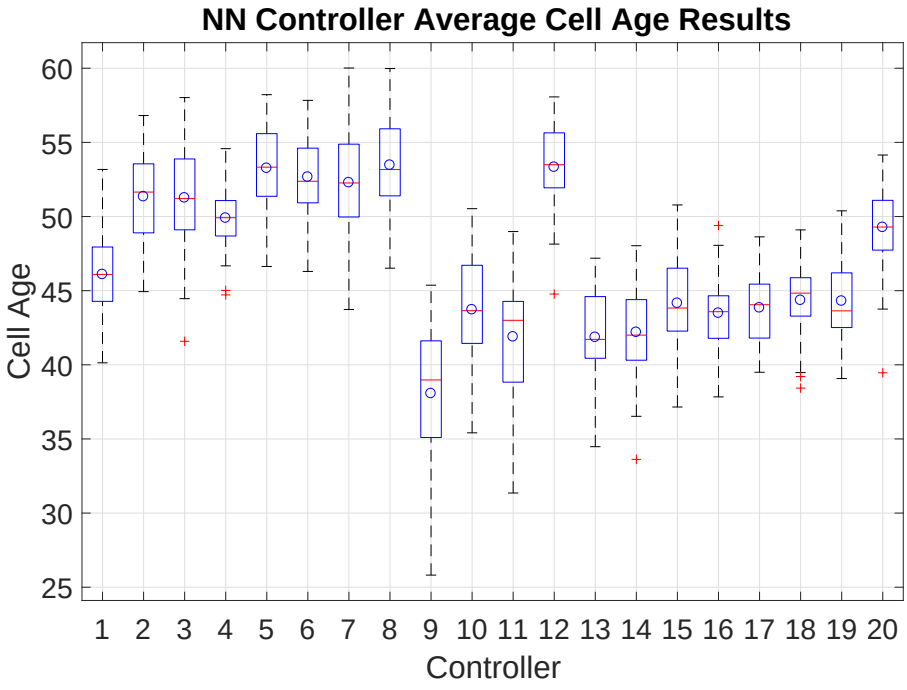
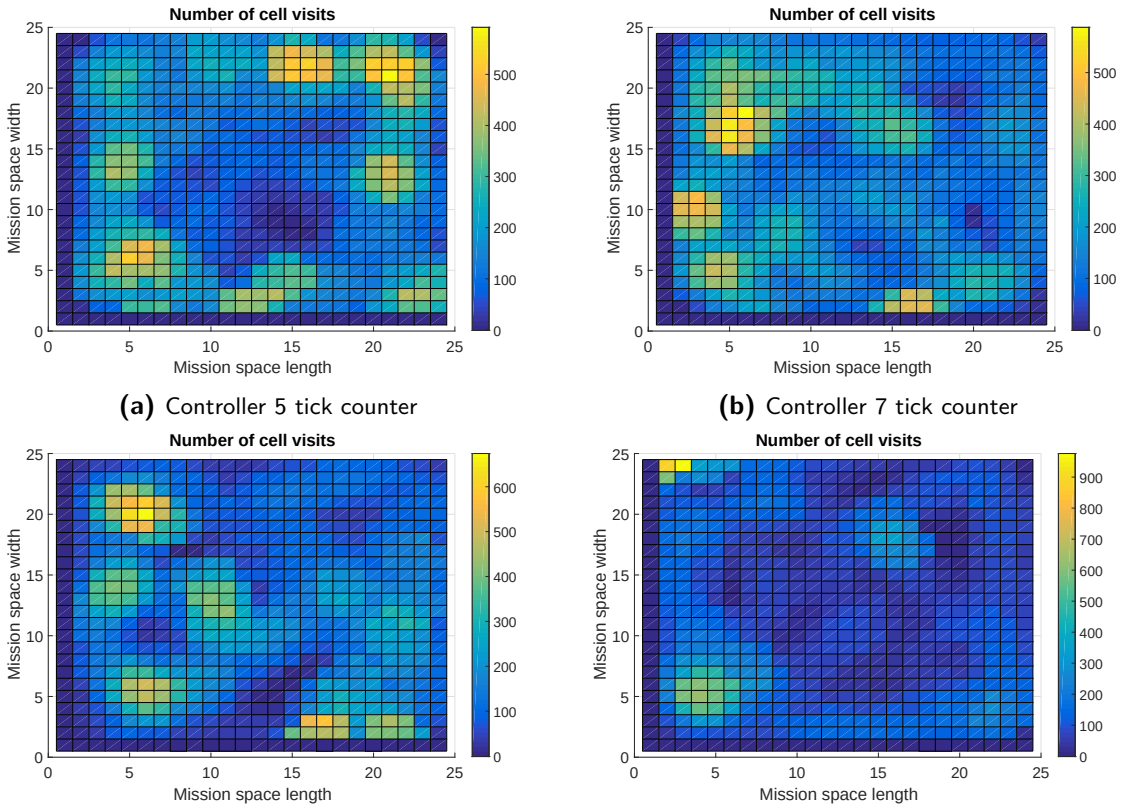


Figure 3-16: Final NN controller: Average cell age



(a) Controller 5 tick counter

(b) Controller 7 tick counter

(c) Controller 8 tick counter

(d) Controller 10 tick counter

Figure 3-17: Final NN controller: Tick counter examples

Scalability Tests

For this solution to be applicable to different greenhouse scenarios it must be scalable to account for varying sizes and number of MAVs used. This property is investigated here. For the first test, the maximum size of the MS for the test was increased from $25m \times 25m$ to $50m \times 50m$ while the number of MAVs was held constant at 8. In the second test, the larger MS was used again and doubled the number of MAVs to 16. The coverage results of the first test are shown in Figure 3-18 with the associated crash data shown in Figure 3-19. As expected, the performance in this case dropped sharply in terms of the coverage levels, with the controllers hardly ever visiting more than 80% of the cells at least once. On average, the controllers cover 61.41% of the area at least once with a standard deviation of $\sigma = 6.280$. In addition, this larger area leads to less congestion which, in turn, reduces the chance of two MAVs colliding during the tests. This is seen in Figure 3-19 where the likelihood of a crash occurring decreased from, on average, 1.78% to 0.56% ($\sigma = 1.090$). This reduction is even clearer when one considers the controllers outlier test results. The majority of these now fall below the 5% mark, with just five exceeding 10%.

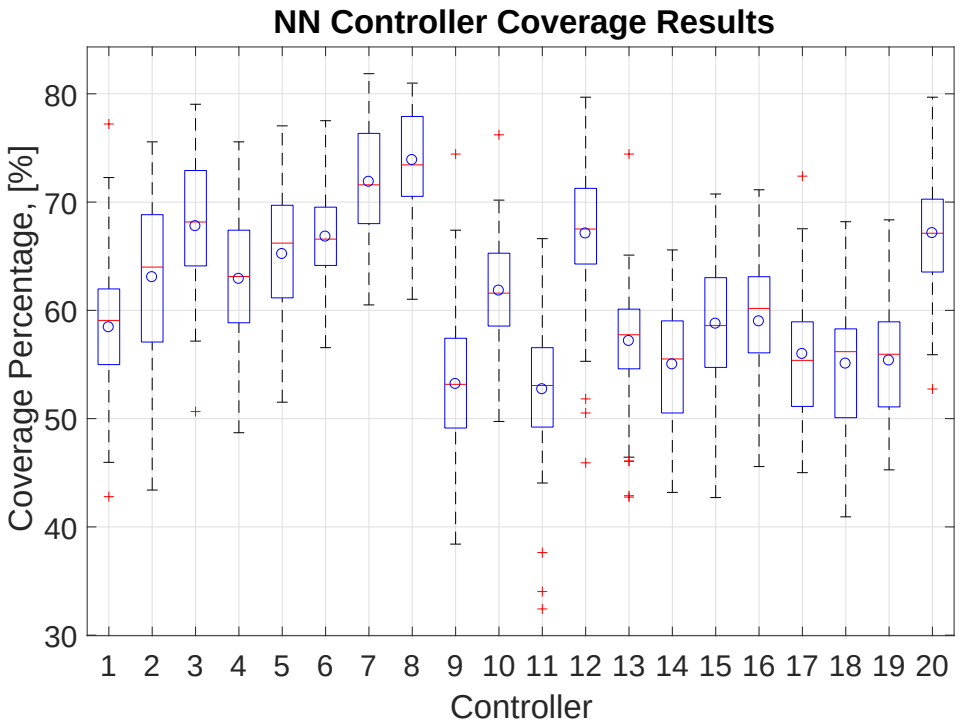


Figure 3-18: Coverage levels for scaled up MS (8 MAVs)

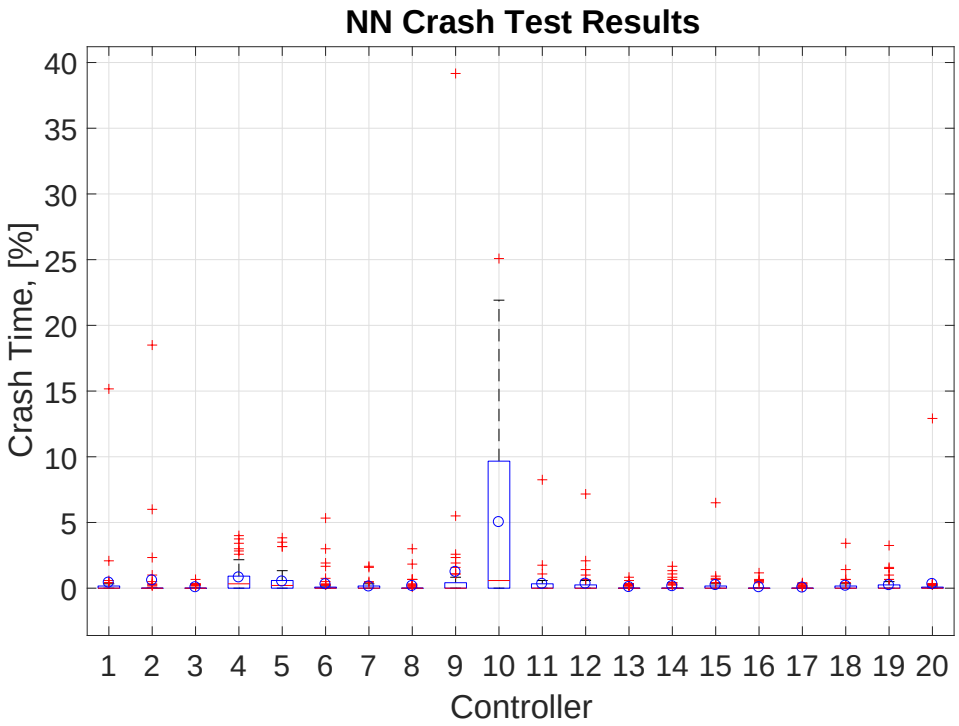


Figure 3-19: Crash results for scaled up MS (8 MAVs)

Naturally, by increasing the number of MAVs in the second set of tests, an increase in the coverage levels is seen in Figure 3-20. Here, the controllers are far more likely to achieve coverage levels of over 80% but they still fall well below the levels produced in the initial tests. An average coverage for the controllers of 82.89% with a $\sigma = 4.964$ can be observed, with the best performing controller producing a coverage of 88.81%. This is 11.8% lower than the coverage results produced for the original $25m \times 25m$ MS. Again, with a more congested airspace, the likelihood of crashes occurring also increases. Figure 3-21 shows comparable results in terms of crashes as those of the original tests with an average crash likelihood of 2.12% ($\sigma = 2.956$) as compared to the 1.78% observed in the initial tests. Again, with the exception of controller 10, all the other controllers do produce runs where not a single MAV collides.

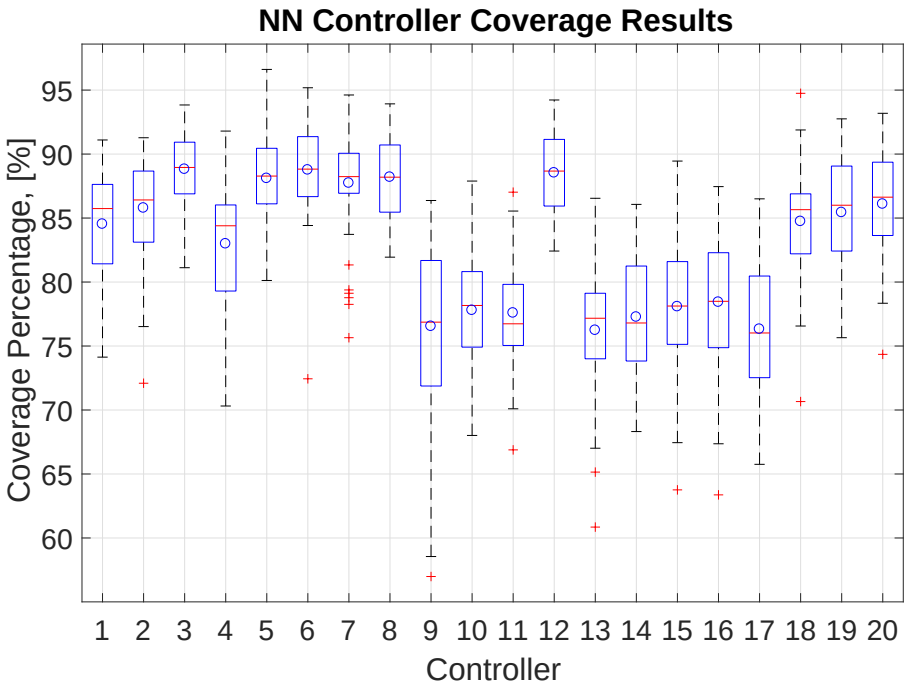


Figure 3-20: Coverage levels for scaled up MS (16 MAVs)

While these results show that the performance of the controllers does suffer when scaling them up to cover much larger areas, this can be reduced by appropriately scaling the number of MAVs used. As such it would be better to evolve controllers for each specific greenhouse, however, reasonable results could still be obtained using the controllers for the general case.

What is more important for the practical implementation of the system, is its performance in the more constricted airspace of the CyberZoo. For this, the five controllers with the highest fitness from each of the evolutionary runs were again post evaluated. This time a fixed $7m \times 7m$ area, discretised into $0.5m \times 0.5m$ cells, was used. Furthermore, only 2 MAVs were used with a sensor range of $3.5m$, a footprint of $1m$ and a maximum speed set to $0.3m/s$. The time horizon of $300s$ was kept the same as that used in previous tests. The results for 15 of the top performing controllers overall are shown in Figure 3-22, Figure 3-23 and Figure 3-24. As expected, for this smaller area, the MAVs are once again able to achieve higher levels of coverage (often $> 90\%$), with an average coverage of 83.23% ($\sigma = 17.711$). If only the top

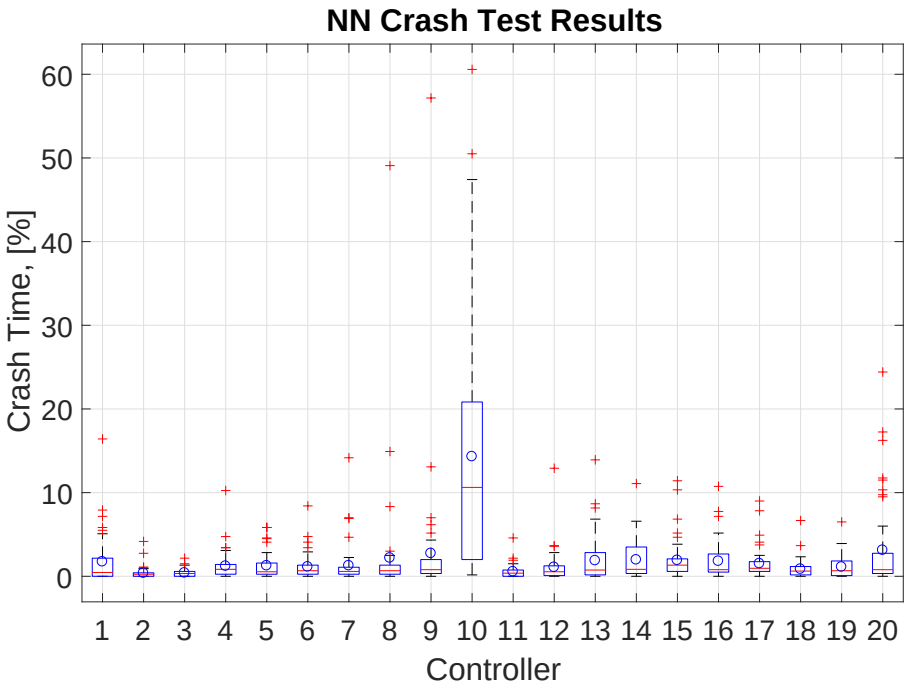


Figure 3-21: Crash results for scaled up MS (16 MAVs)

ten controllers are considered then this is increased to 94.90% ($\sigma = 3.891$).

Surprisingly, the avoidance behaviour exhibited by the MAVs in this smaller area is comparable to that of the original tests (recall Figure 3-15), with the MAVs spending only 0.34% ($\sigma = 0.307$) of the total simulation time within $30cm$ of one another. This is especially clear for controllers 9 and 11 to 14, where the MAVs usually do not collide once during the simulations (average crash percentage of 0.03%, $\sigma = 0.019$). However, from those five controller, only controller 10 obtains a reasonable coverage level. This just highlights the potential trade-off between collision avoidance and area coverage.

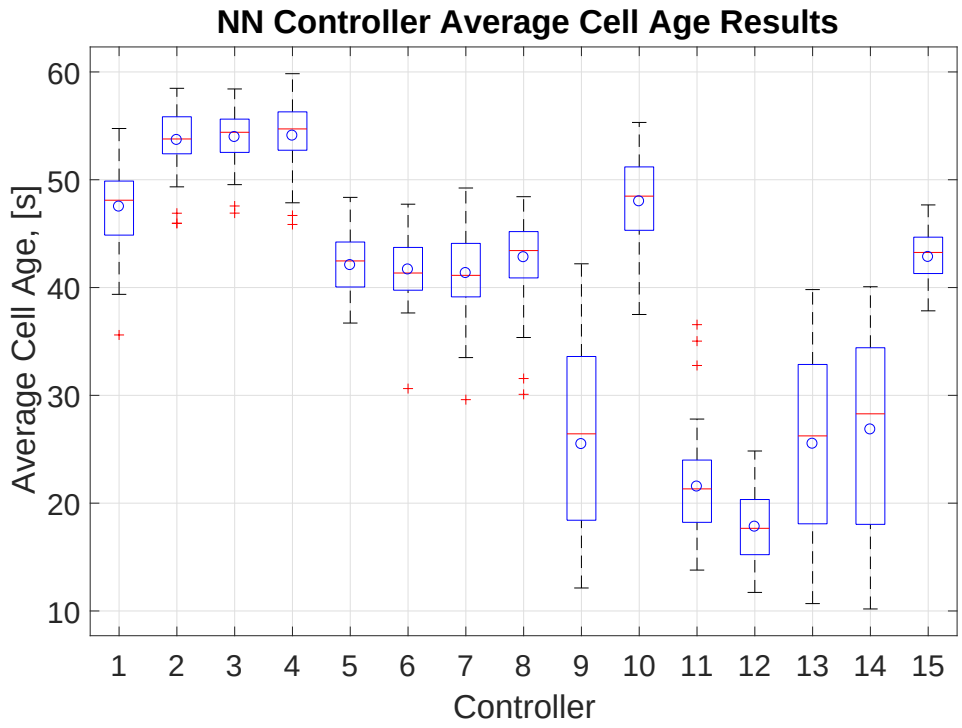


Figure 3-22: Average cell age results for simulated CyberZoo

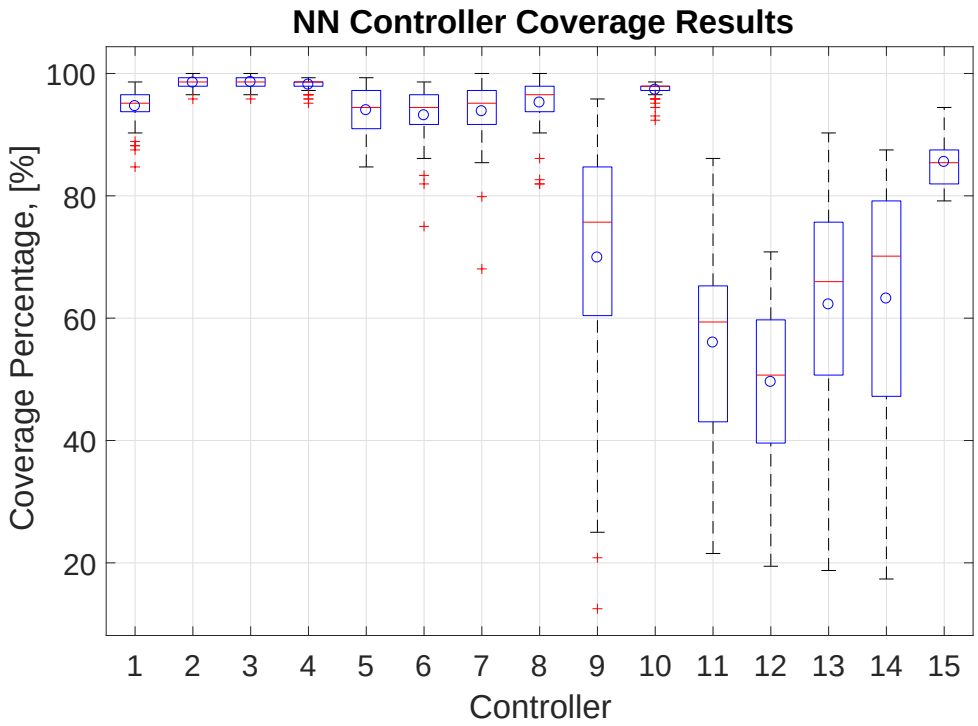


Figure 3-23: Coverage levels for simulated CyberZoo

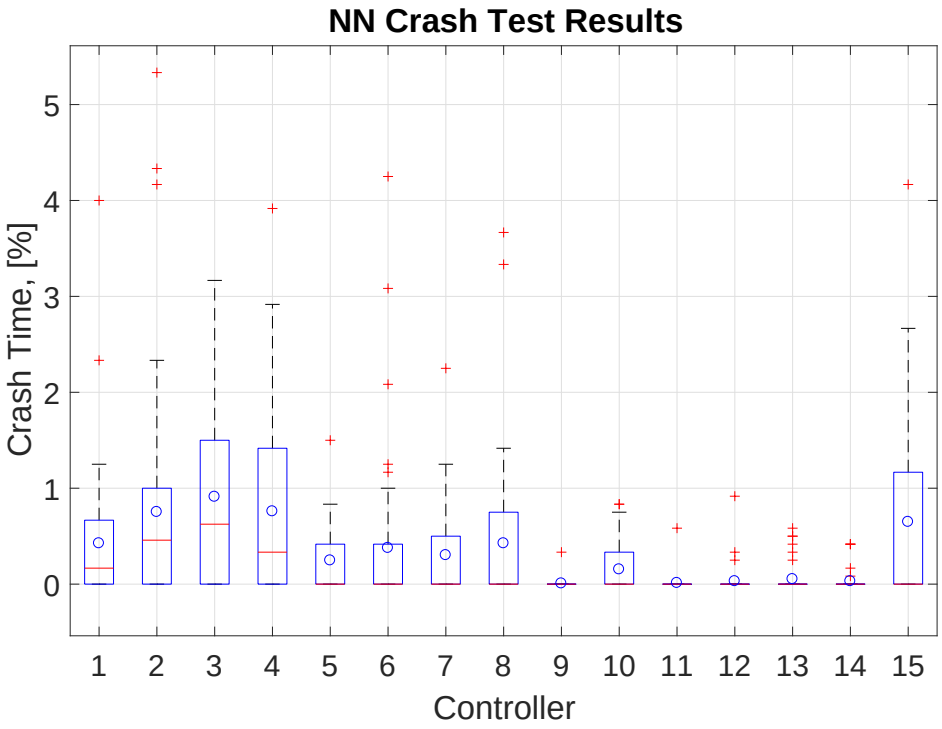


Figure 3-24: Crash results for simulated CyberZoo

When the above tests for the CyberZoo are repeated with three MAVs similar results to those mentioned previously are obtained. These are summarized in Figures 3-25 to 3-27. With more MAVs working together there is an increase in the coverage levels and average cell ages achieved over the course of the simulations. As before, the slightly more congested airspace does create a higher chance for collisions to occur but, from Figure 3-27, it can be seen that this is still usually less than 5%.

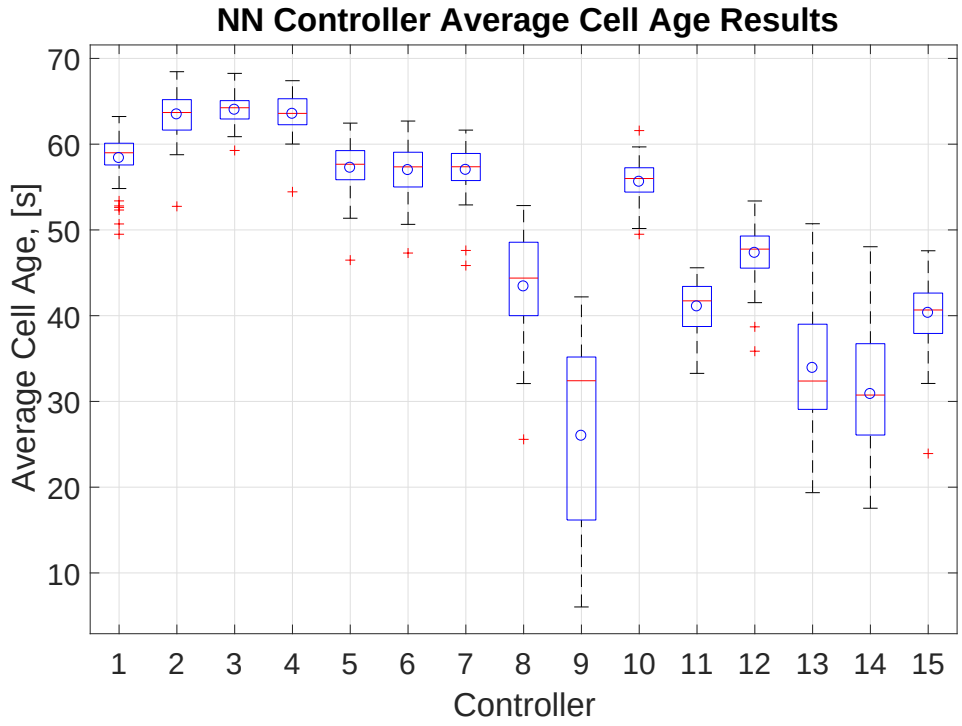


Figure 3-25: Average cell age results for simulated CyberZoo: 3 Agents

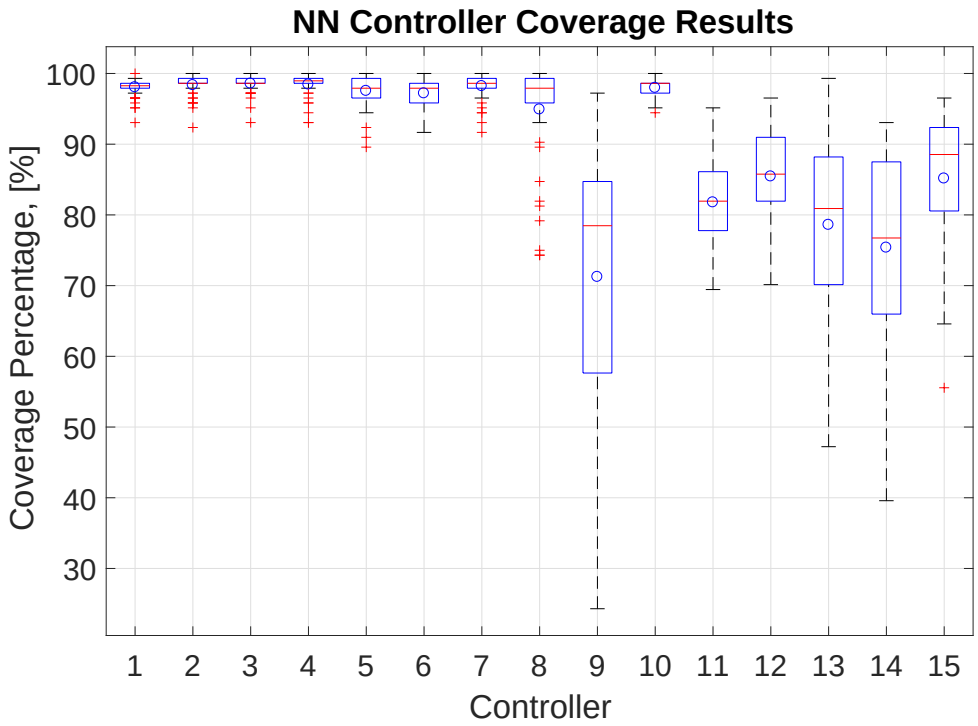


Figure 3-26: Coverage levels for simulated CyberZoo: 3 Agents

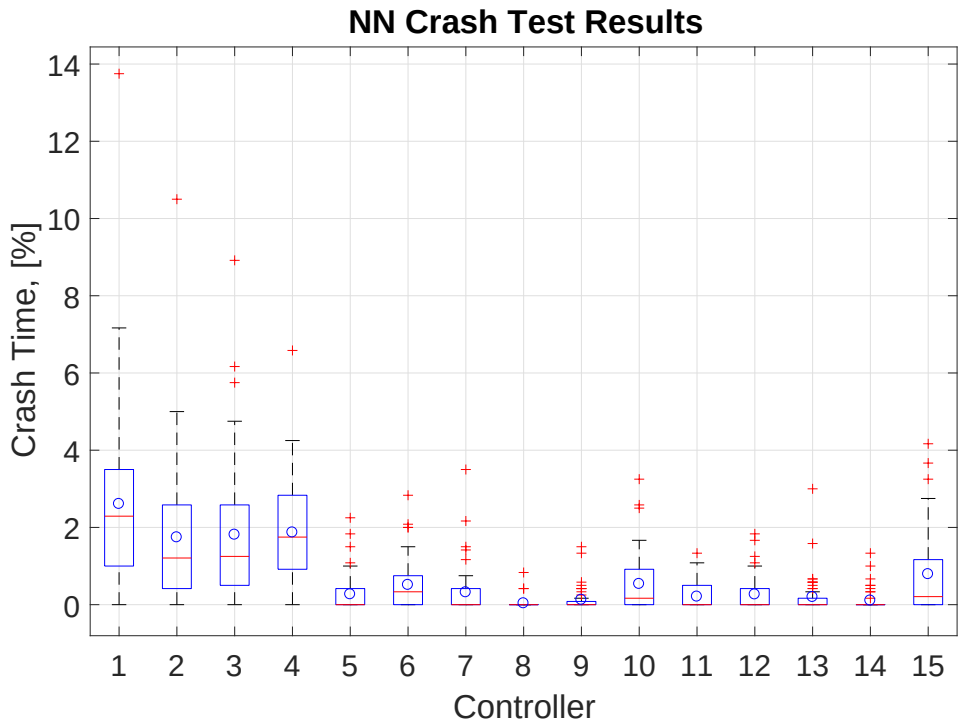


Figure 3-27: Crash results for simulated CyberZoo: 3 Agents

Comparison to Baseline Methods

Now, the NN controllers should have their performance evaluated against methods presented in current literature. For this, two baseline methods will be used. These are; a line-sweep method and a random walk approach. More details about these methods can be found in Appendix B. The test parameters are as follows:

- $25m \times 25m$ MS
- grid resolution of $1m$
- 8 MAVs were used
- $300s$ simulation time
- Time step of $0.25s$

Further, for the random walk implementation, 50 iterations of the test was performed where the MAVs were assigned a random starting position at the beginning of each iteration, while, for the line-sweep method, it was felt that only a single iteration would be sufficient as this method relies on a fixed flight plan so variations between iterations would be negligible. Finally, to make the random walk method more realistic, a simple avoidance measure was implemented to avoid collisions. More information about avoidance measure can be found in Section 4-3.

