# Application of Machine Learning and Artificial Intelligence Techniques to Improve Autonomy in Maritime Surveillance Radar Systems

Angus Brown

Submitted in fulfilment of the requirements for the
Degree of Doctor of Philosophy

School of Engineering
College of Science and Engineering
University of Glasgow

April 2022

# Executive Summary

Current maritime radar surveillance missions are typically carried out using an airborne platform with one or more operators on board. The workload of human operators is a bottleneck in surveillance performance as they can only perform on a single platform discontinuously. Additionally, with progress being made towards the use of remotely operated UAVs for radar surveillance missions, an increase in radar operational autonomy is required to maximise the UAV's surveillance potential.

Consequently, the focus of this research is to improve the autonomy of current maritime radar surveillance missions. By reducing the workload of the current radar operator, the surveillance missions can be performed for longer. This research breaks the autonomy of the radar surveillance mission into two aspects: the platform operator autonomy and the radar operator autonomy. However, the implemented autonomous methods must "complement rather than compete with one another".

In order to implement algorithms for the platform operator autonomy and radar operator autonomy, a maritime radar surveillance simulation and user interface is required. Consequently, this work outlines a real-time maritime surveillance radar simulation and graphical user interface which can be used to carry out missions in the same manner as an operator would with a real system.

For the platform operator autonomy aspect, there is a trade-off between maximising information obtained from the surveillance search area and minimising fuel consumption. The research presented here provides an approach for the optimisation of a UAV's trajectory for maritime radar wide area persistent surveillance to simultaneously minimise fuel consumption, maximise mean probability of detection, and minimise mean revisit time. Quintic polynomials are used to generate UAV trajectories due to their ability to provide complete and complex solutions while requiring few inputs. A wide area search radar model is used within this article in conjunction with a discretised grid in order to determine the search area's mean probability of detection and mean revisit time. The trajectory generation method is then used in conjunction with a multi-objective particle swarm optimisation algorithm to obtain a global optimum in terms of path, airspeed (and thus time), and altitude. The performance of the approach is then tested over two common maritime surveillance scenarios and compared to an industry recommended baseline.

In terms of the radar operator autonomy, imitation learning, as opposed to other forms of machine learning, are advantageous as they act in the same manner as the operator, thus reducing the deviation from the current operational standard and allowing for easier system qualification and human operator interaction. The developed radar simulation and interface is used to obtain operator decision data from a human operator. The operator data is then used with two imitation learning methods, namely Bayesian networks and inverse reinforcement learning, with the methods used in place of the operator with their performance compared and their suitability discussed.

# Contents

# List of Tables

# List of Figures

# Acknowledgements

Firstly, I would like to thank my academic supervisor, Dr Dave Anderson, for his support and guidance throughout this research.

I would also like to thank my industrial supervisors, Dr David Greig and Dr Gavin Halcrow. Their assistance allowed for an invaluable insight into current radar operations for which this research would not have been possible without.

Lastly, I would like to thank Dr Andrew Cusick, for being a constant light along an often dark journey.

This work was supported by Leonardo MW Ltd [grant number SELEX/GU/2015/SOW01] and the ESPRC [grant number EP/N509176/1].

# Declaration

“I declare that, except where explicit reference is made to the contribution of others, that this thesis is the result of my own work and has not been submitted for any other degree at the University of Glasgow or any other institution.”

# Impacts of Coronavirus

The original plan was to use radar operators from Leonardo MW ltd, however, coronavirus prevented this. This plan would also have included questions posed to the operator on how well they trusted the autonomous system and how transparent they found the autonomous system. Specifically, a comparison would have been done between the operator's perceived trust and transparency in each algorithm. Results would include the percentage of data instances in which the operator overrode each algorithm. The results would also have included a measure of the operator's understanding of the algorithm's decisions, based on the questions posed. A comparison would also have been done between the chosen imitation learning algorithms and their non-imitation learning counterparts, namely a manually set Bayesian network and reinforcement learning respectively.

# Nomenclature

**Acronyms**

| | |
|---|---|
| AI | Artificial Intelligence |
| AIS | Automatic Identification System |
| ATR | Automatic Target Recognition |
| BN | Bayesian Network |
| DAG | Directed Acyclic Graph |
| DT | Detection Threshold |
| ECEF | Earth-Centred, Earth-Fixed |
| EM | Expectation Maximisation |
| FAR | False Alarm Rate |
| GA | Genetic Algorithms |
| GAIL | Generative Adversarial Imitation Learning |
| GAN | Generative Adversarial Network |
| GUI | Graphical User Interface |
| IRL | Inverse Reinforcement Learning |
| ISAR | Inverse Synthetic-Aperture Radar |
| LAA | Local Area Average |
| MAR | Missing At Random |
| MAVERIC | Modelling of Autonomous VEhicles using Robust, Intelligent Computing |
| MCAR | Missing Completely At Random |
| MDP | Markov Decision Process |
| MNAR | Missing Not At Random |
| MOPSO | Multi-Objective Particle Swarm Optimisation |

| | |
|---|---|
| NED | North East Down |
| PFA | Probability of False Alarm |
| POMDP | Partially Observable Markov Decision Process |
| PPI | Plan Position Indicator |
| PRF | Pulse Repetition Frequency |
| PRI | Pulse Repetition Interval |
| PSO | Particle Swarm Optimisation |
| RC | Radio Controlled |
| RCS | Radar Cross Section |
| RL | Reinforcement Learning |
| SAR | Synthetic-Aperture Radar |
| SIR | Signal-to-Interference Ratio |
| SNR | Signal-to-Noise Ratio |
| SVM | Support Vector Machine |
| TOW | Take-Off Weight |
| TSFC | Thrust Specific Fuel Consumption |
| UAV | Unmanned Aerial Vehicle |

**Bayesian Networks**

| | |
|---|---|
| $\boldsymbol{D}$ | Data set containing $N$ data instances with each data instance being of length $m$ |
| $\boldsymbol{d}_i$ | A data instance of length $m$ from data set $\boldsymbol{D}$ |
| $\theta_{x\|\boldsymbol{u}}$ | Probabilities of node $x$ given the values of the node's parents $\boldsymbol{u}$ |
| $C(x)$ | The number of data instances $\boldsymbol{d}_i$ that satisfy a given event $x$ |
| $P_D(x \mid \boldsymbol{u})$ | Probability of event $x \mid \boldsymbol{u}$ from data set $\boldsymbol{D}$ |

**Inverse Reinforcement Learning**

| | |
|---|---|
| $\gamma$ | The discount factor which weights future rewards |
| $\pi(s)$ | The policy $\pi$ whereby given a state $s$, an action $a$ is returned |
| $A$ | Finite action space with $k$ actions |
| $Q^{\pi}(s,a)$ | The Q-function |
| $R(s)$ | The immediate reward at state $s$ |

| | |
|---|---|
| $S$ | Finite state space with $N$ states |
| $V^{\pi}(s)$ | The value function |

**Physcial Constants**

| | |
|---|---|
| $c$ | Speed of light in a vacuum |
| $g$ | Standard acceleration due to gravity |
| $k_{\mathrm{B}}$ | Boltzmann constant |
| $r_{\mathrm{e}}$ | Radius of the earth |

**Radar**

| | |
|---|---|
| $\delta$ | Radar cross section |
| $\eta$ | Pulse compression factor |
| $\lambda$ | Radar transmission wavelength |
| $\omega$ | Radar scan rate |
| $\Phi_{\mathrm{c}}$ | Radar grazing angle to surface |
| $\psi_B$ | Radar boresight in azimuth |
| $\theta_{\mathrm{c}}$ | Elevation angle from platform to the earth for a given range |
| $\theta_B$ | Radar boresight in elevation |
| $B_{\mathrm{n}}$ | Radar noise bandwidth |
| $F_{\mathrm{n}}$ | Noise figure |
| $G$ | Radar gain |
| $L_{\mathrm{sys}}$ | Radar system losses |
| $N_{\mathrm{B}}$ | Number of bursts in a dwell |
| $N_{\mathrm{d}}$ | Number of pulses in a dwell |
| $N_{\mathrm{p}}$ | Number of pulses in a burst which is the coherent integration factor |
| $P_{\mathrm{c}}$ | Radar clutter power |
| $P_{\mathrm{d}}$ | Probability of detection |
| $P_{\mathrm{n}}$ | Radar thermal noise power |
| $P_{\mathrm{r}}$ | Radar received power |
| $P_{\mathrm{tx}}$ | Radar transmission power |
| $r_{\mathrm{a}}$ | Radius of the earth adjusted for atmospheric refraction |

| | |
|---|---|
| $r_\mathrm{b}$ | Radar range bin width |
| $r_\mathrm{c}$ | Radar range to the earth for a given elevation |
| $r_\mathrm{h}$ | Radar range to the horizon |
| $r_\mathrm{max}$ | Radar maximum range |
| $r_\mathrm{t}$ | Radar range to a given target |
| $T_\mathrm{d}$ | Time to complete a radar dwell |

**Unmanned Aerial Vehicles**

| | |
|---|---|
| $\gamma$ | Platform elevation |
| $\lambda$ | Longitude |
| $\phi$ | Latitude |
| $\psi$ | Platform heading |
| $a_{\parallel}$ | Platform forward acceleration |
| $a_{\perp}$ | Platform centripetal acceleration |
| $D$ | Platform drag |
| $h^\mathrm{p}$ | Platform altitude |
| $M$ | Mach number |
| $m_\mathrm{C}$ | Platform current total mass |
| $m_\mathrm{fuel}$ | Platform fuel mass |
| $m_\mathrm{TOW}$ | Platform take-off mass |
| $T$ | Platform thrust |
| $V$ | Platform airspeed |
| $X$ | X in ECEF coordinates |
| $x^e$ | North in NED coordinates |
| $Y$ | Y in ECEF coordinates |
| $y^e$ | East in NED coordinates |
| $Z$ | Z in ECEF coordinates |
| $z^e$ | Down in NED coordinates |

# Chapter 1

# Introduction

## 1.1 Motivation

### 1.1.1 Increasing Autonomy

Throughout history, humans have looked towards tools and technology for ways of reducing their workload, and in recent years there has been a surge in the implementation of autonomy and artificial intelligence in order to do so. Every industry from healthcare [1], transport [2], and finance [3] has implemented some form of artificial intelligence to their operations. These implementations have increased operational performance, by some metric, relative to humans in their respective industries.

As per Parasuraman and Riley [4], automation is defined as a machine performing a task that was once performed by humans. What is considered automation will therefore change over time and consequently their definition of automation will eventually encompass high-level autonomy. Generally, there are two reasons to automate a task [5]. The first is to remove human error for high-risk tasks. For example, automation has successfully been used to reduce the rate of accidents [6] during the high-risk landing and take-off stages of flight[7]. The second reason for automating a task is to reduce human operator workload.

However, in order for an autonomous system to be used in place of a human operator there are several questions that must be asked. First and foremost, does the autonomous system offer a benefit (e.g. cost or performance) over the human operator? This question encompasses whether or not the autonomous system offers at least equal performance to the human operator. Secondly, does the system perform any roles that are safety or mission critical? Historically, humans have performed roles where there is a concern to the safety of others (e.g. driving a car) or where mission failure would result in some form of loss (e.g. preventing financial loss from illegal fishing operations). Human operators are often afforded margins of errors for these tasks, and while these errors are mitigated with various safety and operational procedures, mistakes are taken to be part of the operational process. For autonomous systems there is an expectation that these operational errors happen far less often. Consequently, the autonomous system must go through rigorous testing and qualification processes to ensure that it meets the operational requirements set by customers and regulations.

Ideally, an artificially intelligent agent would be developed such that the correct decision is always made. In practice, this is not feasible as there will always be some form of error that results in incorrect decisions being made. For an autonomous system performing safety and mission critical tasks, there are concerns such as the political and societal backlash [8] that may occur if the system fails the task. Additionally, if there is to remain a human in the operational loop then how the remaining operator(s) interact with the system influences how the algorithm should be implemented in the first place. If the operator is in conflict with the autonomous agent, despite the agent's high task performance, the overall gain from the agent is negative.

There are certain industries such as defense where, despite successful testing, actually implementing an autonomous system that replaces a human is a long way off [9]. An additional consideration for an autonomous system with an industrial application is the validation and qualification requirements. However, even in the ideal scenario where the AI agent always makes the correct decision, the validation and qualification process will still have to capture the newly implemented technology and its operational behaviour. If there is a large technological leap and/or a significant deviation from the current operational standards, the system may have a far longer and more rigorous process to go through.

This work outlined in this thesis looks at applying machine learning to increase autonomy in a maritime surveillance radar system. It is therefore important that the algorithms considered minimises the concerns outlined in this section while also obtaining a useful increase in performance.

### 1.1.2 Operator and System Interaction

When implementing AI to a sub-system to improve autonomy for the whole system, it is important to ensure that the rest of the system does not have any bottlenecks that will limit the performance of the AI. In the case of radar surveillance, replacing the human radar operator with machine learning algorithms leaves the airborne platform as a bottleneck. Specifically, the machine learning algorithm can only carry out the decision making processes for as long as the airborne platform is able to maintain surveillance on the search area. Since current airborne radar surveillance methods also have a human piloting the airborne platform, the logical step is to then use an autonomous unmanned aerial vehicle (UAV) in place of a human piloted platform.

UAVs have already been employed for radar surveillance missions [10] with the goal of increasing performance in some manner (e.g. reduced running costs, increased surveillance performance, lower operational risks). However, these missions have involved a radar operator acting from a remote location (see section 4.3 for example operator tasks) rather than an autonomous system replacing the human radar operator. Consequently, research has been carried out on the human-UAV interaction [11], specifically the remote operator's interface with the UAV and onboard systems.

For the foreseeable future of autonomous radar systems, it is still expected that a human operator will remain in the loop, even if remotely. The remote operator will be able to view a display similar to those in current operations, and the operator will command the system at a mission level whilst still having the ability to control and/or override the autonomous system at a lower level (i.e. the current radar operator standard). The ability to control the system at a lower level can also be used to train any machine learning aspects of the system. The machine learning aspects of the system can then operate with partial autonomy and eventually full autonomy as the system is further trained and learns from the experience. Consequently, the operator's input is reduced over time, though the operator will also be tasked with monitoring the system in case intervention is required.

However, there is also the question of which type of UAV to use. Since no human operators are on-board, a UAV that is similar to the current maritime radar surveillance platform may not be optimal. Furthermore, given a UAV and radar, there is then the question of what trajectory the UAV should take to obtain the best surveillance performance.

### 1.1.3 Improvements made by Introducing Autonomy

While some tasks are easier to replace than others, recent advancements in machine learning algorithms has allowed even the most complex tasks to be able to be performed autonomously. The advantage of an artificially intelligent agent over a human operator is the potential to carry out tasks faster, longer, simultaneously, more accurately, and at reduced operational costs.

In applying an autonomous system to surveillance radars, some of the current roles of the time-limited and cost-limited operators would be performed by the system. The omnipresent nature of an autonomous agent allows for the agent to operate at any time of the day on any suitable platform, without break. Consequently, the agent can be deployed to increase surveillance time and/or coverage. Additionally, the computational nature of an autonomous agent allows for decisions making far faster than that of a human operator. The increased rate at which information can be processed and decisions made further increases surveillance performance while also providing faster decision making in time-critical scenarios.

For long surveillance missions, radar operators are often tasked with analysing a constant stream of data from surveilling largely featureless areas of sea/ocean. This type of task can result in operators suffering from an overload of information, and the length of doing this type of task can also result in fatigue [10]. In comparison, an autonomous agent removes human elements such as fatigue from the process resulting in a more more constant performance. Additionally, in a study by Ruff [12], autonomous systems have been shown to reduce workload for operators whether the system is fully autonomous or simply suggesting actions to the operator. In the study, the system was able to maintain this performance under situations too complex for the operator to handle.

As surveillance missions move towards UAVs, the concern of network disruptions, or even attacks, become significant. From tests and studies done for remote UAV operations, data droputs and delays were of a critical issue, preventing information getting to the operator on time [10], [13]. In the case of radars, the remote radar operator would lose communication and control of the radar, impairing mission operations. If an autonomous agent was onboard and ready to take-over if a data dropout occurs, the mission operations can continue.

### 1.1.4 Challenges of Introducing Autonomy

After a series of studies by Riley [14], an operator's reliance on automation was found to be influenced by the operator's confidence in their own abilities, the trust in the autonomous system, operator workload and fatigue, and the perceived level of risk. Whilst most of these characteristics are not directly affected by the way in which an autonomous system is implemented, the trust of the system certainly is.

Increasing autonomy in a system further removes the operator from the system and generally reduces the transparency between the system and the operator. This reduction in transparency can be caused by a number of factors, though they are mostly centered around the system's autonomous decision making. Specifically, the system may process information and make decisions faster than a human could, thus an operator would not be able to supervise the autonomous system as it operates. Additionally, the system may not explain each of the decisions it makes to the operator.

The lack of transparency can lead to the operator unable to interpret the system's decisions or the operator may not even notice that decisions are being made. Consequently, the operator might wrongly interpret the current situation thus reducing his situational awareness and reducing his ability to detect failures in the system's operation [15]. In addition to reducing the operator's situational awareness, the lack of transparency can reduce the operator's trust in the system. If the operator has little trust in the system's autonomous operations, they may be more likely to override the system, negating the benefits of autonomy whilst introducing operator confusion. The converse is also true where the lack of transparency leads the operator to believe the autonomous system performs better than it actually does. The operator's overestimation of the system's performance can result in a reduction in system monitoring by the operator, and thus a reduction in detecting system operational errors [16]. If the operator does not detect these system operational errors, not only can the operator not respond with appropriate corrective actions, machine learning algorithms in the system might learn from flawed data, potentially increasing the likelihood that such an error occurs again.

A infamous case of operator over-trust in an autonomous system is the fatal Airbus A330 crash in 1994. The crash investigation committee concluded that the crew appeared over-confident in the autopilot and had the pilot reacted 4 seconds earlier, the accident would have been prevented [17]. Conversely, an infamous case of operator distrust of autonomy is the Chernobyl nuclear power plant accident. One of the primary factors that influenced the accident was the operators ability to manually disable automatic scram trips, certain safety systems, and alarm signals [18]. Additionally, it was found [19] that it is harder for an autonomous system to regain trust from the operator after a failure than it is for the autonomous system to build that trust initially.

It is therefore apparent that both the operator's awareness of the system and the system's transparency are vital in maintaining the correct level of operator trust in the system which results in the individual benefits of the operator and the system being fully achieved whilst minimising their individual disadvantages. As previously stated, the first argument for automating a task is to eliminate human error in high-risk operations. Consequently, it is necessary for the operator to correctly trust the system in order to reduce the human error that remains in the loop.

The second argument for automating a task is to reduce operator workload. For example, a study by Dixon, Wickens, and Chang on pilots controlling UAVs [20] showed that automation allowed the pilots to reallocate their resources to perform higher-level tasks. However, increasing automation with an operator in the loop can actually increase operator workload and also result in reducing both the operator's situational awareness and trust in the system [12]. In the case of the pilots controlling UAVs, the pilots take on a supervisory role which involves monitoring the system over a long period of time which is a role humans generally perform poorly at [21]. If the system is autonomous and the operator is to be used to supervise the system's operations, new problems will arise from shifting the operator to a role which requires long periods of monitoring. In particular, the move to a supervisory role introduces failures from a lack of vigilance and an over-trust in the autonomous system [22]. Research has already been carried out regarding the training of operators to maintain situational awareness in a role that requires long periods of monitoring and would be considered "boring" [23]. Furthermore, if the surveillance is to be persistent then there may be handovers between human operators. In maritime radar surveillance, the information provided to the operator is the culmination of numerous radar scans which varies from scan to scan. There may be issues in the hand-over as the next operator is required to catch up on the information that had previously been built up over time. In the UAV domain, the hand-over between operators has resulted in major accidents [7].