From the average cell age and the coverage results (Figure 3-28 and Figure 3-29 respectively) it is clear that the line-sweep method provides the best performance overall. It achieves complete coverage of the area in only $170.5s$, has the highest average cell age over time and as each MAV is assigned its own region to cover, there is no chance of collisions between the MAVs. This, however, was to be expected as this is a centralised method were the optimal flightpath can be determined offline and then assigned to each drone. This was just used to illustrate the optimal performance that can be achieved. Of more interest is the comparison to the random walk method. Surprisingly, the random walk method gives a slightly better

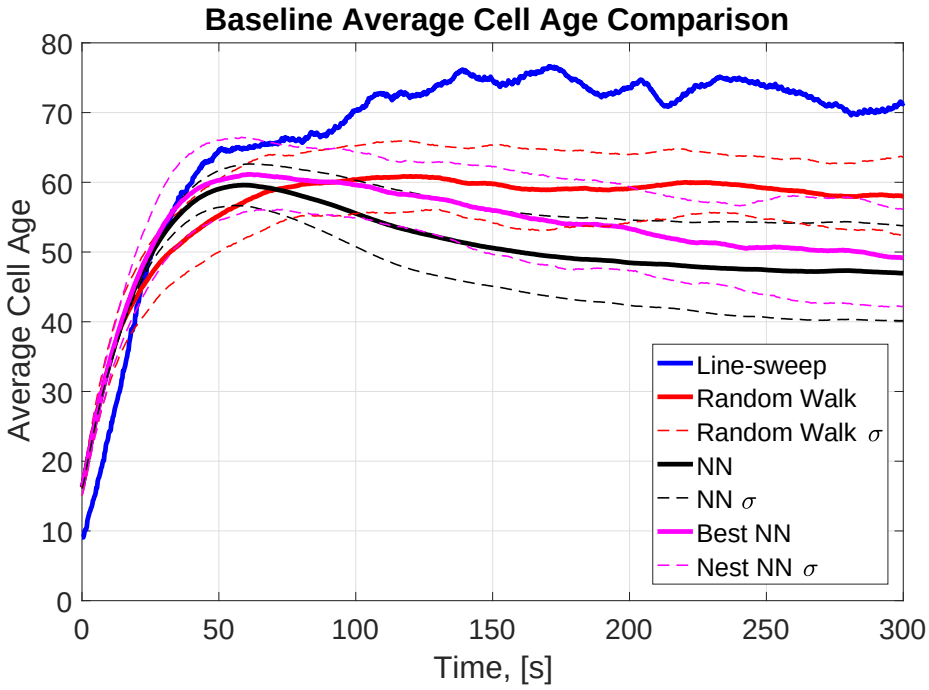


Figure 3-28: Baseline cell age comparison

average cell age than the evolved NN controllers ($56.59s$ with $\sigma = 7.415$ as opposed to $47.03s$ with $\sigma = 4.831$). A possible reason for this reduced average cell age could be a result of the avoidance behaviour of the NN. The random walk approaches avoidance measures are activated any time two MAVs approach closer than $0.8m$ while, during training, the NN apply a penalty to solutions where two MAVs are closer than $3m$ (recall the safety coefficient from Equation (3-5)). As a result of this the NN controllers would likely attempt to maintain larger distances between the MAVs than the random walk would, giving a less optimal trajectory. Luckily, this method of evolved NN is not a complete waste as it does outperform the random walk approach in terms of area coverage. With the best NN the coverage percentage achieved after $300s$ is, on average, 99.10% ($\sigma = 0.860$) while the random walk results in a small reduction of the coverage levels to 95.96% ($\sigma = 2.063$).

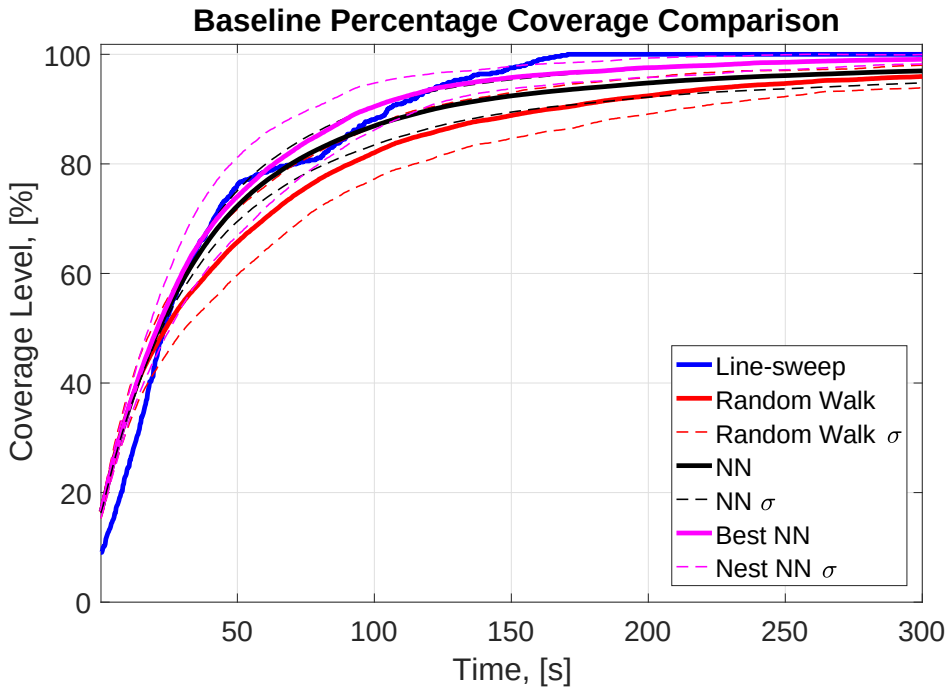
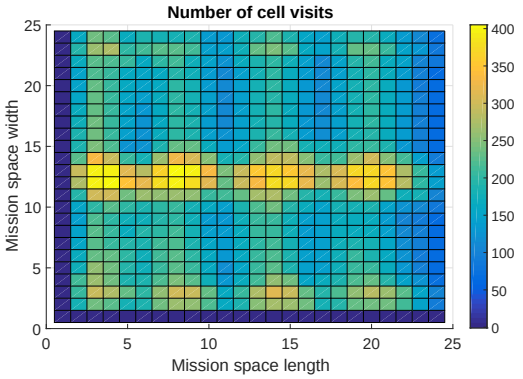
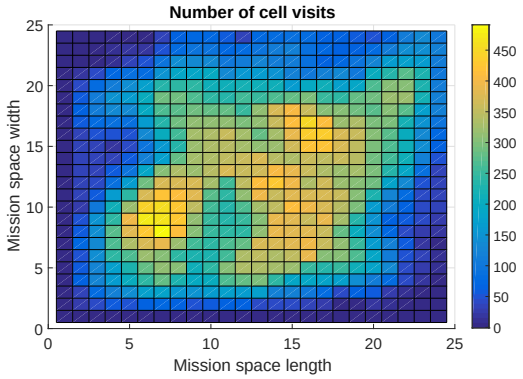


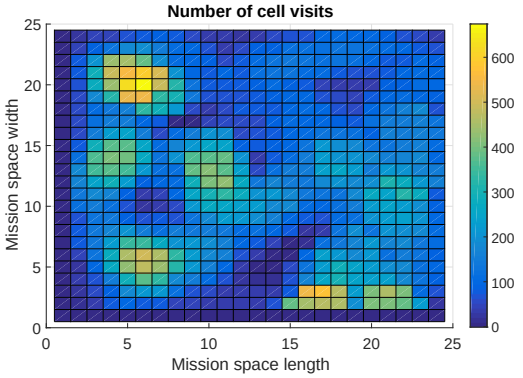
Figure 3-29: Baseline coverage percentage



(a) Line sweep tick counter



(b) Random walk tick counter



(c) Random walk tick counter

Figure 3-30: Line sweep tick counter

Robustness Tests

One of the main strengths of the NN controllers over a predefined search pattern, such as the line-sweep method, is its robustness to failures of the MAVs. In this section two separate tests will be performed on the baseline methods and the NN controllers to show their ability to deal with failures. First, a single agent will 'fail' midway through the simulation at $150s$. In the next test, two agents will fail, one at $100s$ and the other at $200s$. The results of these tests can be seen in Figures 3-31 to 3-34.

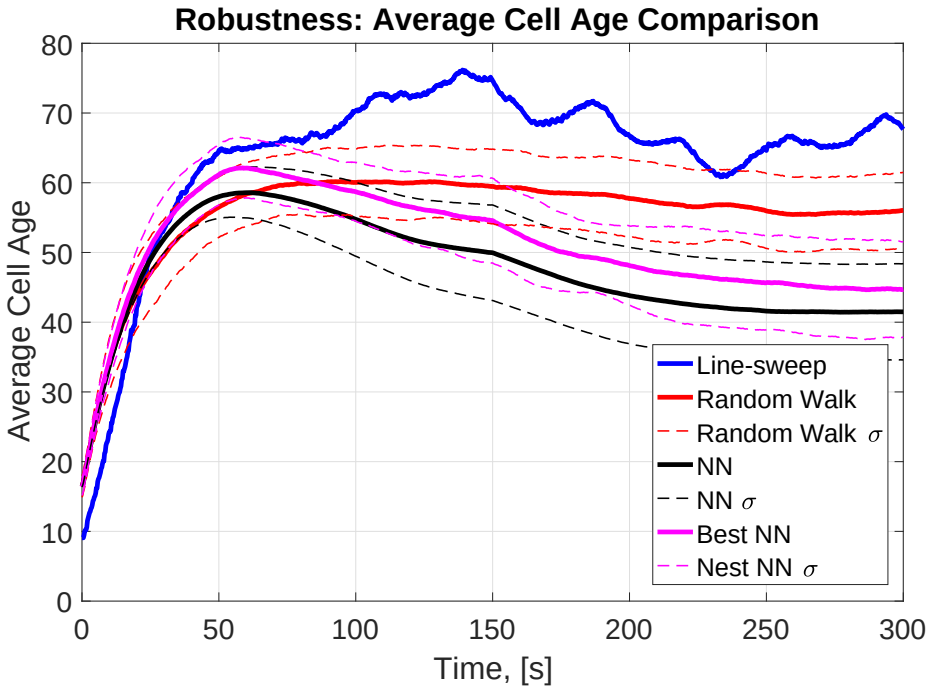


Figure 3-31: Robustness test 1: Cell age comparison to baseline

Figures 3-31 and 3-32 show the controllers performance under a single failure. The most noticeable difference, as expected, is with the sweep-planner. After a failure occurs, the average cell age declines for roughly $84s$ as the MAVs reorganise themselves into their new sub-areas after which it begins to rise again as the MAVs start performing their line sweeps once more. Here, the NN controller is still the worst performer with an average cell age of $47.45s$ ($\sigma = 7.436$) compared to the line-sweeps and random walks ages of $63.96s$ and $55.68s$ (7.119) respectively. Even the best performing NN only achieves an average cell age of $51.0s$ ($\sigma = 5.370$), which is still lower than that of the baseline methods. The effect of the failure is much less noticeable for these reactive methods, with neither the NN nor the random walk displaying more robustness to the failure than the other.

You will also notice that the MAVs are unable to completely cover the area when using the line-sweep method, with a coverage percentage of 97.54% compared to the random walks' coverage of 95.67% ($\sigma = 2.627$) and the NNs of 95.72% ($\sigma = 2.863$). A more efficient breakdown of the sub-areas used by the line-sweep method, as well as alternating sweep directions and dynamically assigned starting positions, could allow this method to handle failures more effectively but was considered beyond the scope of this thesis. In terms of this

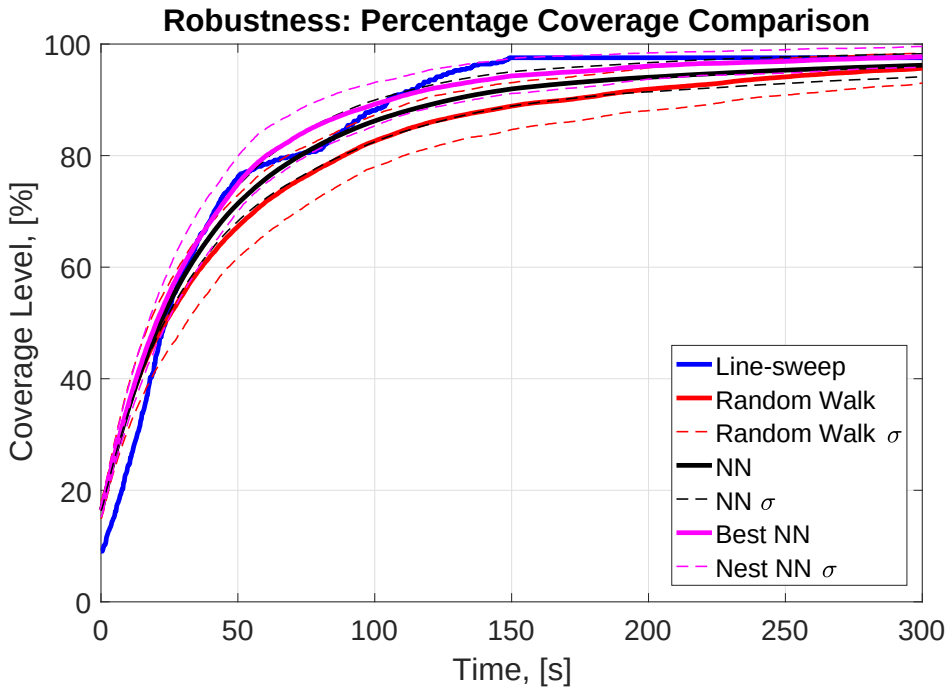


Figure 3-32: Robustness test 1: Coverage percentage comparison to baseline

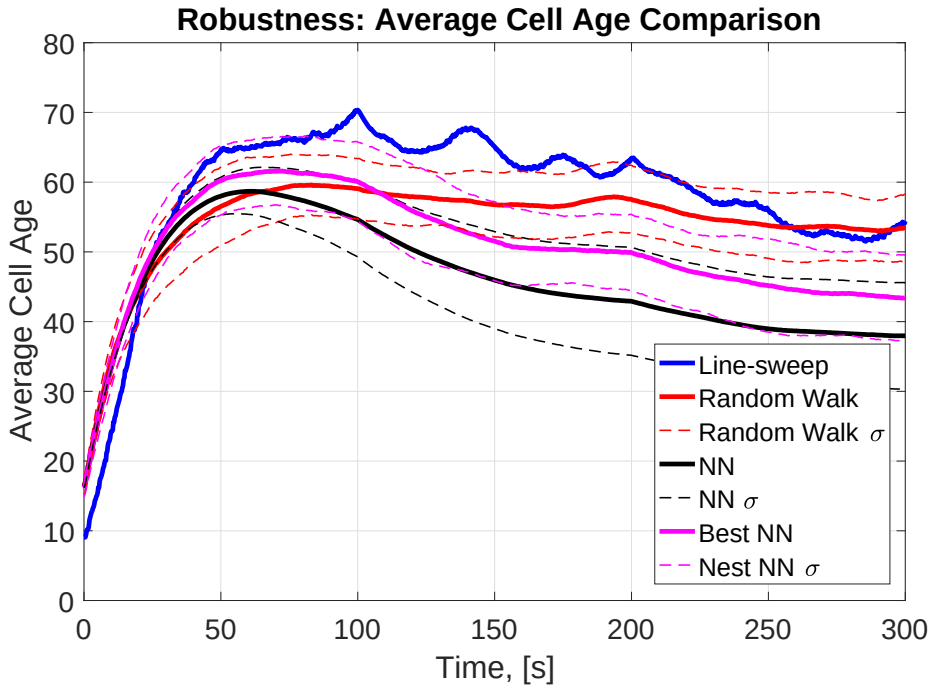


Figure 3-33: Robustness test 2: Cell age comparison to baseline

first robustness test, controller 12 gives the best performance for the NN controllers. This can, at least occasionally, still result in complete coverage of the area over the duration of the

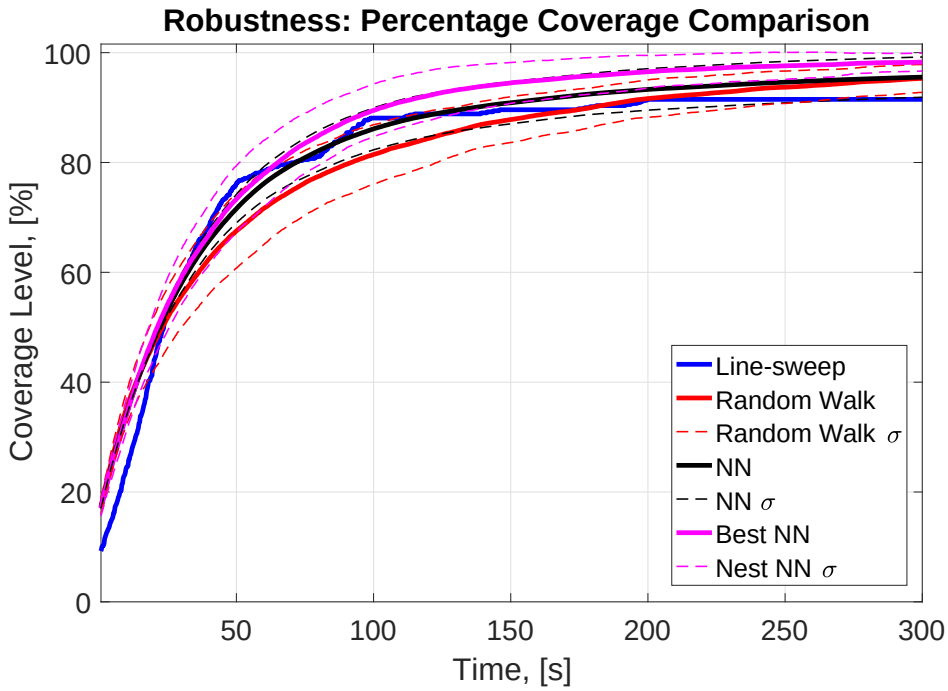


Figure 3-34: Robustness test 2: Coverage percentage comparison to baseline

simulation and has an average coverage of 97.71% ($\sigma = 1.847$).

For the second test were two MAVs fail, shown in Figures 3-33 and 3-34, the trend seen in the previous test continues. The effect of the failures on the line-sweep controller is even more pronounced with its average cell age briefly dropping below that of the random walk towards the end of the simulation time. Again, for the reactive methods there is a slight decrease in their average cell ages but neither of the two methods appear more robust to failures than the other when considering this metric. By analysing the results of the coverage levels, it can be seen that the percentage coverage for the line-sweep drops further from 97.54% in the previous test to 91.49%. Once more the NN and random walk remains nearly unchanged at 95.31% ($\sigma = 3.691$) and 95.30% ($\sigma = 2.438$) respectively. Finally, the top performing NN controller (controller 12), is still able to obtain coverage levels of 100% during some of its tests and its average coverage levels over all 50 tests is 98.30% ($\sigma = 1.605$).

These tests show that the NN controllers are more robust to MAV failures than a globally planned method, represented by a sweep planner in this work. However, it does not offer any advantage, in terms of robustness, over other reactive methods such as the random walk approach.

3-4 Summary

In this chapter, the NEAT algorithm which was used to evolve the NN controllers was described. This is an evolutionary algorithm that is designed to determine the optimal structure for the NN as well the the best weights for each of the connections. An important component

to this algorithm is the cost function used to evaluate the performance of the NNs from each new generation. In this thesis, the cost function presented by Duarte in [2] was used with a small adaption. Instead of penalising the total fitness over the entire duration of the simulation when two MAVs come within $3m$ of one another, the function used here only penalises the fitness gain for the specific time step where the positional constraint was violated. This was shown to produce controllers that resulted in a significant increase in the average cell age over time, in many instances it more than doubled, while also offering a slight increase to the coverage levels.

The next major design choice was selecting the inputs into the NN. For this, two examples of inputs from current literature were tested along with three additional input cases that made use of varying levels of environmental knowledge. Through the post evaluation of these NN controllers it was found that by increasing the amount of environmental knowledge used by the NN the performance of the system could be noticeably improve, which was to be expected. In the end, the best performance was achieved by the fourth input case. This relied on a series of eight virtual feelers (or antennae) that reached outwards from each MAV. These antennae returned the distance to the nearest obstacle and the age of the furthest cell that it could reach. On average, the controllers generated with this input case gave a coverage level of 94.5% with a crash percentage of 1.78%. In addition, these controllers were shown to be robust to failures of MAVs and could be successfully scaled to both larger and smaller MSs and was applicable to varying number of MAVs.

Lastly, the NN controllers were compared to two baseline methods identified from current literature. These were the random walk and the line-sweep method. As expected, the line-sweep method gave the best performance in terms of coverage levels, average cell age and as each of the MAVs are assigned their own subregions, there are no collisions. However, as this is a centralised method that relies on predefined flight paths it is not as robust to failures as the other two methods. Surprisingly, the purely random approach implemented in the random walk controller also outperformed the NN controller in terms of average cell age. Luckily the NN did offer better performance than a random approach in terms of area coverage. As both of these methods are reactive, their ability to deal with failures is nearly identical.

As of yet, the fuel level constraint has not been addressed. This expanded upon in the next chapter where a high level controller is used to determine when the MAV should return to the depot for refuelling.

Chapter 4

Fuel Monitoring Policy

Up until this point the constraint on the MAVs fuel, Equation (2-3), has not been addressed. This leads to an important component of this system, namely, the method for refuelling as the MAVs have a limited flight time. For the MAVs used in this system, this amounts to either charging the battery, or replacing the depleted battery with a charged one while the MAV is stationed at the depot. In this work, it was assumed that the depleted batteries could be manually replaced at the depots.

Next, the aspect of when a MAV must refuel itself forms a vital part of any fuel monitoring policy. Two common methods for achieving this are task assignment and task hand-off. In task assignment ([73, 74, 75, 76], a single flight path that covers the entire MS is created, using a sweep-planner for example. This path is then divided into separate sections that can be completely transversed by a MAV before it returns for refuelling. Covering all these sections, along with the refuelling can be seen as tasks that must be completed. A global planner is then used to efficiently assign MAVs to tasks with the goal of reducing the total time to complete the mission. Similar to this is task hand-off presented in [43, 77, 42, 45]. Again, complete flight paths that cover the area are generated but these paths may require more fuel than the MAVs have available. When a MAV is low on fuel, it will hand off its task to a MAV that is waiting for an assignment who will than continue following the path while the original MAV returns to the depot. Naturally, these methods require a complete flight path to be available for the duration of the mission so they will not be applicable with the reactive nature of the NN controllers presented in this work. In addition, these methods are usually formulated as a Mixed Integer Linear Programming (MILP) which are computationally expensive to solve (they are typically NP-hard [53, 78]). Therefore, these methods were not appropriate for use in this work.

Other methods of refuel scheduling include using motives and artificial emotions [79], an auction based task allocation process [80], formulating the times to start refuelling as a Linear Programming (LP) problem where the goal is to reduce the overlap in refuel times between all the MAVs [32] and fixed threshold scheduling. In the end, the fixed-threshold method was selected as this is highly scalable, it is the most computationally efficient and it requires little to no communication between the MAVs.

4-1 Fixed Threshold Scheduling Method

Currently in papers researching the use of MAVs in PA there is very little work focussing on the aspect of refuelling. In most papers it is assumed that there are enough MAVs to perform complete coverage before refuelling is necessary. [16, 17] are two of the few papers that do not take the above mentioned approach and, instead, they implemented a very simplistic method for incorporating fuel constraints. Each agent was assigned a predefined path to follow (created using a sweep planner) that could potentially require more fuel than was available. Thus, the agents were periodically required to stop following their assigned trajectory and return to base for refuelling. Once refuelled, the agents returned to the last visited point in their trajectory and continued following their original path. This was a purely greedy policy that made no attempt to coordinate refuelling times amongst all the agents which led to congestion at the base.

In literature this is known as Fixed Threshold refuelling. This method states that as soon as the agents energy level, E_{A_k}, drops below a threshold, $E_{threshold}$, it must return to base for refuelling. Naturally the effectiveness of this method is directly proportional to the chosen $E_{threshold}$ and, in a multi-agent setting, this method would likely give a suboptimal result as there is no chance of cooperation.

This purely greedy method is highly scalable as it requires no cooperation between agents. However, the drawback is that this will lead to large levels of congestion at the depots, reducing the efficiency of the system. To combat this, [81] introduced a number of variations to this central method to increase the recharging efficiency. First, is the sanctuary strategy. This attempts to prevent agents from interfering with the others refuelling by limiting the agents that can enter the refuelling area (in other words this prevents congestion at the depot). This was done by "hiding" the depot from the other agents when one of them was refuelling which prevented them from returning to the depot. Next they looked into opportunism, which is the behaviour that agents try to recharge as soon as they see the station, irrespective of their remaining fuel. In their paper this was implemented by relating the frequency at which the agent checks if the depot is "visible" (i.e. not in use) with the agents remaining fuel. The lower their fuel, the more frequent the checks and the more likely that it would be able to refuel.

In this work, similar strategies to those mentioned above were employed. Like [81] this work also made use of the sanctuary principle. The depot is able to broadcast whether or not it is currently in use to all MAVs in range. When the depot is in use, this has the same effect as hiding it from the MAVs and they will no longer attempt to return for refuelling. Next a waiting area was introduced at the depot. This gives MAVs that would have crashed due to lack of fuel a place to safely land while they wait for the depot to become free. This will help deal with the problem of congestion and failure due to lack of fuel which was not addressed in [81]. Finally, opportunism was created by defining two separate threshold values, E_{thresh} and E_{critic}. The first is the low fuel threshold which governs when a MAV can start attempting to return to the depot while E_{critic} indicates a critical fuel level. Once the fuel reaches this critical threshold value, the MAV must return to the depots waiting area. While in this queue, the MAV will not perform the surveillance task or use additional fuel.

This was implemented through the use of a Behaviour Tree (BT) which is described in the next section.

4-2 Behaviour Trees

In the past, finite state machines have been used to describe autonomous behaviours. However, since the introduction of the BT in the gaming industry, they are quickly gaining prominence in designing AI for games and in robotics [82, 83]. BTs are depth-first, directed acyclic graphs that are used to represent the decision process, [49, 84]. Using a hierarchical organisation, the BT can be used to describe complex behaviours by making use of building blocks created from smaller, less complex sub-tasks, [83].

A BT, an example of which is shown in Figure 4-1, has a rooted tree like structure that is composed of nodes and directed edges. If two nodes are connected by and edge, the node connected to the outgoing edge is called the *parent* while the node with an incoming connection is the *child* and if a node has no children it is called a *leaf*. Naturally, the first node will not have any parents and this is known as the *root* node. It is important to note that nodes may have multiple parents which allows sub-trees to be reused but it decreases the readability of the tree, [84]. For this reason, the re-use of sub-trees is usually performed at the nodes that execute a task and not at the control-flow nodes.