From Ruff et al. [12] management-by-consent refers to an automated system whose actions require consent by the operator before they can be carried out. Conversely, management-by-exception refers to an automated system that carries out actions without requiring the operator's permission, but the operator can still interrupt and override the system's actions. From Ruff's study, it was found that under certain conditions management-by-exception performed worse than management-by-consent, thus negating the benefits provided by increased autonomy. This reduction in performance was largely due to the operator being further removed from the decision-making process in the management-by-exception system, resulting in a lower situational awareness and increased difficulty for the operator making decisions when eventually required. For the manage-by-consent system, the operator's began to second-guess decisions made by the autonomous system since it did not have perfect accuracy, resulting in a higher perceived workload than the manage-by-exception system. In the manage-by-exception system, however, the operators missed those decision errors. Their study concluded by highlighting the importance of an active role for the human operator when increasing autonomy in a complex decision-making processes. Furthermore, they state that the level of automation which is of most benefit depends on how they system interacts with the operator.

In summary, the less transparent an autonomous system is, the less trust the operator will have in the system. And if the operator doesn't trust the system, they will be less likely to use it, negating the benefits of the autonomous system. Additionally, the operator's lack of trust in the system will result in conflict, increasing their workload, thus further negating one of the main benefits of autonomy. The previous section described the potential benefits of an autonomous radar system, but those benefits can only be realised if the human factor issues outlined in this section are carefully considered in the design of such a system. Consequently, in order to gain the most benefit from automating a system, it is necessary to consider the order in which tasks should be automated and to what level [6]. Specifically, the interaction between the human operator and the autonomous radar system should be designed such that the advantages of the human operator and the autonomous radar system are maximised whilst the disadvantages minimised.

### 1.1.5 Levels of Autonomy

While the recent surge in high-level autonomous systems and applications of artificial intelligence have shown their performance benefits, there is often no real consensus on the best way to increase autonomy from an industrial point of view. Specifically, the new systems need to navigate existing regulations and potentially bring these regulations up to a suitable standard. Consequently, the new autonomous system will have to transition between the current operational standard and the end-goal of a fully autonomous system. This is particularly the case for safety critical and mission critical applications where current regulations are strict and rigorous. The levels of autonomy therefore provides a series of transition points between a system with no autonomy and a fully autonomous system.

Generally, levels of autonomy indicate a level where the operator has full governance of the system, a level where automation of low level tasks is introduced to enhance and aid the operator, and a level where autonomous operations are introduced which will, either partially or fully, take over higher level operations. Whilst there are several other definitions for levels of autonomy [24], the definition used in the field of self-driving cars is the most widely used and concise.

There are similarities between the goal of replacing human operator carrying out surveillance radar tasks and that of replacing a human driver with a self-driving car. Relative to an autonomous radar system, self-driving cars are a close real-world application that has made significant steps towards autonomy—while also having similar political and qualification issues outlined above. However, radar surveillance applications typically fall under the military domain, for which there is an even greater concern for not only validation but also the way system performance and behaviour.

Self-driving cars have been one of the pillars of the use of AI in society as well as the focus of much AI research. For automated-driving cars, SAE international's J3016 [25] provides a common definition of the levels of automation that a car can have, ranging from "no automation" to "full automation". The report specifies 6 levels of autonomy (outlined in table 1.1) with the human driver monitoring the driving environment in the first three levels whilst the automated driving system monitors the driving environment in the last 3 levels. These levels have been widely adopted within industry, media, and politics [26].

Report J3016 also provides a few key definitions relating to the automated driving task. A "dynamic driving task" involves the operational (e.g. steering, pedals, and monitoring the vehicle and road) and tactical aspects of the driving (e.g. responding to events, lane changes, turn signals), but not the strategic elements (e.g. determining waypoints and destinations). "Driving mode" is a scenario with requirements for the dynamic driving task (e.g. cruise speed, low-speed traffic jam). "Request to intervene" is a notification by the automated system to the human that they should take over from the system with regards to the dynamic driving tasks.

Table 1.1: SAE Levels of Autonomy

| SAE Level | Description |
|---|---|
| 0 | "No Automation" whereby the human driver performs all aspects of the *dynamic driving task* at all times, even with warning or intervention systems enabled. Additionally, the human must monitor the driving environment at all times. |
| 1 | "Driver Assistance" whereby a system provides assistance to either steering or acceleration under certain driving modes by using driving environment information. An example of such a task would be lane keep assist or adaptive cruise control. In this level, there is an expectation that the human driver perform all remaining aspects of the *dynamic driving tasks* whilst monitoring the driving environment at all times, and that the human is the fallback of the assistance system via a *request to intervene*. |

| | |
|---|---|
| 2 | "Partial Automation" whereby one or more systems provide assistance to both steering and acceleration. In other words, the system can control the steering and accelerating simultaneously. For example, lane keep assist and adaptive cruise control operating simultaneously. Again, there is an expectation that the driver performs the remaining aspects of the *dynamic driving tasks* whilst monitoring the driving environment at all times, and responding to a *request to intervene*. |
| 3 | "Conditional Automation" whereby all aspects of the *dynamic driving tasks* are handled by the system. For this level the system is able to replace the system at monitoring the driving environment with the expectation that the human driver responds to a *request to intervene*. A car with level 3 autonomy should be able to drive by itself but under limitations and certain conditions (e.g. in certain weather or *certain driving modes*). |
| 4 | "High Automation" whereby all aspects of the *dynamic driving tasks* are handled by the system. For this level the system monitors the driving environment, however the system must be able to perform if the human does not respond appropriately to a *request to intervene*. In other words, the driver does not have to take over at any point. However, there are still some restrictions on certain *driving modes* of the system relative to a human driver. An example of a level 4 vehicle would be a local driverless taxi. |
| 5 | "Full Automation" whereby the system has full control over all aspects of the *dynamic driving tasks* under all conditions with all *driving modes* enabled. At this level, the system has the same restrictions that a human driver would have. |

In terms of radar mission tasks, the dynamic driving task could be equated to the radar user interface, monitoring of search area, and response to tracks, but not the determination of the mission itself (i.e. where to search and formulated search strategies). The work presented in this thesis looks at methods that can move the maritime radar surveillance mission from partial automation (level 2) to conditional automation (level 3). Specifically, the current operationally standard has the radar operator assisted through various automatic algorithms (e.g. filtering, tracking), but the operator remains in control of the control aspects, particularly the decision making. If a radar system were to become a level 3 autonomous system, all aspects of the radar mission would have to be handled by the radar system with an operator expected to be able to respond to a "request to intervene".

## 1.2 Operation of an Artificially Intelligent System for Mission Critical Applications

There is a wide choice of machine learning algorithms that could be applied to increase the decision making autonomy for maritime radar surveillance missions. Whilst the obvious choice of algorithm would be the one that obtains the highest accuracy of decision making, the concerns outlined previously need to be heavily considered when choosing the algorithm. As per a report from the DSTO [10]:“The actions of operators and the automated system complement rather than compete with one another". In order to avoid any conflict, it is necessary that the operator is aware of and understands the decisions made by the autonomous system.

Several algorithms that fall into the imitation learning category of machine learning algorithms were chosen for this research. In simple terms, imitation learning learns to perform a task in the same way as the expert demonstrations that it learns from. This section outlines the benefits imitation learning algorithms offer over other machine learning algorithms in regards to their application to radar surveillance autonomy.

For certain processes, data collection can be a difficult and expensive process. This is particularly the case for a maritime radar surveillance mission due to the cost incurred by flying the platform. Furthermore, there is also the chance that on a particular flight, there will be few meaningful interactions for the radar operator to make (e.g. few samples will be collected if there are few vessels at sea on the chosen day). Certain machine learning algorithms that use a “trial and error" approach (e.g. reinforcement learning) which can require numerous trials to obtain even a basic operational performance. Imitation learning approaches use expert demonstrations which typically require far fewer runs, or even no runs at all, to achieve a basic operational standard [27]. Consequently, using an imitation learning approach also reduces, or removes, the required trials leading to faster learning in terms of flight time.

In the case of the Global Hawk UAV, automation was found to be central [7] to many of the human factor issues, as the operators found it difficult to closely monitor the autonomous system for long periods of time. This inattention resulted in a reduction in situational awareness and a decreased ability to handle any faults or failures caused by the autonomous system. Furthermore, the human operator typically monitors areas they would expect to see issues or actions appear, and if the autonomous system does not behave in the same way, the human operator will not pick up any faults as easily [28]. If the operator is part of the autonomous system’s training process as well as monitoring its behaviour post-training, then the operator may be more engaged, thus increasing their situational awareness, as they are taking on an additional role of acting as a teacher. Additionally, the operator may be more likely to determine if the system is behaving correctly if the operator is trying to determine if the system’s behaviour matches their own, rather than if the system’s behaviour is objectively correct.

Returning to concern of data dropouts for remote radar operations. If the autonomous agent acts in the same manner as the human radar operator, then the current operational methods will be maintained during the dropout making the mission operations more synchronous. This benefit also applies to the issue of operator hand-over, as an AI agent can continue operation whilst the next operator reviews the current information.

If the machine behaves differently, even slighty, then this can cause operator confusion which, when there are tasks and decisions that need to be made within seconds, is an overall detriment to the operation. This concern is still the case when an engineer has determined that the machines behaviour is more optimal than the human's behaviour which it replaced. Similarly, if the behaviour of the system deviates too far from the current operational standard, the operator's situational awareness can be reduced. For example, the operator might expect to observe radar tracks in a particular search area, but with the 'new' system, the detections were correctly deemed to be spurious and not therefore not tracked. This changed behaviour may go against the operator's typical understanding of the situation leading to the operator to doubt or mistrust the system [29]. Consequently, in terms of increasing autonomy in radar systems, the main hurdle is building the trust of the operator that the machine will behave in an expected and wanted manner. Additionally, if the AI can be shown to act as per the current operational standard, the validation and qualification processes may be more streamlined [30] as there is less of a change from current methods. Imitation learning is therefore an appealing option as it's operational behaviour is the behaviour expected by the radar operator, thus does not deviate from the current operational standard.

When the autonomous solution is replacing a human, it is nearly always necessary that the autonomous solution offers at least equal performance by some metric. Furthermore, for certain types of tasks (e.g. mission critical applications) it is important that the solution has transparency in the way it operates. Not only is it important for the system to be transparent for the benefit of the operator, it is also important for the system to be transparent to get an accurate view of the decision making process. Specifically, increased system transparency allows for an accurate view of the system's learning leading to the ability to easily diagnose and correct when and where incorrect decisions were made.

## 1.3 Background

### 1.3.1 Machine Learning and Artificial Intelligence

In order to define artificial intelligence (AI), it is first required to define intelligence itself. Whilst AI researchers have provided many varied answers to the question of 'what is intelligence?' [31], a comprehensive definition is provided by Poole, Mackworth, and Goebel [32] as: "[An intelligent agent does what] is appropriate for its circumstances and its goal, it is flexible to changing environments and changing goals, it learns from experience, and it makes appropriate choices given perceptual limitations and finite computation."

As per Poole, Mackworth, and Goebel, AI is then defined as the study of the design of intelligent agents. “The methodology is to design, build, and experiment with computational systems that perform tasks commonly viewed as intelligent." A common misconception is that the goal of AI is to simulate intelligence. In actual fact, “The goal is to understand real (natural or synthetic) intelligent systems by synthesising them. A simulation of an earthquake isn’t an earthquake; however, we want to actually create intelligence, as you could imagine creating an earthquake." In other words, the created AI is to be used functionally within the real world.

Machine learning is a subset of AI that involves the machine utilising algorithms to learn how to preform a task using experience, without being explicitly programmed. As per Mitchell [33], [34], “A computer program is said to learn from experience $E$ with respect to some class of tasks $T$ and performance measure $P$, if its performance at tasks in $T$, as measured by $P$, improves with experience $E$."

The task $T$ is not the process of learning itself, but rather the learning is the means of attaining the ability to perform the task $T$. Machine learning tasks are usually described by how they would process an example, where an example is defined by a collection of quantitively measured features. Common examples include the pixels within an image or, in the radar domain, the measurements (distance, speed, *SNR*) of a target or targets. Additionally, common machine learning tasks include classification, regression, and decision processes.

The performance $P$ of a task $T$ must be quantified. Typically this is done by measuring the accuracy of the program carrying out the task $T$. The accuracy is often simply the proportion of examples for which the program produces the correct output. Note that typically the performance of a program on unseen data is of more interest than the performance on already seen data. Specifically, the performance on unseen data is a better measure of how the program will perform when applied in practice to the real world. As such, the set of data used to test the performance $P$ of the program is usually separate from the data used for training the program [33].

There are a variety of machine learning methods, each suited to their own niche of problems. Figure 1.1 shows some of the machine learning branches and their corresponding algorithm types. Though, in general, problems that are solved using machine learning algorithms can be broken down into the following three categories: supervised learning, unsupervised learning, and reinforcement learning. Furthermore, it is the type experience $E$ that the program learns from that defines which category the learning falls under.

Supervised learning problems experience a labelled data set (where a knowledgable external supervisor labels each example in the data set). Each example (such as the pixels or radar measurements mentioned above) is paired with a label, which indicates the correct action the program should take. The overall objective of the supervised learning program is to be able to respond correctly to any given example. Specifically, this means acting correctly to examples not seen whilst training. The supervised learning problems include the regression and classification problems. From figure 1.1, methods used to solve these types of problems include neural networks, deep learning, regularisation, decision trees, Bayesian methods, and ensemble methods.

Conversely, unsupervised learning problems experience unlabelled datasets. Unsupervised learning algorithms learn useful properties of the structure of this dataset without supervision (i.e. no information is provided beyond the data itself). From figure 1.1, the unsupervised learning problems include the clustering and dimensionality reduction categories.

Lastly, reinforcement learning is a category where the program does not experience a fixed data set. Instead, the algorithm explores a state-action-reward space with aim of maximising a user-defined reward in which there is feedback loop between the program learning and its experience.

### 1.3.2 Optimisation

Optimisation is a field closely related to machine learning with many machine learning algorithms utilising an optimisation sub-routine to learn parameters that optimise the given objective function from the given data. Additionally, certain subsets of optimisation fall under the banner of artificial intelligence. Namely, evolutionary optimisation algorithms (e.g. genetic algorithms (GA)) and swarm intelligence optimisation algorithms (e.g. particle swarm optimisation (PSO)). Chapter 3 provides an in-depth overview of optimisation algorithms.

### 1.3.3 Radar Surveillance Autonomy

Surveillance of a given search area is a task with many useful applications. In a broad sense, example missions include exploring a new area or patrolling an existing area. In comparison to other sensors, radars can operate in low visibility conditions and at long ranges, thus making them ideal for surveillance missions. Consequently, radars have been used in surveillance missions ranging from space, maritime, and air scenarios. The wide variety of applications of surveillance radar highlights their importance in modern society.

For security classification and availability of information, the research presented here focuses on maritime radar surveillance. A typical maritime surveillance mission carried out today involves a manned airborne platform navigating a defined search with a radar operator performing actions based on information observed from the radar display. Specifically, the radar would perform a 360° search scan, pick up detections, then form tracks based on those detections. The radar operator would then use the track information and decide which tracks required further investigation. When a track is investigated, the radar operator uses parameters such as the track's position, velocity, and abnormalities in order to determine whether or not a closer look is required on the target. When a closer look is required, the radar operator requests the manned platform to move towards the target. By moving closer, the radar operator can increase the radar resolution and potentially get a synthetic aperture radar (SAR) image or inverse synthetic aperture radar (ISAR) image on the target. When moving towards a target, there is a point at which the operator has confidence in what the target it and what action is required for the target. This whole process would need to be performed by an autonomous agent replacing the human(s). Section 4.3.3 outlines the process of an autonomous agent performing this task. Currently, applications of machine learning to radar systems have generally been in the form of classification problems, cognitive radar, or in tracking.

Recent developments in convolution neural networks and deep learning that have allowed for human-level performance at image classification tasks [36], and these developments have been applied in many forms to radar classification tasks. For surveillance radar, the classification problems focus on determining target types typically using a synthetic aperture radar (SAR) image [37].

Most radar systems employ a feed-forward processing chain in which they first perform some low-level processing of received sensor data to obtain target detections and then pass the processed data on to some higher-level processor such as a tracker, which extracts information to achieve a system objective. The concept behind cognitive radar [38], [39] is that system performance can be improved using adaptation between the information extracted from the sensor/processor and the design and transmission of subsequent illuminating waveforms. Essentially, information present in a single dwell is insufficient for controlling the radar, and by employing long and short term memory the system can adapt to new environments. Cognitive radar techniques using reinforcement learning [40], [41] have been applied to adaptive waveform as well as improving the tracking and detection process.

However, to the author's knowledge, there has been no work done with regards to replacing the human operator for any type of aerospace application decision process. Furthermore, to the author's knowledge there has been no research into the use of imitation learning as a means of reducing operator mistrust in an autonomous system, and also as a means of streamlining the validation phase of the system. Chapter 4 provides greater insight into the algorithms and applications of imitation learning.

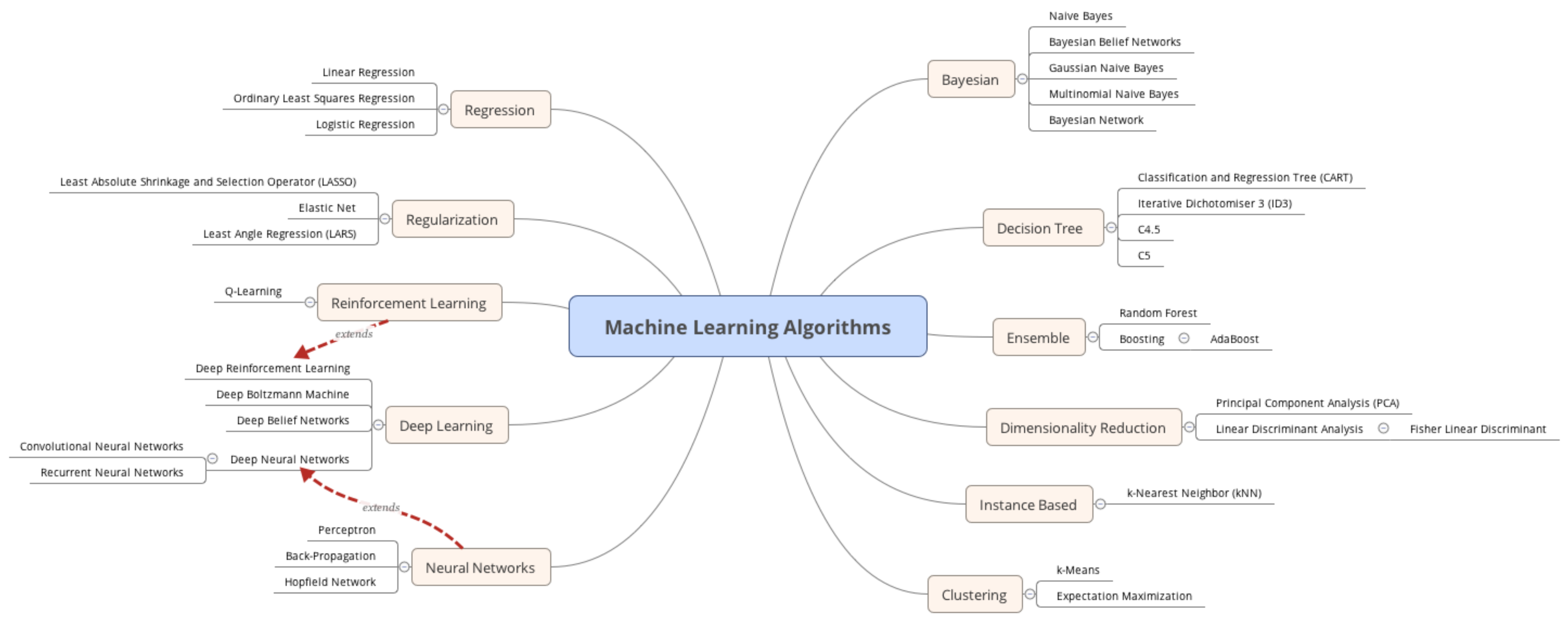


Figure 1.1: A brief overview of machine learning algorithms [35].

## 1.4 Aims

The problem statement of this research is: to increase autonomy in radar surveillance systems to allow for increased surveillance performance. This performance can come in many forms such as surveillance time, surveillance coverage, surveillance reliability, or even the cost of running a surveillance mission. However, increasing autonomy needs to be done in a way that meets the rigorous qualification process, navigates any potential societal or political pressure, and is able to compliment, not compete, with the remaining operator(s) in the loop.

The overall aim of this research is to provide a method in which the task of radar surveillance can move towards a greater level of autonomy. By considering the previously mentioned issues, imitation learning methods provide an appealing solution, one which increases autonomy, compliments the remaining operator in-the-loop, streamlines the qualification process, and potentially navigates any societal or political issues. The imitation learning methods will then be investigated for their suitability as a replacement to a human operator, in terms of decision making accuracy and also their suitability to address the issues outlined.

As previously mentioned, there is the question of what trajectory a UAV should fly for a radar surveillance mission with a remote operator. Whilst there has been work done on UAV trajectory optimisation for radar coverage, to the author's knowledge, none of this work deals with large area surveillance or uses a sufficient radar model within the optimisation. Consequently, another aspect of this research is to develop an algorithm that will determine optimal trajectories for an autonomous UAV carrying out a maritime surveillance radar mission. This algorithm will consider the UAV in terms of dynamics and fuel consumption, and radar surveillance requirements in terms of probability of detection and revisit time.

In order to apply the imitation learning algorithms for radar surveillance, a radar simulation is required. This radar simulation and operator interface is required to replicate the experience that a human operator would have, such that a human operator could then use the simulation in order to obtain training data for the imitation learning algorithms. Once data is obtained, the imitation learning algorithms can be applied and results obtained and conclusions drawn.

## 1.5 Contributions of Research

### 1.5.1 Simulation and GUI for Maritime Radar Surveillance Missions

1. A radar simulation is derived for maritime wide area surveillance. This model includes the power returns, target detections, target tracking, and accounts for the earth's curvature and the radar's sector scan.

2. A radar GUI is developed to run complex maritime surveillance missions with the ability to capture both the scenario information and the human operator's decisions.

### 1.5.2 Trajectory Optimisation for Radar Surveillance

1. Firstly, a polynomial trajectory generation method is derived that provides complex trajectories while requiring few inputs. This method accounts for the fixed-wing platform dynamics and propulsion while also accounting for the requirements of a persistent surveillance mission (i.e. the start pose is equal to the end pose).

2. A method for determining the search area coverage in terms of probability of detection and revisit time is outlined.

3. The considerations of the required platform for this mission type are accounted for. These considerations include the high altitude region of operation, long mission time, and high fuel to weight ratio of the platform.

4. Lastly, the surveillance trajectory is used with a multi-objective global optimisation algorithm. Results are compared to an industry recommended baseline which shows the performance of this method. Additionally, the geometric similarity of trajectories within the Pareto front are highlighted.

### 1.5.3 Imitation Learning for Radar Surveillance

1. Firstly, the requirements of an autonomous radar system at the operational level are defined. Primarily, this includes the required behaviour of radar as well as the interaction between the autonomous radar and the operator.

2. Three typical maritime surveillance radar tasks, representing various stages of the surveillance process, are outlined and setup within the simulation.

3. Two imitation learning approaches, namely Bayesian Networks and Inverse Reinforcement Learning, were implemented to imitate the radar operator's decision making process from the collected operator data.

4. The two approaches are then compared relative to each other in terms of accuracy of decisions made relative to the operator.

The following publications have resulted from the work shown in this thesis:

- A. Brown and D. Anderson, “Trajectory optimization for high-altitude long endurance UAV maritime radar surveillance," IEEE Transactions on Aerospace and Electronic Systems, 2019. DOI: 10.1109/TAES.2019.2949384. [42]

- A. Brown and D. Anderson, “Imitating Radar Operator Decisions for Maritime Surveillance Missions Using Bayesian Networks." 2019 International Radar Conference, Toulon, France, 23-27 Sept 2019. DOI: 10.1109/RADAR41533.2019.171229. [43]

## 1.6 Thesis Outline

Chapter 2 outlines the real-time radar simulation and user interface developed. This includes an outline of the radar mathematical model, specifically the equations used for the radar returns, detection, and tracking. Additionally, the outline includes an overview of the graphical user interface (GUI) that the operator uses to interact with the radar simulation and thus carry out typical maritime surveillance radar missions.

Chapter 3 describes the first step in improving autonomy for maritime surveillance radar. This chapter outlines an algorithm developed to obtain optimal trajectories for wide area persistent maritime surveillance using a radar on board a high-altitude long endurance platform. Specifically, the trajectory generation method is formulated, a method of obtaining the probability of detection and revisit time for a given trajectory is shown, and the optimisation problem is mathematically defined. The simulation results for the optimisation of the trajectory is then presented.

Chapter 4 gives a literature review of imitation learning and specifies the choice of algorithms and why they were chosen. Additionally, the tasks which the human operator perform for data collection are outlined in this chapter.

Chapter 5 outlines the specific Bayesian network algorithm chosen and the reasoning behind the choice. A derivation of the algorithm is also presented with assumptions noted. Additionally, this chapter states how the tasks were represented within the Bayesian network and how the network is implemented within the radar simulation.

Chapter 6 outlines the specific inverse reinforcement algorithms chosen and the reasoning behind the choice. A derivation of each algorithm is also presented with assumptions noted. Additionally, this chapter states how the tasks were represented within the framework of a Markov Decision Process (MDP) and how the algorithms were implemented within the radar simulation.

Chapter 7 presents the results of the algorithms and compares their performance in terms of how well they imitate the actions of a typical maritime radar operator. The results are also discussed in terms of the algorithms suitability for operation.

Chapter 8 summarises the thesis and conclusions. Additionally, avenues of potential future work are suggested.

# Chapter 2

# Radar Modelling

## 2.1 Introduction

Maritime surveillance radar operations are complex and often require an operator to control the radar system with the aide of a graphical user interface (GUI). This chapter outlines a representative real-time maritime surveillance radar simulation in conjunction with a GUI which is used to control the radar system, and consequently perform the surveillance mission. This simulation allows for the testing of such a maritime radar system, or even a supplementary system, which would otherwise require not only the radar system itself, but a platform and operational environment.

This simulation was developed and integrated as part of the MAVERIC (Modelling of Autonomous VEhicles using Robust, Intelligent Computing) simulation engine [44], [45] for the purpose of simulating real-time maritime radar surveillance missions. MAVERIC allows for the simulation of multiple entities (e.g. UAVs, helicopters, and boats) within a set 3D operational environment. Each entity can contain multiple sub-entities (e.g. sensors and payload) with the ability for both the entity and the sub-entities to have simulation models of varying fidelity. MAVERIC, and consequently this simulation, was developed in C++ with the use of Qt [46] for the graphical components. Note that the simulation and GUI were tested with a i7-6700 processor and 16GB of RAM.

The real-time radar simulation includes power returns from targets with the use of the radar range equation, an approximated beam pattern, and the curvature of the earth accounted for. Sea clutter is also modelled by considering the grazing angle to the surface and using a 'constant gamma' model. Targets are then detected using both local area average (LAA) and binary integration, after which a Kalman filter is used to track the detected targets. Furthermore, the entities under surveillance (i.e the boats) have the ability to contain a transponder. The operator can then fuse the radar tracks with the transponder data to obtain further information—or lack of information—on the track.

In addition to the radar simulation, a graphical user interface (GUI) was simultaneously developed. This interface allows for an operator to carry out a maritime radar surveillance mission in a similar way to an operator using an actual radar system within a real environment. The GUI contains a plan position indicator (PPI) to display the radar returns after LAA detection has occurred. The current tracks are overlayed on the PPI display with the operator able to select a track resulting in its information being displayed. The GUI also contains radar controls to allow, for example, the operator to set the range resolution and/or the angular position and width of the radar sector scan. For the maritime radar missions considered here, the platform is airborne and mobile, and in an actual mission, the platform position would be influenced by the radar operator (e.g. requesting a direction of movement or requesting movement to a specific coordinate). Consequently, the simulation developed here also allows for the operator to input guidance commands to change the platform's trajectory.

### 2.1.1 Main Contributions

The contribution of this chapter is the definition of a real-time maritime surveillance radar simulation, display and interface. Specifically, the contributions are as follows:

1. A radar simulation is derived for maritime wide area surveillance. This model includes the power returns, target detections, target tracking, and accounts for the earth's curvature and the radar's sector scan.
2. A radar GUI is developed to run complex maritime surveillance missions with the ability to capture both the scenario information and the human operator's decisions.
3. This radar simulation and GUI builds upon and contributes to the MAVERIC framework. Consequently, any number of radar simulations and GUIs can be deployed within a MAVERIC scene containing multiple autonomous and human agents.

## 2.2 Simulation Outline

### 2.2.1 Overview

Fig. 2.1 shows an overview of the radar simulation and interface. The core of the simulation lies within the 'Scene' component. This component houses all of the entities including the radar and its platform.

The 'Radar Power Returns' component requests the absolute entity positions from the 'Scene' in order to ascertain the entity positions relative to the radar for the radar power measurement calculations. These power measurements are then passed through several components ('LLA Detection' and 'Binary Integration') in order to determine target detections. The target detections are then passed to the Kalman filter tracker where the 'Associate Track' component either creates a new track or requests an update to an existing track via the 'Kalman Filter

Update' component. The 'Kalman Filter Predict' component provides a track pose estimate for the current simulation time which is then passed to the 'GUI' component.

The 'GUI' component displays the radar tracks in addition to the radar LAA detections. Additionally, the 'Scene' component provides the 'GUI' component with the entities' transponders. These can then be observed by the operator and the information used in conjunction with the radar data.

The operator's main interface is with the GUI. The operator's radar commands (e.g. changing the radar sector scan) are passed to the 'Radar Power Returns' component where the radar system is updated. Additionally, the operator's platform commands (e.g. changing the platform direction) are passed to the 'Platform Guidance' component. This component determines the required guidance commands which are then passed to the 'Platform Control' component. This component determines the required platform control inputs in order to achieve the guidance commands.

Lastly, the 'Platform Dynamics' component uses the control inputs from the 'Platform Control' component to calculate the platform pose at the current simulation time. This pose is then used to update the scene. Note that the entities beside the radar platform also have their own guidance, control, and dynamics which update the scene. These components were hidden from the diagram for simplicity and are not discussed in detail within this thesis. Information on the other vehicle guidance, control, and dynamics can be found in the MAVERIC documentation [44], [45].

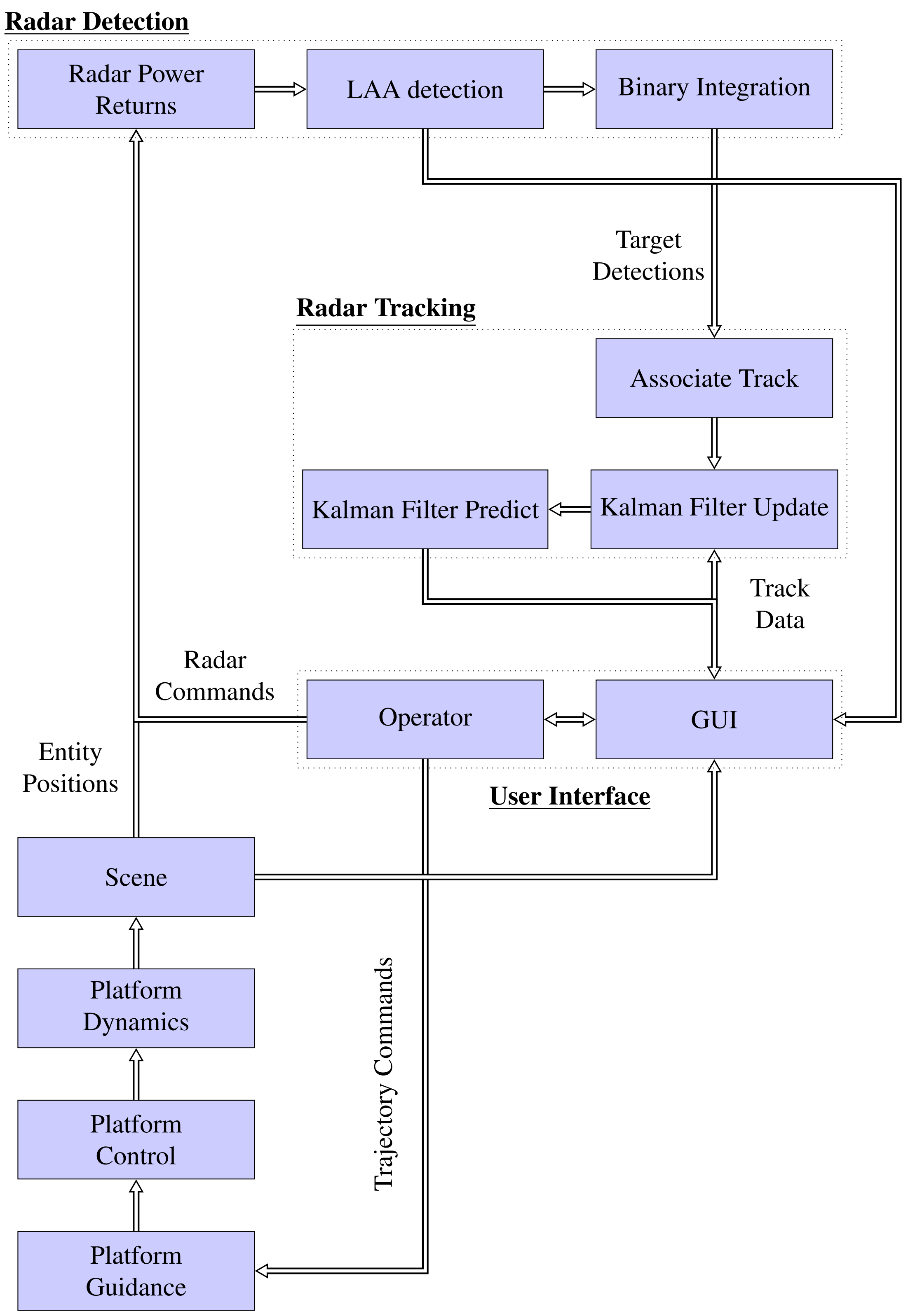


Figure 2.1: Overview of the radar simulation and interface within MAVERIC.

### 2.2.2 Scenario

It is assumed that the operational environment is completely at sea with the earth assumed to be spherical with a radius $r_e = 6318137\,\text{m}$. Furthermore, it is assumed that the scene is initialised at a latitude of $\phi_0 = 0\,\text{rad}$ and a longitude of $\lambda_0 = 0\,\text{rad}$. The scene consists of $N$ targets (where in this case all the targets are maritime vessels) and a single UAV containing the radar system (though any number of radar system and platform pairs can exist within the scene). Thus, there are $M$ entities (where $M = N + 1$) within the scene. The entities' dynamic controls consist of a commanded velocity, heading, and elevation which is used to update the entity's position in north east down (NED) coordinates. In the case of the vessels, the elevation angle and commanded elevation angle are assumed to be 0 rad while the platform height is assumed 0 m (i.e. sea level).

A UAV is are considered as the radar platform for this research, and UAVs typically operate at high altitudes. Consequently, the variation in speed of sound, air density, and air temperature with altitude needs to be accounted for as these properties affect various aspect of the overall system. For example, the atmospheric temperature affects the radar's thermal noise, and the air density affects the UAV's aerodynamics. Within the simulation, the variation in temperature follows the International Standard Atmosphere (ISA) model and as a result varies linearly from 288.15 K at sea level to 216.65 K at an altitude of 11000 m—with the temperature assumed constant until an altitude of 20000 m. Note that it also assumed that the platform's altitude does not exceed 20000 m.

The state of entity $m$ is given by $\boldsymbol{x}_m = [x^{\mathrm{e}}_m, y^{\mathrm{e}}_m, z^{\mathrm{e}}_m, V_m, \psi_m, \gamma_m]$. where $x^{\mathrm{e}}_m$, $y^{\mathrm{e}}_m$, $z^{\mathrm{e}}_m$ are the north east down (NED) coordinates of the platform. $V_m$ is the forward speed of the entity, while $\psi_m$ and $\gamma_m$ are the heading and elevation angles respectively. The state derivatives are then obtained as

$$\dot{\boldsymbol{x}}_m = \begin{bmatrix} V_m \cos(\gamma_m)\cos(\psi_m) \\ V_m \cos(\gamma_m)\sin(\psi_m) \\ -V_m \sin(\gamma_m) \\ (V_{\mathrm{c}_m} - V_m)/V_{\tau_m} \\ (\psi_{\mathrm{c}_m} - \psi_m)/\psi_{\tau_m} \\ (\gamma_{\mathrm{c}_m} - \gamma_m)/\gamma_{\tau_m} \end{bmatrix}, \tag{2.1}$$

where, for entity $m$, $V_{\tau_m}$ is the velocity time constant and $V_{\mathrm{c}_m}$ is the commanded forward speed. Similarly, $\psi_{\tau_m}$ and $\gamma_{\tau_m}$ are the heading and elevation time constants respectively, whilst $\psi_{\mathrm{c}_m}$ and $\gamma_{\mathrm{c}_m}$ are the commanded heading and elevation angles respectively. Note that a time constant characterises the response of an input to a first order system.

Given a simulation time step $\Delta t$, the state of entity $m$ at time step $k$ can then be updated with the use of Euler's integration method as follows:

$$\boldsymbol{x}_{\mathrm{m}_{k+1}} = \boldsymbol{x}_{\mathrm{m}_k} + \dot{\boldsymbol{x}}_{\mathrm{m}_k} \Delta t. \tag{2.2}$$

The radar dynamics consist of a scan rate $\dot{\omega}$. Similarly to the entity dynamics, the radar boresight azimuth position at time step $k$ is updated as $\psi_{\mathrm{B}_{k+1}} = \psi_{\mathrm{B}_k} + \dot{\omega}\Delta t$. It should be noted that when sector scan is in use, the radar azimuth angular position instantaneously moves to the lower sector scan angle once the upper sector scan limit is reached.

A commonly used feature of maritime radar systems is the incorporation of transponders. A transponder is on board a maritime vessel which relays information such as the vessel type and position to the radar system. Within the simulation, there is the option for each maritime vessel to have–or not have–a transponder on board. In the case that a vessel has a transponder, the transponder then continually transmits the current state $\boldsymbol{x}_m$ of the vessel.

Lastly, each vessel is assumed as a Swerling 1 target [47], consequently, the radar cross section (RCS) of the target changes on a scan-to-scan basis. Thus, each vessel has a mean and standard deviation for its RCS which is used to update the current RCS on each new radar scan. Williams, Cramp, and Curtis [48] outline the typical RCS range for various maritime vessels, and the results from this study were used to set the RCS values of the vessels within this simulation, depending on their size and type.

### 2.2.3 Radar Power Returns

For a radar performing a search on a sea surface, the radar is pointed towards the sea surface and at a set interval emits a series of bursts of energy, also known as a dwell, in the form of radio waves. After a time, the emitted bursts bounce off either the sea surface or an entity at sea, and return to the radar. The length of time it takes the bursts to return to the radar and the energy with which they return is dependant on several factors. These factors broadly include the range to point at which the burst reflected, how reflective the object which the burst intersected was, and how easily the burst was able to pass through the atmosphere at time of emission.