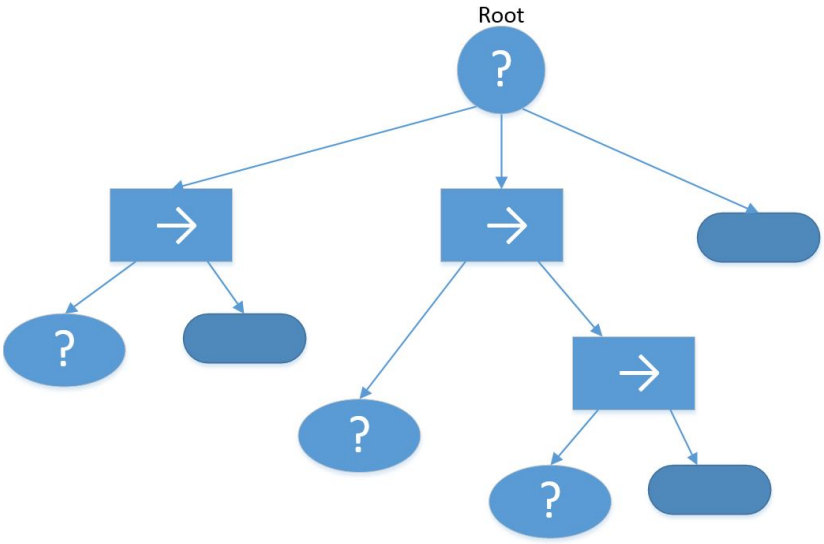
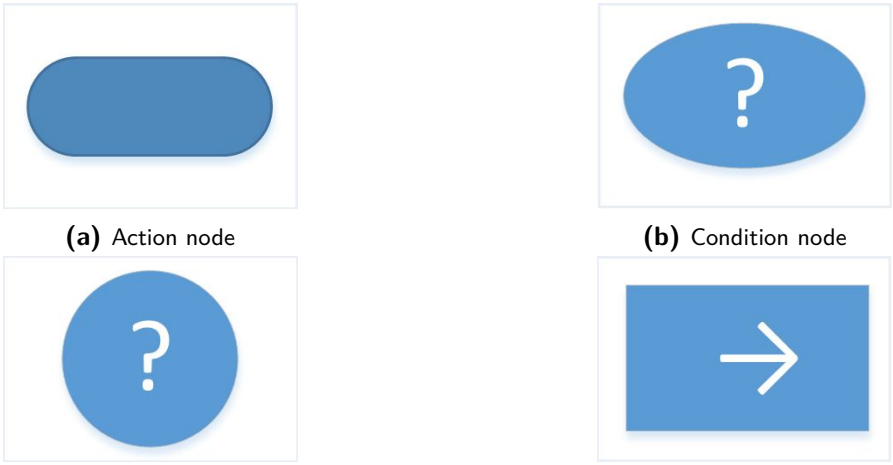


Figure 4-1: Behaviour tree example

The nodes in a BT can be further classified according to their types [84, 4]. There are six possibilities, namely; Selector, Sequence, Parallel, Decorator, Action or Condition. The first four types mentioned form composite (or control-flow) nodes which are used to determine how the BT will be executed, the Action type dictates how the MAV interacts with the environment and the Condition type tests a property of the environment. The Condition and Action types can only be applied to a leaf node while composite nodes must have at least one child node. For this work, only the action, condition, selector and sequence node types will be used and their symbols are shown in Figure 4-2.

When executing the BT, the root node periodically generates a signal, known as a *tick*, which propagates through the branches of the tree, according to the node type. This *tick* starts at the root node and moves from left to right through its child branches, propagating through the



(a) Action node **(b)** Condition node

(c) Selector node **(d)** Sequence node

Figure 4-2: Node type examples

entire branch before moving on to the next one. Once the tick reaches a node, it is evaluated and returns one of the following statuses; *success*, *failure* or *running*. The combination of these *success* and *failure* values are responsible for routing the tick to the desired sub-task. An execution is considered complete once the root node receives a *success* status from one of its branches or all of its branches returns *failure*. A list of the node types and their returned statuses is given in Table 4-1.

Table 4-1: Node types and their statuses, [4]

Node Type	Success	Failure	Running
Selector	If one child succeeds	If all children fail	If one child is running
Sequence	If all children succeed	If one child fails	If one child is running
Action	Upon completion	When impossible to complete	During completion
Condition	If true	If false	Never

4-3 Implemented Behaviour Tree

Using the above mentioned node types a BT to coordinate the refuelling aspect was created by hand. This BT acts as a higher level controller to that of the NN and is used to determine the MAVs current behaviour. The desired behaviour can be broken down into five sub-tasks/states which are:

1. Avoidance
2. Homing
3. Charge
4. Wait

5. Surveillance

The full BT is seen in Figure 4-3 and more a more detailed explanation of the different branches of the tree follows below.

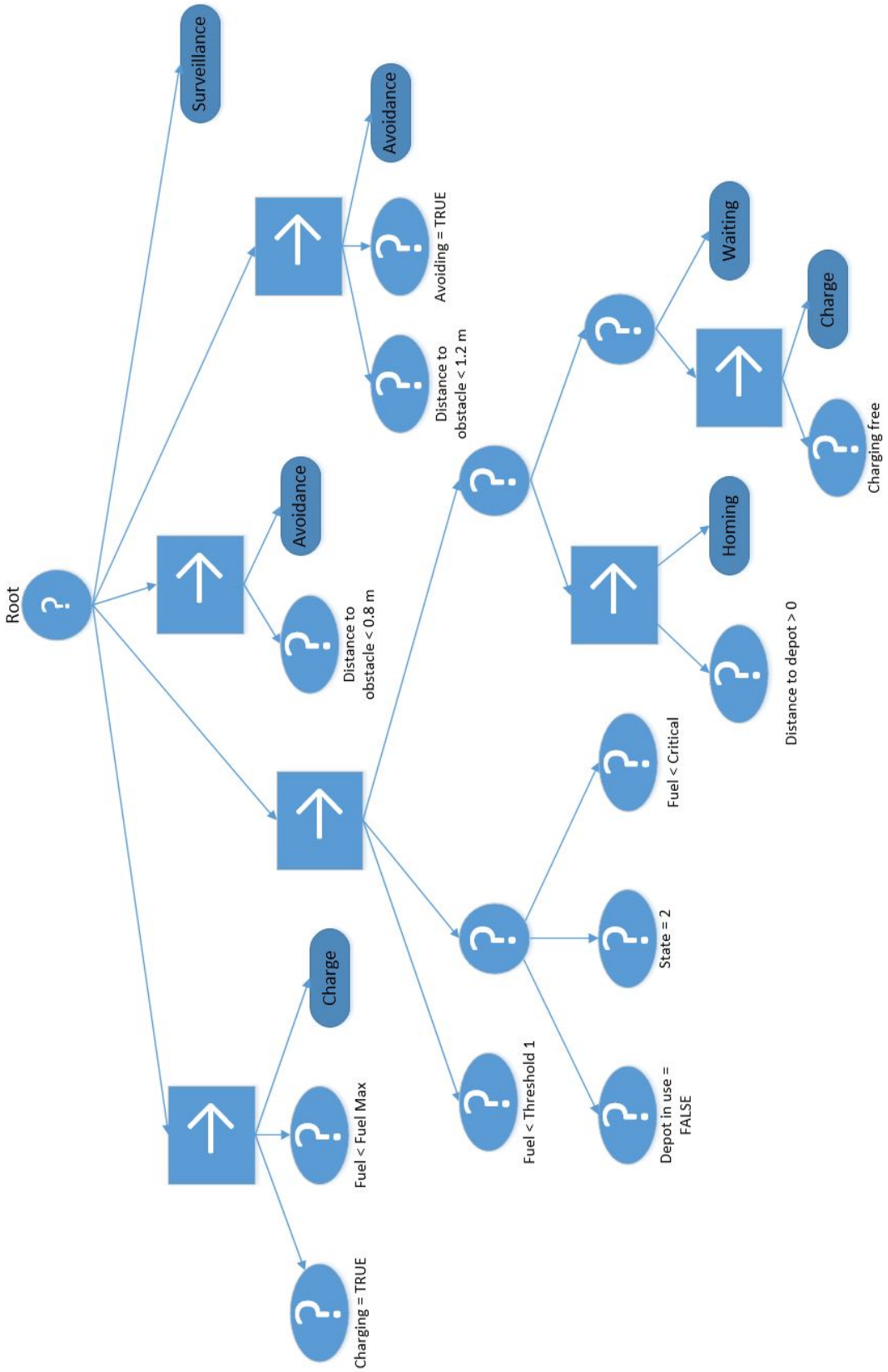


Figure 4-3: Implemented behaviour tree

The first child node of the BT forms part of the *charging* subtask. This ensures that if the MAV is in its *charge* state (charging = TRUE) and the fuel level is still less than the maximum, then it must remain in the *charge* state.

This is combined with the second child branch, shown in Figure 4-4, which manages the *homing, charging* and *waiting* states. First, the current fuel level of the MAV is checked and if this is lower than a given threshold, then refuelling may be considered. Further, one of three other conditions must be satisfied before the MAV can return to the depot for recharging. The first condition is given as 'Depot in use = FALSE' which is related to the 'visibility' of the depot discussed earlier (recall Section 4-1). If any of the MAVs are in the *homing, charging* or *waiting* states then the depot is invisible to the other MAVs and the Depot in use flag is set to TRUE otherwise it is FALSE. This prevents multiple MAVs from attempting to recharge themselves at the same time. Once the 'Depot in use' flag is set to TRUE and the depot is hidden from all the MAVs, the second condition is used to ensure that if a MAV is in the *homing* state, it will remain in the homing state until it reaches the depot. Finally, the last condition will allow a MAV to return to the depot when its current fuel level has reached the critical threshold.

Once one of these conditions are met, the MAV is placed into either the *homing, charging* or *waiting* state through the third child branch seen in Figure 4-4. Here, if the MAV is not currently at the depot, it enters the *homing* state to return to the depot. Otherwise, the MAV lands and it either enters the *charge* state if no other MAV is currently being refuelled or it joins the queue waiting to be refuelled. While waiting, it is assumed that the MAVs do not consume any further fuel.

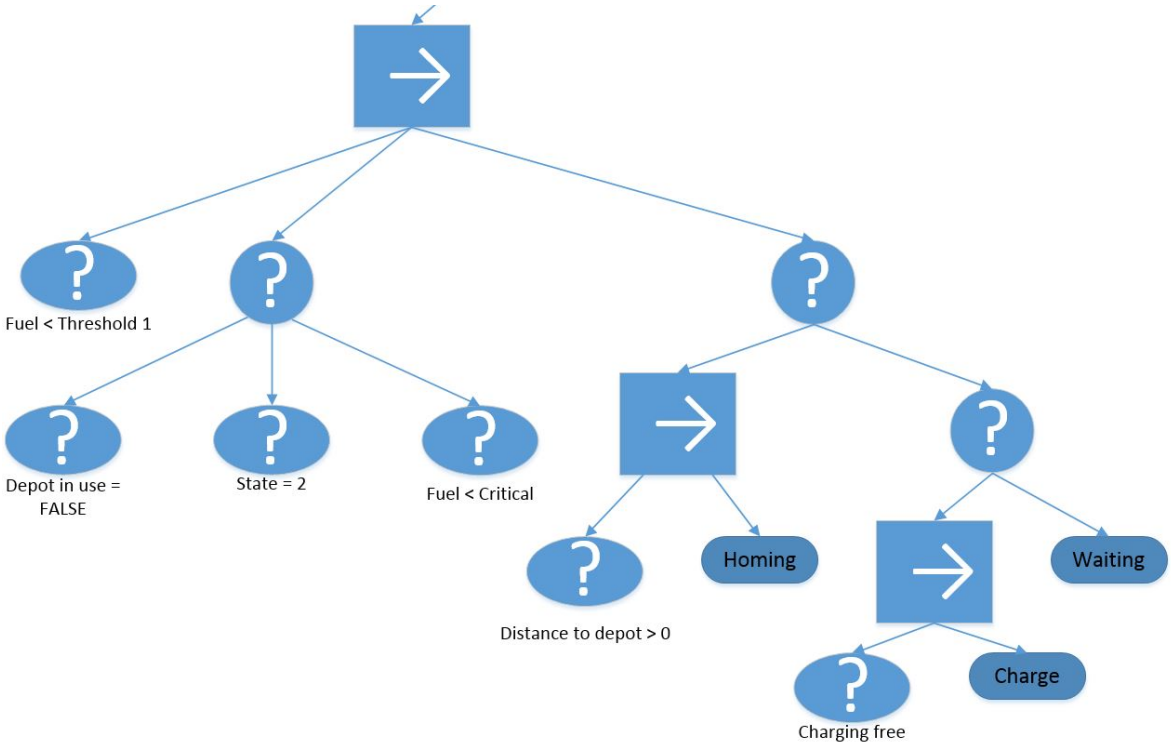


Figure 4-4: Charging decision process

While the NN does evolve basic avoidance behaviour, as seen in Section 3-3-3, it is still desirable to have a hard coded strategy in place to avoid collisions and prevent the MAV from leaving the MS. This is depicted in Figure 4-5. This checks the distance to the nearest obstacle and if this is less than $0.8m$ then the MAV will implement an avoidance maneuver. Once the *Avoidance* state has been activated, it will remain in this state until the distance to the obstacle is greater than $1m$.

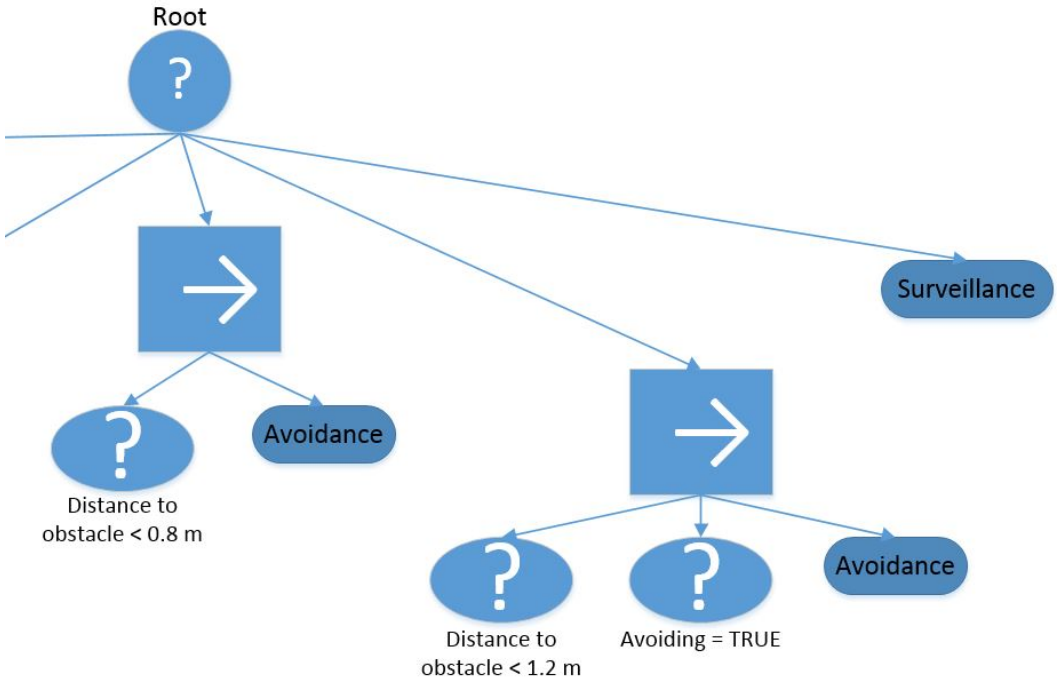


Figure 4-5: Avoidance decision process

The avoidance behaviour implemented here is based off of an evolved collision avoidance strategy presented by Szabó in [59]. This relies on the fact that if two MAVs are on a collision course, the conflict can be prevented if at least one of them reverses its flight direction [59]. For convenience, the proof is included in Appendix C. If both MAVs reverse their respective flight directions, the collision will be avoid but the potential conflict may still remain. For example, once the avoidance maneuver has been completed, both MAV could return to their original flight paths which would again lead to a collision. As such, Szabó implemented a turning maneuver during the reverse flight to prevent the conflict reoccurring. This ensures that successive conflicts are always slightly different from on another which, through a repetitive process, will lead to a collision free flight path.

As the MAVs used by Szabó could alter their headings during the course of the experiments, the turning maneuver could be accomplished by specifying a rotation about the yaw axis. However, this is not applicable to the experiments performed in this work as it is assumed that during operation, the MAVs heading is kept constant. As a result, the approach had to be modified slightly to account for this fact. Instead of a rotation about the yaw axis, the actual reverse flight direction was adjusted by $\alpha = 45°$, shown in Figure 4-6. Like the previously mentioned turning maneuver, this ensures that successive conflicts are slight different from one another.

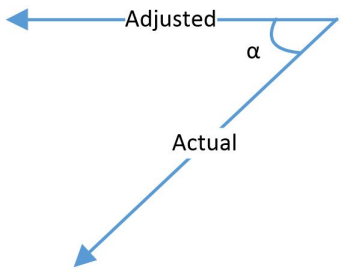


Figure 4-6: Example of adjusted flight path for collision avoidance

An example of the collision avoidance maneuver for two MAVs is shown in Figure 4-7. In the image below, the MAVs start at $(5, 2)$ and $(5, 9)$ respectively and initially they fly straight towards one another. Once the distance between them is less than the required threshold of $0.8m$ the avoidance strategy is activated. This is indicated by the stars in the figure. As can be seen, the MAVs can safely avoid the initial collision and, using the turning maneuver, they avoid repetition of the same conflict. This also shows the by chaining multiple avoidance movements, the MAVs are able to safely move past one another and find a collision free flight path.

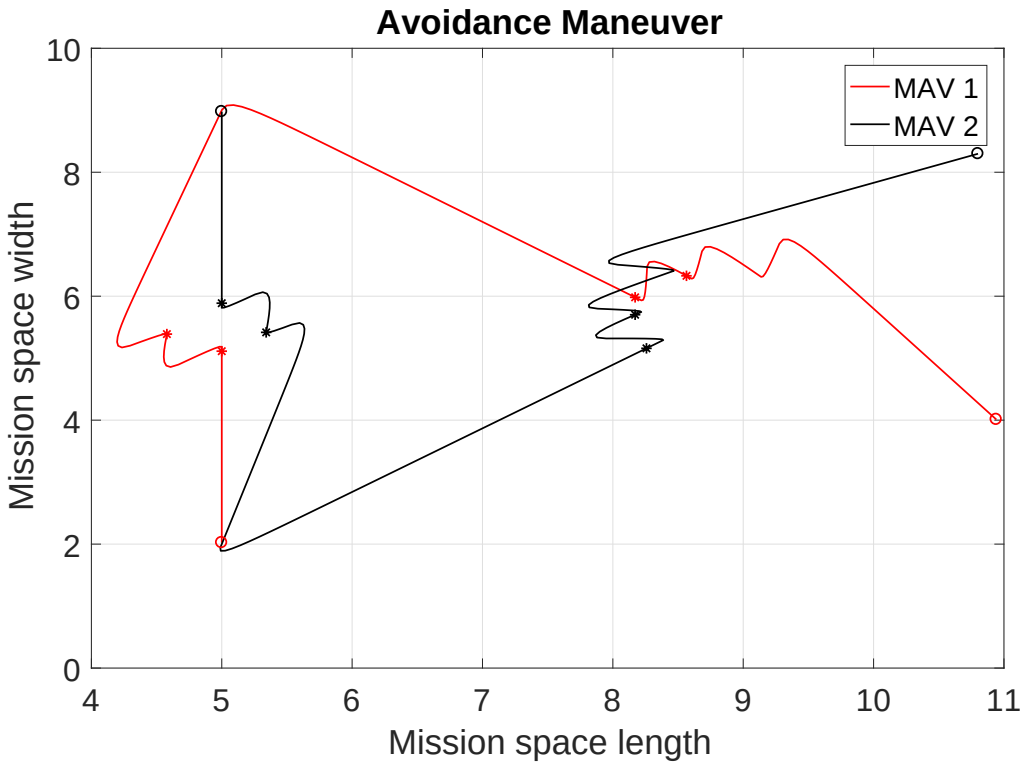


Figure 4-7: Example of collision avoidance behaviour

4-4 Simulation Results

In this section, the performance of the aforementioned BT is analysed in simulation. For these tests, the 20 controllers identified in Section 3-3-3 will again be tested in a $25m \times 25m$ simulated environment discretised into $1m \times 1m$ cells.

4-4-1 Avoidance Behaviour

For the first tests, the impact of the additional avoidance measures are investigated. For this, the BT is implemented in the simulations but it is assumed that the MAVs have infinite fuel. Box plots showing the results for the 50 tests performed on each of the 20 controllers is shown in Figures 4-8 to 4-10. As in the previous plots, the red crosses show the outlier results, the blue circle shows the average result for each controller and the red line shows the median value.

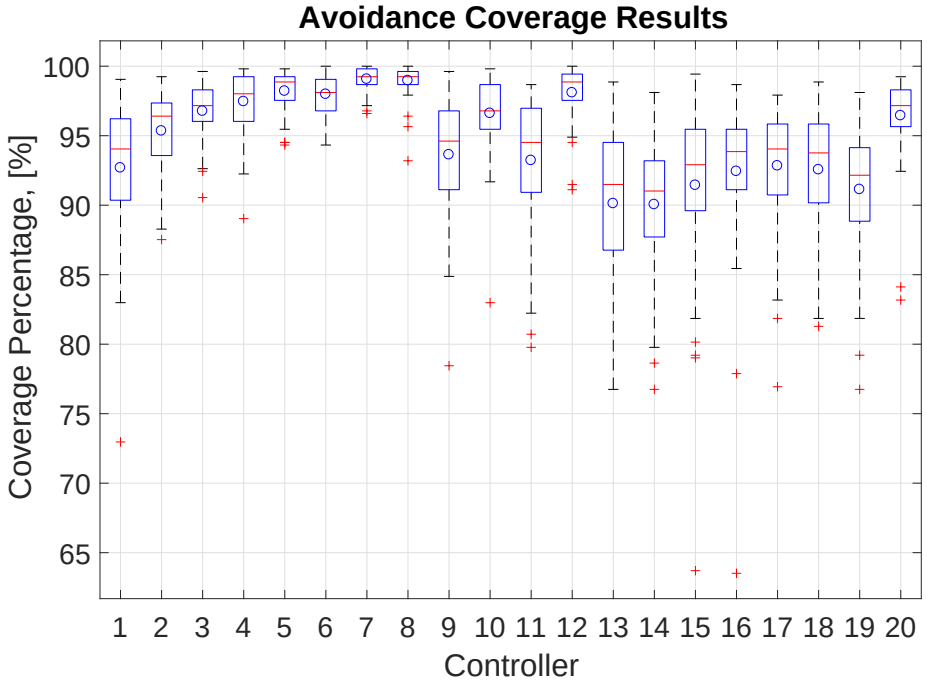


Figure 4-8: Coverage levels with avoidance measures

In terms of coverage levels achieved (Figure 4-8), no significant changes can be observed. On average, the controllers acheive a coverage level of 94.75% ($\sigma = 3.056$) after $300s$ as opposed to the 94.69% ($\sigma = 3.239$) achieved previously. This was to be expected though. In the previous tests it was observed that the NN controllers were able to avoid collisions for most, if not all, of the simulation. As a result, the additional avoidance measures implemented here would only occasionally be needed. Thus their impact on the system should be negligible in terms of the coverage and average cell age. The plots for the average cell age (Figure 4-9) again confirm this expectation. There is very little difference between these plots and the original results from Figure 3-16 shown earlier. Here, on average, the controllers obtain a

mean cell age over the simulations duration of $46.96s$ ($\sigma = 4.480$) as opposed to the mean age of $47.03s$ ($\sigma = 4.831$) that was achieved without the use of the additional avoidance measures.

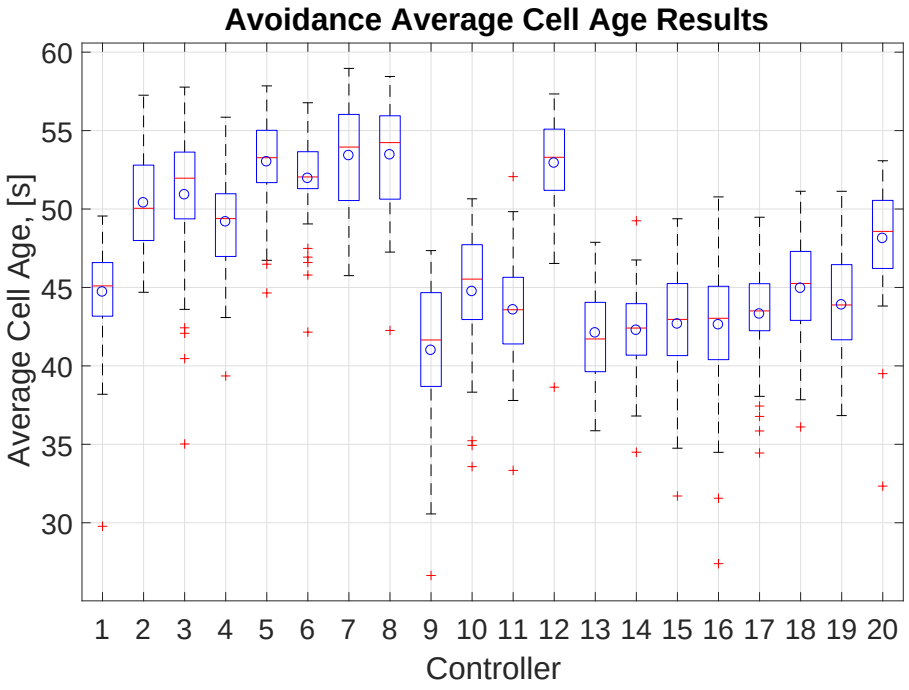


Figure 4-9: Average cell ages with avoidance measures

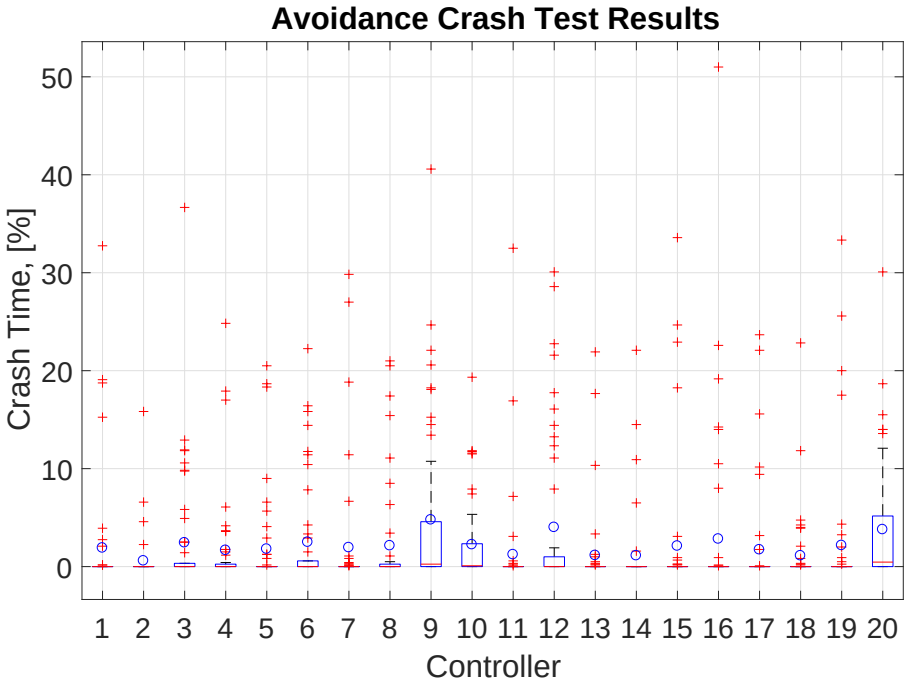


Figure 4-10: Crash instances with avoidance measures

Unsurprisingly, the main difference in the performance is seen when comparing the number of crashes between MAVs (Figure 4-10). If one was to consider only the average crash percentage for the 20 controllers, this could be misleading. For the case where the additional avoidance measures are used, the average crash percentage is 2.15% ($\sigma = 0.145$) while 1.78% ($\sigma = 1.593$) was achieved in the original tests. This would seem to indicate that there is not any benefit to the avoidance measure, that it could in fact reduce the performance. However, this in not the case as shown in Table 4-2. This table shows the number of tests, out of 50, where not a single collision occurred. From this it is clear that the additional avoidance measures adds a significant improvement to the systems performance in terms of the crash time. In many cases it more than doubles the amount of simulations where not a single collision takes place.

Table 4-2: Number of tests with no collisions

Controller	With Avoidance	Without Avoidance
1	41	19
2	46	34
3	35	33
4	35	10
4	38	15
6	31	12
7	38	11
8	35	14
9	16	15
10	25	17
11	42	30
12	32	13
13	40	39
14	45	33
15	40	18
16	40	25
17	42	23
18	39	24
19	40	20
20	20	3

What can also been seen from Figure 4-10 is that in this case it appears that the effect of any crashes is more pronounced resulting in a higher crash percentage than shown previously. This is indicated by the much greater amount of outlier results and their respective magnitudes. To understand why this is, one must look at what caused the avoidance strategy to fail. Three main scenarios were identified that led to the algorithm failing. These are:

1. Two MAVs could have their starting positions initialised to within $30cm$ of one another. While this case is very rare, it did occur in a number of simulations which caused the MAVs to begin in a crashed state. As the MAVs are initialised with a velocity equal to $0m/s$ the collision avoidance method would cause both MAV to move off in the same direction, never separating.