The dwell time of a radar is the time the radar beam is on a given target. In other words, it is the time it takes for the radar to scan an angular width equal to the azimuth 3 dB beamwidth. This can be stated as $T_\mathrm{d} = \psi_{3\,\mathrm{dB}}/\dot{\omega}$. Within a dwell, several bursts of pulses are transmitted by the radar with the pulse repetition interval equated as $PRI = (2r_\mathrm{max})/c$, where $c$ is the speed of light (taken to be $3e8\,\mathrm{m\,s^{-1}}$). Therefore, the number of pulses within a dwell is obtained as $N_\mathrm{d} = T_\mathrm{d}/PRI$. Furthermore, the number of pulses in a burst is given by $N_\mathrm{p}$ which is also the coherent integration factor. Therefore, the number of bursts in a dwell is simply $N_\mathrm{B} = N_\mathrm{d}/N_\mathrm{p}$. As a result, the time step of the radar simulation is set to the time taken to complete one burst: $\Delta t = N_\mathrm{p}PRI$.

The radar within the simulation was based on a typical maritime surveillance radar [49] with the specifications for this radar stated in table 2.1. For maritime surveillance radars, the main operational mode is 'search and track'. The aim of this mode is to search a defined area on the sea surface and form tracks on any detections. Additionally, tracks in this mode are only updated with repeat detections from the search scan, so no additional antenna time is dedicated to the tracks and the radar maintains a continuous rotation. Note that this mode is non-coherent and thus has no doppler processing. For this simulation, 'search and track' is the primary mode used, though inverse synthetic aperture radar (ISAR) is partially considered (and outlined in section 4). ISAR allows for a high-resolution image to be formed using a target's motion, thus giving the operator a visual on the target and allowing for target

Table 2.1: Radar specifications

| Parameter | Unit | Value |
|---|---|---|
| Power, $P_{\text{tx}}$ | W | 3000 |
| Wavelength, $\lambda$ | m | 0.031 |
| Antenna height, $A_h$ | m | 0.33 |
| Antenna length, $A_l$ | m | 0.875 |
| Pulse compression factor, $\eta$ | | 185.2 |
| Coherent integration factor, $N_{\text{p}}$ | | [128 / 32 / 16] |
| Noise figure, $F_n$ | dB | 7 |
| System losses, $L_{\text{sys}}$ | dB | 7.5 |
| Probability of false alarm, $PFA$ | | $1e-6$ |
| Maximum range, $r_{\text{max}}$ | km | [18.52 / 92.6 / 185.2] |
| Range bin, $r_{\text{b}}$ | m | [10 / 50 / 100] |
| Noise Bandwidth, $B_n$ | Hz | $[15 / 3 / 1.5] \times 10^6$ |

recognition.

Rather than simulate the radar emissions and their reflections within the environment, it is typical to use the radar range equation for both simplicity and real-time requirements in order to calculate the radar power returns. The radar equations outlined in this section are based on basic principles of radar [50]–[53].

The power returned from a target is partially dependant on the range, elevation from radar boresight, and azimuth from radar boresight. In order to obtain these variables, the NED coordinates of both the platform and the target are converted into earth-centered, earth-fixed coordinates (ECEF). For a given entity $m$, the ECEF coordinates (denoted by $X_m$, $Y_m$, and $Z_m$) are converted from the NED coordinates as follows:

$$\begin{aligned} X_m &= (r_{\text{a}} - z^{\text{e}}_m)\cos(x^{\text{e}}_m/r_{\text{a}} + \phi_0)\cos(y^{\text{e}}_m/r_{\text{a}} + \lambda_0)\,, \\ Y_m &= (r_{\text{a}} - z^{\text{e}}_m)\cos(x^{\text{e}}_m/r_{\text{a}} + \phi_0)\sin(y^{\text{e}}_m/r_{\text{a}} + \lambda_0)\,, \\ Z_m &= (r_{\text{a}} - z^{\text{e}}_m)\sin(x^{\text{e}}_m/r_{\text{a}} + \phi_0)\,, \end{aligned} \tag{2.3}$$

where the commonly used ‘four-thirds earth model’ is used to account for atmospheric refraction. Thus, for the radar equations the value of the radius of the earth is given by $r_{\text{a}} = (4/3)r_{\text{e}}$.

For brevity, the majority of the remaining derivation for obtaining the range and azimuth from boresight is summarised instead. The platform position in ECEF coordinates and in latitude, longitude, and height can be used to translate and rotate the target ECEF coordinate to obtain a local Cartesian position (denoted by $\Delta X$, $\Delta Y$, and $\Delta Z$). This local Cartesian coordinate can then be used to obtain the range to target $n$ from the platform as follows:

$$r_{\text{t}_n} = \sqrt{(\Delta X_n)^2 + (\Delta Y_n)^2 + (\Delta Z_n)^2}\,. \tag{2.4}$$

The azimuth from the target is then obtained as

$$\Psi_{t_n} = -(\psi_B + \psi_p) + \arctan2(\Delta Y_n, \Delta Z_n), \tag{2.5}$$

where $\psi_B$ is the current azimuth of the radar boresight and $\psi_p$ is the heading angle of the platform.

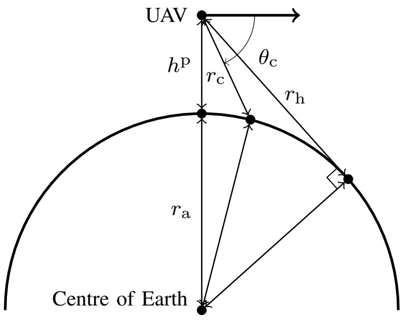


Figure 2.2: Geometry of UAV radar boresight with respect to Earth. $r_a$, $r_h$, and $r_c$, are the radius of the earth (accounting for atmospheric refraction), the slant range to the horizon, and the slant range for a given elevation $\theta$ from the UAV respectively. Furthermore, $h^p$ is the platform altitude equal to $-z^e$.

Obtaining the elevation from boresight requires the use of the geometry outlined in Fig. 2.2. Firstly, the slant range to the horizon $r_h$ is simply obtained using Pythagoras' theorem as:

$$r_h = \sqrt{(r_a + h^p)^2 - r_a^2}. \tag{2.6}$$

Continuing, if the slant range $r_c$ to a point on the ground is known then the elevation angle from the UAV to that point can be obtained with the use of the cosine rule:

$$\theta_c = -\arcsin\left(\frac{r_c^2 + (r_a + h^p)^2 - r_a^2}{2r_c(r_a + h^p)}\right), \tag{2.7}$$

where it should be noted that elevation is taken as negative in the downward direction. Alternatively, (2.7) can be rearranged and solved for $r_c$ using the quadratic formula. As a results, for a given elevation angle $\theta_c$ the slant range to a point on the ground can be calculated as

$$r_c = -(r_a + h^p)\sin\theta_c - \sqrt{(r_a + h^p)^2 \sin^2\theta_c - r_h^2}. \tag{2.8}$$

For surveillance operations, the boresight of the radar beam is usually set such that it maximises the area of coverage. This is done by pointing the lower edge of the 3 dB beam in elevation at the maximum range. By setting $r_c = r_{\max}$ (and accounting for the beamwidth), equation (2.7) can be used to set the boresight for maximum area for a given altitude as follows:

$$\theta_B = -\frac{\theta_{3\,\mathrm{dB}}}{2} - \arcsin\left(\frac{r_{\max}^2 + (r_a + h^p)^2 - r_a^2}{2r_{\max}(r_a + h^p)}\right). \tag{2.9}$$

For target $n$, the elevation from boresight can then be obtained with further use of equation

(2.7). This is simply obtained by setting $r = r_{\mathrm{t}_n}$ and subtracting the radar boresight elevation:

$$\Theta_{\mathrm{t}_n} = -\theta_{\mathrm{B}} - \arcsin\left(\frac{r_{\mathrm{t}_n}^2 + (r_{\mathrm{a}} + h^{\mathrm{p}})^2 - r_{\mathrm{a}}^2}{2r_{\mathrm{t}_n}(r_{\mathrm{a}} + h^{\mathrm{p}})}\right). \tag{2.10}$$

The radar cross section $\sigma$ is a measure of how much radar energy an object reflects back to the source. The factors that affect the radar cross section include the object's size, materials, and angle relative to the source. For the scenarios considered here, a given target $n$ is assumed to have a radar cross section $\sigma_n$ of $100\,\mathrm{m}^2$.

The radar gain $G$ is a function of elevation and azimuth from radar boresight [53]. An approximation for the radar gain in both azimuth and elevation is made by firstly scaling the elevation from boresight (assuming the radar's length is greater than its height) as follows: $\hat{\Theta}_{\mathrm{t}_n} = \Theta_{\mathrm{t}_n}(A_h/A_l)$. Another approximation is made by taking the hypotenuse of both angles: $\Gamma_{\mathrm{t}_n} = \sqrt{\hat{\Theta}_{\mathrm{t}_n}^2 + \Psi_{\mathrm{t}_n}^2}$. The radar gain $G$ is then approximated as

$$G(\Gamma_{\mathrm{t}_n}) = \begin{cases} \frac{4\pi A_l A_h}{\lambda^2}\operatorname{sinc}^2(\pi\frac{A_l}{\lambda}\sin(\Gamma_{\mathrm{t}_n})), & \text{if } \Gamma_{\mathrm{t}_n} < \frac{\pi}{4} \\ 0, & \text{otherwise} \end{cases} \tag{2.11}$$

Note that angles $\Gamma$ further than $45^\circ$ from the boresight are not considered to reduce computational load.

The radar used here is assumed to have a rectangular antenna, and thus the beamwidth in elevation and azimuth is approximated with $\theta_{3\,\mathrm{dB}} = 0.88(\lambda/A_h)$ and $\psi_{3\,\mathrm{dB}} = 0.88(\lambda/A_l)$ respectively. The radar sensor is represented using the radar range equation and consequently, the power received by the radar from target $n$ is given by

$$P_{\mathrm{t}_n} = \frac{\eta N_{\mathrm{p}}^2 P_{\mathrm{tx}} G^2(\Gamma_{\mathrm{t}_n}, \Psi_{\mathrm{t}_n})\sigma_n\lambda^2}{(4\pi)^3 L_{\mathrm{sys}} r_{\mathrm{t}_n}^4}, \tag{2.12}$$

Power emitted by the radar is returned from all reflective surfaces. In addition to any maritime vessels, power is also returned from the sea surface, known as sea clutter. The clutter power is obtained by discretising the surface in both range and azimuth to create cells for which the returned power can be calculated for each range bin $r_{\mathrm{b}}$. The step size in range is simply the range bin width $r_{\mathrm{b}}$ while the step size in azimuth is the azimuth beamwidth $\psi_{3\,\mathrm{dB}}$. To reduce computational load, the number of steps in azimuth $n_\psi$ was set to 9 (i.e. 4 steps on either side of the beam centre). Furthermore, range cells less than the platform height $h^{\mathrm{p}}$ were ignored.

The range value of the radar range bins start at $r_{\mathrm{b}}$ and end at $r_{\max}$ resulting in a total of $n_{\mathrm{r}}$ range bins. This is also the case for the radar clutter cells where a given clutter cell range $r_{\mathrm{c}}$ can be used to obtain the area covered on the ground by that cell. This area is then obtained by accounting for the curvature of the earth as

$$A = \frac{\psi_{3\,\mathrm{dB}}}{2}\left((r_\mathrm{c}+r_\mathrm{b})^2 - r_\mathrm{c}^2\right)\frac{r_\mathrm{a}}{r_\mathrm{a}+h^\mathrm{p}}\,. \tag{2.13}$$

Besides range, another significant factor that determines the returned surface clutter power is the grazing angle $\Phi$ to that surface. The grazing angle (also known as the angle of incidence) is defined as the angle to the surface along slant range $r_\mathrm{c}$. Further use of Fig. 2.2 provides the following equation for a clutter cell indexed in range and azimuth with $u$ and $v$ respectively:

$$\Phi_{\mathrm{c}_{uv}} = -\arcsin\left(\frac{r_{\mathrm{c}_{uv}}^2 - (h^\mathrm{p}+r_\mathrm{a})^2 + r_\mathrm{a}^2}{2r_{\mathrm{c}_{uv}}r_\mathrm{a}}\right). \tag{2.14}$$

For the power returned by the surface clutter, the 'constant gamma' model [53] is used to approximate the radar cross section of each clutter cell using the grazing angle:

$$\sigma_\mathrm{c}(\Phi_\mathrm{c}) = \gamma A \sin(\Phi_\mathrm{c})\,, \tag{2.15}$$

where $\gamma$ is a value from a Gaussian distribution dependant on the sea state. Within the simulation, the sea state can vary from dead calm to stormy waters, which, in terms of the distribution results in a mean that varies from $1e-5$ to $1e-4$ and a standard deviation that varies from $2.5e-6$ to $2.5e-5$.

In a similar manner to the target power, the clutter power is obtained as:

$$P_{\mathrm{c}_{uv}} = \frac{\eta N_\mathrm{p}^2 P_\mathrm{tx} G^2(\Theta_{\mathrm{c}_{uv}},\Psi_{\mathrm{c}_{uv}})\sigma_{\mathrm{c}_{uv}}(\Phi_{\mathrm{c}_{uv}})\lambda^2}{(4\pi)^3 L_\mathrm{sys} r_{\mathrm{c}_{uv}}^4}\,, \tag{2.16}$$

where $\Theta_{\mathrm{c}_{uv}}$ is obtained with the use of equation (2.10). $\Psi_{\mathrm{c}_{uv}}$ is dictated by the clutter cell index number (for example, if $n_\psi = 9$ and $v = 5$, then the azimuth from boresight would equal 0 rad).

The total clutter power for each range bin is obtained by summing along the azimuth of each range bin to form the following vector:

$$\boldsymbol{P}_\mathrm{c} = \sum_{v=1}^{n_\psi} P_{\mathrm{c}_{uv}}\,. \tag{2.17}$$

The power received by the radar is also susceptible to thermal noise within the receiver itself. This incoherent noise power is formulated as

$$\boldsymbol{P}_\mathrm{n} = -N_\mathrm{p} k_\mathrm{B} T B_n F_n \log_e(1-\boldsymbol{p})\,, \tag{2.18}$$

where $\boldsymbol{p}$ is a vector containing $n_\mathrm{r}$ uniform random numbers with interval $[0,1]$. $k_\mathrm{B}$ is Boltzmann's constant with value $1.38064852e-23\,\mathrm{J\,K^{-1}}$ and $T$ is the current air temperature at the platform's altitude.

The vector containing the total power received by the radar for each range bin for a given burst is therefore obtained with:

$$\boldsymbol{P}_{\mathrm{r}} = \sum_{n=1}^{N} \boldsymbol{P}_{\mathrm{t}_m} + \boldsymbol{P}_{\mathrm{c}} + \boldsymbol{P}_{\mathrm{n}} \,. \tag{2.19}$$

## 2.2.4 Detection

In order to discern targets from the sources of noise (i.e. clutter or receiver noise), the power received by the radar requires post processing. The simplest way is to introduce a threshold for which a detection can be declared if a power value exceeds this threshold. There will still be spikes in noise that will exceed the threshold value, but the value can be set to achieve a particular false alarm rate (*FAR*). The false alarm rate is related to the probability of the power recieved from any range cell exceeding the threshold over the dwell time:

$$FAR = \frac{n_{\mathrm{r}} PFA}{T_d} \,, \tag{2.20}$$

where *PFA* (probability of false alarm) is the probability that a single range cell exceeds the threshold.

Assuming the noise fluctuation is of a Rayleigh distribution, a detection is then declared if the received power $P_{\mathrm{r}_u}$ for range cell $u$ is greater than the threshold $DT_u$. The threshold for every range cell is obtained as [50]:

$$DT = -\bar{\boldsymbol{P}} \log_e(PFA) \,, \tag{2.21}$$

where $\bar{\boldsymbol{P}}$ is the vector containing the interference level for each range cell obtained with a local area average (LAA). For range cell $u$, the local area average takes the average $P_{\mathrm{r}}$ of the surrounding cells excluding the cell under test (i.e. range cell $u$) and the guard cells. The number of surrounding cells was taken to be 3 in each direction. The guards cells are simply the cells directly neighbouring the cell under test, and in this case 1 guard cell in each direction was chosen. This is illustrated in Fig. 2.3.

As a result, the detected signal-to-interference ratio *SIR* for range cell $u$ after post processing is given by

$$SIR_u = \begin{cases} P_{\mathrm{r}_u}/\bar{P}_u, & \text{if } \bar{P}_u > DT_u \\ 0, & \text{otherwise} \end{cases} \tag{2.22}$$

The *PFA* affects the LAA threshold which determines whether or not a given radar bin's *SIR* is considered a detection. A high *PFA* will result in a lower threshold and thus more receiver noise and clutter will be detected, but there is also more potential for targets to be detected. Conversely, a low *PFA* results in less receiver noise and clutter being detected at the cost of a decreased chance of a potential target detection.

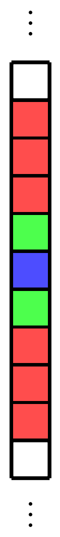


Figure 2.3: Diagram of local area average detection where each cell represents a range bin. The blue cell is the cell under test, the red cells are the surrounding cells, and the green cells are the guard cells.

For surveillance operations, tracking and predicting a specific target's path is of significant use. One use, for example, could be determining if a target is heading towards a zone that is off limits. Accurate and precise target measurements are therefore required to maintain an accurate track, allow for missed detections, and reduce the false alarm rate. Whilst the LAA threshold reduces the PFA, detections from non-target sources will still be declared. A further detection method is therefore applied to the detections being passed to the tracker in order to reduce spurious tracks being formed.

Since several bursts are transmitted in a dwell, several target detections can also be made during the dwell. As previously mentioned, there are $N_\mathrm{B}$ bursts in a dwell. The 'binary integration' method can then be applied across this dwell to further refine the detections for the tracker. This method results in a detection being passed to the tracker if it has been detected using the LAA method $M$ out of $N_\mathrm{B}$ times.

Before the detections from the 'binary integration' method are passed to the tracker, the measurements in azimuth can be further refined to improve tracking accuracy. As the radar steps in azimuth, it is unlikely that the boresight will perfectly align with the target resulting in bursts to either side of the target. This is illustrated in Fig. 2.4. The centroid method can therefore be used to refine the azimuth measurement by weighting the detection azimuth values against their corresponding power value as follows:

$$\psi_\mathrm{d} = \frac{\sum_{i=1}^{M} P_{\mathrm{r}_i} \cdot \psi_{\mathrm{r}_i}}{\sum_{i=1}^{M} P_{\mathrm{r}_i}}, \tag{2.23}$$

where $\psi_\mathrm{d}$ is the refined azimuth of the target detection, and $P_{\mathrm{r}_i}$ and $\psi_{\mathrm{r}_i}$ are the power and azimuth for detection $i$ out of $M$ respectively.

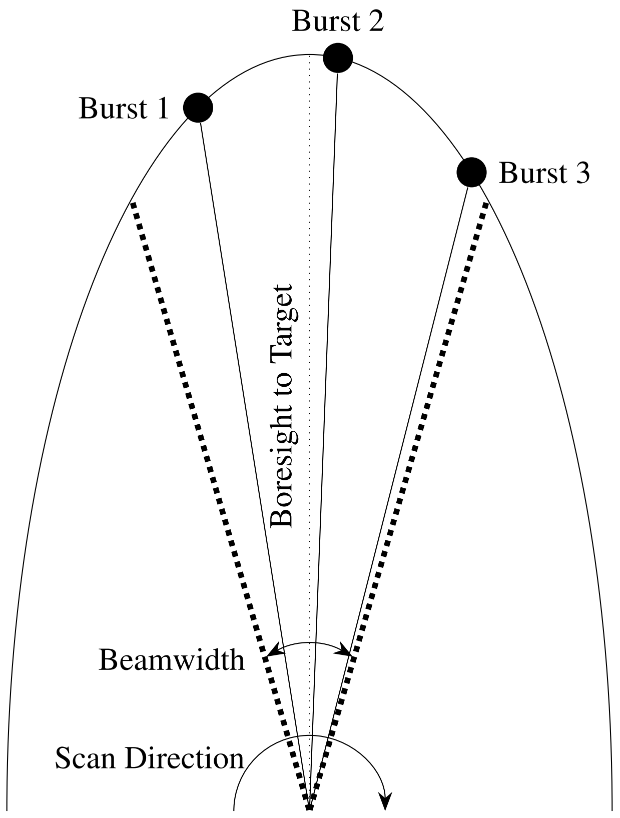


Figure 2.4: Illustration of bursts relative to a target.

### 2.2.5 Tracking

The processed detection in terms of target azimuth $\psi_\mathrm{d}$ and the range bin $r_\mathrm{d}$ is received by the tracker. These values are converted to latitude and longitude using the radar platform's navigational data. The detection is compared with the gates of existing tracks to determine if it is a new target or if the detection can be associated with an existing track.

A Kalman filter [54] is used to track the position of the target in terms of latitude and longitude. For brevity, the workings of the Kalman filter will not be discussed here as extensive work has already been carried out for radar target tracking with Kalman filters and other filter types [55]. The Kalman filter is outlined in algorithm 1. The 'predict' stage is called at each simulation time step to obtain an up-to-date track position estimate whilst the 'update' stage is only called when there is a new measurement for that track. Note that a given detection was associated to an existing track if its position was within a radius of 200 m of the track. If a given detection was not within any track's radius, a new track was created.

The global position estimate in latitude and longitude of track $i$ at time-step $k$ is then given by $\hat{\boldsymbol{x}}_k^i = [\phi, \lambda]_k^i$. The full state of the Kalman filter track $i$ at time-step $k$ is given by equation (2.24). Note that a linear motion model was used for the Kalman filter in these scenarios, however different motion models could be used [56].

```
Algorithm 1: KALMAN FILTER
Input: $\hat{\boldsymbol{x}}_{k-1}^{-}, \boldsymbol{P}_{k-1}^{-}$
Output: $\hat{\boldsymbol{x}}_k, \boldsymbol{P}_k$
Predict:
    A priori state estimate:     $\hat{\boldsymbol{x}}_k^{-} = \boldsymbol{F}_k \hat{\boldsymbol{x}}_{k-1}^{-}$
    A priori error covariance:   $\boldsymbol{P}_k^{-} = \boldsymbol{F}_k \boldsymbol{P}_{k-1}^{-} \boldsymbol{F}_k^T + \boldsymbol{Q}_k$
end
Update:
    Innovation matrix:               $\boldsymbol{S}_k = \boldsymbol{H}_k \boldsymbol{P}_k^{-} \boldsymbol{H}_k^T + \boldsymbol{R}_k$
    Optimal Kalman gain:             $\boldsymbol{K}_k = \boldsymbol{P}_k^{-} \boldsymbol{H}_k^T \boldsymbol{S}_k^{-1}$
    A posteriori state estimate:     $\hat{\boldsymbol{x}}_k = \hat{\boldsymbol{x}}_k^{-} + \boldsymbol{K}_k (\boldsymbol{z}_k - \boldsymbol{H}_k \hat{\boldsymbol{x}}_k^{-})$
    A posteriori error covariance:   $\boldsymbol{P}_k = \boldsymbol{P}_k^{-} - \boldsymbol{K}_k \boldsymbol{H}_k \boldsymbol{P}_k^{-}$
end
```

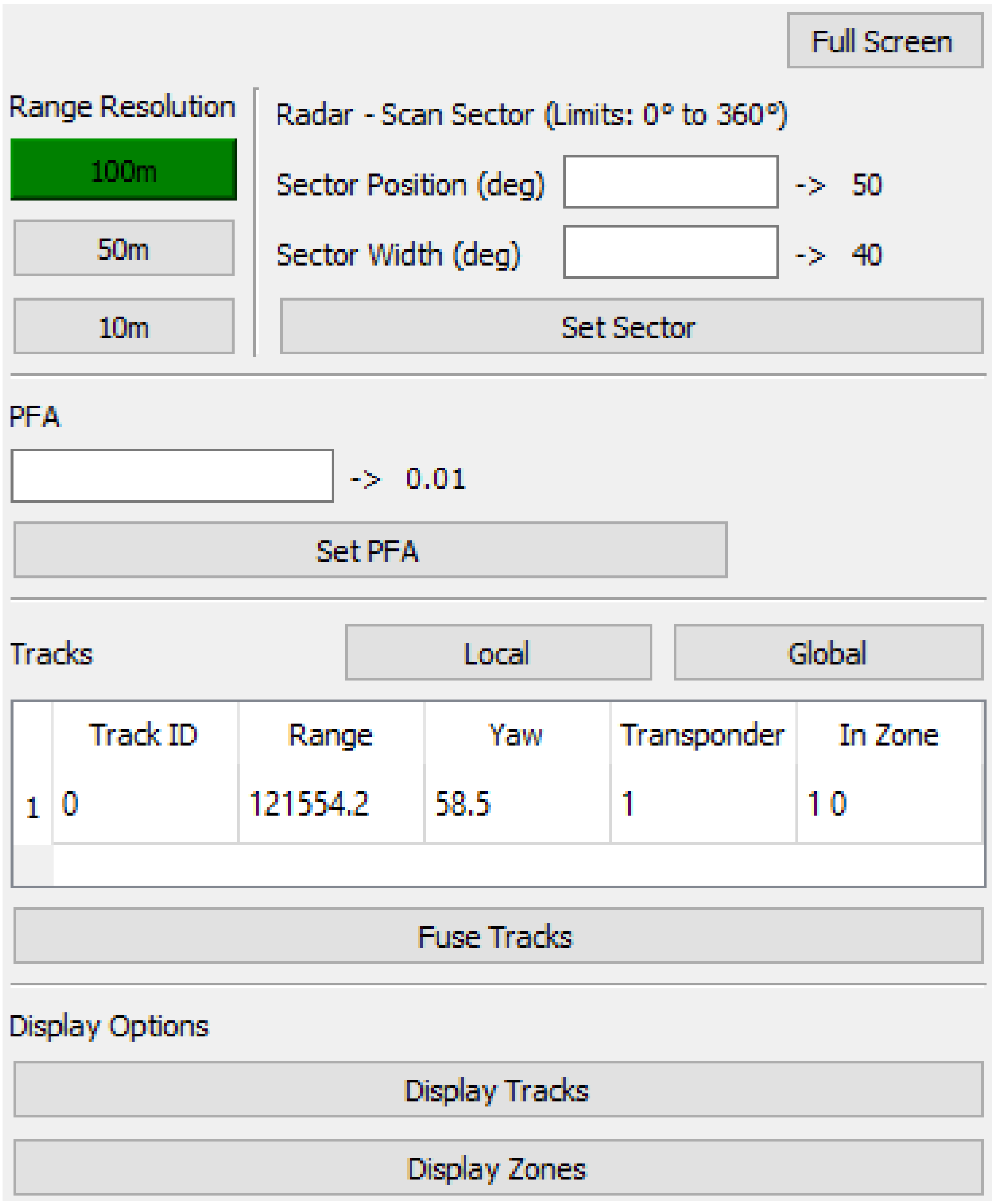
$$\begin{aligned}
\text{State estimate, } \hat{\boldsymbol{x}}_k^i &= \begin{bmatrix} \hat{\phi}_k^i & \hat{\lambda}_k^i \end{bmatrix} \\
\text{State transition matrix, } \boldsymbol{F}_k &= \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
\text{Process noise covariance matrix, } \boldsymbol{Q}_k &= \begin{bmatrix} \zeta_\phi^2 & \zeta_\phi \zeta_{\dot{\phi}} & 0 & 0 \\ 0 & \zeta_{\dot{\phi}}^2 & 0 & 0 \\ 0 & 0 & \zeta_\lambda^2 & \zeta_\lambda \zeta_{\dot{\lambda}} \\ 0 & 0 & 0 & \zeta_{\dot{\lambda}}^2 \end{bmatrix} \\
\text{Observation noise matrix, } \boldsymbol{R}_k &= \begin{bmatrix} \delta_\phi & 0 \\ 0 & \delta_\lambda \end{bmatrix} \\
\text{Observation model matrix, } \boldsymbol{H}_k &= \boldsymbol{I}_2
\end{aligned} \tag{2.24}$$

Note that the values for $\delta$ were set by transforming the local coordinate errors as a result of range bin size and the centroid method into global coordinates. Furthermore, the values for $\zeta$ were heuristically chosen.

### 2.2.6 Graphical User Interface

In order for an operator to carry out typical maritime radar missions using this simulation, it was necessary to develop a graphical user interface. The controls for this interface are shown in Fig. 2.5 while a section of the display is shown in Fig. 2.6.

A plan position indicator (PPI) is used to display the radar data in terms of *SIR* for a given range and azimuth. In a similar manner to the radar range bins and azimuth step size, the PPI display is discretised into cells in range and azimuth. The PPI circular display has a


Full Screen

Range Resolution

100m

50m

10m

Radar - Scan Sector (Limits: 0° to 360°)

Sector Position (deg) -> 50

Sector Width (deg) -> 40

Set Sector

PFA

-> 0.01

Set PFA

Tracks

Local

Global

| | Track ID | Range | Yaw | Transponder | In Zone |
|---|---|---|---|---|---|
| 1 | 0 | 121554.2 | 58.5 | 1 | 1 0 |

Fuse Tracks

Display Options

Display Tracks

Display Zones

Figure 2.5: Radar GUI within MAVERIC. In addition to the described controls, the operator can also opt to show or hide the track and zone overlays.

500 pixel radius and as such, there are effectively 500 display range bins for each displayed azimuth sector. The angular resolution of the display $\Delta\psi_{\mathrm{d}}$ is the angular distance covered by the radar within a burst: $\Delta\psi_{\mathrm{d}} = \dot{\psi}\Delta t = \dot{\psi}N_{\mathrm{p}}PRI$. However, given that there are 500 display range bins while the radar itself has 1852 range bins, there is thus more radar range bins than can be displayed for a given azimuth sector. As such, the maximum *SIR* from the range bins that are within the current display range space is plotted. In figure 2.6, it appears as if there is more noise at further ranges. These ranges are further off the main-beam so the thermal noise starts to have a greater affect on the LLA. However, the thermal noise is completely

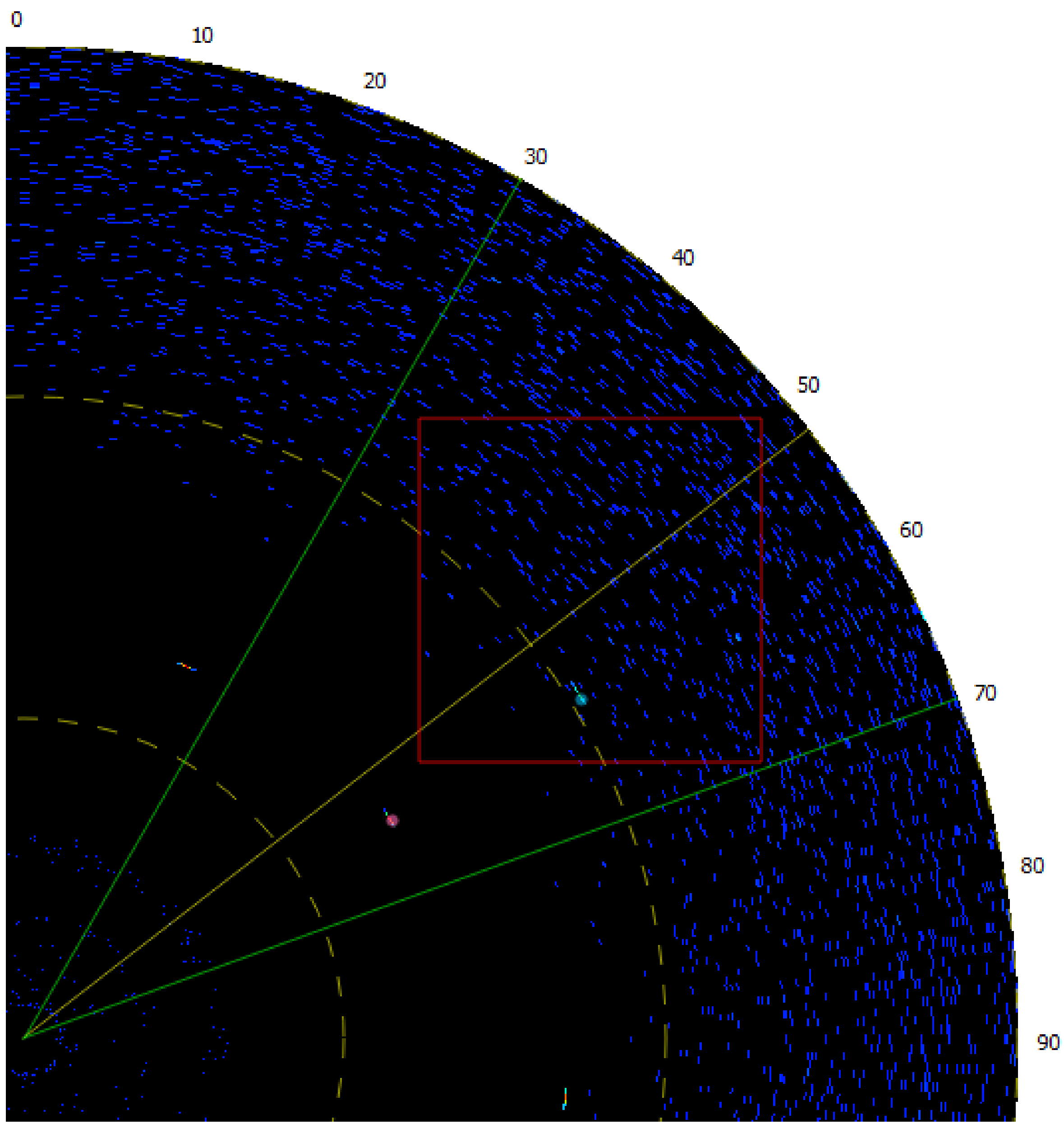


Figure 2.6: Radar PPI display within the GUI. The yellow solid line indicates the current azimuth of the radar boresight while the green lines indicate the current sector scan limits.

random, and it would be very unlikely for the binary integration to declare a detection.

The GUI receives the tracks from the tracker which are subsequently converted back to range and azimuth to be overlayed on the PPI display. From figure 2.6, the pink dot (shown on the outside of the bottom left of the red box) indicate tracks while the blue dot (shown centrally on the inside of the red box) indicates the track currently selected by the operator.

The operator has several options in terms of controls, one of which is the ability to select the range resolution. Changing the range resolution also changes the maximum range resulting in the operator being able to perform long range surveillance of a large area at low resolution or performing short range surveillance of a small area at high resolution.

It is often the case that the operator wishes to scan a small section of the area covered by the radar. This is done with the use of sector scan where within the simulation the operator

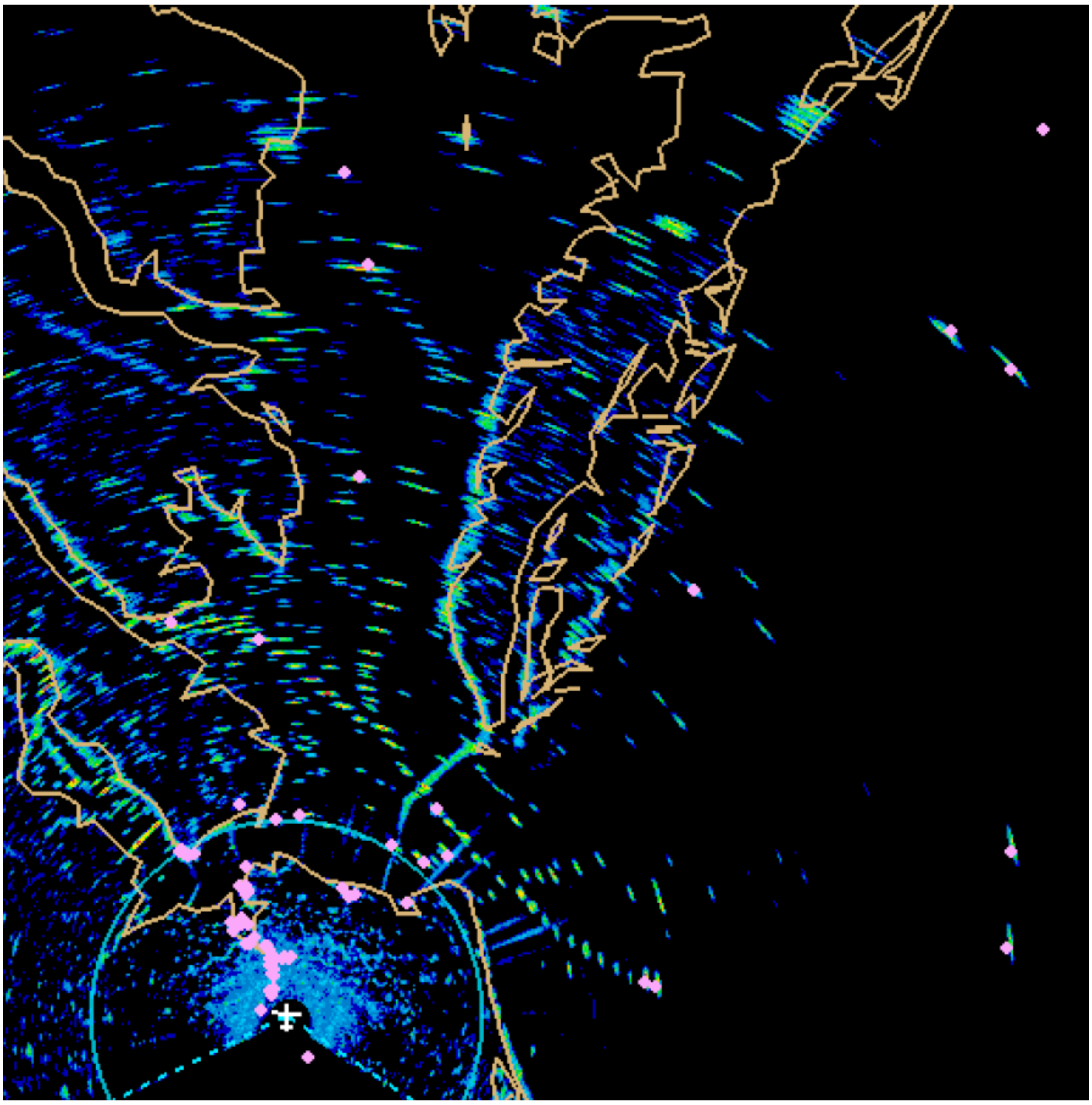


Figure 2.7: An example of an actual maritime radar display [49].

controls the angular segment to be scanned by setting its angular position and width.

As previously mentioned, the *PFA* affects the detection threshold which in turn affects the *SIR* values displayed on the operator's screen. Consequently, the controls allow for the operator to set the value of the *PFA*.

The operator can also create a number of zones defined by coordinates in latitude and longitude. The operator can then visually determine a target's location relative to a given zone. For example, the operator can determine if a boat performing a fishing operation is within an illegal area. From figure 2.6, the red polygon outlines an operator defined zone.

Using a cursor, the operator can select a track. The current track information is displayed in the 'Tracks' box with the operator able to display the targets position in either range and azimuth ('Local') or latitude and longitude ('Global'). In addition to the track position, the information also indicates whether or not a track is within any of the defined zones. Furthermore, the operator can fuse the current tracks with the available transponders and the display indicates if a track does or does not have a transponder. A track without a transponder, for example, could potentially indicate an illegal operation.

Fig. 2.7 shows an example of a typical maritime radar display. Note that this is not intended to match the simulation as there will be various differences in operational parameters (e.g. altitude, target types). Furthermore, the simulation's intent is to provide representative behaviour of a maritime radar and control system rather than provide a realistic performance model.