2. While performing an avoidance maneuver, the MAVs do not check for additional collision

courses with other MAVs. As a result, two MAVs that are currently performing an avoidance movement could collide with one another while trying to avoid their original conflicts.

3. In the last case, two MAVs could be commanded to travel in a similar flight direction as a result of the avoidance maneuver. While in theory they will never collide, they will also never separate from one another and resolve the conflict. This can occur when two MAVs are approaching one another and the first is already performing an avoidance movement. Depending on the second MAV's original flight direction, it could be placed on a parallel flight path by the avoidance strategy. Alternatively, as the method works on the MAVs commanded velocity, it is also possible for the NN inputs to give two MAVs roughly the same velocity outputs just before the avoidance measures are activated. This will also cause the MAVs to perform the same avoidance motion.

These cases can cause the MAV to never resolve their collision courses, resulting in the higher crash times seen in the boxplot.

4-4-2 Refuelling

Arguably, the most important factor for determining the success of this solution is the MAVs fuel level over the duration of the mission. When the BT is implemented, this should remain above 0 at all times. In Figure 4-11 the fuel levels for each of the eight MAVs is shown over the course of a typical persistent surveillance mission. It is important to note that the discharge and refuelling rates were exaggerated in this figure so that multiple refuelling cycles could be illustrated. This figure shows that none of the MAVs become completely discharged over the course of the simulated mission. You will notice that all MAVs begin with the same fuel level, then, as they reach the first fuel threshold they start attempting to refuel. With the large number of MAVs sharing the depot, initially, not all MAVs will be able to charge themselves before running out of fuel. This is where the queueing feature comes in, which is represented by the sections where the fuel level is held constant for one or more MAVs. As mentioned, this prevents crashed due to low fuel levels while it has the added benefit of staggering the refuel times of all the MAVs which will make future refuelling cycles more efficient.

As mentioned earlier, exaggerated charging and discharging rates were used for the simulated tests. These were $\dot{e}_{depot} = 5 units/s$ and $\dot{e}_m = 0.5 units/s$ while $E_{thresh} = 50\%$ and $E_{critic} = 34\%$ were used as the first and second refuelling thresholds. For the physical implementation on the other hand, the refuelling was based on the batteries current voltage. Table 4-3 shows the actual values used for the physical tests on the Parrot bebop 2. Here, the threshold values correspond to the low battery voltage warning given by Paparazzi. It is also important to note that for the physical tests, the batteries are not charged while the MAV is at the depot but, rather, they are replaced by new batteries. Therefore, in the physical tests, the depot recharging rate is replaced with a fixed refuelling time of roughly $40s$.

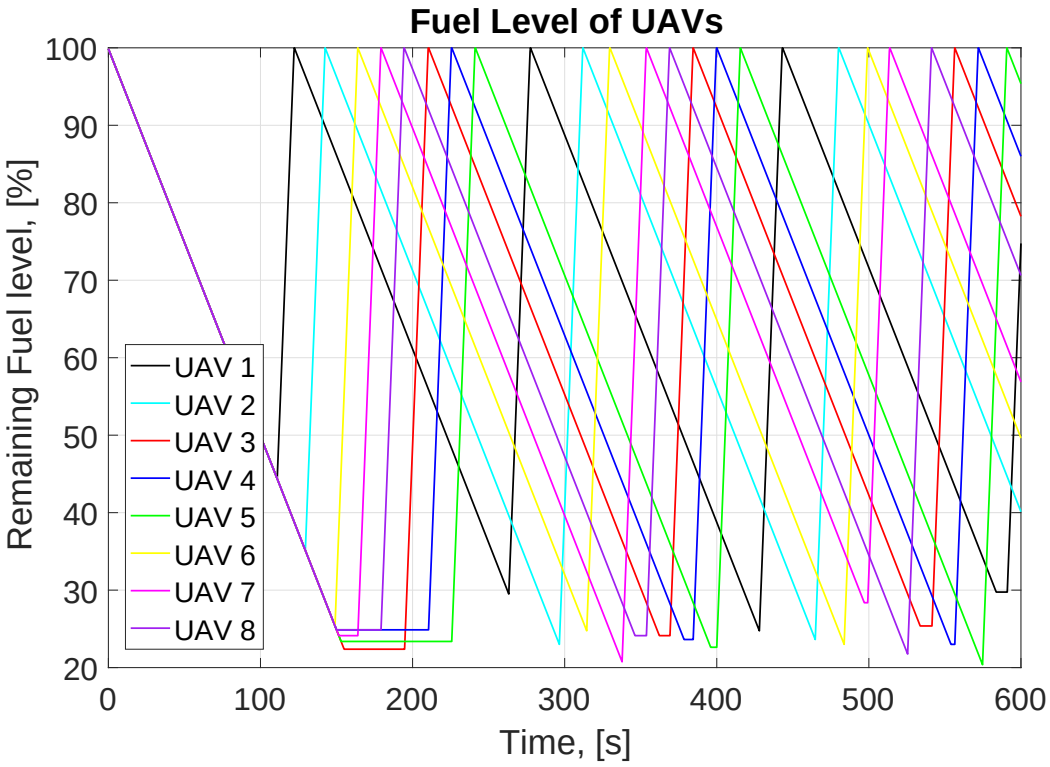


Figure 4-11: MAV fuel levels

Table 4-3: Parrot Bebop 2 refuelling parameter values

Parameter	Bat. Voltage, [V]
$E_{m,max}$	12.4
E_{thresh}	10.7
E_{critic}	10
E_{min}	9.6

Comparison to Original Performance

Now that we know what the impact of the additional avoidance measures is, we can now analyse how the refuelling aspect affects the systems performance. In these tests, the exaggerated refuelling parameters were again used and the refuelling station (or depot) was located at the centre of the MS. The results of the coverage levels for the 50 iterations of each of the 20 controllers is given in Figure 4-12. On average, we observe a slight increase in the coverage levels achieved, 97.41% ($\sigma = 1.613$), as compared to the original results of 94.69% ($\sigma = 3.239$) seen in Section 3-3-3.

To explain why there is this increase in coverage levels, one has to explore the explorative behaviours created by the NN. In general, the MAVs tend to move towards the edges of the MS and they circle the MSs borders with a slight zigzag pattern. Occasionally, a MAV would return to the centre and explore before flying off towards the edges once again. While this behaviour ensures that most cells, particularly those near the border of the MS, are regularly

visited, it does happen that cells nearer to the centre are visited with less frequency or, in some cases, not at all. However, when the BT is applied, the homing task breaks the MAVs out of their normal search behaviour by drawing the MAV back towards the centre of the MS. This allows the MAV to (potentially) visit cells that were not observed when no fuel constraint was applied. Figure 4-13 shows an example of the flight paths for six MAVs, first for the case when the BT is not applied followed by the case when it is. In Figure 4-13a there are clear open areas that no MAV passes through. Even with the range of the MAVs sensor footprint there are cells in the open spaces that will not be visited. After the BT is applied in Figure 4-13b, the MS is clearly covered more effectively by the MAVs.

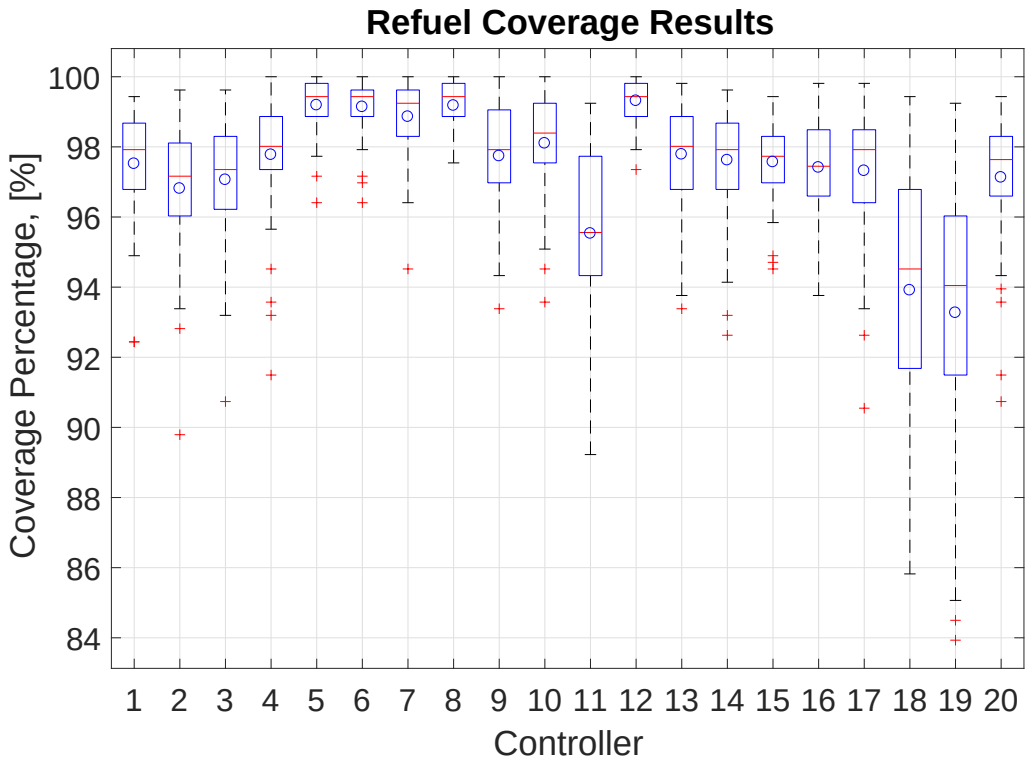
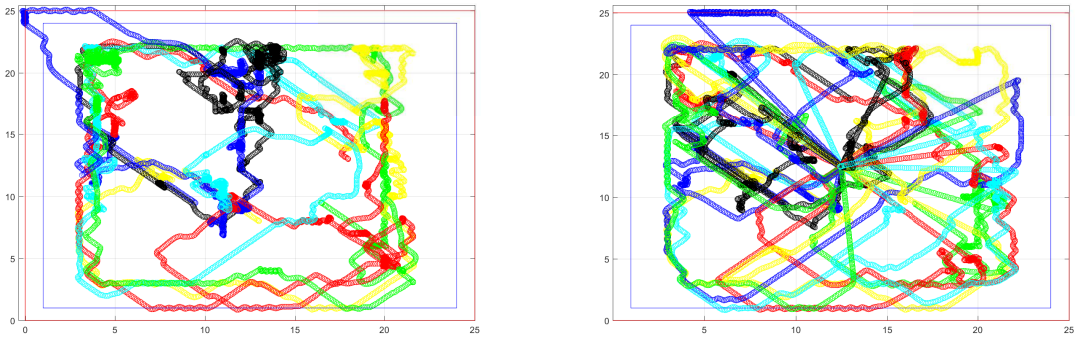


Figure 4-12: Coverage levels with refuelling

For the average age results, there is a significant drop in the cell age as the MAVs return for their first refuelling cycle. Hence, in order to gain a clearer understanding of the average cell age over the duration of the simulation, the total simulation time was doubled from 300s to 600s when testing this metric. This allowed the system to settle after the shock of the initial refuel cycle. As with the coverage levels, the average cell age also experienced a slight improvement when applying the fuel constraints. The average cell age increased from 47.03 to 52.39s and the standard deviation more than halved to $\sigma = 2.32$. The increase in performance can be attributed to the homing action periodically drawing the MAVs towards the centre.

As expected from the previous analysis on the avoidance measures, it can be seen in Figure 4-15 that there is a clear reduction in the number of crashes when using the BT. What is interesting to see is that in this case, the magnitude of the outlier results seems is smaller than the case where only the additional avoidance measures were used. However, if one considers how the avoidance measures fail then these results do make sense. Previously it



(a) Without BT **(b)** With BT

Figure 4-13: 6 MAV flight path examples

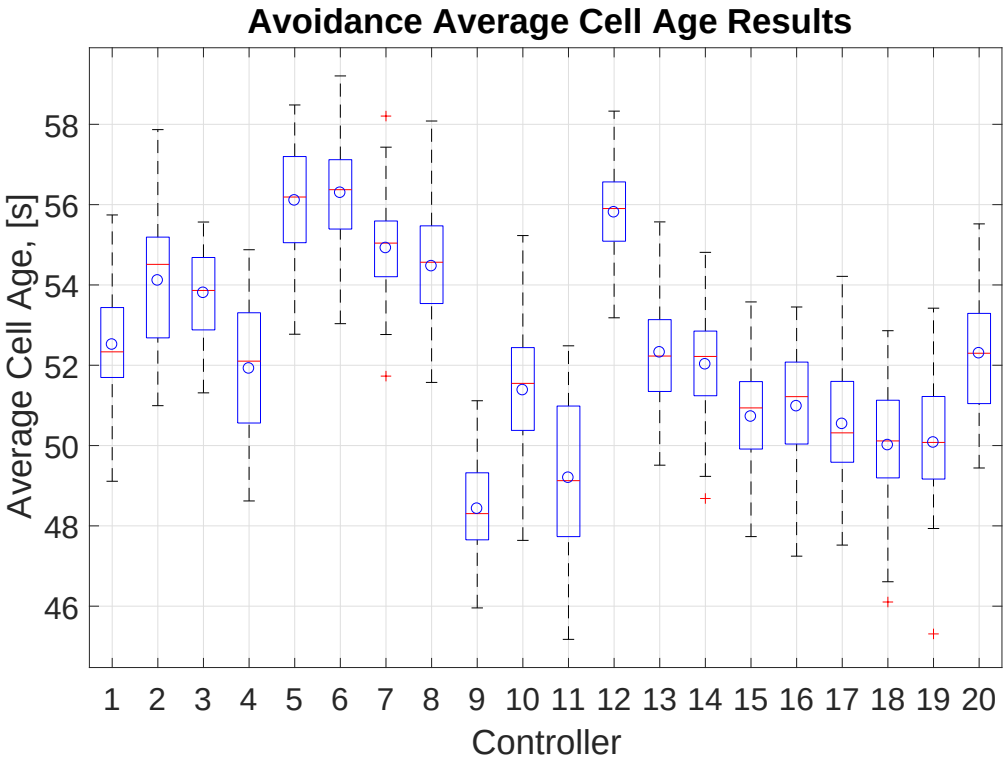


Figure 4-14: Average cell ages with refuelling

was mentioned that in some instances the avoidance algorithm causes two or more MAVs to have the same (new) heading when attempting an avoidance maneuver. As such, the MAV are never able to separate from one another, causing the high crash percentage. Now, with the fuel constraint activated, it gives the MAVs in this situation a chance to move further away from one another when one returns for refuelling. This is one possible explanation for the reduction in the outlier results.

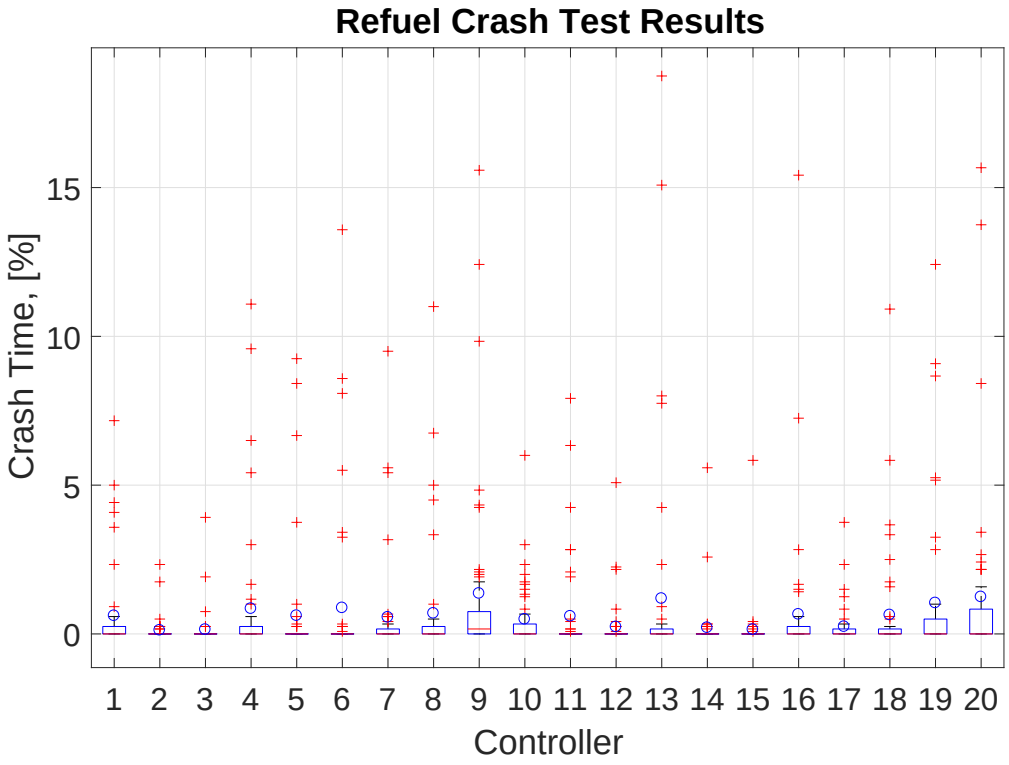


Figure 4-15: Number of crash instances with refuelling

Comparison to Random Walk

Figures 4-16 and 4-17 illustrates how the coverage and average cell age evolve over time. For these figures, the simulation time was increased from $300s$ to $600s$ to shown the evolution over time more clearly. Included in these figures is the performance of the system if the random walk baseline approach was used instead of the evolved NN. The coverage graph is exactly as would be expected from previous baseline and robustness comparisons. The coverage rises sharply in the beginning after which it slowly approaches 100%. Again it can be seen that the NN offers slightly improved performance over the random walk.

By closely examining this plot at $100s$, this is the time at which the first MAV returns for refuelling, a slight change in the plots behaviour can be observed. At this point there is a faster than expected rise in the coverage levels (compare to that shown previously in Figure 3-29) as the MAV flies towards the depot. This trend continues as the other MAVs return to the depot at $132s$. It is only after all the MAVs have returned to the depot for the first time, at roughly $t = 145s$, that the behaviour changes. Now, the coverage levels return to their more gradual increase as the MAVs once again rely on the NN outputs to fly through the MS.

The more interesting case is that of the average cell age shown in Figure 4-17. As was the case in the plots of the coverage levels, there is a clear change in the behaviour of these plots as the first MAV returns to the depot. This is particularly noticeable for the graph of the best performing NN. Again, this is most likely a result of the homing task breaking the MAV out of it normal pattern, which in turn allowed it to visit a number of cells that were previously unvisited. Next, the most noticeable difference to the original comparison

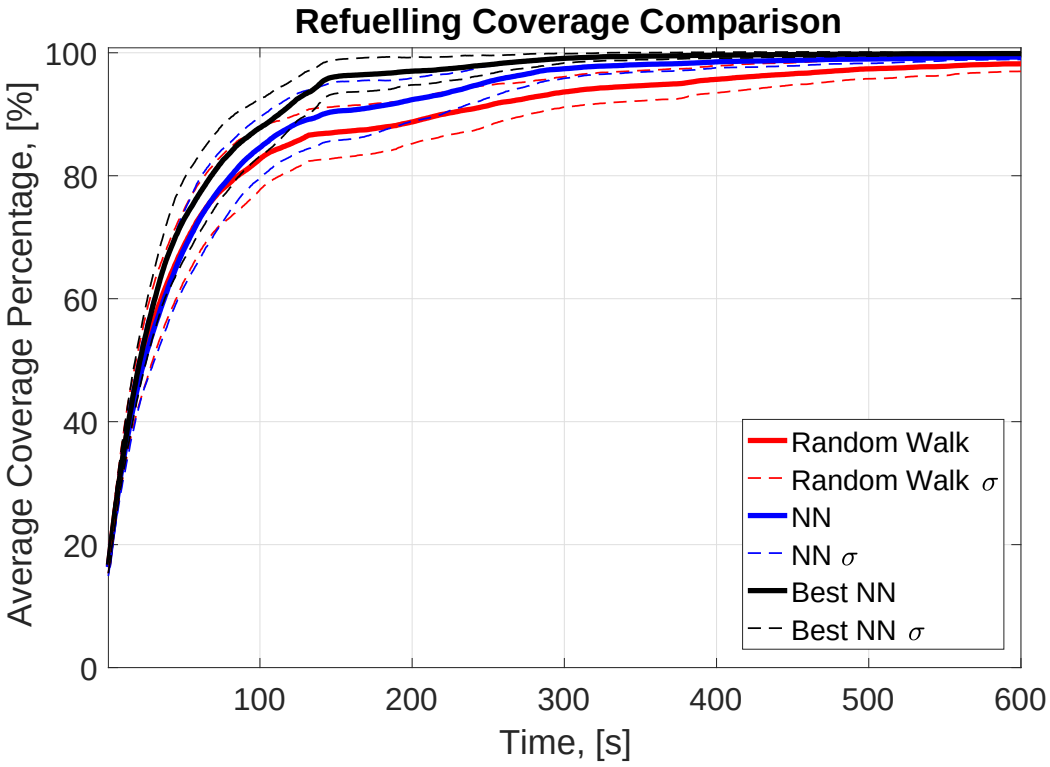


Figure 4-16: Coverage levels with refuelling

plot (recall Figure 3-28) is the large decrease in the cell age as the rest of the MAVs return to the depot at wait for refuelling. Luckily as more MAVs are refuelled this quickly returns to its original levels. In the next refuelling cycle, indicated by the slight decrease in cell age at roughly $t = 300s$, the impact of the refuelling is greatly reduced as the MAVs have now staggered their refuel times reducing the time spent waiting at the depot.

It is also interesting to note that in this figure, the results of the random walk and the NN controllers are far more similar, with the best NN even outperforming the random walk. This was not the case in the original baseline comparison. While this homing aspect has a positive benefit to the NN which normally tries to move away from the centre, this negatively impacts the random walk controller. For this controller, MAVs would often pass through/near the centre of the MS during its normal operation. As a result of this, the system does not benefit from the homing task periodically drawing the MAVs towards the centre as it does when the NN controllers are used. In addition, as the centre of the MS becomes more crowded, the MAVs must naturally spend more time avoiding one another rather than exploring. This leads to further decreases in the performance. One would expect this to have a greater impact on the random walk controllers as the NNs already account for collision avoidance in their decision process. Further, with the NNs tendency to direct the MAVs towards the edges, MAVs should experience less congestion in their airspace.

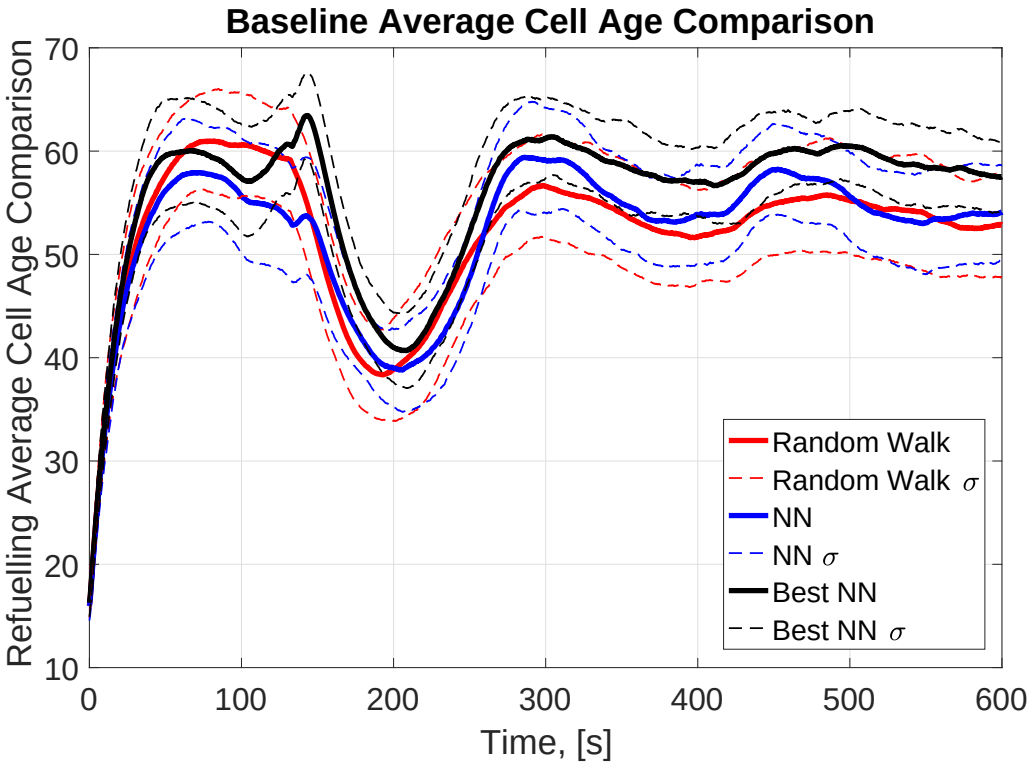


Figure 4-17: Average cell ages with refuelling

4-4-3 Performance with Real World Fuel Constraints

Finally, it was decided to include results from a simulation with a more realistic time horizon. For this test, four MAVs were used to perform the persistent surveillance task in a $25m \times 25m$ MS. Further, it was assumed that each MAV had a total flight time of $480s$ and the relationship between the battery levels and the total flight time are shown in Table 4-3. To account for the longer flight time of the MAVs, the tests were run over a time horizon of 24 minutes and the results are shown in Figures 4-18 to 4-20.

Table 4-4: Real-world refuelling parameter values

Parameter	Time Remaining, [s]	Bat. Voltage, [V]	% Fuel Remaining
$E_{m,max}$	480	12.4	100
E_{thresh}	192	10.7	40
E_{critic}	72	10	15
E_{min}	0	9.6	0

Again, these results show that the designed BT can be successfully implemented on a system working with the actual fuel level constraint from an actual flight platform.

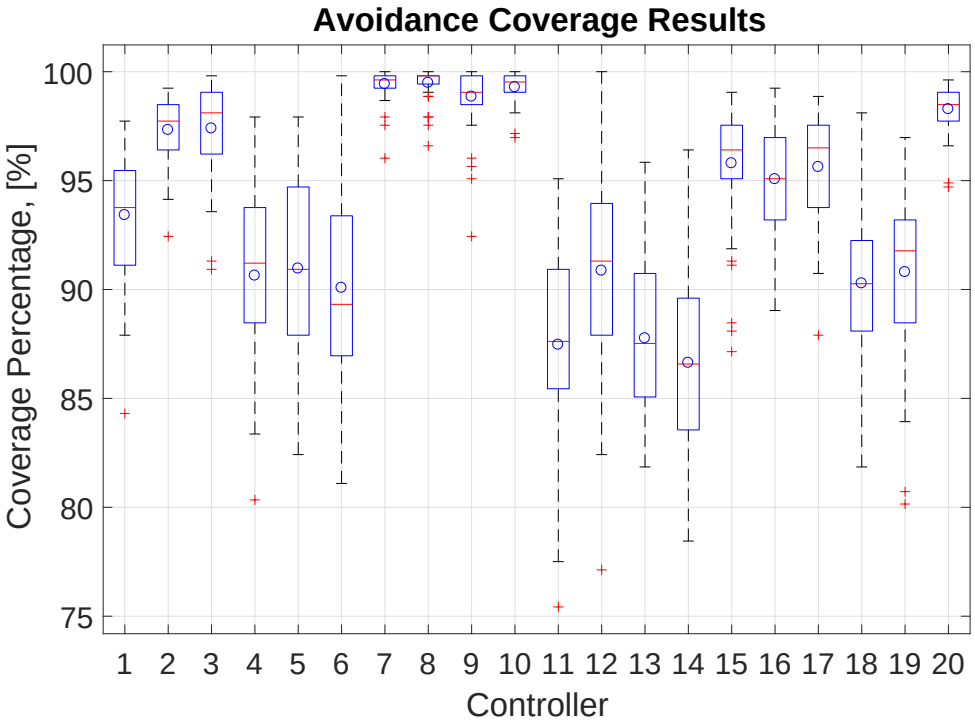


Figure 4-18: 4 MAV refuelling flight results: coverage percentage

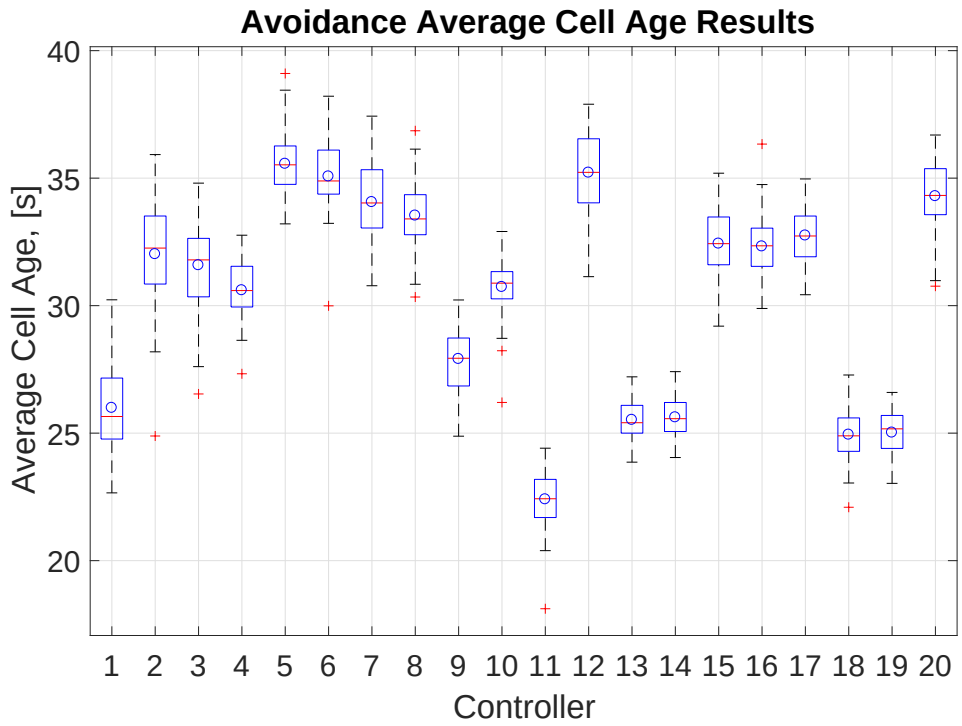


Figure 4-19: 4 MAV refuelling flight results: average cell age

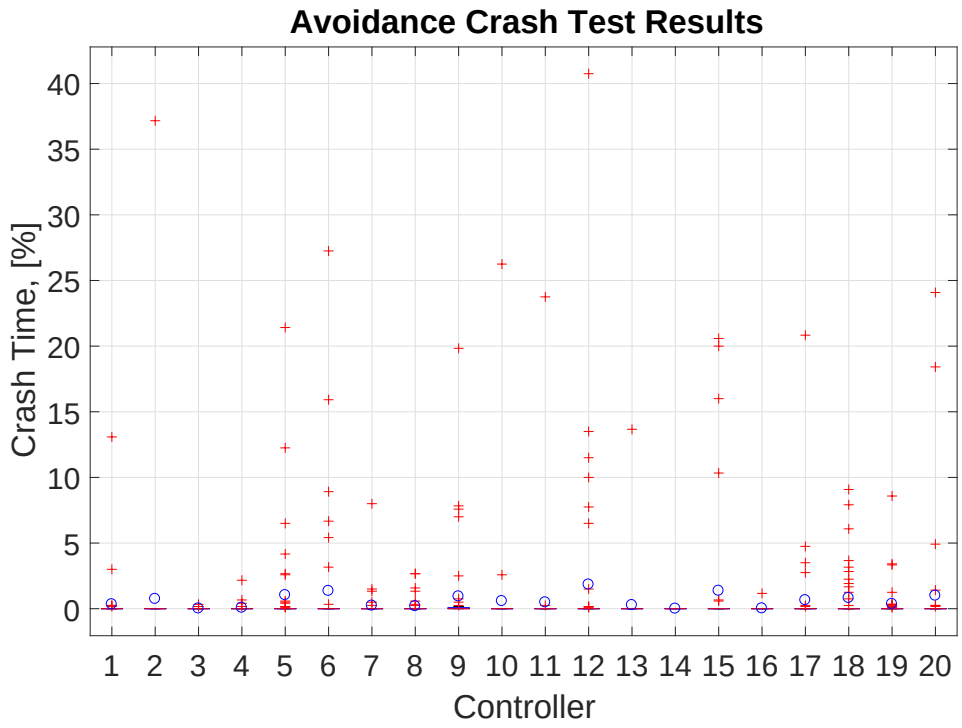


Figure 4-20: 4 MAV refuelling flight results: crash percentage

4-5 Summary

This section focussed on how the real-world fuel level constraint was taken into account in this thesis. A BT was chosen to implement the high level control needed over a finite state machine due to its simplicity and clear readability. The persistent surveillance mission was broken up into 5 tasks (or states) and the BT was responsible selecting the appropriate task for each MAV. Theses were; Avoidance, Homing to depot, the Charging task, Waiting at the depot and surveillance of the MS.

The avoidance measures implemented in this thesis is based off of the strategy employed by Szabó in [59]. It makes use of the principle that if two MAVs are on a collision course, the MAVs will not collide it at least one of them reverses its current flight direction. One small adaption was made in this work, instead of rotating the MAVs about their yaw axis after reversing their flight direction, the reversed flight direction was adjusted by angle α. This was to account for the fact that the NN assumes that the MAVs body coordinate frame matches the global coordinate frame. This method was able to greatly increase the number of runs where no collision between the MAVs occurred while it had a negligible impact on the coverage levels and average cell age.

With the BT applied, the MAVs were able to uphold the constraint placed on their fuel levels with no MAVs failing during any of the simulations. Interestingly, including the fuel level constraints had the opposite effect on the systems performance than originally expected. It was initially assumed that by including refuelling, both the coverage and cell age metrics would be noticeably reduced as was the case with the robustness tests. However, that was not what

was observed during the tests. These showed that both metrics experienced improvements. This was attributed to the flight paths created by the homing behaviour. During normal operation, the NN controllers tended to command the MAVs to circle near the border of the MS. By including the homing behaviour, the MAVs would move towards the centre of the MS far more frequently allowing them to survey the centre cells more frequently and to visit cells that might have remained unobserved.

Now that the separate components to the system have been tested and verified in simulation, they can now be implemented in the physical system. This process and the results will be explained in the next chapter.

Chapter 5

Practical Implementation

Now that the solution has been verified in simulation, it is time to move on to the practical implementation.

5-1 UWB Positioning Results

Unfortunately, the MAVs could not fly while using the position estimates obtained using the NLLS. Figure 5-1 shows the estimated position of the MAV while it is held stationary on the ground. In the figure, it can be seen that even with the moving average filter there are still large variations in the estimated position. This is a problem for the autopilots internal positional controller which attempts to correct for what it sees as the MAV drifting off course. This causes the MAV to fly erratically around the area.

In an attempt to further reduce the noise in the parameter estimation, a Kalman filter was added on the estimated position. First, the measurement noise was estimated by collecting ranging data for the case where the MAV was held stationary on the ground. The Matlab *cov* command was then used to determine the associated co-variance of the data. This was computed as:

$$R_{Kalman} = 10^{-3} \begin{bmatrix} 0.618 & 0.346 \\ 0.346 & 0.572 \end{bmatrix} \tag{5-1}$$

As is typical for physical systems, the process noise covariance matrix, Q_{Kalman}, is used as a tuning parameter, while the cross-covariance matrix, S_{Kalman}, was assumed to be $\mathbf{0}$. The Q_{Kalman} was initially assumed to be $Q_{Kalman} = 10^{-3}\mathbf{I}$. While this did allow very fast convergence between the estimate and the NLLS position, its reliance on the noisy NLLS solutions was still too great. As such, the Q_{Kalman} was decreased slightly to increase its reliance on

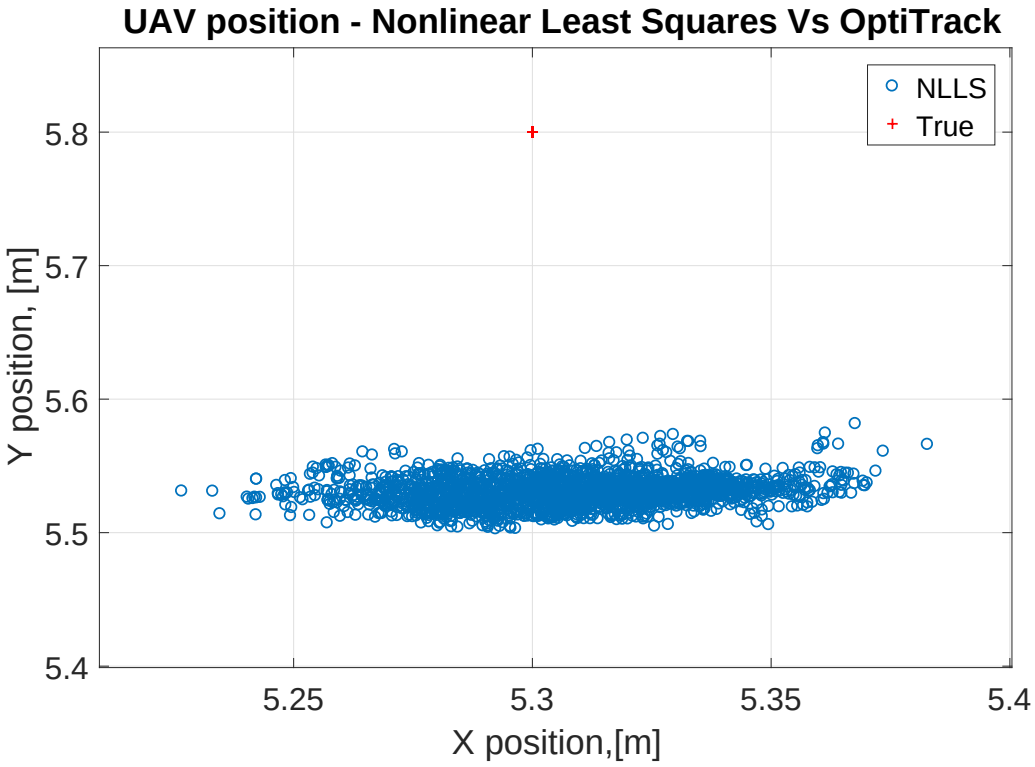


Figure 5-1: UWB position noise example

the modelled position. The final Q_{Kalman} used in the system is shown in Equation (5-2).

$$Q_{Kalman} = 10^{-6} \begin{bmatrix} 0.001 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.001 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.001 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.001 & 0 & 0 \\ 0 & 0 & 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 0 & 0 & 10 \end{bmatrix} \tag{5-2}$$

This along with the R_{Kalman} shown previously was used to determine the Kalman gain, K_{Kalman}. Through the use of the Matlab *dare* command, this was determined as:

$$K_{Kalman} = \begin{bmatrix} 0 & 0 \\ 0.0002 & -0.0001 \\ 0 & 0 \\ -0.0001 & 0.002 \\ 0.0172 & 0.0053 \\ 0 - 0.0053 & 0.0180 \end{bmatrix} \tag{5-3}$$

For the case where the MAV was stationary the results shown in Figure 5-2 were obtained. From this it can be seen that the noise on the estimated position has been noticeably reduced.

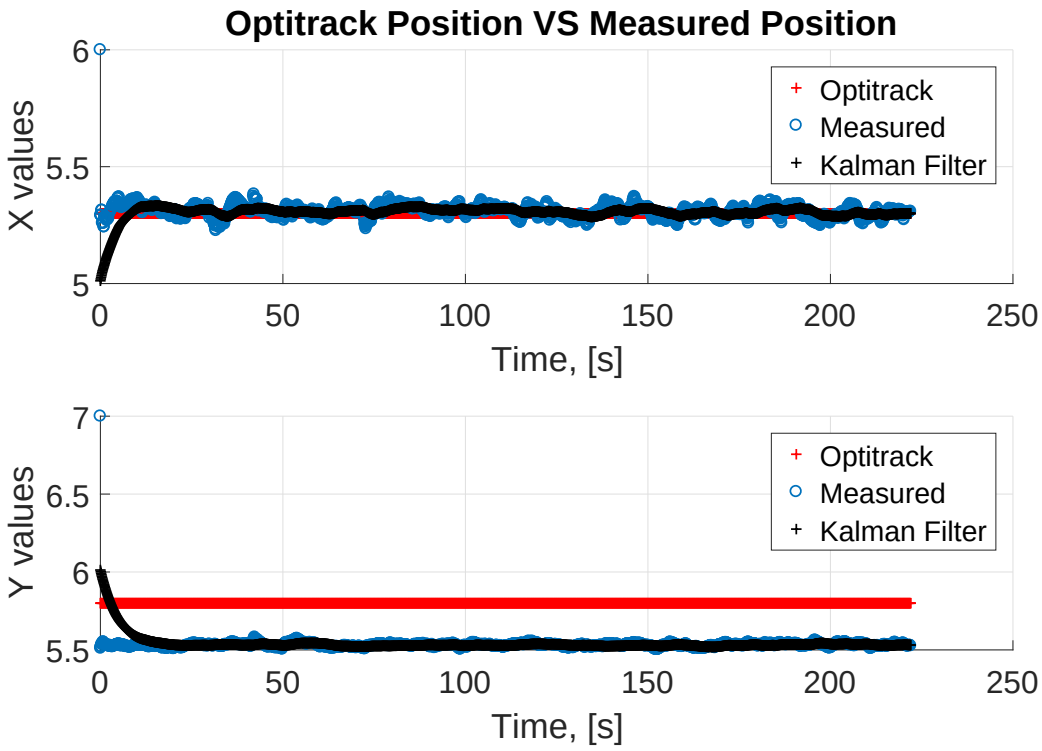


Figure 5-2: Kalman filter performance: Stationary MAV

Next its performance must be analysed while the MAV is flying through the CyberZoo. For this test, the MAV is controlled using the OptiTrack position while ranges and the positional outputs from the Kalman filter are recorded. This is shown in Figure 5-3. Initially, it seems to be working very well but at roughly $t = 35s$ the second major issue with the positioning system was discovered. Occasionally one of the anchors would suddenly stop performing the ranging procedure with the tag or it will start obtaining unrealistic values such as $-408365m$.

At first it was assumed that this was a hardware issue as it always seemed to affect the first anchor. However, when the module was replaced the error persisted. Further, when the original Arduino code from [68] was flashed to the modules, the error did not occur. From this it was concluded that there is a bug in the software that was introduced when the adaptions were made to the code. In the end the bug could not be found and corrected in time for the UWB positioning system to be used for the NN tests.

Once this bug has been fixed, actual flight tests can be performed where the positional information used by the autopilot is supplied by the UWB positioning system. The first test must check whether or not the Kalman filter reduces the noise sufficiently to allow the MAV to hover in place. If the data is still too noisy for the internal positional controller then the Kalman filter could be further tuned to rely more heavily on the models position. Alternatively, the positional control could be bypassed altogether. There are two methods of accomplishing this. The first, and probably best, way would be to search through the Paparazzi code to locate where autopilot calls the internal controller and to disable the controller there. A simpler

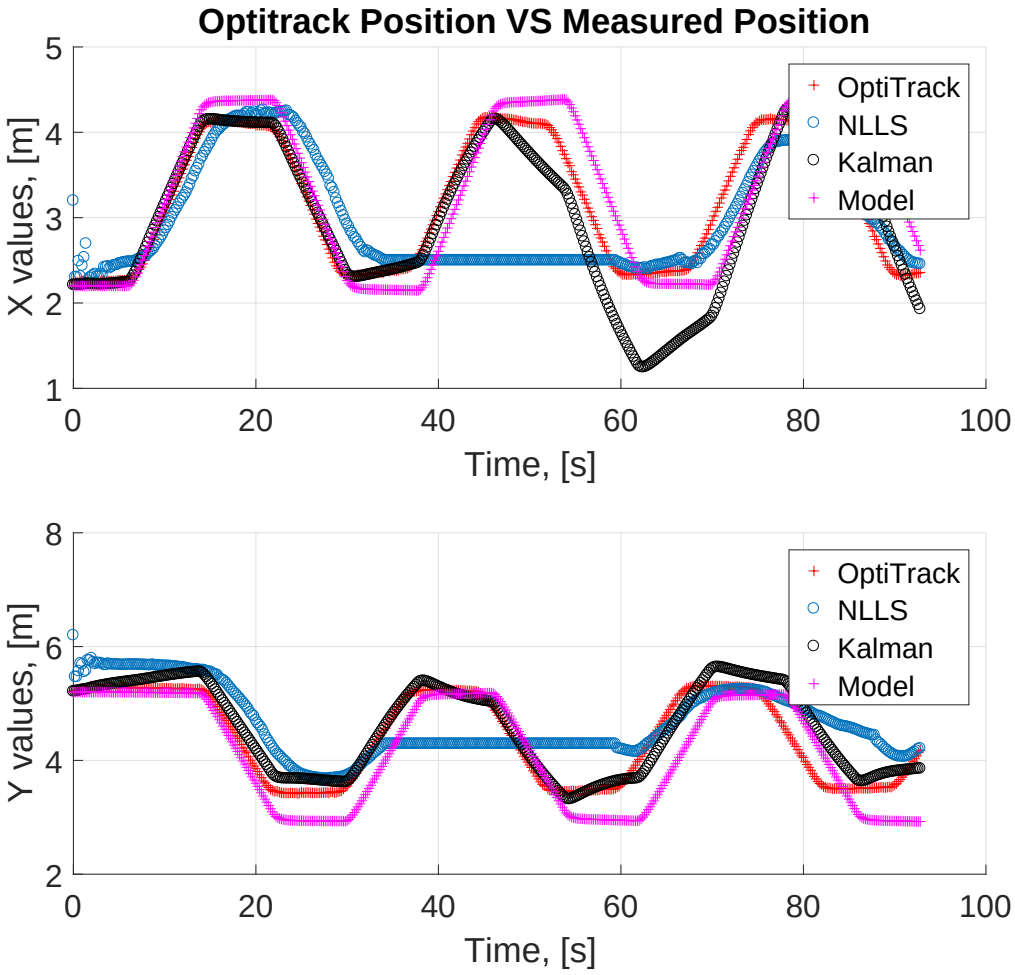


Figure 5-3: Kalman filter performance: Mobile MAV

method would be to store the calculated positions as a global variable instead of initialising the UWB system as a replacement for the GPS. In this case, the autopilot can be told to ignore the GPS signal and the modules (in this work it would be the NN and UWB communication modules) will need to call on the global variable as needed. This remains a task for future work.

5-2 High Fidelity Simulation

Before the NN controllers were used on the actual hardware, a high fidelity simulation was run. This was done to give a safe environment for debugging the Paparazzi implementation to ensure that no unexpected behaviour would cause damage to the hardware during flight. For this, a Robot Operating System (ROS) model developed in [85] was used. This made use of the *Gazebo* physics engine and the *Hector-quadrotor* model, which together provide a

validated platform [85].[1]

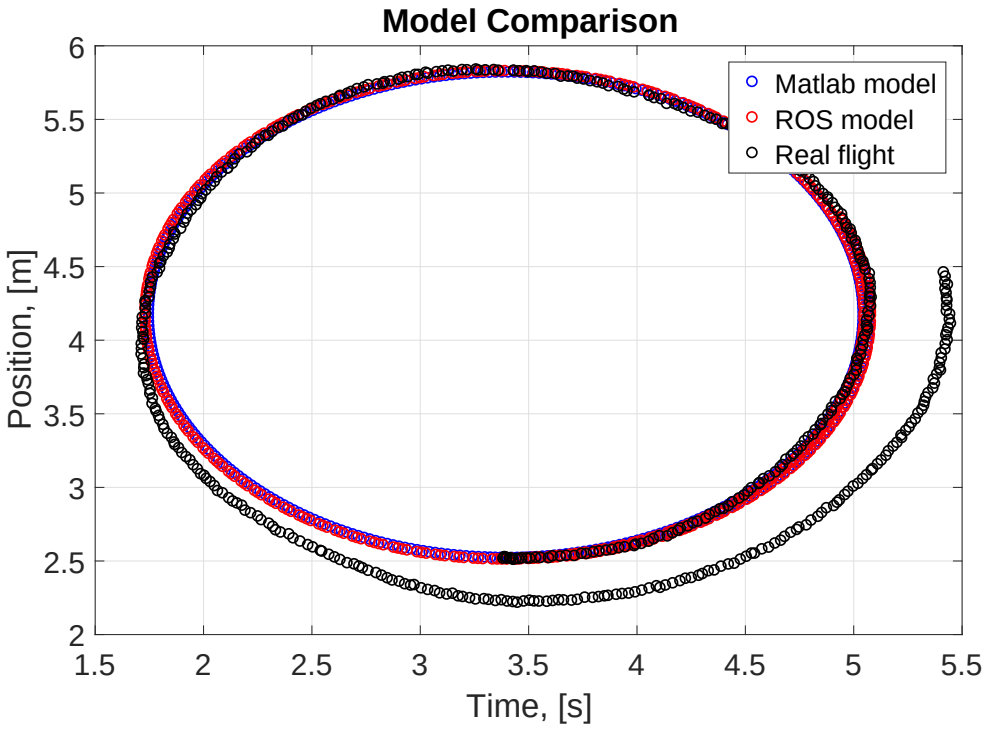


Figure 5-4: Model comparison for a circular flight pattern

Figures 5-4 and 5-5 shows a comparison between the Matlab model used, the ROS model and the actual flight results for two different velocity input sequences. The first figure shows the results for a circular flight path which shows that the two models have very similar characteristics for gradual changes in the commanded velocity. In Figure 5-5 the differences between the models becomes clearer. The Matlab model struggles to represent the dynamics of the more aggressive turning manoeuvres, resulting in the slightly rounded corners of the square pattern. The ROS model on the other hand, is able to account for sudden changes in velocity allowing it to correctly represent the sharp turns.

With the high degree of similarity between the Matlab and the ROS models, one would expect similar outputs from the NN controllers. This can be seen in Figures 5-6 and 5-7 which shows the x and y positions of two MAVs in a $7m \times 7m$ MS. While the resulting flight paths of the two MAVs are not identical, they resemble on another closely enough that they produce similar behaviours in terms of avoidance and area coverage. One final note about the differences between the models is that the Matlab model does not take the aerodynamic effects of the propellers into account. This will not affect the accuracy of the model while flying with a single MAV or when there is a large enough distance between the MAVs. However, it was observed that in the ROS simulations, if two or more MAVs approached too close to one another, the propeller effects would often prevent the MAVs from separating, leading to a slightly higher chance of collisions.

[1] ROS, Gazebo and the Hector-quadrotor package are freely available at: www.ros.org/, www.gazebosim.org and wiki.ros.org/hector_quadrotor respectively.

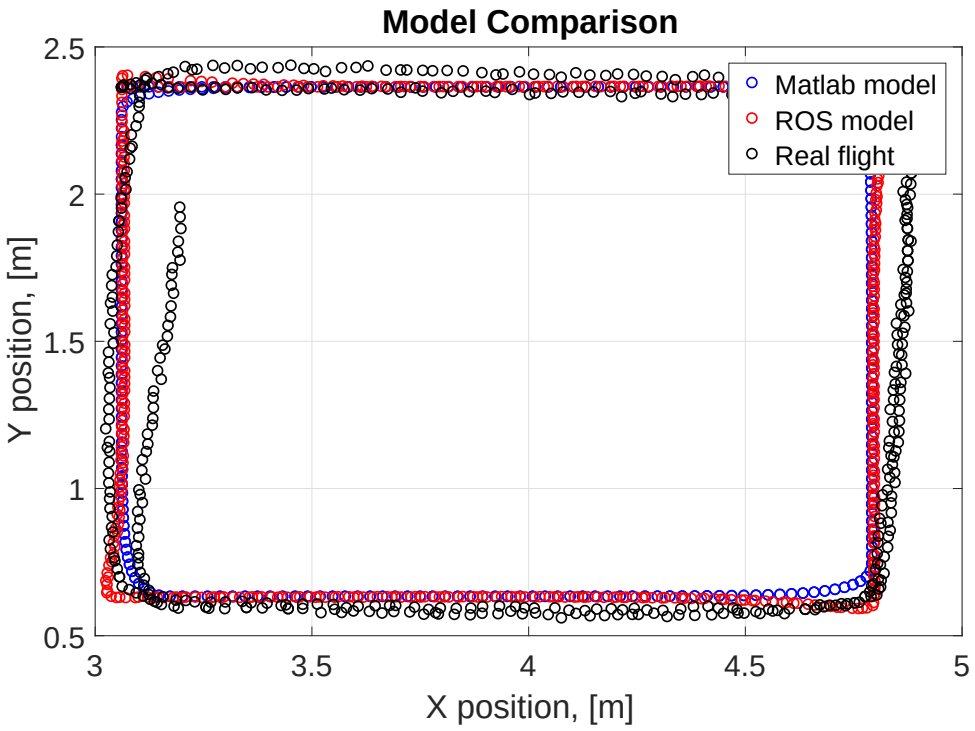


Figure 5-5: Model comparison for square flight pattern

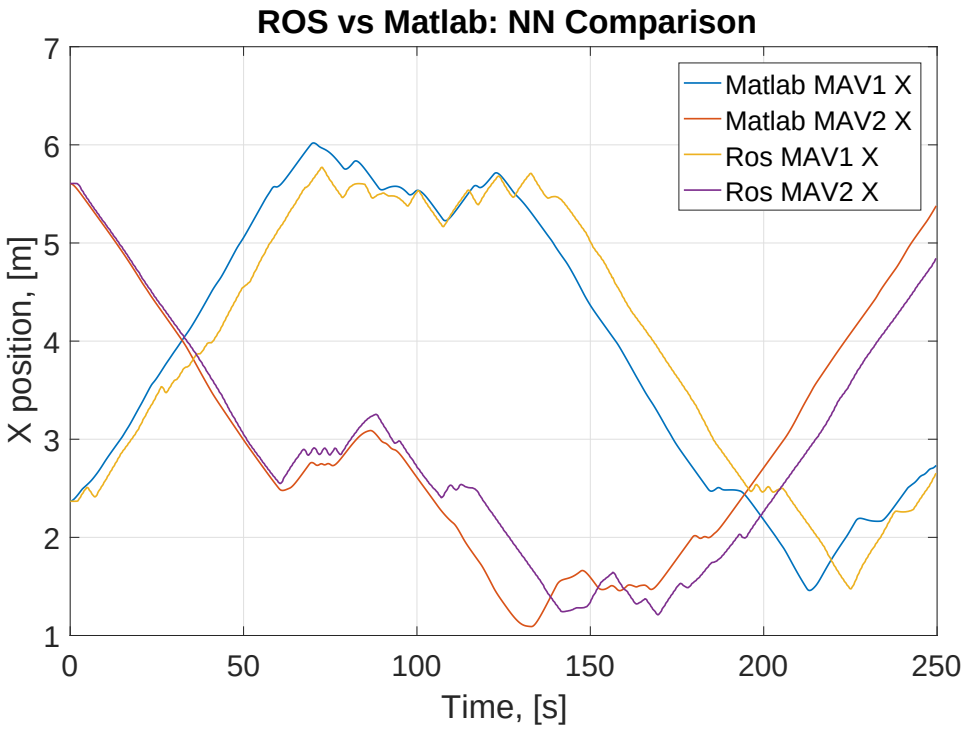


Figure 5-6: Model comparison for a circular flight pattern

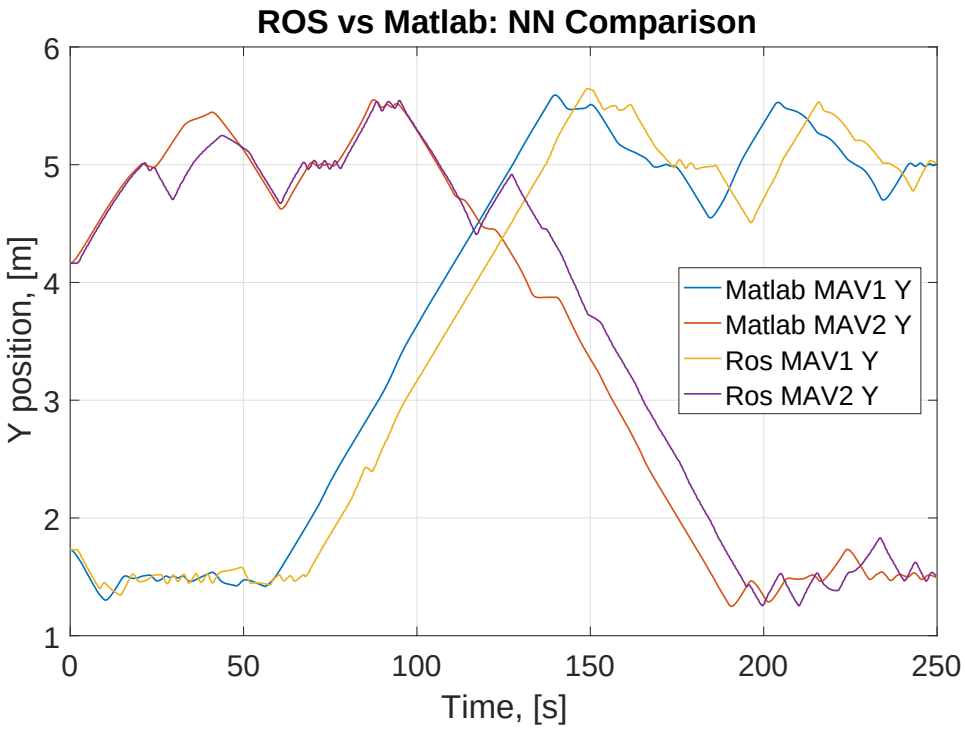


Figure 5-7: Model comparison for square flight pattern

5-3 NN Results

After proving that the NN code worked on the ROS model, it was time to test the controllers on the physical flight platform. As mentioned earlier, the UWB positioning could not be made to work correctly, as such, the evolved NN controllers were tested with the positional information supplied by the OptiTrack system. In these tests, the high level control provided by the BT was not used (i.e. no refuelling was taken into account and the avoidance behaviours relied solely on the NN controllers).

For the first round of tests, the controllers generated in Section 3-3-3 for the simulated CyberZoo were implemented in the Paparazzi autopilot software. As in the simulated tests, the maximum velocity was limited to $0.3m/s$, the range of the virtual antennae was set to $3.5m$ and the NN function (and cell ageing) was called at a rate of $4Hz$ (i.e. $\Delta T = 0.25s$). If needed, this update frequency can be increased as the on-board runtime of the NN code is $0.1ms$. However, it is important to remember that the UWB modules broadcast their positions at a rate of $10.74Hz$ (recall Table 2-1) and, as such, this places an upper bound on the update frequency of the NN controller. When using the UWB to provide positional information, this bound is further reduced to $5.3Hz$ to account for the slower update rate of the ranging estimates.

The average cell age and the coverage level results for the tests can be seen in Figures 5-8 and 5-9 respectively. From these figures, it can be seen that the NN controller has very similar performance to that of the simulations. This was to be expected as the OptiTrack system can provide exact positional information so there is. In addition, the CyberZoo is an enclosed area

which prevents disturbance inputs, such as wind, from affecting the MAVs. Unfortunately longer tests were not possible as a result of the battery packs used. In nearly all the tests, the batteries voltage would drop too low within 90s after take-off and the MAVs would be forced to land.