In terms of platform controls, the operator can either select a global coordinate to move towards. This coordinate could be either a track or an operator designated zone. Alternatively, the operator can move in the direction of a track in terms of increasing and decreasing the range to the target, or the operator can maintain range but increase or decrease the heading towards the target. Both options allow the operator to get a better/alternative view of a target which can help the operator determine the target's intent and therefore the actions that should be taken.

## 2.3 Conclusion

This chapter outlines a simple and computationally efficient maritime radar simulation in conjunction with a graphical user interface to allow for an operator to control a radar system, for surveillance scenarios, in a similar manner to an actual system. The scene consists of a number of targets and a radar platform (though multiple platforms can simultaneously exist). The radar was modelled with the use of the range equation, by accounting for the curvature of the earth, and approximating a radar beam pattern in order to compute power returns from both targets and sea clutter.

Detections were obtained with the use of a local area average and consequently displayed on the user's interface. Binary integration was then applied to further refine the detections which could then be passed to the Kalman filter tracking algorithm. The tracks can be shown on the display with the operator able to use a cursor to select a specific track which results in that track's information being shown. This information contains the position (in either local or global coordinates) of the track, whether or not the track has an associated transponder (after fusion), and whether the track is in any of the user designated zones. In addition to selecting track information, the operator can change range resolution, set the angular position and width of a sector scan, set the probability of false alarm, and fuse tracks with available transponders. Lastly, the operator has the ability to input guidance commands to the simulated platform. This allows for the operator to move towards a specific target or to move to a specific position.

# Chapter 3

# Trajectory Optimisation for Radar Surveillance

## 3.1 Introduction

### 3.1.1 Motivation

If the decisions of an operator are to be imitated, and to make full use of that autonomy, it is necessary for the platform itself to be autonomous. Specifically, if the artificially intelligent agent that replaces the human operator can operate at all times and operate on multiple platforms simultaneously, then the overall mission is limited by the availability of pilots in terms of both maximum flight time for an individual pilot and number of pilots. The use of autonomous platforms removes this bottleneck and frees the artificially intelligent agent to operate at all times and on multiple platforms.

In order for the platform to be autonomous, the required control inputs to the platform are needed. These control inputs represent the operator trajectory commands shown in Fig. 2.1. While classic radar surveillance trajectories (e.g. circular path, race track) could be used, surveillance time as well as the revisit time and probability of detection are of the highest importance. Additionally, obtaining a trajectory that achieves optimal surveillance performance will reduce the surveillance down time, making full use of the AI. Furthermore, as mentioned in 1.1.2, the chosen platform for this work was a high altitude UAV which differs significantly from typical radar surveillance platforms such as helicopters. As such, finding the trajectories that are optimal for the this platform type for various search areas is of benefit in order to maximise the benefit of the artificially intelligent agent.

For an unmanned aerial vehicle (UAV) carrying out a maritime radar surveillance mission, there is a trade-off between maximising information obtained from the search area and minimising fuel consumption. This chapter presents an approach for the optimisation of a UAV's trajectory for maritime radar wide area persistent surveillance to simultaneously minimise fuel consumption, maximise mean probability of detection, and minimise mean revisit time.

Quintic polynomials are used to generate UAV trajectories due to their ability to provide complete and complex solutions while requiring few inputs. Furthermore, the UAV dynamics and surveillance mission requirements are used to ensure a trajectory is realistic and mission compatible. A wide area search radar model is used within this research in conjunction with a discretised grid in order to determine the search area's mean probability of detection and mean revisit time. The trajectory generation method is then used in conjunction with a multi-objective particle swarm optimisation (MOPSO) algorithm to obtain a global optimum in terms of path, airspeed (and thus time), and altitude. The performance of the approach is then tested over two common maritime surveillance scenarios and compared to an industry recommended baseline (outlined in 3.4.1 and 3.4.2).

Airborne surveillance is of significant interest for both military and civilian applications. A radar, in comparison to other remote sensors, provides large area surveillance in both adverse weather conditions (e.g. rain, fog) and varying light conditions (i.e. day and night). An airborne platform is often required for large area maritime radar surveillance. As such, the length of surveillance time is dependent on the fuel consumption of said platform, which in turn is dependent on the platform trajectory. However, the platform trajectory also affects the surveillance of the search area. In particular, it affects the visibility of the radar and where the main beam intersects the surface. Specifically, the platform airspeed, altitude, and path will affect the fuel consumption and how long the radar's beam stays within a given area, thus affecting both the probability of detection and revisit time.

For persistent surveillance missions, the aim is not necessarily to search for a specific target but to patrol a region of interest where complete coverage is required (i.e. every point in the search area is visited to some degree at least once during the trajectory). Furthermore, the platform trajectory is required to be as continuous as possible with little to no down time to allow for constant surveillance. Therefore, minimising the fuel consumption of the platform allows for a given surveillance trajectory to be flown on for longer. High altitude UAVs are typically designed for long endurance, and as such are able to perform surveillance missions for significant amounts of time before a refuel is required [57]. Additionally, UAVs have a significant advantage over other airborne platforms, in that they are able to fly any time of day with minimal human input. For both these reasons, a high altitude UAV was selected as the platform for this problem. Note that high altitude is generally defined as above commercial altitudes (taken as 11000 m).

There is therefore a trade-off between minimising the UAV fuel consumption and maximising search area coverage, where the coverage is defined in terms of both the probability of detecting a given target and the revisit time (i.e. the time between covering the same point in the search area). Increasing the coverage of the surveillance area requires a larger trajectory, resulting in more fuel consumed. However, there is also a trade-off between the revisit time and the probability of detection. If the trajectory covers a large area at a low speed, a lot of time is spent at each point in the search area which increases the overall probability of detection. However, this trajectory type results in a greater time between revisiting a given point in the search area. Thus, the aim is to obtain a trajectory that simultaneously minimises fuel consumption, maximises the average probability of detection, and minimises average revisit time for a given search area. These three criteria were quantified with cost functions for which a multi-objective global optimisation algorithm was used to obtain trajectories that minimise the costs. Currently within industry [58], only specific trajectories are considered, which may not be optimal for a given search area and mission. The optimised trajectories were compared with industry recommended trajectories to highlight the performance of this approach.

Note that this problem deals with finding an optimal trajectory for the surveillance of a pre-specified area (e.g. a fishing zone). As such, the solution for a given area does not need to be solved in real-time [59], and consequently an offline solution is adopted. Additionally, in reference to section 1.1.4, it is vital that the autonomous system replacing a human is transparent in its operations. By having the algorithm work offline, the exact route of the UAV will be known beforehand. In contrast to an offline algorithm, the UAV trajectory generated by an online algorithm is subject to change as it is being performed. An offline solution therefore removes the unpredictability and ambiguity that could potentially lead to operator confusion.

Finding an optimal trajectory is an optimisation problem with many coupled variables under complex constraints. As such, the cost function associated with this problem is multimodal and local search algorithms (e.g. hill climbing) are unsuited [60]–[64]. Instead, global optimisation algorithms were explored for this problem. It should be noted that for complex global optimisation problems, finding the absolute optimal variables is usually impossible due to the large and complex search space. Instead, it is common to use global optimisation algorithms to obtain solutions that are more optimal than a baseline.

Global optimisation problems are generally solved with the use of modern metaheuristic algorithms. These algorithms are typically biologically inspired and exploit two important features of evolution: intensification and diversification [65]–[67]. Intensification attempts to improve on the nearby current best solutions, whereas diversification attempts to explore the search space in an efficient manner. Within nature, these features of evolution have led to biological systems becoming the fittest within their environment, so algorithmically imitating these features allows for more optimal solutions to be found within the problem search space. The most commonly used metaheuristic algorithms for solving global optimisation problems are genetic algorithms (GA) [68] and particle swarm optimisation (PSO) [69].

### 3.1.2 Related Work

A simple way to define a UAV trajectory is to use polynomials in time [70]. In particular, quintic polynomials provide a simple method for the representation of velocities and accelerations for a complete trajectory of a given platform. The main advantage of polynomials, relative to other trajectory methods, is the ability to specify the start and end conditions which is a requirement in this case. These polynomials have been used for modelling both fixed-wing UAV [71] and multi-rotor [72], [73] trajectories. For optimising a UAV's trajectory for surveillance, it would be typical to consider the problem as 4D (space and time). However, for maritime surveillance the surface is level, thus there would be no need to change altitude which is therefore assumed constant throughout a given trajectory. There have been numerous implementations of polynomials for the use of trajectory optimisation. The majority of the work focuses on minimising path risk [74], minimising path length in the face of objects [75], maximising threat avoidance while minimising fuel consumption [76], or a combination of fuel minimisation, threat avoidance, and reconnaissance [77]. In addition, multi-objective optimisation has been implemented for polynomial UAV trajectory optimisation in order to maximise threat avoidance while also incorporating terrain constraints [78] and also fuel consumption [79].

Both genetic algorithms (GA) [80] and particle swarm optimisation (PSO) [81] have been used for trajectory optimisation and path planning. From Hassan [82], PSO appears to outperform GA in terms of computational efficiency when applied to constrained nonlinear problems with continuous design variables, as is the case with the proposed problem. For these problems, PSO also achieves higher quality solutions in general. While GA generally outperforms PSO [83] for discrete trajectory optimisation cases, according to Besada-Portas [84] PSO outperforms the GA for continuous trajectory scenarios which involve a series of waypoints and areas of significance. Additionally, when a polynomial trajectory generation method has been used, PSO has been found to outperform GA [85], [86]. Similarly, [81] uses PSO for b-spline trajectory optimisation. For these reasons, PSO was chosen.

Current literature has not considered sensor trajectory optimisation for a large area under persistent surveillance. For sensor trajectory optimisation (with sensor modelling), up to now the work has focused on either a downward looking sensor [87]–[90] or synthetic aperture radar (SAR) [91]. In the aforementioned studies, the input to the UAV is some form of discrete series of heading changes or waypoints. This method generates continuous polynomials for smooth, efficient circular turns. Additionally, in the above papers, the trajectory is optimised for searching certain locations of interest within the search area. This problem differs from the problem presented in this chapter, where trajectories are obtained that maximise coverage for a whole search area. Additionally, due to the cyclic nature of the requirement that the UAV returns to its starting pose, the type of search algorithm that can be used is limited.

For the papers mentioned above, both the search area and sensor are modelled at a much lower resolution. For large area problems, higher resolution is required which is a source of great computational cost. This chapter outlines the trajectory optimisation method in a way that makes obtaining optimal solutions practicable.

Specifically, in both Li [91] and Ergezer [87], there are multiple small regions with the start point, end point, and visiting order of the regions pre-specified. In the case of Li [91], an

A* search algorithm is employed which cannot be applied to this problem for a multitude or reasons, primarily due to the problems cyclic nature. In the case of Ergezer [87], the optimiser is pre-seeded which, while increasing computational efficiency, does not perform a truly global search. This chapter presents the surveillance trajectory problem as a global optimisation problem by allowing a large degree of freedom in terms of path, altitude, and airspeed.

In Perez-Carabaza [90], a receding horizon approach is used which does not provide a global search. Additionally, the UAV height, commanded airspeed, initial position, and initial heading are fixed. These constraints sacrifice potential solutions in favour of computational efficiency as the computational efficiency is more advantageous in their case. However, in the cases presented within this chapter, optimality is a priority over computational efficiency.

### 3.1.3 Main Contributions

The contribution of this chapter is the optimisation of trajectories for large area radar surveillance using the following:

1. Firstly, a polynomial trajectory generation method is derived that provides complex trajectories while requiring few inputs. This method accounts for the fixed-wing platform dynamics and propulsion while also accounting for the requirements of a persistent surveillance mission (i.e. the start pose is equal to the end pose).

2. The radar mathematical model in 2 is expanded upon for use in optimising trajectories. This model includes the earth's curvature and sector scan. Furthermore, a method for determining the search area coverage in terms of probability of detection and revisit time is outlined.

3. The considerations of the required platform for this mission type are accounted for. These considerations include the high altitude region of operation, long mission time, and high fuel to weight ratio of the platform.

4. Lastly, the surveillance trajectory is used with a multi-objective global optimisation algorithm. Results are compared to an industry recommended baseline which shows the performance of this method. Additionally, the geometric similarity of trajectories within the Pareto front are highlighted.

Section 3.2 outlines the trajectory generation method and formulates the fuel consumption for a given trajectory. This section also derives the maritime radar wide area surveillance model, and how it is used to obtain the probability of detection and revisit time at a given point in the search area. Section 3.3 describes the cost function and the optimisation method. Section 3.4 presents simulation results, and conclusions are drawn in section 3.5.

## 3.2 UAV Surveillance Trajectory Generation

### 3.2.1 Polynomial Trajectory

The polynomials used to define the UAV's trajectory can be a function of either position on a constant altitude plane with Cartesian coordinates (x,y), or velocity which is defined by airspeed and heading, with each method having their advantages and disadvantages. The advantage of using a polynomial in $x$ and $y$ is that specific coordinates can be defined such that the UAV passes through them. In the case of a surveillance trajectory, a polynomial in $x$ and $y$ allows for the UAV to return to its initial coordinate, thus allowing for a continuous trajectory. However, one of the most common UAV manoeuvrers is the banked turn where a polynomial defined in $x$ and $y$ essentially approximates sinusoidal functions. For banked turns that result in a large change in heading, the polynomial can not approximate the large trigonometric segment as effectively as it can for a shorter segment. This insufficient approximation can result in unrealistic trajectories.

For a polynomial in heading and velocity, the banked turns are easily represented. However, the major drawback of these polynomials is the inability to control specific coordinates. This drawback prevents a repeated surveillance trajectory where the start and end position and heading are equal. Thus, both polynomials will be used here with the majority of the trajectory defined by a series of $n_\mathrm{h}$ heading polynomials with an $x$ and $y$ polynomial used to return the UAV to the starting coordinate. In this case, the velocity polynomial is ignored to simplify the problem given that there would be little reason for the UAV to change airspeed during a surveillance mission.

Starting with the return polynomial defined in $x$ and $y$ (where $x$ and $y$ are in NED (north, east, down) coordinates with $x$ parallel to lines of constant latitude and $y$ parallel to lines of constant longitude), the position $\boldsymbol{p}$, velocity $\boldsymbol{v}$, and acceleration $\boldsymbol{a}$ can be obtained as follows:

$$\boldsymbol{p}(\tau) = \boldsymbol{c}\tau^5 + \boldsymbol{d}\tau^4 + \boldsymbol{e}\tau^3 + \boldsymbol{f}\tau^2 + \boldsymbol{g}\tau + \boldsymbol{h}, \tag{3.1}$$

$$\boldsymbol{v}(\tau) = \frac{\mathrm{d}\boldsymbol{p}(\tau)}{\mathrm{d}\tau} = 5\boldsymbol{c}\tau^4 + 4\boldsymbol{d}\tau^3 + 3\boldsymbol{e}\tau^2 + 2\boldsymbol{f}\tau + \boldsymbol{g}, \tag{3.2}$$

$$\boldsymbol{a}(\tau) = \frac{\mathrm{d}\boldsymbol{v}(\tau)}{\mathrm{d}\tau} = 20\boldsymbol{c}\tau^3 + 12\boldsymbol{d}\tau^2 + 6\boldsymbol{e}\tau + 2\boldsymbol{f}, \tag{3.3}$$

where $\boldsymbol{p}(\tau) = [x(\tau), y(\tau)]$, $\boldsymbol{v}(\tau) = [\dot{x}(\tau), \dot{y}(\tau)]$, and $\boldsymbol{a}(\tau) = [\ddot{x}(\tau), \ddot{y}(\tau)]$. In addition, $\tau \in [0, \tau_f]$ where $\tau_f = t_f - t_0$ with $t_0$ and $t_f$ being the respective start and end times of the polynomial. The $1 \times 2$ vectors, $\boldsymbol{c}$, $\boldsymbol{d}$, $\boldsymbol{e}$, $\boldsymbol{f}$, $\boldsymbol{g}$, and $\boldsymbol{h}$, define the polynomial with the first column defining the $x$-axis and the second column defining the $y$-axis.

The start point of the polynomial consists of a position, velocity, and acceleration denoted by $\boldsymbol{p}_0$, $\boldsymbol{v}_0$, and $\boldsymbol{a}_0$. Similarly, the end point consists of $\boldsymbol{p}_f$, $\boldsymbol{v}_f$, and $\boldsymbol{a}_f$. These values can be obtained by recalling that the return polynomial connects the last coordinate from the heading polynomials to the initial starting coordinate. As a result the following can be stated: $\boldsymbol{p}_0 = \boldsymbol{p}_E$ where $\boldsymbol{p}_E$ indicates the last coordinate reached by the heading polynomials; $\boldsymbol{p}_f = \boldsymbol{p}_S$ where

$\boldsymbol{p}_S$ is the initial starting coordinate of the whole trajectory; $\boldsymbol{v}_0 = [V_E \cos(\psi_E),\ V_E \sin(\psi_E)]$ where $V_E$ and $\psi_E$ are the airspeed and heading of the UAV at the last coordinate reached; $\boldsymbol{v}_f = [V_E \cos(\psi_S),\ V_E \sin(\psi_S)]$ where $\psi_S$ is the initial heading of the UAV for the whole trajectory; $\boldsymbol{a}_0 = [a_{\parallel_E} \cos(\psi_E) - a_{\perp_E} \sin(\psi_E),\ a_{\parallel_E} \sin(\psi_E) + a_{\perp_E} \cos(\psi_E)]$ where $a_{\parallel_E}$ and $a_{\perp_E}$ are the respective parallel and normal accelerations to the UAV's forward direction at the last coordinate reached; $\boldsymbol{a}_f = [a_{\parallel_S} \cos(\psi_S) - a_{\perp_S} \sin(\psi_S),\ a_{\parallel_S} \sin(\psi_S) + a_{\perp_S} \cos(\psi_S)]$ where $a_{\parallel_S}$ and $a_{\perp_S}$ are the initial UAV's respective parallel and normal accelerations (where $a_{\parallel_S}$ is taken to be 0). Fig. 3.1 outlines the components of the return polynomial trajectory.

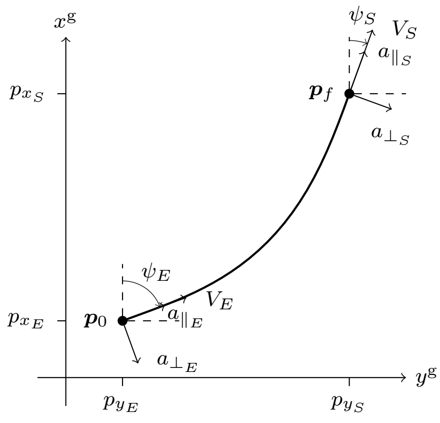


Figure 3.1: Illustration of the return polynomial components. $p_{x_E}$ and $p_{y_E}$ indicate the last $x$ and $y$ position of the combined heading polynomials while $p_{x_S}$ and $p_{y_S}$ indicate the initial starting position of the whole trajectory.