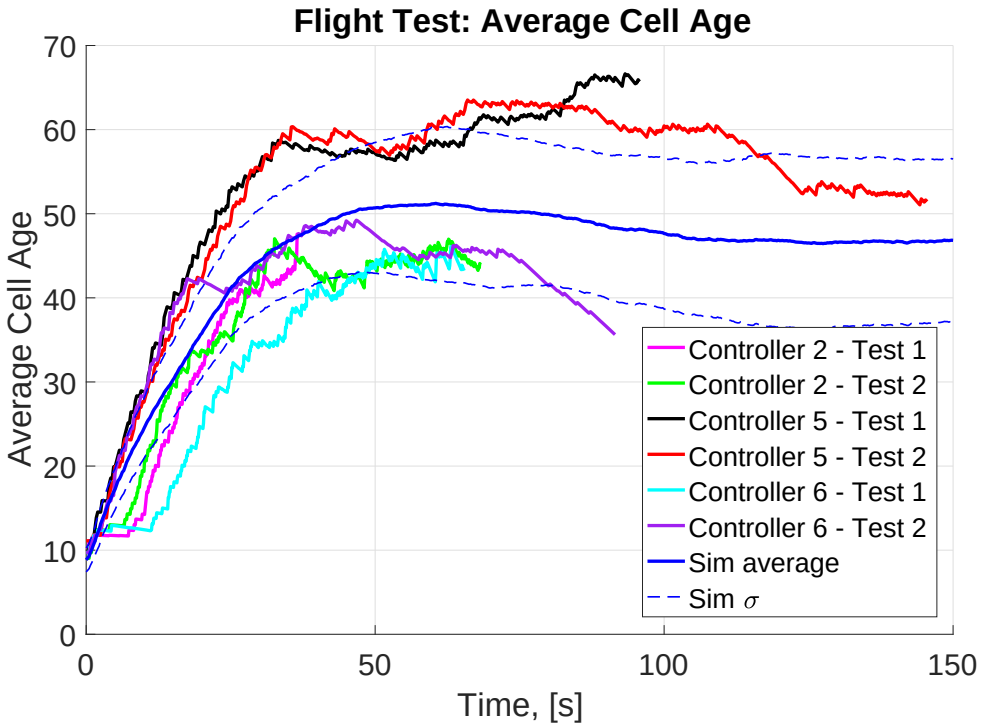


Figure 5-8: Physical Flight Results: Average Age

As was the case with two MAVs, similar performance between the actual and simulated results can be seen when 3 using MAVs. The results comparing the actual flights to the simulated flights performance are shown in Figures 5-10 and 5-11. For these tests only controller 5 was considered as this showed the best performance in both the ROS models and in the flight tests for two MAVs. Again, the limited battery life was a problem during testing.

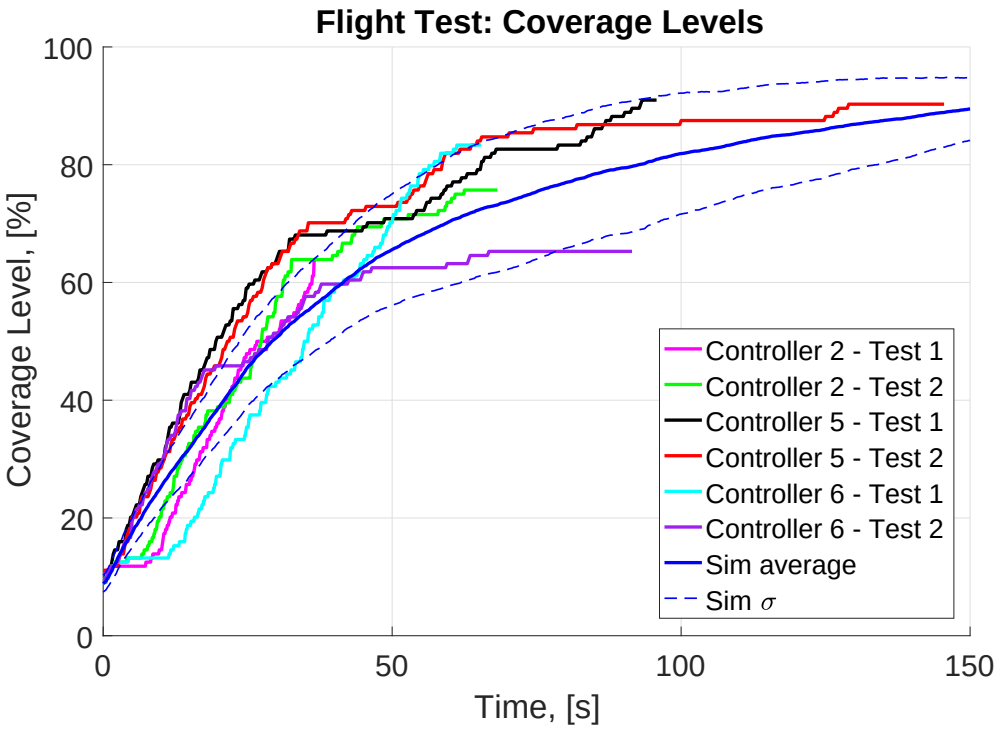


Figure 5-9: Physical Flight Results: Coverage Percentage

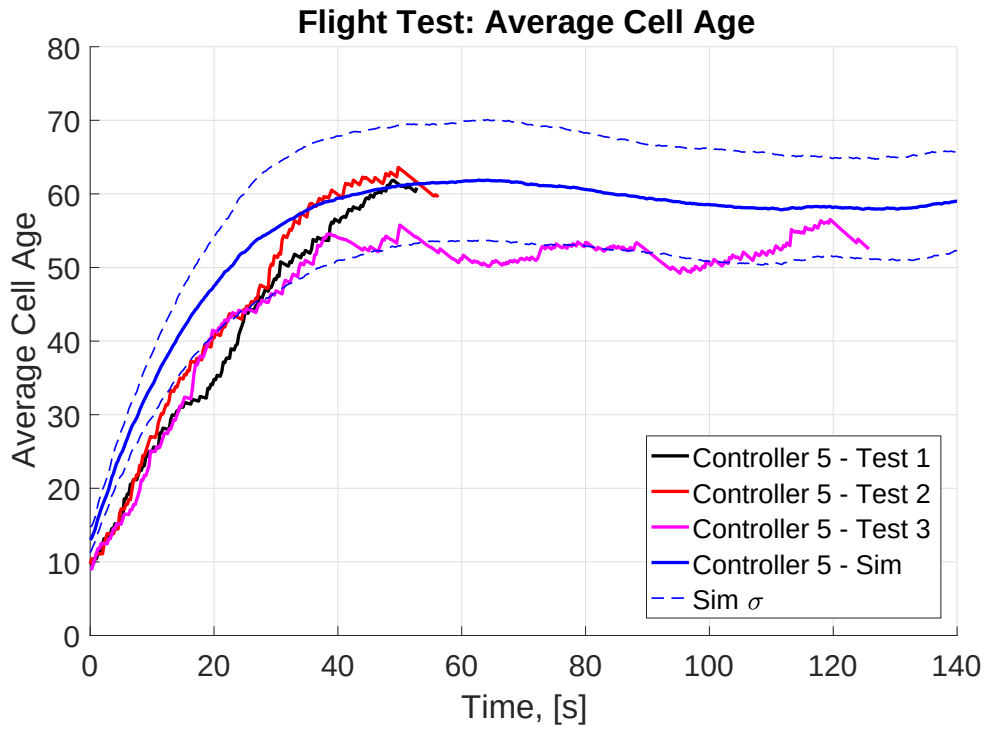


Figure 5-10: 3 MAV flight Results: Average Age

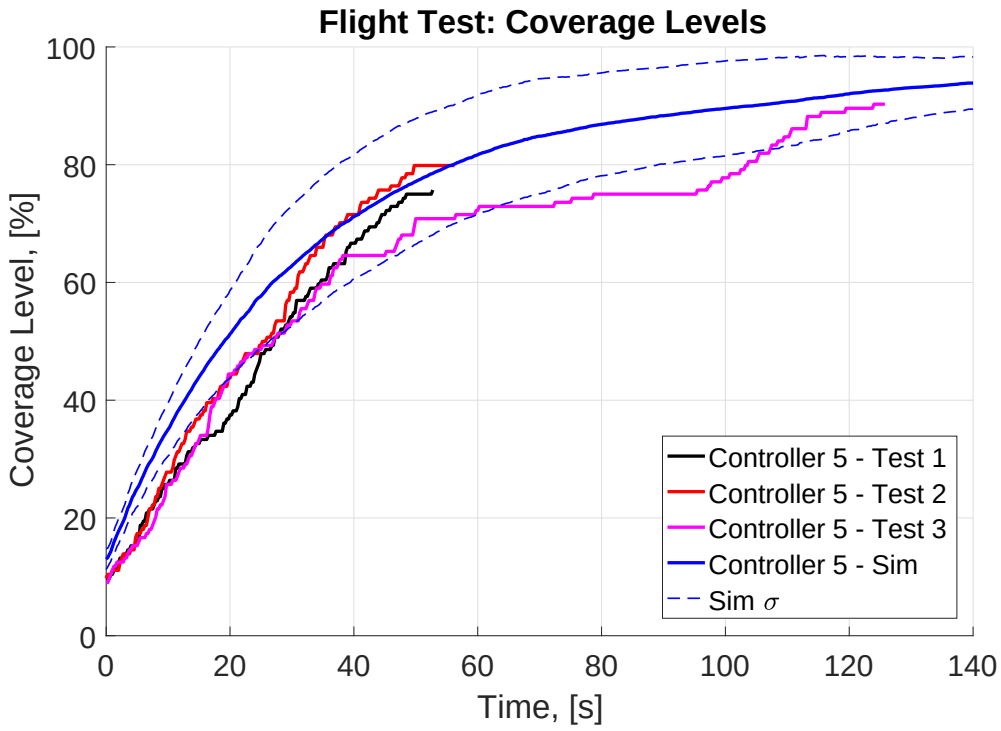


Figure 5-11: 3 MAV flight Results: Coverage Percentage

5-4 BT Results

As mentioned in the previous chapter, the threshold values used in the physical implementation of the refuelling BT were based on the low battery voltage warnings displayed by Paparazzi. This assumes a maximum supply voltage of $12.4V$ while the first warning is activated at $10.5V$ and the second at $9.8V$. Ideally the MAV should already be at (or near) the depot by the time the second warning is activated. As such the $E_{thresh} = 10.7V$ and $E_{critic} = 10V$.

This was implemented in the autopilot software, and an example of the resultant flight paths for two MAVs is shown in Figure 5-12. Again the batteries were a major hurdle when performing the tests. In most attempts, the battery supply voltage would drop below the first threshold value immediately after take-off. As a result, the MAV would often never leave the depot after its battery was replaced. This can be seen occurring with MAV 2s second flight.

In Figure 5-13 the coverage and average cell ages from the flight in Figure 5-12 are given. However, with the limited battery life causing difficulties with the testing, it is difficult to draw any meaningful conclusions from only these results. As in the simulated tests, the average cell age experienced a sharp drop at roughly $60s$ as the second MAV also started to return to the depot. Then, as the first MAV returns to its surveillance task the coverage and average age start to slowly increase once again. Unfortunately multiple refuel cycles were not possible so it could not be verified if the impact of future refuelling trips is reduced by the BT as it was in simulation.

With the physical implementation, there could be a significant factor that will affect the

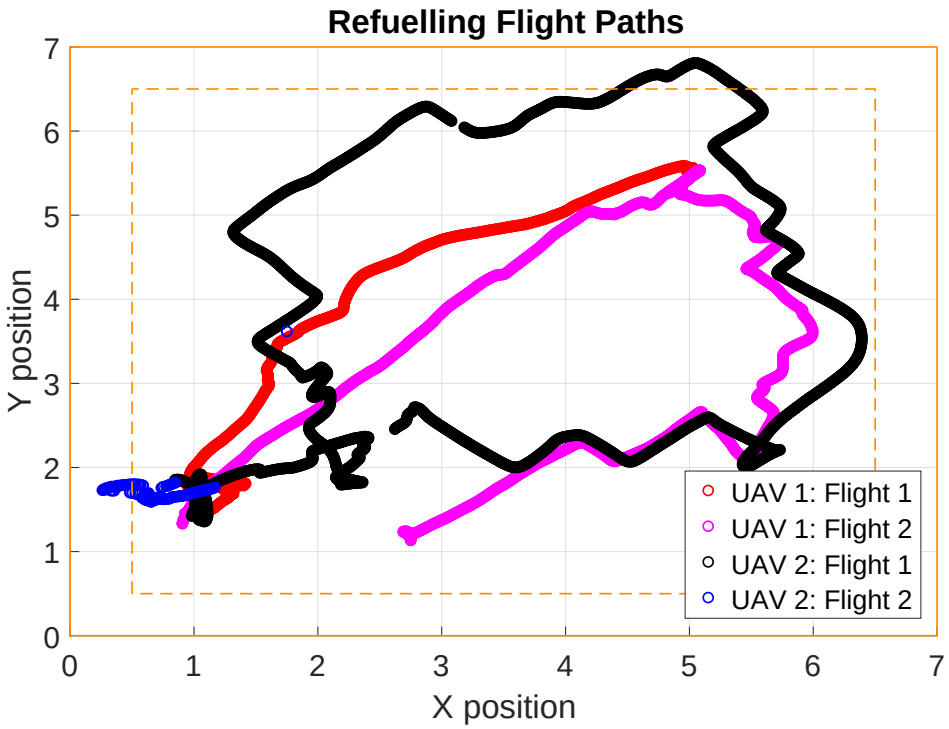
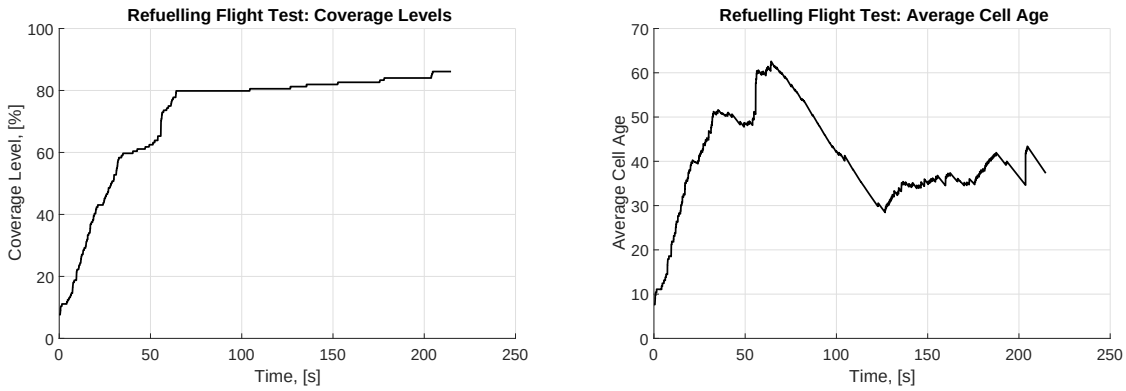


Figure 5-12: 2 MAV refuelling flight paths



(a) Refuelling physical implementation: Coverage **(b)** Refuelling physical implementation: Average cell age

Figure 5-13: Individual controller comparisons

systems performance which was not taken into account during the simulations. During the actual tests, the MAVs must be restarted as a result of swapping the batteries. While the Parrot website does state that the batteries are hot-swappable it was neglected to test whether the age map stored on each MAV was reset after the battery was replaced. This must be verified in future and the simulations should be adapted to include this factor if necessary.

5-5 Summary

The UWB positioning and communication system was partially successful. The inter-MAV communication of the current positions and the status of the depot were reliably transmitted by the UWB module. In addition, the NLLS position estimation, a Kalman filter on the position and an UWB GPS structure were programmed in C and verified for the Paparazzi autopilot software. The main issue with this positioning system was in the adaptions made to the Arduino code that controlled the UWB modules. There is a bug in the code that periodically causes one of the modules to stop performing the ranging procedure. This made the system too unreliable for the physical tests.

The NN controllers were successfully implemented and tested on the actual flight platform. During the tests comparable results in terms of the performance metrics were recorded. In addition, the MAVs exhibited the same explorative behaviours as those recorded in the simulated tests. This was to be expected as the OptiTrack system used to determine the position of the MAVs offers a high degree of accuracy which eliminates the noise being input into the NN. Further, the enclosed are of the CyberZoo limits the disturbance forces that act on the MAVs.

The BT designed in the previous chapter was successfully implemented on the physical system. However, further testing will be needed to thoroughly verify that the performance on the actual setup is comparable to that of the simulations.

The main issue faced when performing the physical tests was with the batteries used to power the MAVs. The short flight times made it difficult to draw meaningful conclusions about the overall performance of the controllers.

Chapter 6

Conclusions

In this final chapter, the work presented in this thesis and the main results will be summarised. This is followed by an outline of the potential aspects for future work.

6-1 Summary

The main goal of this thesis was to design and implement a controller that could be used to perform a multi-agent persistent surveillance task. The solution could not rely on a centralised controller and had to account for the limitations placed on the system by the flight platforms fuel constraints. It was decided to use the NEAT algorithm to evolve NN controllers that were responsible for solving the persistent surveillance task while a BT was used to account for the limited fuel level. The NN controller and BT could then be implemented on a swarm of MAVs which would then be used to constantly fly throughout a greenhouse, collecting environmental data as they go for use in PA. Further, as this system is originally intended to operate in indoor environments, an inexpensive and reliable alternative to GPS is needed to supply the MAVs with their positional data. This thesis investigated the use of an OptiTrack motion capture system and an UWB localisation system that gave the position relative to a set of fixed anchors.

In Chapter 2 the chapter started by giving a detailed explanation of the problem, and its simplifying assumptions, that was solved during this thesis. It concluded with a description of the physical hardware that was used during the testing phase of this work. Here, the UWB positioning system was introduced and its performance was further described in Section 5-1. For this system a series of four stationary anchor modules were installed at known locations around the perimeter of the test area while each MAV was equipped with its own module. By measuring the time of flight for messages sent between the modules, the ranges between the MAVs and the stationary nodes could be determined. A NLLS algorithm was then used to estimate the positions of the MAVs based on these ranges.

This proved to be able to obtain reasonable ($\text{MSE}_x = 0.139$ and $\text{MSE}_y = 0.074$) positional estimates provided that the autopilot software used positional information supplied by the

OptiTrack system in the flight controller. When attempting to use the positional information supplied by the UWB system it was found that there was too much noise on the estimated position. This caused the flight controller to try correct for what it saw as the MAV drifting. This resulted in the MAV moving erratically around the test area instead of hovering in place. To combat this, a Kalman filter was added to reduce the noise present in the positioning system. However, an issue with the module prevented this from being further tuned and tested to ensure that it does solve the problem. Due to the problems faced with this system it was decided to rely on the OptiTrack system for all physical tests.

The NEAT algorithm was selected to generate the NN controllers. This is an evolutionary algorithm that optimises the structure of the NN as well as the connection weights between nodes. In current literature, this method has not been widely used in the field of persistent surveillance. The few papers that do make use of this focus on showing that it is applicable as a method for solving the persistent surveillance problem and do not present comparisons to existing strategies or possible improvements.

The first change made in this work compared to the current literature was with the fitness function used to analyse the performance of the NN controllers. Instead of multiplying the total average fitness, over the entire duration, by the safety coefficient, in this work the safety coefficient was multiplied by the individual fitness scores for each time step. This penalised the system each time the positional constraint was violated instead of penalising it only once at the end. For the test case, this produced an average increase in the coverage levels of 3.14% while the difference between th best performing controller for each case was 9.35s and 7.26% for the average cell age and coverage percentage respectively.

Next, five different sets of input values for the NN were tested. Two of the input sets were from current literature, while the remaining three used varying levels of information retrieved from the surrounding environment. This was done in an attempt to increase the controller's performance. During the testing procedure, it was found that the fourth set of input values produced the best controllers in terms of the performance metrics identified in Chapter 2. This made use of 18 inputs comprised of the ranges returned by the 8 virtual antennae, the cell ages of the furthest cell reached by each of the 8 antenna, a boolean input that was set to 1 if the MAV was inside the MS and a bias node. With these inputs, the post evaluation of the controllers showed that the best controller had an average coverage of 99.1%, cell age of 53.46s and a crash percentage of 2.6% over the 50 tests. The average values over the 20 controllers tested were 94.96% for the coverage, 47.03s for the average cell age and 1.78% for the crash percentage.

The performance of the NN controllers compared to the baseline methods was somewhat disappointing. As expected, the (near) optimal performance obtained from the centralised line-sweep method clearly gave the best performance. However, the NN and the random walk offered greater robustness to MAV failures. For the random walk, this method showed better results in terms of the average cell age (56.59s). However, its ability to completely cover the area was slightly lower than that of the NN, with an average coverage level of 95.96%.

In Chapter 4 a BT was created by hand to assign the MAV to one of five tasks. These were avoidance, homing, charging, waiting or surveillance. During testing this was shown to ensure that the real-world fuel constraints that the MAVs were never violated. In addition, the added avoidance measures further reduced the crash percentage more than doubling the number of tests that did not result in a single collision. Surprisingly the inclusion of the fuel

constraints and the BT actually improved the performance of the system as a whole when considering the coverage and cell age. On average, these were increased to 97.41% and 52.39s respectively. This increased performance was attributed to the fact that the homing task 'broke' the MAVs out of their flight patterns around the border of the MS and periodically drew them back to the centre.

Finally, in Chapter 5 the BT and the NN controllers were implemented in Paparazzi for the actual flight tests in the CyberZoo. These tests showed that the NN controllers offered similar performance on the physical flight platform as they did in simulation. However, the length of the tests were relatively short as a result of the batteries used during testing. This made it difficult to draw conclusions of the controller's avoidance ability. Further, this also hindered the refuelling tests. Here, the MAVs fuel would immediately drop below the first threshold as soon as they took-off. As a result, the MAVs would usually not move away from the depot after their batteries were swapped out.

In conclusion, this thesis presented an evolved NN controller that could be implemented on each individual MAV. By using the adapted fitness function and the new input values, a clear improvement was observed compared to the NNs presented in current literature. While this work focussed on applying this system to PA in greenhouses, it is also directly applicable to security related tasks such as crowd monitoring or patrolling an area.

6-2 Future Work

In this last section, a few topics for future research are described.

6-2-1 UWB Positioning

Research into UWB positioning systems have been well documented in current literature [86, 87, 65, 88]. However, as mentioned in Section 5-1 more work is needed to further develop the Arduino code controlling the ranging procedure between modules. Specifically, this should focus on implementing a more reliable ranging procedure between multiple anchors and multiple tags.

A further issue with the current UWB system is the update rate of the range estimates. Currently the system is based on the TWR method where asynchronous two way ranging is used to determine the distance between two modules. However, this requires the modules to broadcast a total of four messages before it can calculate the range. As more modules are added, this can greatly reduce the system's ability to produce results in (near) real-time. In addition, this can introduce more errors to the range estimates as the increasing time between the ranging messages give the MAVs more time to change their positions during the ranging process [67]. An alternative would be to implement a system based on TDOA, for example. In this case, the position is determined using the difference in propagation times between signal sent by the anchors [68]. As a result, only the anchors are required to broadcast signals and the position can be determined after each anchor broadcasts a single message. This will allow the system to operate with a much larger swarm of MAVs. The main drawback is that this method requires the anchors system clocks to be synchronised with on another [68, 89].

6-2-2 NN Performance

During evolution the NNs usually evolved behaviour where they would fly around near the edge of the MS and would only occasionally pass through the centre of the area. When analysing the performance of the system with the BT implemented, it was observed that by breaking the MAVs out of their naturally formed flight patterns an increase in the area coverage and cell age could be obtained. As a topic for future work it would be useful to determine whether the same benefit could be obtained by including an element of randomness into the NNs inputs. In addition an analysis between the performance of the system using the standard feedforward structure (as in this work) and the performance when recurrent connections are allowed in the NN will be beneficial.

Further, research into the impact of differing age maps is necessary and has not been addressed in current literature. During refuelling, as the batteries are swapped, the MAV must be restarted this could cause it to lose the information currently stored in its age map. This can lead to a duplication of effort as the MAVs survey areas that were recently visited instead of flying towards the unvisited areas. This can also be related to communication constraints placed on the system. Differences between the age maps will start developing as the MAVs move out of communication range, or if the period of communication is longer that the NNs update rate.

6-2-3 Refuelling

While this thesis focussed on a single refuelling station, the method should be directly applicable to the multiple depot case. A minor topic for future research could look into the effect that this will have on the overall systems performance. Related to this, one could then look further to determine the ideal placement for the depots.

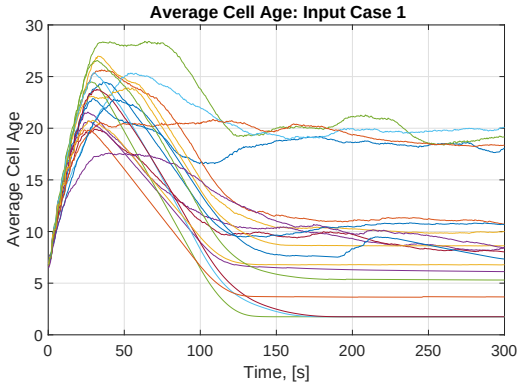
Lastly, this work assumed that each MAV must be completely recharged before it is allowed to leave the depot. While this makes sense for the case where the MAVs have their batteries replaced at the depot, future research could look into recharging the MAVs at the depot instead. In this case it would then be possible to only partially recharge a MAV before it continues with the surveillance mission. This has not received much attention in current literature and may offer increased system performance, particularly during the first recharge cycle where a long queue forms at the depot.

Appendix A

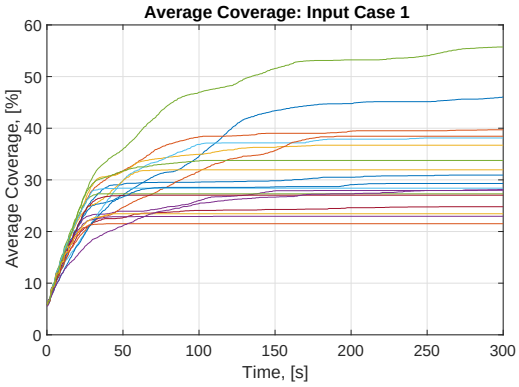
Input Case Results

This appendix gives the individual results of the average cell age and coverage levels achieve by each of the 20 controllers analysed for the different input cases.

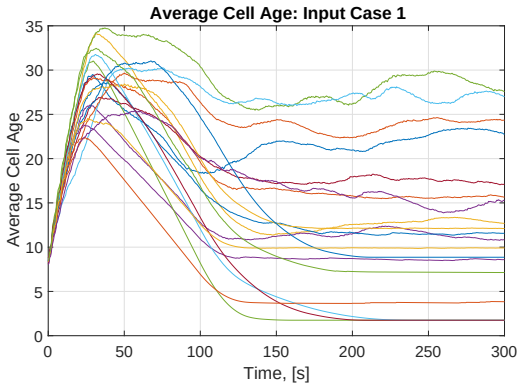
A-1 Input Case 1



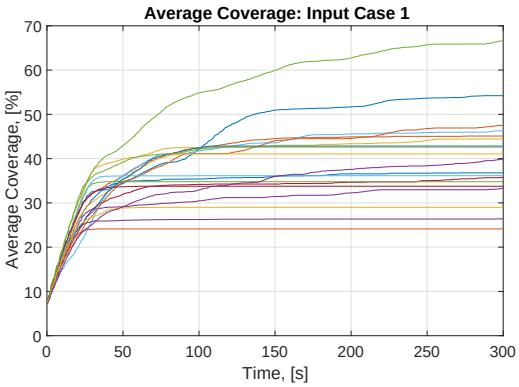
(a) Average cell age: 4 UAVs



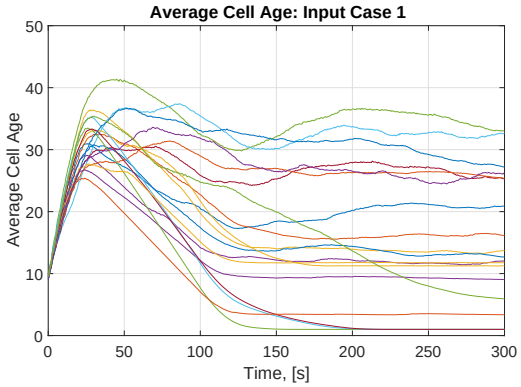
(b) Coverage percentage: 4 UAVs



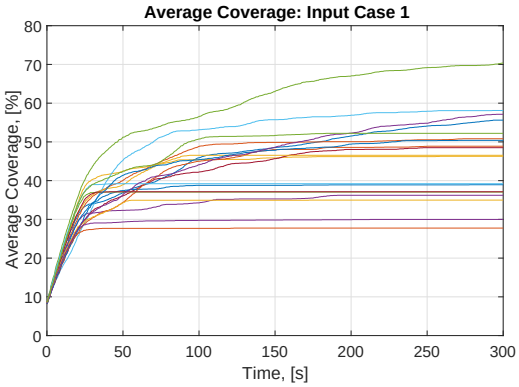
(c) Average cell age: 6 UAVs



(d) Coverage percentage: 6 UAVs



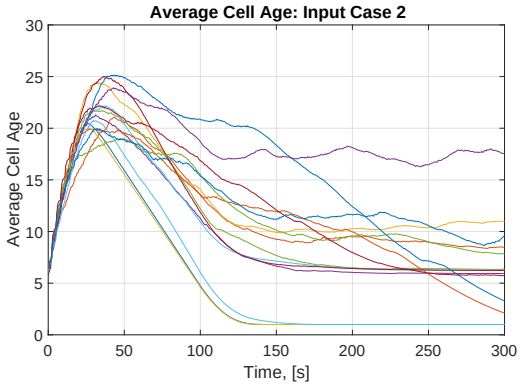
(e) Average cell age: 8 UAVs



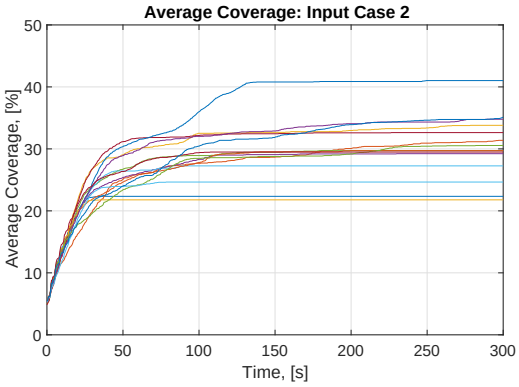
(f) Coverage percentage: 8 UAVs

Figure A-1: Input case 1 results

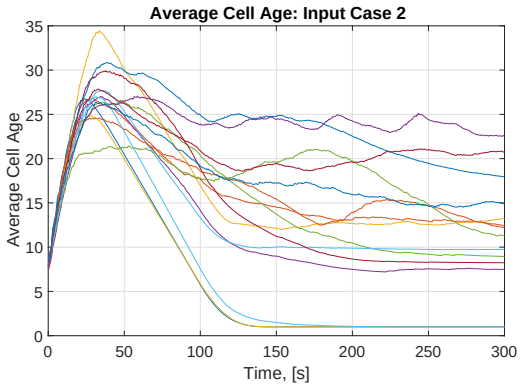
A-2 Input Case 2



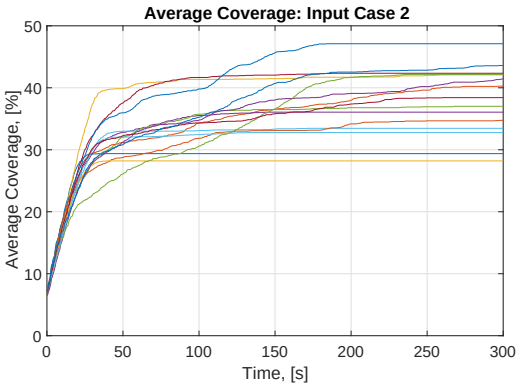
(a) Average cell age: 4 UAVs



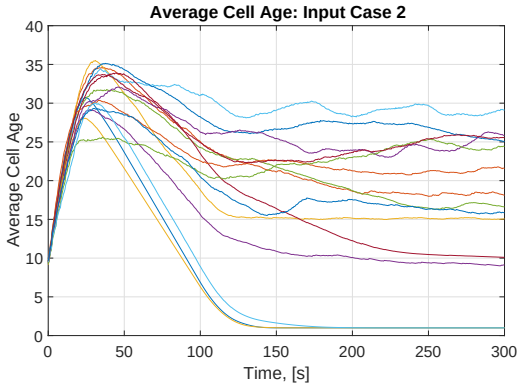
(b) Coverage percentage: 4 UAVs



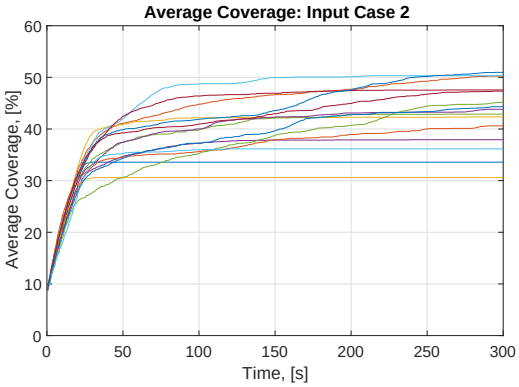
(c) Average cell age: 6 UAVs



(d) Coverage percentage: 6 UAVs



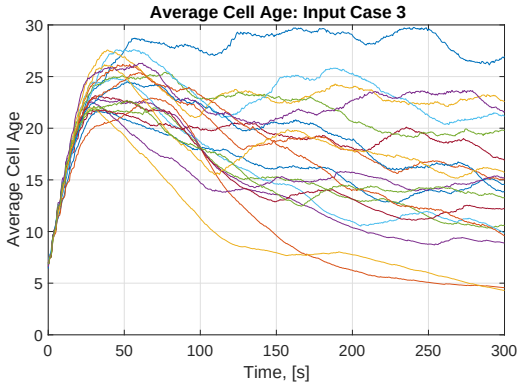
(e) Average cell age: 8 UAVs



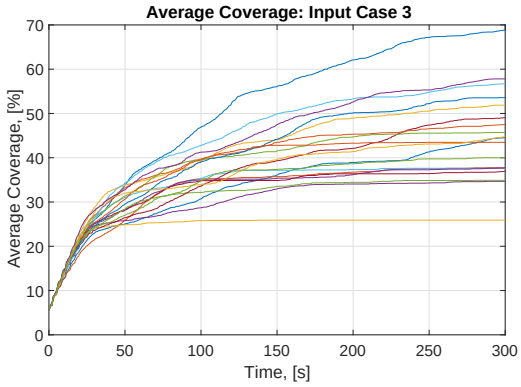
(f) Coverage percentage: 8 UAVs

Figure A-2: Input case 2 results

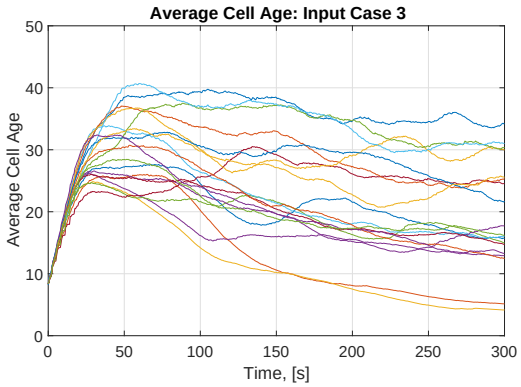
A-3 Input Case 3



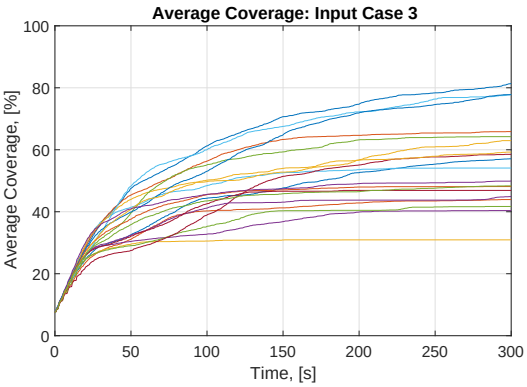
(a) Average cell age: 4 UAVs



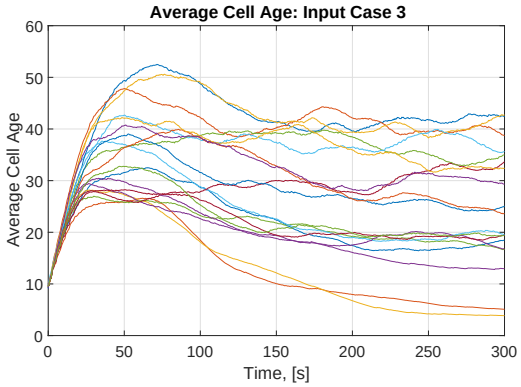
(b) Coverage percentage: 4 UAVs



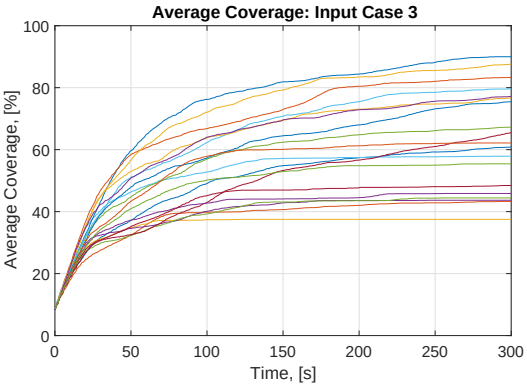
(c) Average cell age: 6 UAVs



(d) Coverage percentage: 6 UAVs



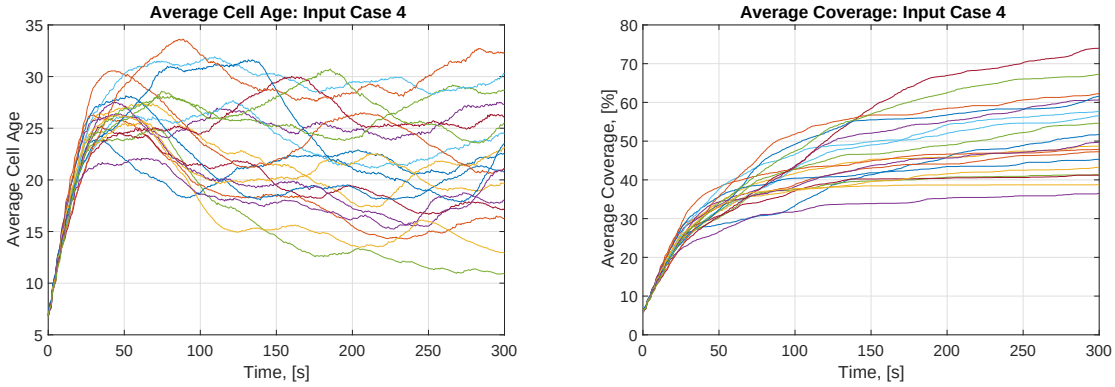
(e) Average cell age: 8 UAVs



(f) Coverage percentage: 8 UAVs

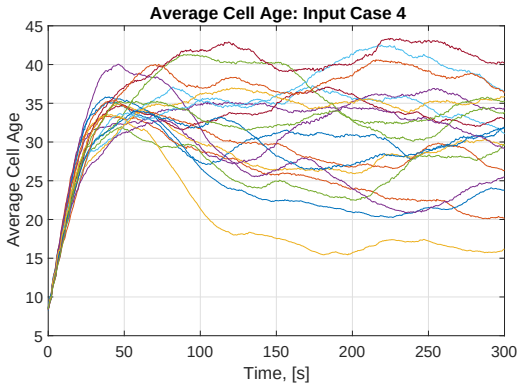
Figure A-3: Input case 3 results

A-4 Input Case 4

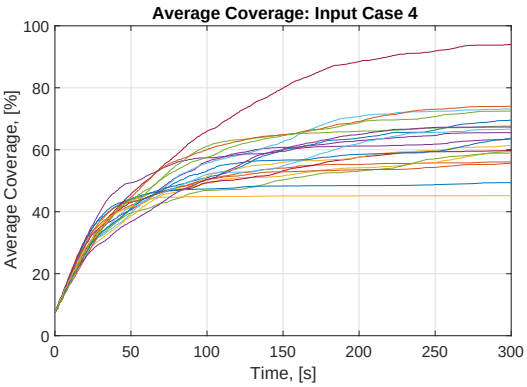


(a) Average cell age: 4 UAVs

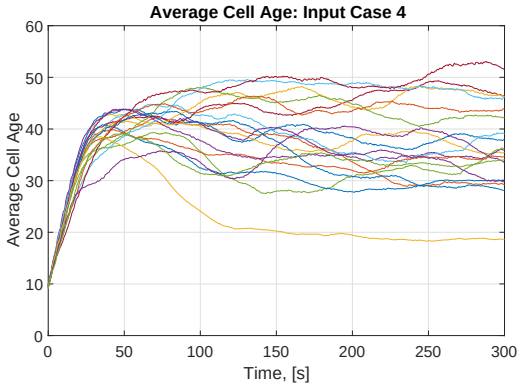
(b) Coverage percentage: 4 UAVs



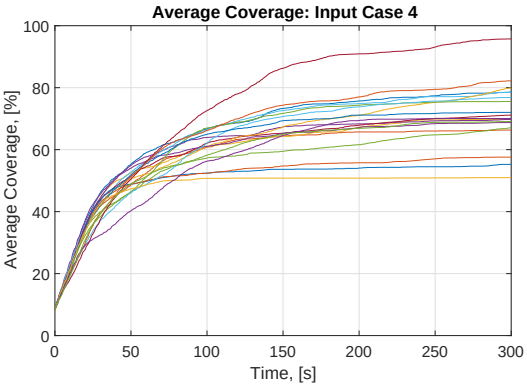
(c) Average cell age: 6 UAVs



(d) Coverage percentage: 6 UAVs



(e) Average cell age: 8 UAVs



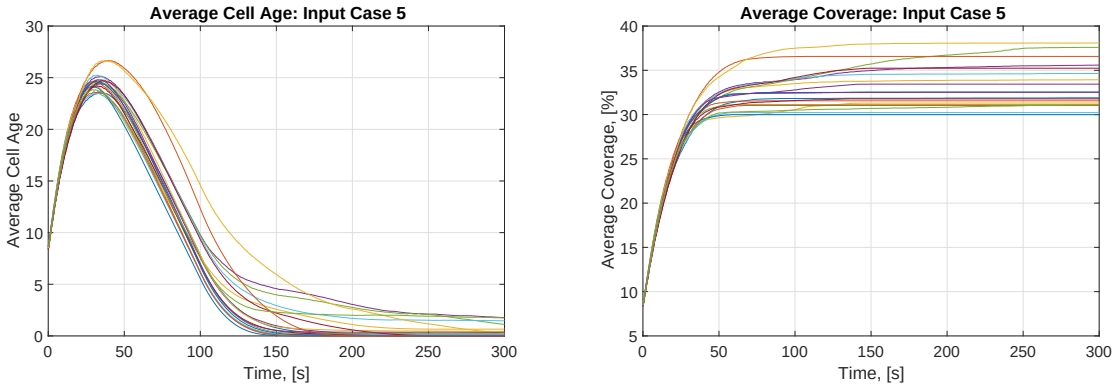
(f) Coverage percentage: 8 UAVs

Figure A-4: Input case 4 results

A-5 Input Case 5

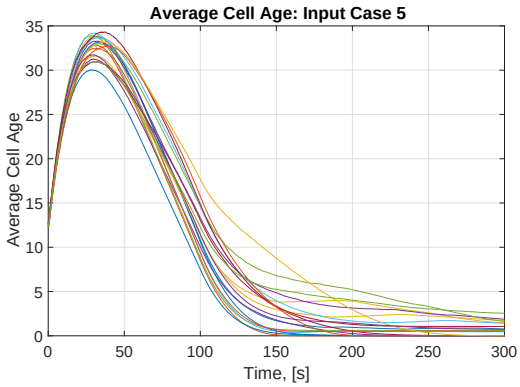
This input case is based off of the work presented in [72]. It makes use of three inputs into the NN, namely; the distance to the nearest other MAV, the distance to the nearest boundary and a bias node. While the authors of [72] were able to acheive reasonable results using these inputs, it could not be replicated here. In this attempt, the NN controllers could not evolve any avoidance behaviours and never learnt to stay within the MS. The original results can

be seen in Figure A-5.

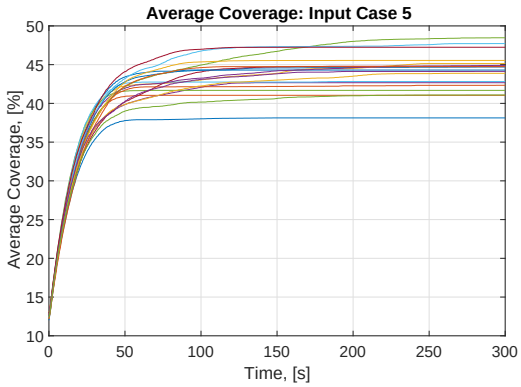


(a) Average cell age: 4 UAVs

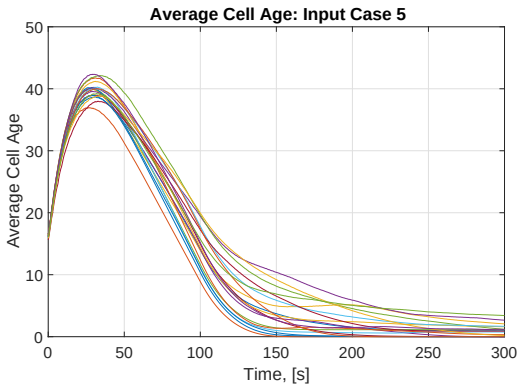
(b) Coverage percentage: 4 UAVs



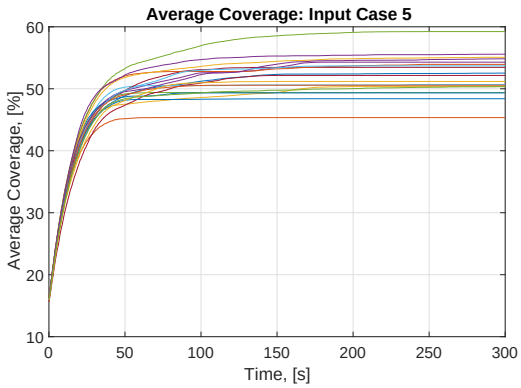
(c) Average cell age: 6 UAVs



(d) Coverage percentage: 6 UAVs



(e) Average cell age: 8 UAVs

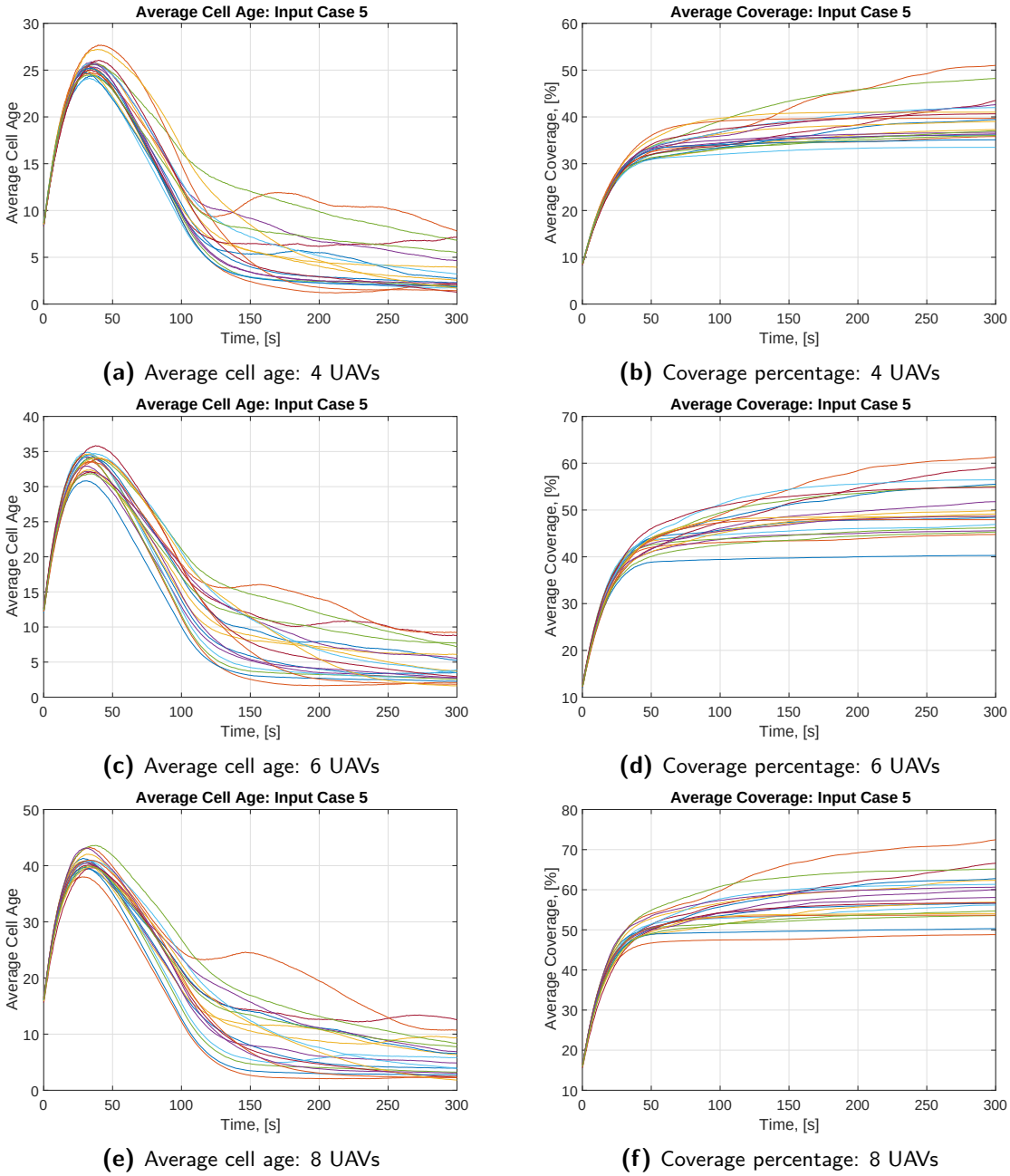


(f) Coverage percentage: 8 UAVs

Figure A-5: Input case 5 results

In an attempt to increase the performance for this case, a high level controller was included to keep the MAVs within the MS. Whenever a MAV left the defined MS, the high level controller would ignore the NN commands and force the MAV to fly towards the centre of the area for 4 time steps. The results of the post evaluation for these evolved NNs is shown in Figure A-6. This does give a slight improvement to the performance but it is still significantly lower than any of the other cases. Again, the NN are unable to evolve avoidance behaviours.

This is not so surprising as the limited inputs used here do not have a directional component. As a result, there is no way of differentiating, for example, between an obstacle on the left of the MAV to one on the right. One method that the original authors could have dealt with this problem is by allowing recurrent connections in their NN. This would allow previous node values to be fed back into the NN during the next iteration. This, in essence, give the NN a limited form of memory which could improve the evolved avoidance and exploration behaviours. However, no mention of this was made in their extended abstract and was not implemented in this work.



(a) Average cell age: 4 UAVs

(b) Coverage percentage: 4 UAVs

(c) Average cell age: 6 UAVs

(d) Coverage percentage: 6 UAVs

(e) Average cell age: 8 UAVs

(f) Coverage percentage: 8 UAVs

Figure A-6: Input case 5 results

Appendix B

Baseline Persistent Coverage Methods

This appendix contains the extracts from the literature review that deals with the two types of baseline methods that were used as a comparison to the evolved NN. The methods are; a sweep planner and the random walk.

B-1 Sweep Planner

Current solutions in PA lean heavily towards a sweep planner approach as seen in [30, 16, 31, 14, 17] for example. This method, also known as a lawnmower search pattern or the line-sweep method, searches the given area using the zig-zag trajectory shown in Figure B-1. However, if there are obstacles or no-go areas present, it is first necessary to decompose the area into smaller subregions (known as exact cellular decomposition) which are then individually searched with these back and forth sweeps. A common approach to decompose the target area is given in subsection B-1-1 and the order in which they are searched can be seen as a Vehicle Routing Problem (VRP).

B-1-1 Boustrophedon Decomposition

For simple environments, rectangular with no obstacles for example, it is very easy to determine the coverage path using this line sweep method. In practice, however, the algorithm will need to account for more complex environments. This is done by decomposing the original area into smaller subregions (or cells) as seen in Figure B-2, which can then be covered with the back and forth sweeps. In literature, the most common method for creating these subregions is know as *Boustrophedon Decomposition*, which is an improvement on the trapezoidal decomposition method.

In the trapezoidal decomposition method described in [3], a vertical line, known as a *slice*, moves from left to right over the entire area to be considered. The cells are then formed using a sequence of *open* and *close* operations which occur when the slice intersects a vertex of a

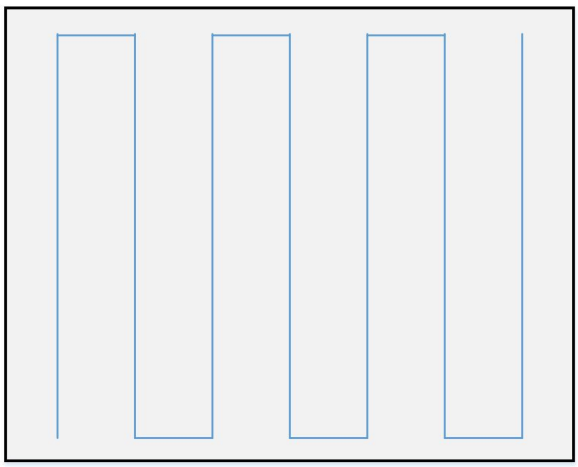


Figure B-1: Lawnmower search pattern

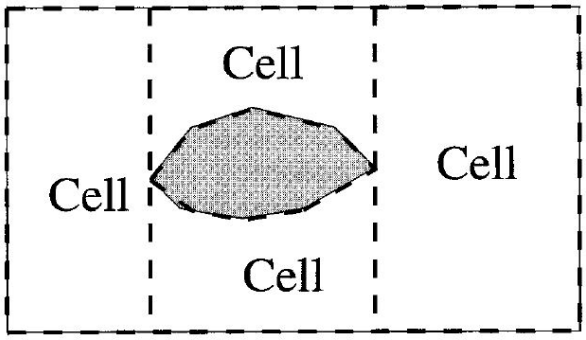


Figure B-2: Boustrophedon decomposition example, [3]

polygonal obstacle. This creates either an IN, OUT or a MIDDLE event. For an IN event, the current cell is *closed* while two new cells are *opened*. An OUT event is the opposite of this, two cells are *closed* while a new one is *opened*. Finally, a MIDDLE event occurs when the current cell is *closed* and a new cell is *opened*. An example of an IN and OUT event is shown in Figure B-3a and Figure B-3b respectively.

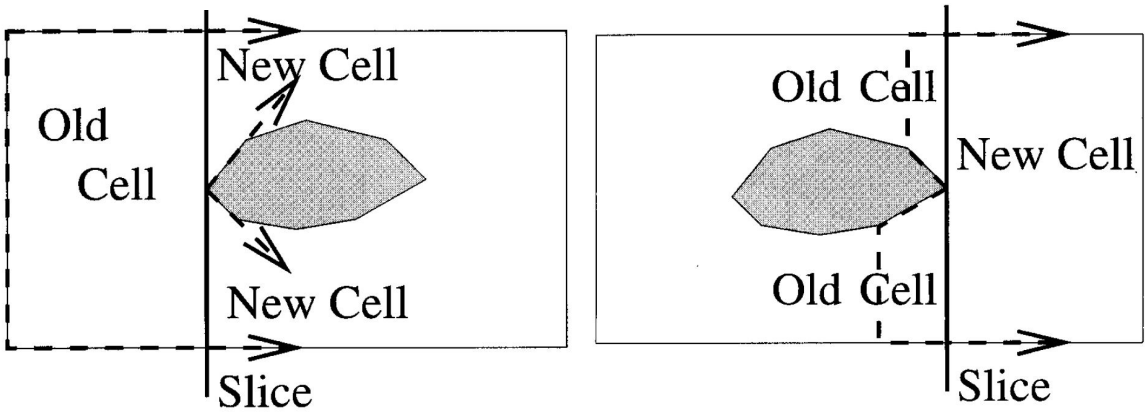
The contribution of the Boustrophedon decomposition is to group together consecutive cells created with the MIDDLE event into a single cell, which in turn reduces the total number of cells. To accomplish this, the MIDDLE event is altered so that instead of opening a new cell, it now updates the current cell.

[90] noted that the cells created are highly dependant on the travelling direction of the *slice*. Therefore, the author proposed a heuristic method for determining the optimal travel direction of the slice. First, the performance of the coverage path was defined as

$$performance = \frac{\text{time performing turns}}{\frac{\text{path length}}{\text{nominal speed}} + \text{time performing turns}} \tag{B-1}$$

Then, the problem was solved by the following procedure:

1. Calculate the performance of the area decompositions for each of the following slice



(a) IN event **(b)** OUT event

Figure B-3: Example of IN and OUT events given in [3]

angles; 0, 30, 60, 90, 120 and 150°. (i.e. a step size of 30°)

2. Select the three best performing angles and discard the rest
3. Half the step size and add this to the angles selected in point 2
4. Calculate the performance of the three new directions
5. Repeat points 2-4 until the the step size is less than 1°

Just as in [91], the author of [90] does not consider the cost of travelling from one subregion to another.

B-1-2 Extension to the Multiple UAV Case

In current literature, there are a number of approaches for extending the sweep planner to account for a multi-agent search the target area. Most commonly, the area is split into M subregions (where M is the number of MAVs in the system) either using a decomposition method, as in [31, 30], or manually as in [92, 93]. Each agent is then assigned its own subregion which it searches using back and forth sweeps. This approach is known as spacial decomposition.

Another method is shown in [16, 17]. Here, the total coverage path is determined using Boustrophedon decomposition with a pre-selected sweep direction and it is then divided into M equal sections (depending on either distance covered or travel time). Each of these sections is then assigned to individual agents.

Lastly, the least common method, used in [94], is to specify a formation that the agents must travel in, for example a line, and have this formation preform the line-sweep. This effectively increases the width of the area covered by each sweep.

B-2 Random Walk

Arguably, the simplest method for implementing a reactive motion control strategy is to use a random walk. In this work, the, the MAV decides on the next cell to visit through the use of a probability distribution like a Gaussian or uniform distribution. However, it is possible to adapt this method to include environmental data into the selection process.