Using the 3 polynomial start conditions in conjunction with equations (3.1), (3.2), and (3.3), the following can be obtained:

$$\boldsymbol{p}(0) = \boldsymbol{p}_0 = \boldsymbol{h}, \tag{3.4}$$

$$\boldsymbol{v}(0) = \boldsymbol{v}_0 = \boldsymbol{g}, \tag{3.5}$$

$$\boldsymbol{a}(0) = \boldsymbol{a}_0 = \boldsymbol{f}. \tag{3.6}$$

Substituting the 3 polynomial end conditions ($\boldsymbol{p}(\tau_f) = \boldsymbol{p}_f$, $\boldsymbol{v}(\tau_f) = \boldsymbol{v}_f$, and $\boldsymbol{v}(\tau_f) = \boldsymbol{v}_f$) and the 3 polynomial start conditions ((3.4), (3.5), and (3.6)) into equations (3.1), (3.2), and (3.3) gives 3 simultaneous equations. Solving the simultaneous equations gives the following:

$$\begin{aligned}\boldsymbol{e} =& (-{\tau_f}^2(\boldsymbol{a}_0 - 3\boldsymbol{a}_f) - 4\tau_f(3\boldsymbol{v}_0 + 2\boldsymbol{v}_f) \\ &- 20(\boldsymbol{p}_0 - \boldsymbol{p}_f))/(2{\tau_f}^3),\end{aligned} \tag{3.7}$$

$$\begin{aligned}\boldsymbol{d} =& ({\tau_f}^2(3\boldsymbol{a}_0 - 2\boldsymbol{a}_f) + 2\tau_f(8\boldsymbol{v}_0 + 7\boldsymbol{v}_f) \\ &+ 30(\boldsymbol{p}_0 - \boldsymbol{p}_f))/(2{\tau_f}^4),\end{aligned} \tag{3.8}$$

$$\begin{aligned}\boldsymbol{c} =& (-{\tau_f}^2(\boldsymbol{a}_0 + \boldsymbol{a}_f) - 6\tau_f(\boldsymbol{v}_0 + \boldsymbol{v}_f) \\ &- 12(\boldsymbol{p}_0 - \boldsymbol{p}_f))/(2{\tau_f}^5).\end{aligned} \tag{3.9}$$

For the heading polynomials, similar methods are used. The polynomials obtained are for $\psi_i(\tau_i)$, $\dot{\psi}_i(\tau_i)$, and $\ddot{\psi}_i(\tau_i)$ for $i = 1, ..., n_{\mathrm{h}}$. Note that for each segment $\ddot{\psi}_0$ and $\ddot{\psi}_f$ are taken to be 0. Also note that the end conditions of a given polynomial $i$ are equal to the start conditions of subsequent polynomial $i+1$. This continuity allows for each polynomial to be defined in the time domain by $\tau$ given that the initial polynomial starts at time 0. Furthermore, given continuity at the polynomial boundaries, the $n_{\mathrm{h}}$ heading polynomials can be combined to form one continuous entity. Additionally, including the return polynomial (for a total of $n_{\mathrm{w}}$ polynomials) forms another continuous entity. It is necessary to differentiate between the two as certain properties are evaluated differently for the return polynomial (in $x$ and $y$) but the full trajectory is still required.

In order to obtain various properties (such as the aerodynamic and propulsive components) along the trajectory in a computationally efficient manner, both the combined heading polynomials and the return polynomial were discretised with a spacing of $\Delta\tau_{\mathrm{f}}$ (taken to be 1 s). The total number of discrete points along the combined heading polynomials is given by $k_h = \lfloor t_{n_{\mathrm{h}f}}/\Delta\tau_{\mathrm{f}} \rfloor + 1$ while the total number of discrete points along the full trajectory is given by $k_w = \lfloor t_{n_{\mathrm{w}f}}/\Delta\tau_{\mathrm{f}} \rfloor + 1$. The evaluated position and velocity vectors for the combined heading polynomials at discrete point $h$ (for $h = 1, ..., k_h$) in $x$ and $y$ are then given by $\boldsymbol{p}_h$ and $\boldsymbol{v}_h$ respectively.

For the heading polynomials, the velocity in $x$ and $y$ can be obtained for a given heading $\psi_h$ and airspeed $V_h$ (equation (3.12)) at a given step $h$ with the following:

$$\boldsymbol{v}_h = [V_h \cos(\psi_h), V_h \sin(\psi_h)]\,. \tag{3.10}$$

In order to determine the return polynomial, the final coordinate in $x$ and $y$ needs to be calculated. With the use of the trapezoidal rule for numerical integration applied to equation (3.10), the position can be updated as follows:

$$\boldsymbol{p}_h = \boldsymbol{p}_{h-1} + \Delta\tau \frac{\boldsymbol{v}_h + \boldsymbol{v}_{h-1}}{2}\,. \tag{3.11}$$

As previously stated, each polynomial is defined by its start and end conditions as well as the value of $\tau$. For the polynomial in $x$ and $y$, the start and end conditions are already known, and as such the polynomial is simply defined by the value of $\tau$ which for this polynomial is denoted as $\tau_{\mathrm{R}}$. A drawback of heading polynomials not previously mentioned is that the nature of a polynomial makes transitions overly smooth and slow (i.e. a change in heading lasting the whole segment). These transitions are not representative of UAV flight and so for this reason it is assumed that each segment has a constant heading rate such that a singular segment is defined by $\tau$ and $\Delta\psi$ (which is equivalent to $\psi_f - \psi_0$). The heading rate is then taken as $\Delta\psi/\tau$. However, this assumption results in discontinuous segments. In order to remedy this discontinuity, transitional segments were introduced to blend the heading rates. These segments are defined as having a fixed value of 20 s for $\tau_{\mathrm{tr}}$ with the start and end heading rates for the transitional segments defined as $\dot{\psi}_{\mathrm{tr}_0} = \dot{\psi}_{i_f}$ and $\dot{\psi}_{\mathrm{tr}_f} = \dot{\psi}_{i+1_0}$. The average heading rate between polynomial $i$ and polynomial $i+1$ can then be used to obtain the change in heading for the transitional segment: $\Delta\psi_{\text{tr}} = \tau_{\text{tr}}(\dot{\psi}_{i_f} + \dot{\psi}_{i+1_0})/2$.

For the sake of computational time, ways of reducing the search space were explored. One such way was to replace the setting of the speed with a normalised value $\mu_V$ using the stall speed $V_{\text{stall}}$ and the minimum drag speed $V_{\text{md}}$. The commonly used formulations for these speeds were used. The normalization was done on a linear scale such that for $\mu_V = 0$, the speed would equal the stall speed, and for $\mu_V = k_V$, the speed would equal the minimum drag speed. The value of $k_V$ was set to 0.288 and was chosen as such because it allows the UAV to reach the maximum airspeed $V_{\text{max}}$ at any altitude. The equation to obtain the UAV airspeed from the normalised value at a given step $k$ along the combined heading polynomials is given as follows:

$$V_h = V_{\text{stall}_h} + \mu_V \frac{V_{\text{md}_h} - V_{\text{stall}_h}}{k_V}, \quad \{\mu_V \in \mathbb{R} | \mu_V \in [0,1]\}. \tag{3.12}$$

This equation would allow for the optimiser to change altitude without offsetting the UAV's speed while also being used to update the UAV's airspeed throughout the heading polynomials. Given that the return polynomial is defined in $x$ and $y$, it is not simple to adjust the velocity throughout its trajectory. Therefore, for the return segment it is assumed that both the start and end velocity will be equal to the value of the velocity at the end of the last heading polynomial.

To further heuristically reduce the search space, the value of $\tau_{\text{R}}$ for the return segment can be replaced with a $\tau$ modifier value $k_\tau$. Since the start and end velocities are equal, the straight line distance between the start and end coordinates of the return segment can be used to estimate a straight line time $t_{\text{sl}}$. The value of $k_\tau$ is then used to modify the straight line distance such that the value of $\tau_{\text{R}}$ is obtained as

$$\tau_{\text{R}} = k_\tau t_{\text{sl}}, \quad \{k_\tau \in \mathbb{R} | k_\tau \in [1, 1.1579]\}, \tag{3.13}$$

where the upper value of $k_\tau$ is set such that the UAV can perform a 45° turn with constant speed.

In conjunction with the value of $k_\tau$ for the return polynomial and the values of $\tau_i$ and $\Delta\psi_i$ for the heading polynomials, additional parameters are used to define the trajectory. These consist of the initial coordinate $\boldsymbol{p}_S$, normalised airspeed $\mu_V$, initial heading $\psi_S$, and altitude of platform operation $h^{\text{P}}$. The whole polynomial trajectory can then defined by vector $\boldsymbol{x}$ such that

$$\boldsymbol{x} = [\boldsymbol{\tau}, \boldsymbol{\Delta\psi}, \boldsymbol{p}_S, \mu_V, \psi_S, k_\tau, h^{\text{P}}], \tag{3.14}$$

where $\boldsymbol{\tau} = [\tau_1, ..., \tau_{n_{\text{h}}}]$, $\boldsymbol{\Delta\psi} = [\Delta\psi_1, ..., \Delta\psi_{n_{\text{h}}}]$, and $\boldsymbol{p}_S = [x_{1_0}, y_{1_0}]$.

### 3.2.2 Fixed-Wing UAV Fuel Consumption

For surveillance missions, the UAV will fly for as long as possible while carrying a radar unit. For this reason, the UAV specifications were loosely based on the Northrop Grumman

Table 3.1: UAV specifications

| Parameter | Unit | Value |
|---|---|---|
| Operational ceiling, $h^{\mathrm{p}}_{\max}$ | m | 19800 |
| Minimum altitude, $h^{\mathrm{p}}_{\min}$ | m | 11000 |
| Maximum airspeed, $V_{\max}$ | $\mathrm{m\,s^{-1}}$ | 186 |
| Maximum load factor, $n_{\max}$ | | 1.035 |
| Maximum coefficient of lift, $C_{l_{\max}}$ | | 1.65 |
| Wing reference area, $S$ | $\mathrm{m}^2$ | 50 |
| Wing aspect ratio, $AR$ | | 25 |
| Zero-lift drag coefficient, $C_{d_0}$ | | 0.0135 |
| Oswald efficiency, $e_0$ | | 0.75 |
| Take-off thrust, $T_0$ | N | 34000 |
| Take-off mass, $m_{\mathrm{TOW}}$ | kg | 11600 |
| Fuel capacity, $m_{\mathrm{fuel}}$ | kg | 6500 |

RQ-4 Global Hawk [57], [92]. Further to these specifications, it is also necessary to consider some mission specific limits. Primarily, the UAV needs to fly above commercial aircraft altitudes, and so the minimum altitude was set to 11000 m. It is also assumed that there is no wind acting on the UAV. The UAV specifications are then outlined in table 3.1.

From take-off to landing, the flight is assumed to be broken into 3 stages. The first stage encompasses the process of take-off, climb, and cruising to the start of the surveillance mission. The second stage involves carrying out the surveillance mission and the third and final stage encompasses the process of returning to base and landing. It is assumed that the fuel fraction (i.e. the fraction of fuel consumed during a given stage of flight) for the first and third stage is 0.9. These fractions also account for any excess fuel required for safety. Therefore, the initial mass of the UAV at the start of the trajectory is denoted by $m_{\mathrm{I}}$ which is equal to $\frac{m_2}{m_1}m_{\mathrm{TOW}}$, where $m_{\mathrm{TOW}}$ is the take-off mass of the UAV. Furthermore, the minimum required weight of the UAV at the end of the trajectory is denoted by $m_{\mathrm{F}}$ and equals $m_{\mathrm{I}}(m_{\mathrm{TOW}} - m_{\mathrm{fuel}})/(\frac{m_2}{m_1}\frac{m_4}{m_3}m_{\mathrm{TOW}})$.

The rate of fuel consumption of the UAV can be calculated (equation (3.15)) as the product of the thrust specific fuel consumption $TSFC$ and the required thrust $T_{\mathrm{required}}$, both of which are dependent on the setting of the UAV. The rate of fuel consumption can then be integrated to obtain the total fuel consumed for a given trajectory. Given the large fuel-to-weight ratio along with the potentially long flight times, it is important to consider the effects of the change in UAV mass along the trajectory. This effect is achieved by incorporating the current UAV mass $m_{\mathrm{C}}$ into the required thrust equation. Since the path is discretised, the trapezoidal method for numerical integration is then applied to update the current UAV mass at each point in the discretised trajectory using the rate of fuel consumption as shown below.

$$\dot{m}_{\mathrm{f}_w} = T_{\mathrm{required}_w} TSFC_w\,, \tag{3.15}$$

$$m_{\mathrm{C}_w} = m_{\mathrm{C}_{w-1}} - \Delta\tau_{\mathrm{f}} \frac{\dot{m}_{\mathrm{f}_w} + \dot{m}_{\mathrm{f}_{w-1}}}{2}\,. \tag{3.16}$$

The required thrust depends on the setting of the UAV which includes the altitude, airspeed, current mass, bank angle, and acceleration. For the heading polynomials, the acceleration is obtained with respect to the local coordinates. In other words, the tangential component is tangent to the path curvature defined by the heading polynomial and is simply given by $a_{\|_h} = (v_h - v_{h-1})/\Delta\tau_f$. The centripetal component is defined along the outward normal to the curve and is obtained with the radius of curvature ($\nu = v/\dot{\psi}$) as follows: $a_{\perp_h} = -{v_h}^2/\nu_h$.

For the return segment, the polynomial is defined in $x$ and $y$ coordinates. However, the UAV will fly with forward velocity along the longitudinal axis. It is then necessary to resolve the return polynomial in the direction of the velocity vector [93]. By denoting a given velocity vector as $\boldsymbol{v}$ and a given acceleration vector as $\boldsymbol{a}$, the airspeed $V$ for the return polynomial is then simply calculated as $V = \|\boldsymbol{v}\|$.

The $x$ and $y$ acceleration of the UAV can be resolved along the longitudinal axis (parallel to the airspeed) to obtain the forward acceleration for the return polynomial as follows:

$$a_{\|} = \frac{\boldsymbol{a}^T \boldsymbol{v}}{\|\boldsymbol{v}\|} . \tag{3.17}$$

Using Pythagoras' theorem, the centripetal acceleration for the return polynomial can then be obtained with

$$a_{\perp} = \sqrt{\|\boldsymbol{a}\|^2 - a_{\|}^2} \times sgn(a_x v_y - a_y v_x) , \tag{3.18}$$

where the term $sgn(a_x v_y - a_y v_x)$ is used to determine the direction of the centripetal acceleration with the $x$ and $y$ subscripts indicating the $x$ and $y$ component of the respective vector.

For obtaining the radar pointing direction, it is necessary to obtain the UAV heading. For the return polynomial, the heading is obtained as: $\psi^{\mathrm{p}} = \operatorname{arctan2}(p_y, p_x)$ where $p_y$ and $p_x$ refer to the respective $x$ and $y$ elements of $\boldsymbol{p}$ (the position of the UAV), and arctan2 is the four-quadrant inverse tan.

It is then necessary to determine the required thrust for a given setting of the UAV [93]. For a fixed-wing UAV, a banked turn is used to change heading. By banking, a horizontal component of the lift force is generated which forms the centripetal acceleration. The UAV's drag during a steady level flight banked turn is then defined as

$$D = c_1 V^2 + \frac{c_2 n^2}{V^2} , \tag{3.19}$$

where

$$c_1 = \frac{1}{2}\rho S C_{d_0},\ c_2 = \frac{2W^2}{\pi e_0 AR \rho S},\ n = \sqrt{1 + a_{\perp}^2/g^2} , \tag{3.20}$$

where $W = m_{\mathrm{C}} g$, and $g$ is the gravitational acceleration with value $9.80665\,\mathrm{m\,s^{-2}}$, and $\rho$ is the air density. The required thrust is then obtained as follows:

$$T_{\text{required}} = D + m_{\mathrm{C}} a_{\|} , \tag{3.21}$$

Equation (3.15) stated that the rate of fuel consumption is the product of the required thrust and the thrust specific fuel consumption. The next steps are to determine the available thrust in order to determine that the required thrust does not exceed it, and also to obtain the thrust specific fuel consumption.

The engine is assumed to be a high-bypass turbofan with an available thrust that decreases with altitude and Mach number $M$. The thrust available is assumed to vary linearly with Mach number given that the UAV operates within a fairly small band of Mach numbers (roughly between 0.3 and 0.6). It is assumed that at $M = 0.3$ there is 75% of the take-off thrust $T_0$ and at $M = 0.6$ there is 60% of $T_0$. These values were selected to provide sufficient thrust at maximum altitude in order to maintain the maximum speed while also fitting within reasonable variations of thrust [94]. The variation in available thrust as a function of altitude and Mach number [95] is given by:

$$T_{\text{available}} = T_0(\mu(M-0.3)+0.6)\sigma_1^{b_1}\sigma_2^{b_2}\,, \tag{3.22}$$

where $\sigma_1 = \rho_\text{t}/\rho_0$ with $\rho_0$ the air density at sea level (with value $1.225\,\text{kg}\,\text{m}^{-2}$) and $\rho_\text{t}$ the air density at the end of the troposphere (with value $0.3639\,\text{kg}\,\text{m}^{-2}$). Similarly, $\sigma_2 = \rho/\rho_\text{t}$. Significant lack of engine data resulted in the values for $b_1$ and $b_2$ assumed to be 0.5 and 0.8 respectively. These values were selected in a similar manner to the values of $M$. The gradient of the thrust-Mach line is denoted as $\mu$ with value $(0.6-0.75)/(0.6-0.3)$, and $M = V/a$, where $a$ is the speed of sound which is assumed constant given the UAV's region of altitude operation.

The thrust specific fuel consumption can be obtained as a function of altitude and Mach number [95]. However, the effect of altitude on $TSFC$ is a function of temperature and in this case (within the region of 11000 m and 20000 m) the temperature is taken as constant. Thus, the thrust specific fuel consumption can be calculated as

$$TSFC = TSFC_0 M^{\alpha}\,, \tag{3.23}$$

where $TSFC_0$ was taken to be $2.55e-5\,\text{kg}\,\text{N}^{-1}\,\text{s}^{-1}$ and $\alpha$ was taken to be 0.6 which is typical for a high-bypass turbofan.

The value of $TSFC$ and $T_{\text{required}}$ can now be evaluated at any point on the whole trajectory, and thus the equations to obtain fuel consumed and change in UAV mass (equations (3.15) and (3.16) respectively) can be used to obtain the total fuel consumed:

$$m_{\text{consumed}} = m_\text{I} - m_{\text{C}_{k_w}}\,, \tag{3.24}$$

where $m_{\text{C}_{k_w}}$ is the UAV weight at the end of the trajectory.

### 3.2.3 Radar Coverage

To obtain the coverage of the radar, a method of representing the sensor and how it covers a given area needs to be implemented. Two aspects will be considered with regards to coverage. The first, the probability of detection at a point within the area, and the second, the revisit time to a point within the area. The probability of detection, as the name suggests, mathematically encapsulates the probability of detecting a specific target. The revisit time is the time between repeated coverage of a given point in the search area. As the UAV moves around the search area, the main beam of the radar will intersect the surface, thus increasing the probability of detection and resetting the current revisit time to zero for those points within the beam. Ideally, the probability of detection is to be maximised while the revisit time is to be minimised. There is thus an inherit trade off between the two. It should be noted that the target interest and probability of target location are uniformly distributed across the search area.

In order to determine the global coverage for a given trajectory in an efficient manner, the search area was discretised into an evenly spaced $x$-$y$ search grid consisting of $n^{\mathrm{g}}$ nodes (where $x$ is along north, $y$ is along east, and $z$ is assumed sea level (i.e. $0\,\mathrm{m}$)). The search grid is defined in terms of the node spacing $\Delta G$ (which sets in meters how far apart each node is along each axis), and the number of nodes $n_x^{\mathrm{g}}$ and $n_y^{\mathrm{g}}$ in the $x$ and $y$ axis respectively. Both node numbers are taken to be odd such that there is a definite center node (note that the center of the search grid is always taken to be $0\,\mathrm{rad}$ in latitude and longitude, and $(0,0,0)$ in NED coordinates). The total number of nodes $n^{\mathrm{g}}$ in the search grid is therefore equal to $n_x^{\mathrm{g}} n_y^{\mathrm{g}}$.

At each node in the search grid, the probability of detection and revisit time needs to be evaluated for the full trajectory. These components can be evaluated by firstly representing the probability of detection grid and the revisit time grid as matrices $\boldsymbol{D}$ and $\boldsymbol{T}$ respectively, both of which have dimensions $n_y^{\mathrm{g}} \times n_x^{\mathrm{g}}$.