For example, in [9] the authors adapted the method slightly to include knowledge of the environment so that the UAVs would focus more on the unexplored or partially explored areas. The goal of this paper was slightly different to that of persistent surveillance in that instead of attempting to minimise the total age of the age map, the authors were trying to reduce the uncertainty of measurements captured in the target area. Successive visits to a cell increased its certainty and once it became completely certain, it was no longer visited by any of the agents.

An easy way to adapt the pure random walk to the problem at hand will be to include knowledge of the cell ages, and maybe the distances between cells and agents. This can be accomplished by using a weighted roulette wheel approach where the probability of a cell being selected is proportional to its age. The relevant weights of the cells can be calculated as in Equation (B-2) below. Note the age of cell 0 is not considered as this cell contains the depot and, therefore, does not need to be covered by the agents.

$$weight(n) = \frac{age(n)}{\sum_{i=1}^{N} age(i)}, \ \forall n \in 1, 2, \ldots, N \tag{B-2}$$

Appendix C

Collision Avoidance Proof

As mentioned in Section 4-3 the avoidance strategy used in this work is based off of [59]. It relies on the principle that if two MAVs are flying on a collision course with an identical speed V, then the MAVs will not collide provided that at least one MAV reverses its flight direction. It must continue along this reversed path, flying with a speed of v, until it reaches the point where it entered into the conflict. The proof for this statement is taken from [59] and is given in below for two MAVs operating on 2-D plane.

Collision Avoidance Proof. If there are two MAVs, $A_m \in \{1, 2\}$, where their initial positions are given by $\vec{P}_m(0)$ and they are flying at a constant velocity vector where $|\vec{v}_1| = |\vec{v}_2|$. Their positions over time and the relative position vector $\vec{D}(t)$ is given as:

$$\begin{aligned}
\vec{P}_1(t) &= \vec{P}_1(0) + \vec{v}_1 t \\
\vec{P}_2(t) &= \vec{P}_2(0) + \vec{v}_2 t \\
\vec{D}(t) &= \vec{P}_1(t) - \vec{P}_2(t) = (\vec{P}_1(0) - \vec{P}_2(0)) + (\vec{v}_1 - \vec{v}_2)t
\end{aligned}$$

By splitting the vectors into their x and y components, the magnitude of $\vec{D}(t)$, $d(t)$, is given as

$$d(t) = \sqrt{(D_x + v_x t)^2 + (D_y + v_y t)^2}$$

where

$$\begin{aligned}
D_x &= P_{x,1}(0) - P_{x,1}(0) \\
D_y &= P_{y,1}(0) - P_{y,1}(0) \\
v_x &= v_{x,1} - v_{x,1} \\
v_y &= v_{y,1} - v_{y,1}
\end{aligned}$$

Now, let t_c denote the time where $d(t)$ is at its minimum (i.e. the point where the two MAVs

are closest). This equals the time at which $\dot{d}(t) = 0$.

$$\dot{d}(t) = \frac{v_x(D_x + V_x t_c) + v_y(D_y + v_y t_c)}{\sqrt{(D_x + v_x t)^2 + (D_y + v_y t)^2}}$$
$$t_c = -\frac{D_x v_x + D_y v_y}{v_x^2 + v_y^2}$$

thus

$$d(t_c) = \sqrt{\left(D_x - V_x \frac{D_x v_x + D_y v_y}{v_x^2 + v_y^2}\right)^2 + \left(D_y + v_y \frac{D_x v_x + D_y v_y}{v_x^2 + v_y^2}\right)^2}$$

For a collision between the MAVs to occur, the distance between them must be less than the diameter of the MAVs defined as D. As the minimum distance between the MAV occurs at t_c, this translates to $d(t_c) \leq D$, or:

$$\sqrt{\left(D_x - V_x \frac{D_x v_x + D_y v_y}{v_x^2 + v_y^2}\right)^2 + \left(D_y + v_y \frac{D_x v_x + D_y v_y}{v_x^2 + v_y^2}\right)^2} \leq D$$
$$\therefore D_x^2 + D_y^2 - \frac{(D_x v_x + D_y v_y)^2}{v_x^2 + v_y^2} \leq D$$

from this it follows that:

$$\frac{v_x}{v_y} = \frac{D_x D_y \pm D\sqrt{D_x^2 + D_y^2 - D^2}}{D_y^2 - D^2}$$

Now, since the terms on the left hand side of the inequality are constant and one of the and one of the solutions corresponds to a negative value of t_c, the following relationship between the velocities can be determined

$$v_x \leq C v_y$$

By substituting the original velocity vectors back into this equation we get:

$$v_{x,1} - v_{x,2} = C(v_{y,1} - v_{y,2})$$

If MAV2 was to reverse its velocity vector such that $\vec{v}_{2,new} = -\vec{v}_2$. In this case, for the MAVs to remain on a collision course, the following equations must be true:

$$v_{x,1} - v_{x,2} \leq C(v_{y,1} - v_{y,2})$$
$$v_{x,1} - v_{x,2,new} \leq C(v_{y,1} - v_{y,2,new}) \Rightarrow v_{x,1} + v_{x,2} \leq C(v_{y,1} + v_{y,2})$$

This can only be true if

$$\frac{v_{x,1}}{v_{y,1}} = \frac{v_{x,2}}{v_{y,2}}$$

Using the earlier definition that $|\vec{v}_1| = |\vec{v}_2|$ and the above equation it follows that $\vec{v}_1 = \vec{v}_2$. Therefore, if the initial positions were different then the two MAVs could not have initially been on a collision course. AS this violate the initial assumption, this shows that reversing the velocity of one MAV will prevent the collision. □

Bibliography

[1] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary computation*, vol. 10, no. 2, pp. 99–127, 2002.

[2] M. Duarte, V. Costa, J. Gomes, T. Rodrigues, F. Silva, S. M. Oliveira, and A. L. Christensen, "Evolution of collective behaviors for a real swarm of aquatic surface robots," *PloS one*, vol. 11, no. 3, p. e0151834, 2016.

[3] H. Choset, "Coverage of known spaces: The boustrophedon cellular decomposition," *Autonomous Robots*, vol. 9, no. 3, pp. 247–253, 2000.

[4] P. Ogren, "Increasing modularity of UAV control systems using computer game behavior trees," in *Aiaa guidance, navigation, and control conference*, p. 4458, 2012.

[5] M. Mazur, R. Abadie, J. Smith, A. Wisniewski, V. Huff, and S. Stroh, *Clarity from above: PwC global report on the commercial applications of drone technology.* PwC Polska Sp, 2016.

[6] C. A. Rokhmana, "The potential of UAV-based remote sensing for supporting precision agriculture in indonesia," *Procedia Environmental Sciences*, vol. 24, pp. 245–253, 2015.

[7] A. Barrientos, J. Colorado, J. d. Cerro, A. Martinez, C. Rossi, D. Sanz, and J. Valente, "Aerial remote sensing in agriculture: A practical approach to area coverage and path planning for fleets of mini aerial robots," *Journal of Field Robotics*, vol. 28, no. 5, pp. 667–689, 2011.

[8] D. J. Mulla, "Twenty five years of remote sensing in precision agriculture: Key advances and remaining knowledge gaps," *Biosystems engineering*, vol. 114, no. 4, pp. 358–371, 2013.

[9] D. Albani, J. IJsselmuiden, R. Haken, and V. Trianni, "Monitoring and mapping with robot swarms for agricultural applications," in *Advanced Video and Signal Based Surveillance (AVSS), 2017 14th IEEE International Conference on*, pp. 1–6, IEEE, 2017.

[10] D. Gómez-Candón, A. De Castro, and F. López-Granados, "Assessing the accuracy of mosaics from unmanned aerial vehicle (UAV) imagery for precision agriculture purposes in wheat," *Precision Agriculture*, vol. 15, no. 1, pp. 44–56, 2014.

[11] J. A. Berni, P. J. Zarco-Tejada, L. Suárez, and E. Fereres, "Thermal and narrowband multispectral remote sensing for vegetation monitoring from an unmanned aerial vehicle," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 47, no. 3, pp. 722–738, 2009.

[12] M. Erena, S. Montesinos, D. Portillo, J. Alvarez, C. Marin, L. Fernandez, J. Henarejos, and L. Ruiz, "Configuration and specifications of an unmanned aerial vehicle for precision agriculture.," *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*, vol. 41, 2016.

[13] J. Primicerio, S. F. Di Gennaro, E. Fiorillo, L. Genesio, E. Lugato, A. Matese, and F. P. Vaccari, "A flexible unmanned aerial vehicle for precision agriculture," *Precision Agriculture*, vol. 13, no. 4, pp. 517–523, 2012.

[14] G. Ristorto, P. D'Incalci, R. Gallo, F. Mazzetto, and G. Guglieri, "Mission planning for the estimation of the field coverage of unmanned aerial systems in monitoring mission in precision farming," *CHEMICAL ENGINEERING TRANSACTIONS*, vol. 58, pp. 649–654, 2017.

[15] T. Oksanen and A. Visala, "Coverage path planning algorithms for agricultural field machines," *Journal of Field Robotics*, vol. 26, no. 8, pp. 651–668, 2009.

[16] D. Richards, T. Patten, R. Fitch, D. Ball, and S. Sukkarieh, "User interface and coverage planner for agricultural robotics," in *Proceedings of ARAA Australasian Conference on Robotics and Automation (ACRA)*, 2015.

[17] D. Ball, P. Ross, A. English, T. Patten, B. Upcroft, R. Fitch, S. Sukkarieh, G. Wyeth, and P. Corke, "Robotics for sustainable broad-acre agriculture," in *Field and Service Robotics*, pp. 439–453, Springer, 2015.

[18] B. H. Y. Alsalam, K. Morton, D. Campbell, and F. Gonzalez, "Autonomous UAV with vision based on-board decision making for remote sensing and precision agriculture," in *Aerospace Conference, 2017 IEEE*, pp. 1–12, IEEE, 2017.

[19] J. J. Roldán, G. Joossen, D. Sanz, J. del Cerro, and A. Barrientos, "Mini-UAV based sensory system for measuring environmental variables in greenhouses," *Sensors*, vol. 15, no. 2, pp. 3334–3350, 2015.

[20] J. J. Roldán, P. Garcia-Aunon, M. Garzón, J. de León, J. del Cerro, and A. Barrientos, "Heterogeneous multi-robot system for mapping environmental variables of greenhouses," *Sensors*, vol. 16, no. 7, p. 1018, 2016.

[21] K. Ferentinos, N. Katsoulas, A. Tzounis, C. Kittas, and T. Bartzanas, "A climate control methodology based on wireless sensor networks in greenhouses," in *XXIX International Horticultural Congress on Horticulture: Sustaining Lives, Livelihoods and Landscapes (IHC2014): 1107*, pp. 75–82, 2014.

[22] K. P. Ferentinos, N. Katsoulas, A. Tzounis, T. Bartzanas, and C. Kittas, "Wireless sensor networks for greenhouse climate and plant condition assessment," *Biosystems Engineering*, vol. 153, pp. 70–81, 2017.

[23] J. Hwang, C. Shin, and H. Yoe, "A wireless sensor network-based ubiquitous paprika growth management system," *Sensors*, vol. 10, no. 12, pp. 11566–11589, 2010.

[24] G. Aiello, I. Giovino, M. Vallone, P. Catania, and A. Argento, "A decision support system based on multisensor data fusion for sustainable greenhouse management," *Journal of Cleaner Production*, vol. 172, pp. 4057–4065, 2018.

[25] K. Langendoen, A. Baggio, and O. Visser, "Murphy loves potatoes: Experiences from a pilot sensor network deployment in precision agriculture," in *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International*, pp. 8–pp, IEEE, 2006.

[26] T. Ahonen, R. Virrankoski, and M. Elmusrati, "Greenhouse monitoring with wireless sensor network," in *Mechtronic and Embedded Systems and Applications, 2008. MESA 2008. IEEE/ASME International Conference on*, pp. 403–408, IEEE, 2008.

[27] D. Chaudhary, S. Nayse, and L. Waghmare, "Application of wireless sensor networks for greenhouse parameter control in precision agriculture," *International Journal of Wireless & Mobile Networks (IJWMN) Vol*, vol. 3, no. 1, pp. 140–149, 2011.

[28] H. M. Jawad, R. Nordin, S. K. Gharghan, A. M. Jawad, and M. Ismail, "Energy-efficient wireless sensor networks for precision agriculture: A review," *Sensors*, vol. 17, no. 8, p. 1781, 2017.

[29] N. Nigam, "The multiple unmanned air vehicle persistent surveillance problem: A review," *Machines*, vol. 2, no. 1, pp. 13–72, 2014.

[30] G. S. Avellar, G. A. Pereira, L. C. Pimenta, and P. Iscold, "Multi-UAV routing for area coverage and remote sensing with minimum time," *Sensors*, vol. 15, no. 11, pp. 27783–27803, 2015.

[31] I. Maza and A. Ollero, "Multiple UAV cooperative searching operation using polygon area decomposition and efficient coverage algorithms," in *Distributed Autonomous Robotic Systems 6*, pp. 221–230, Springer, 2007.

[32] N. Nigam, S. Bieniawski, I. Kroo, and J. Vian, "Control of multiple UAVs for persistent surveillance: algorithm and flight test results," *IEEE Transactions on Control Systems Technology*, vol. 20, no. 5, pp. 1236–1251, 2012.

[33] A. Wallar, E. Plaku, and D. A. Sofge, "Reactive motion planning for unmanned aerial surveillance of risk-sensitive areas," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 3, pp. 969–980, 2015.

[34] P. Trodden and A. Richards, "Multi-vehicle cooperative search using distributed model predictive control," in *AIAA Guidance, Navigation and Control Conference and Exhibit*, p. 7138, 2008.

[35] M. M. Polycarpou, Y. Yang, and K. M. Passino, "A cooperative search framework for distributed agents," in *Intelligent Control, 2001.(ISIC'01). Proceedings of the 2001 IEEE International Symposium on*, pp. 1–6, IEEE, 2001.

[36] Y. Yang, M. M. Polycarpou, and A. A. Minai, "Opportunistically cooperative neural learning in mobile agents," in *Neural Networks, 2002. IJCNN'02. Proceedings of the 2002 International Joint Conference on*, vol. 3, pp. 2638–2643, IEEE, 2002.

[37] X. Lin and C. G. Cassandras, "An optimal control approach to the multi-agent persistent monitoring problem in two-dimensional spaces," *IEEE Transactions on Automatic Control*, vol. 60, no. 6, pp. 1659–1664, 2015.

[38] C. G. Cassandras, X. Lin, and X. Ding, "An optimal control approach to the multi-agent persistent monitoring problem," *IEEE Transactions on Automatic Control*, vol. 58, no. 4, pp. 947–961, 2013.

[39] C. Franco, G. López-Nicolás, C. Sagüés, and S. Llorente, "Adaptive action for multi-agent persistent coverage," *Asian Journal of Control*, vol. 18, no. 2, pp. 419–432, 2016.

[40] M. D. Richards, D. Whitley, J. R. Beveridge, T. Mytkowicz, D. Nguyen, and D. Rome, "Evolving cooperative strategies for UAV teams," in *Proceedings of the 7th annual conference on Genetic and evolutionary computation*, pp. 1721–1728, ACM, 2005.

[41] N. Zhou, C. G. Cassandras, X. Yu, and S. B. Andersson, "Decentralized event-driven algorithms for multi-agent persistent monitoring," *arXiv preprint arXiv:1708.06432*, 2017.

[42] B. D. Song, J. Kim, and J. R. Morrison, "Rolling horizon path planning of an autonomous system of UAVs for persistent cooperative service: Milp formulation and efficient heuristics," *Journal of Intelligent & Robotic Systems*, vol. 84, no. 1-4, pp. 241–258, 2016.

[43] J. Kim, B. D. Song, and J. R. Morrison, "On the scheduling of systems of UAVs and fuel service stations for long-term mission fulfillment," *Journal of Intelligent & Robotic Systems*, pp. 1–13, 2013.

[44] B. Bethke, J. P. How, and J. Vian, "Multi-UAV persistent surveillance with communication constraints and health management," in *AIAA Guidance, Navigation, and Control Conference (GNC)*, 2009.

[45] M. Valenti, D. Dale, J. How, D. Pucci de Farias, and J. Vian, "Mission health management for 24/7 persistent surveillance operations," in *AIAA guidance, navigation and control conference and exhibit*, p. 6508, 2007.

[46] B. Bethke, M. Valenti, and J. P. How, "UAV task assignment," *IEEE robotics & automation magazine*, vol. 15, no. 1, 2008.

[47] N. K. Ure, G. Chowdhary, Y. F. Chen, M. Cutler, J. P. How, and J. Vian, "Decentralized learning-based planning for multiagent missions in the presence of actuator failures," in *Unmanned Aircraft Systems (ICUAS), 2013 International Conference on*, pp. 1125–1134, IEEE, 2013.

[48] V. Trianni and S. Nolfi, "Engineering the evolution of self-organizing behaviors in swarm robotics: A case study," *Artificial life*, vol. 17, no. 3, pp. 183–202, 2011.

[49] K. Y. Scheper, S. Tijmons, C. C. de Visser, and G. C. de Croon, "Behavior trees for evolutionary robotics," *Artificial life*, vol. 22, no. 1, pp. 23–48, 2016.

[50] M. Duarte, S. M. Oliveira, and A. L. Christensen, "Hybrid control for large swarms of aquatic drones," in *Proceedings of the 14th International Conference on the Synthesis & Simulation of Living Systems. MIT Press, Cambridge, MA*, pp. 785–792, Citeseer, 2014.

[51] M. Duarte, J. Gomes, V. Costa, S. M. Oliveira, and A. L. Christensen, "Hybrid control for a real swarm robotics system in an intruder detection task," in *European Conference on the Applications of Evolutionary Computation*, pp. 213–230, Springer, 2016.

[52] Y. Cao, W. Yu, W. Ren, and G. Chen, "An overview of recent progress in the study of distributed multi-agent coordination," *IEEE Transactions on Industrial informatics*, vol. 9, no. 1, pp. 427–438, 2013.

[53] Z. Yan, N. Jouandeau, and A. A. Cherif, "A survey and analysis of multi-robot coordination," *International Journal of Advanced Robotic Systems*, vol. 10, no. 12, p. 399, 2013.

[54] S. Gezici, Z. Tian, G. B. Giannakis, H. Kobayashi, A. F. Molisch, H. V. Poor, and Z. Sahinoglu, "Localization via ultra-wideband radios: a look at positioning aspects for future sensor networks," *IEEE signal processing magazine*, vol. 22, no. 4, pp. 70–84, 2005.

[55] P. Volf, D. Šišlák, D. Pavlíček, and M. Pěchouček, "Surveillance of unmanned aerial vehicles using probability collectives," *Holonic and Multi-Agent Systems for Manufacturing*, pp. 235–245, 2011.

[56] Parrot, "Parrot bebop 2." https://www.parrot.com/us/drones/parrot-bebop-2#lightweight, 2018. Accessed: 2018-11-14.

[57] J. Lipsky, "Parrot unveils bebop 2 drone." https://www.eetimes.com/document.asp?doc_id=1328291, 2015. Accessed: 2018-11-14.

[58] Paparazzi, "Paparazzi: The free autopilot." https://wiki.paparazziuav.org/wiki/General, 2016. Accessed: 2018-11-14.

[59] T. Szabó, "Autonomous collision avoidance for swarms of MAVs," MSc thesis, Delft University of Technology, 2015.

[60] OptiTrack, "OptiTrack sytems." https://optitrack.com/systems/#robotics, 2018. Accessed: 2018-11-14.

[61] A. Pancham, N. Tlale, and G. Bright, "Literature review of SLAM and DATMO," 2011.

[62] A. Nemra, L. M. Bergasa, E. López, R. Barea, A. Gómez, and Á. Saltos, "Robust visual simultaneous localization and mapping for MAV using smooth variable structure filter," in *Robot 2015: Second Iberian Robotics Conference*, pp. 557–569, Springer, 2016.

[63] K. Tateno, F. Tombari, I. Laina, and N. Navab, "CNN-SLAM: Real-time dense monocular SLAM with learned depth prediction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, 2017.

[64] DecaWave Ltd, *Product Information: DWM 1000*, 2018. V. 1.2.

[65] K. Guo, Z. Qiu, C. Miao, A. H. Zaini, C.-L. Chen, W. Meng, and L. Xie, "Ultra-wideband-based localization for quadcopter navigation," *Unmanned Systems*, vol. 4, no. 01, pp. 23–34, 2016.

[66] B. D. Schutter and T. van den Boom, *Optimization in Systems and Control*. TU Delft, 2015.

[67] Y. Jiang and V. C. Leung, "An asymmetric double sided two-way ranging for crystal offset," in *Signals, Systems and Electronics, 2007. ISSSE'07. International Symposium on*, pp. 525–528, IEEE, 2007.

[68] S. van der Helm, "On-board range-based relative localization," (The Netherlands), MSc thesis, Delft University of Technology, 2018.

[69] X. Yao, "A review of evolutionary artificial neural networks," *International journal of intelligent systems*, vol. 8, no. 4, pp. 539–567, 1993.

[70] M. Buckland and M. Collins, *AI techniques for game programming*. Premier press, 2002.

[71] A. L. Nelson, G. J. Barlow, and L. Doitsidis, "Fitness functions in evolutionary robotics: A survey and analysis," *Robotics and Autonomous Systems*, vol. 57, no. 4, pp. 345–370, 2009.

[72] J. Butterworth, B. Broecker, K. Tuyls, and P. Paoletti, "Evolving coverage behaviours for MAVs using NEAT," in *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 1886–1888, International Foundation for Autonomous Agents and Multiagent Systems, 2018.

[73] M. Darrah, W. Niland, and B. M. Stolarik, "Increasing UAV task assignment performance through parallelized genetic algorithms," in *Proceedings from*, 2007.

[74] T. Shima and C. Schumacher, "Assignment of cooperating UAVs to simultaneous tasks using genetic algorithms," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, p. 5829, 2005.

[75] A. Richards, J. Bellingham, M. Tillerson, and J. How, "Coordination and control of multiple UAVs," in *AIAA guidance, navigation, and control conference, Monterey, CA*, 2002.

[76] P. Goyal, E. Smeur, and G. de Croon, "Mission planning for sensor network deployment using a fleet of drones," *Delft University of Technology, Delft, Zuid-Holland*, vol. 2629, 2016.

[77] J. Kim, B. D. Song, and J. R. Morrison, "On the scheduling of systems of UAVs and fuel service stations for long-term mission fulfilment," *Journal of Intelligent & Robotic Systems*, pp. 1–13, 2013.

[78] M. B. Dias, R. Zlot, N. Kalra, and A. Stentz, "Market-based multirobot coordination: A survey and analysis," *Proceedings of the IEEE*, vol. 94, no. 7, pp. 1257–1270, 2006.

[79] F. Michaud, E. Robichaud, and J. Audet, "Using motives and artificial emotions for prolonged activity of a group of autonomous robots," in *Proceedings of the AAAI Fall Symposium on Emotions. Cape Code Massachussetts*, 2001.

[80] J. Leonard, A. Savvaris, and A. Tsourdos, "Energy management in swarm of unmanned aerial vehicles," *Journal of Intelligent & Robotic Systems*, vol. 74, no. 1-2, pp. 233–250, 2014.

[81] F. Sempé, A. Munoz, and A. Drogoul, "Autonomous robots sharing a charging station with no communication: a case study," in *Distributed Autonomous Robotic Systems 5*, pp. 91–100, Springer, 2002.

[82] C.-U. Lim, R. Baumgarten, and S. Colton, "Evolving behaviour trees for the commercial game defcon," in *European Conference on the Applications of Evolutionary Computation*, pp. 100–110, Springer, 2010.

[83] A. Klöckner, "Interfacing behavior trees with the world using description logic," in *AIAA Guidance, Navigation, and Control (GNC) Conference*, p. 4636, 2013.

[84] A. Marzinotto, M. Colledanchise, C. Smith, and P. Ögren, "Towards a unified behavior trees framework for robot control," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pp. 5420–5427, IEEE, 2014.

[85] M. Coppola, K. Mcguire, K. Scheper, and G. Croon, "On-board bluetooth-based relative localization for collision avoidance in micro air vehicle swarms," 09 2016.

[86] J. Li, Y. Bi, K. Li, K. Wang, F. Lin, and B. M. Chen, "Accurate 3D localization for MAV swarms by UWB and IMU fusion," in *2018 IEEE 14th International Conference on Control and Automation (ICCA)*, pp. 100–105, IEEE, 2018.

[87] M. W. Mueller, M. Hamer, and R. D'Andrea, "Fusing ultra-wideband range measurements with accelerometers and rate gyroscopes for quadrocopter state estimation," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pp. 1730–1736, IEEE, 2015.

[88] T. M. Nguyen, A. H. Zaini, K. Guo, and L. Xie, "An ultra-wideband-based multi-UAV localization system in GPS-denied environments," in *Proc. Int. Micro Air Vehicle Conf. Competition*, pp. 1–6, 2016.

[89] A. M. Khalel, "Position location techniques in wireless communication systems," 2010.

[90] T. Oksanen *et al.*, *Path planning algorithms for agricultural field machines*. Helsinki University of Technology, 2007.

[91] W. H. Huang, "Optimal line-sweep-based decompositions for coverage algorithms," in *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, vol. 1, pp. 27–32, IEEE, 2001.

[92] R. Bähnemann, D. Schindler, M. Kamel, R. Siegwart, and J. Nieto, "A decentralized multi-agent unmanned aerial system to search, pick up, and relocate objects," *arXiv preprint arXiv:1707.03734*, 2017.

[93] R. Bähnemann, M. Pantic, M. Popvić, D. Schindler, M. Tranzatto, M. Kamel, M. Grimm, J. Widauer, R. Siegwart, and J. Nieto, "The ETH-MAV team in the MBZ international robotics challenge," *arXiv preprint arXiv:1710.08275*, 2017.

[94] P. Vincent and I. Rubin, "A framework and analysis for cooperative search using UAV swarms," in *Proceedings of the 2004 ACM symposium on Applied computing*, pp. 79–86, ACM, 2004.

Glossary

List of Acronyms

BT	Behaviour Tree
CPP	Coverage Path Planning
EA	Evolutionary Algorithm
ER	Evolutionary Robotics
GA	Genetic Algorithm
IPM	Integrated Pest Management
LP	Linear Programming
MAV	Micro Aerial Vehicle
MILP	Mixed Integer Linear Programming
MPC	Model Predictive Control
MRP	Multi-agent Reactive Policy
MS	Mission Space
MSE	Mean-Squared Error
NEAT	Neuro Evolution of Augmenting Topologies
NLLS	Non-Linear Least Squares
NN	Neural Network
PA	Precision Agriculture
ROS	Robot Operating System
SLAM	Simultaneous Localization and Mapping

TDOA Time Difference of Arrival

TWR Two Way Ranging

TU Delft Delft University of Technology

TWEANN Topology and Weight Evolving Artificial Neural Networks

UAV Unmanned Aerial Vehicle

UWB Ultra-wideband

VRP Vehicle Routing Problem

WSN Wireless Sensor Network

List of Symbols

α	Adjustment angle for collision avoidance
$\delta(i,j)$	Specisation compatibility distance
δ_t	Compatibility threshold
ω	Natural frequency
σ	Standard deviation
θ	Direction of travel
ζ	Model damping factor
$\bar{H}(X_k)$	Hessian of current state
$\bar{W}_{gene}$	Average weight differences of matching genes
ΔT	Time step
$\dot{e}_{depot}$	Depot recharge rate
$\dot{e}_m$	MAV energy depletion rate
$\mathcal{FP}$	Set of flight plans
$\mathcal{G}$	Continuous mission space
$\mathcal{I}_{age}$	Information age
$\mathcal{O}$	Set of obstacles in MS
$\nabla e(X_k)$	Jacobian matrix
$\vec{D}(t)$	Relative position vector
A	Set of agents or MAVs
$age(n,k)$	Cell age, discretised MS
$age(x,y,\mathcal{FP},t)$	Age of a point in the mission space, continuous space
c_i	Speciation weighting factors
D_{gene}	Number of disjoint genes
$Dist(n,m)$	Distance between MAV k and cell n
$e(X_k)$	Error vector for current state

E_{critic}	Critical fuel threshold
E_{gene}	Number of excess genes
$E_{m,max}$	Maximum MAV energy level
$E_m(t)$	MAV energy level at time t
E_{thresh}	Low fuel threshold
f	NN fitness (or cost) function
f_i'	Adjusted fitness
$G(N)$	Approximate cellular decomposition of mission space
$g(n)$	Cell in discretised age grid
L_1	Length of MS
L_2	Breadth of MS
L_3	Obstacle length
L_4	Obstacle breadth
M	Number of MAVs
m	MAV index, $m \in \{1, 2, \ldots, M\}$
$minDist$	Minimum distance between any two MAVs
N	Total number of cells in discretised MS
N_{gene}	Number of genes
out_i	NN output i
$P_{depot,q}$	Depot position
$P_m(0)$	Initial position of MAV m
Q	Number of depots
q	Depot index, $q \in \{1, 2, \ldots, Q\}$
Q_{Kalman}	Process noise covariance matrix
R_i	Range between beacon i and a MAV
R_{Kalman}	Measurement noise covariance matrix
S	Safety coefficient
S_{Kalman}	Cross covariance matrix
t	Time
t_f	Final time
t_s	Start time
V	MAV footprint
v_{max}	MAV maximum velocity
$v_{x,m}$	MAV x velocity
$v_{y,m}$	MAV y velocity
x	X position
X_k	Current State
X_{k+1}	Future state
y	Y position
z	Altitude (or position along the z axis)