For efficiency, the radar was similarly discretised into an $x$-$y$ radar grid (and as before $z$ is taken to be 0) with $n^{\mathrm{r}}$ nodes. Each node represents the probability of detecting a specific target on the ground at a given relative NED coordinate to the UAV. Furthermore, due to the relatively small size of search grids, it is assumed that the radar grid is calculated at a latitude and longitude of $0\,\mathrm{rad}$. To account for the curvature of the earth, the nodes in NED coordinates are converted to ECEF (earth-centered, earth-fixed) coordinates to obtain the absolute range to the UAV as outlined by equations (3.25) and (3.26).

For both the search grid and the radar grid, it is accepted that there will be some warping effects (i.e. the spacing between nodes will not be equal) particularly near the poles of the earth. However, there is warping effects for all methods of mapping a sphere to a 2D grid, and this effect is only of concern if the search area (or radar range) is significantly large. In this case, the diagonal distance between the cells of the grid deviated less than $1\,\mathrm{m}$ when wrapped around the earth. This error was deemed well within reasonable limits.

As is often the case, the full 360° scan of the radar may not be needed. For example, if a thin coastal stretch is to be monitored, the maximum angular sector of the radar need only cover the length of the coastal strip. Accordingly, sector scan is introduced to allow for this scenario, and is defined by the sector scan angular position $\delta$ and the sector scan angular width $\gamma$.

The radar grid represents a full scan of the radar (for any $\gamma$). Given the radial nature of a radar scan, the number of nodes in the $x$-axis is equal to the number of the nodes in the $y$-axis, and is given by $n^{\mathrm{r}}_{xy} = 2\lfloor r_{\max}/\Delta G\rfloor + 1$. Thus, $n^{\mathrm{r}} = n^{\mathrm{r}}_{xy} n^{\mathrm{r}}_{xy}$. The matrices containing the range, elevation from boresight, elevation from UAV, signal-to-noise ratio, and probability of detection are denoted $\boldsymbol{r}$, $\boldsymbol{\Theta}$, $\boldsymbol{\theta}^{\mathrm{r}}$, $\boldsymbol{SNR}$, and $\boldsymbol{P}_{\mathrm{d}}$ respectively.

The ECEF coordinates (denoted by $X$, $Y$, and $Z$) of each node in the radar grid, converted from NED coordinates (denoted $x^{\mathrm{r}}$ and $y^{\mathrm{r}}$), are obtained by modifying equation (2.3) as follows:

$$\begin{aligned} X_{uv} &= r_{\mathrm{a}} \cos(x^{\mathrm{r}}_{uv}/r_{\mathrm{a}}) \cos(y^{\mathrm{r}}_{uv}/r_{\mathrm{a}})\,, \\ Y_{uv} &= r_{\mathrm{a}} \cos(x^{\mathrm{r}}_{uv}/r_{\mathrm{a}}) \sin(y^{\mathrm{r}}_{uv}/r_{\mathrm{a}})\,, \\ Z_{uv} &= r_{\mathrm{a}} \sin(x^{\mathrm{r}}_{uv}/r_{\mathrm{a}})\,, \\ \text{for } u = 1,\ldots,n^{\mathrm{r}}_{xy} &\quad \text{and} \quad \text{for } v = 1,\ldots,n^{\mathrm{r}}_{xy}\,, \end{aligned} \tag{3.25}$$

where subscripts $u$ and $v$ indicate the row and column index of a given node in the radar grid.

The range from the UAV to an individual radar grid node in matrix $\boldsymbol{r}$ is obtained by modifying equation (2.4) as follows (recalling that the radar is calculated at a latitude and longitude of 0 rad):

$$r_{uv} = \sqrt{(X_{uv} - h^{\mathrm{p}})^2 + {Y_{uv}}^2 + {Z_{uv}}^2}\,. \tag{3.26}$$

The radar system outlined in section 2 was based on a typical airborne maritime surveillance radar suitable for a long range UAV mission. This chapter deals with wide area surveillance. Hence, the radar range was set to the maximum value (i.e. 185200 m), however any combination of radar parameters could be used with this method.

For computational requirements, the surface clutter was omitted. This omission has no effect on the optimisation problem as clutter would only affect the absolute probability of detection and not the relative probability of detection. Consequently, the radar signal-to-noise ratio $SNR$ is considered rather than the signal-to-interference ratio $SIR$. With the use of equation (2.12) and equation (2.19), each element in matrix $\boldsymbol{SNR}$ is given by

$$SNR_{uv} = \frac{\eta N_p P_{\mathrm{tx}} G^2(\Theta_{uv}) \sigma_{\mathrm{tgt}} \lambda^2}{(4\pi)^3 k_{\mathrm{B}} T B_n F_n L_{\mathrm{sys}} {r_{uv}}^4}\,, \tag{3.27}$$

where $\sigma_{\mathrm{tgt}}$ is taken to be 100 m$^2$.

Since the radar grid represents a full scan of the radar, equation (2.11) can be modified to remove the azimuth component. Consequently, the radar gain $G$ is a function of elevation from radar boresight given in matrix form as

$$G(\boldsymbol{\Theta}) = \frac{4\pi A_l A_h}{\lambda^2} \operatorname{sinc}^2(\pi \frac{A_h}{\lambda} \sin(\boldsymbol{\Theta})), \tag{3.28}$$

where $\boldsymbol{\Theta} = \boldsymbol{\theta}^{\mathrm{r}} - \theta_{\mathrm{B}}$. $\boldsymbol{\theta}^{\mathrm{r}}$ is a matrix containing the elevation angle of each node from the UAV, and $\theta_{\mathrm{B}}$ is the the radar boresight elevation angle. Using the range to each node in the radar grid $r_{uv}$ in conjunction with equation (2.8) and equation (2.7), the elevation from boresight for each node can be obtained.

The *SNR* can now be obtained for any node, and thus the probability of detection $\boldsymbol{P}_{\mathrm{d}}$ can be obtained (an example of which is shown in Fig. 3.2). Given that the target can be approximated by Swerling 1 models, each element in the matrix can then be approximated [47] as

$$P_{\mathrm{d}uv} = PFA^{(1+SNR_{uv})^{-1}}. \tag{3.29}$$

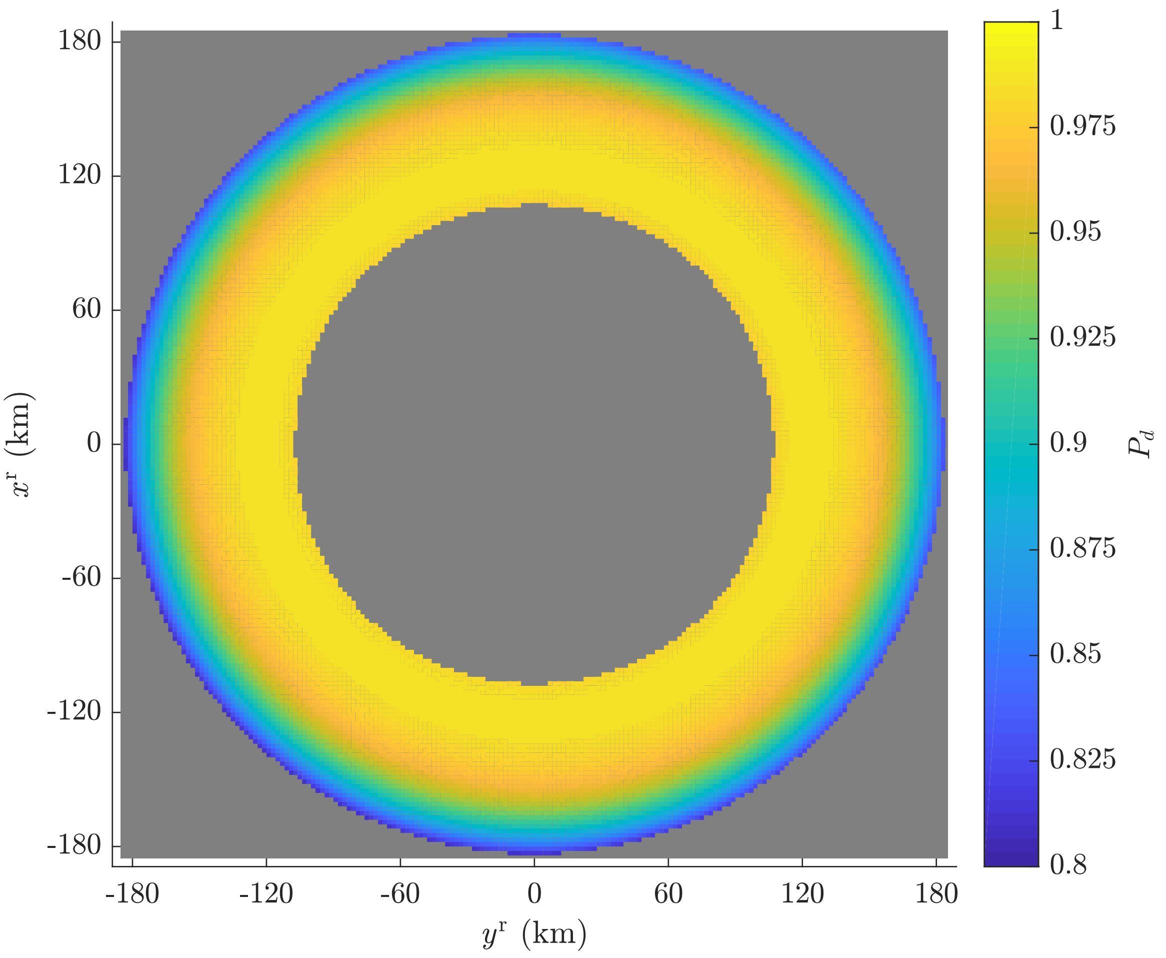


Figure 3.2: Probability of detection radar grid at altitude $h^{\mathrm{p}} = 15000\,\mathrm{m}$. The gray area indicates excluded nodes.

It should be noted that in order to improve computational time, several conditions were applied during the creation of matrix $\boldsymbol{P_d}$. The first of which was to disregard the radar side lobes and ensure that only the main beam of the radar was considered. This consideration

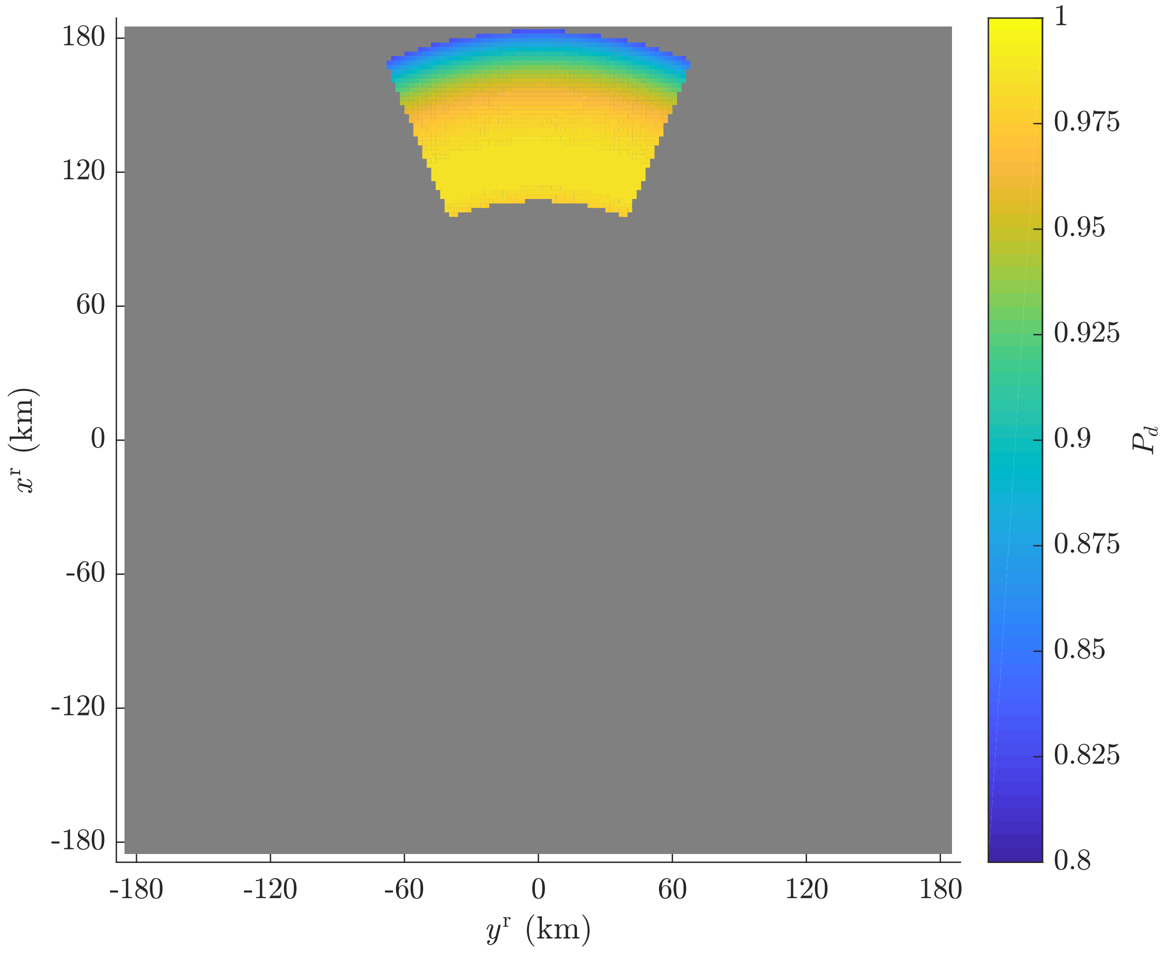


Figure 3.3: An example of the probability of detection radar grid with the following parameters: $\delta = \pi/2\,\mathrm{rad}$, $\gamma = \pi/4\,\mathrm{rad}$, and $h_p = 15000\,\mathrm{m}$. In this case, the scan time $\Delta\tau_r = \gamma/\dot{\omega} = 3\,\mathrm{s}$.

was implemented by removing any nodes that had a range $r_{uv}$ less than the upper and lower slant range of the radar main beam such that $r^-_{3\,\mathrm{dB}} \leq r_{uv} \leq r^+_{3\,\mathrm{dB}}$. In order to obtain the slant ranges, equation (2.8) was used where $\theta^-_{3\,\mathrm{dB}} = \theta_{3\,\mathrm{dB}} + \theta_{3\,\mathrm{dB}}/2$ and $\theta^+_{3\,\mathrm{dB}} = \theta_{3\,\mathrm{dB}} - \theta_{3\,\mathrm{dB}}/2$. The probability of detection for the ignored nodes was 0.

When sector scan is applied only the nodes within the sector need to be considered. The following condition can then be used on each node in the radar grid to determine if it lies within the sector: $|\psi^{\mathrm{r}}_{uv} - \psi^{\mathrm{p}} - \delta| \leq \gamma/2$.

The matrix $\boldsymbol{P}_{\mathrm{d}}$ provides the probability of detection for the radar grid at a given point in the trajectory. However, it is necessary to obtain the accumulated probability of detection at each node in the global search grid for a given trajectory. In order to calculate the accumulated probability of detection, it is first necessary to obtain the current search grid nodes that are aligned with the current radar grid nodes.

Given the long surveillance range, the maximum speed of the UAV, and the fast scan rate, it is assumed that a full sector scan (for any $\gamma$) is applied at a time step equal to the equivalent scan time. Note that the distance covered by the UAV during the time taken to scan is not significant relative to the range covered by the radar. This assumption greatly improves computational efficiency by allowing the radar to be represented by one matrix. Similar to obtaining the fuel consumption, the trajectory is discretised in $\tau$. In this case however, the spacing $\Delta\tau_{\mathrm{r}}$ was set to match the scan rate of the radar and was thus equal to $\gamma/\dot{\omega}$.

A given discrete position ($\boldsymbol{p}$ and $h^{\mathrm{p}}$) in NED coordinates along the trajectory can be projected to determine the NED position on the ground that is equivalent to the position of the center of the radar grid. This ground position, obtained with equation (3.30), is then rounded to the nearest node on the global search grid. At this point, the radar grid will be aligned with the global search grid and thus the current probability of detection of the radar grid can be used to update the global search grid probability of detection (or revisit time).

$$\boldsymbol{p}^{\mathrm{g}} = \boldsymbol{p}\frac{r_{\mathrm{e}}}{r_{\mathrm{e}} + h^{\mathrm{p}}}\,. \tag{3.30}$$

For a given discrete position of the UAV projected to the ground $\boldsymbol{p}^{\mathrm{g}}$, the lower and upper matrix indices of the global search grid that are occupied by the radar grid are given by the $1\times 2$ vectors $\boldsymbol{m}$ and $\boldsymbol{n}$ respectively. The index of the center node for the search grid and the radar grid are denoted by $\boldsymbol{g}^{\mathrm{g}} = \lceil [n_x^{\mathrm{g}}/2, n_y^{\mathrm{g}}/2] \rceil$ and $g^{\mathrm{r}} = \lceil n_{xy}^{\mathrm{r}}/2 \rceil$ respectively, where $\lceil x \rceil$ denotes the ceiling function which returns the lowest integer greater than or equal to $x$. $\boldsymbol{m}$ and $\boldsymbol{n}$ are then obtained as follows:

$$\boldsymbol{m} = \mathrm{nint}(\frac{\boldsymbol{p}^{\mathrm{g}}}{\Delta G}) + \boldsymbol{g}^{\mathrm{g}} - g^{\mathrm{r}} + 1\,, \tag{3.31}$$

$$\boldsymbol{n} = \mathrm{nint}(\frac{\boldsymbol{p}^{\mathrm{g}}}{\Delta G}) + \boldsymbol{g}^{\mathrm{g}} + g^{\mathrm{r}} - 1\,, \tag{3.32}$$

where, for example, $m_1$ and $m_2$ indicate the respective column and row index of the lower bounds of the global search grid that is within the radar grid. Note that the function nint, obtains the nearest integer.

The probabilities of detection for the current radar grid can then be combined with the probabilities of detection for the global search grid. Noting that for events that are independent and not mutually exclusive, the probability of event $A$ or $B$ occurring is given by $P(A\cup B) = P(A) + P(B) - P(A)P(B)$. The appropriate nodes in the probability of detection search grid can then be updated as

$$\boldsymbol{D}_{jk} \leftarrow \boldsymbol{D}_{jk} + \boldsymbol{P}_{\mathrm{d}w} - \boldsymbol{D}_{jk} \circ \boldsymbol{P}_{\mathrm{d}w}\,, \tag{3.33}$$

where $j = m_1, ..., n_1$ and $k = m_2, ..., n_2$ which indicate the respective row and column indices of the search grid nodes that are within the radar grid. Note that the Hadamard product is used for element wise multiplication. An additional check of $j$ and $k$ is applied to ensure that each value lies within the bounds of the search grid such that $1 \le j \le n_y^{\mathrm{g}}$ and $1 \le k \le n_x^{\mathrm{g}}$. Values that do not satisfy this condition are ignored. $\boldsymbol{P}_{\mathrm{d}w}$ indicates the radar grid probability of detection $\boldsymbol{P}_{\mathrm{d}}$ at discrete point $w$ on the trajectory (with a time step of $\Delta\tau_{\mathrm{r}}$ used).

In order to obtain the average revisit time, three quantities are required for each node in the search area grid: the number of visits by the radar $T^{\mathrm{n}}$, the total revisit time $T^{\mathrm{t}}$, and the current revisit time $T^{\mathrm{c}}$.

Due to the discretisation of the search area, radar area, and scan time, it is possible for a node to enter coverage, then exit coverage, then re-enter coverage all within 3 simulation steps. While this event is rare, a smoothing technique was nevertheless employed. This technique ensures that nodes will only have their visit number incremented if there has been two steps since the last increment. The update rules for the number of visits and total revisit time for each node in the search area grid are given in equation (3.34) and (3.35) respectively.

$$\boldsymbol{T}^{\mathrm{n}}_{pq} \leftarrow \boldsymbol{T}^{\mathrm{n}}_{pq} + 1\,, \tag{3.34}$$

$$\boldsymbol{T}^{\mathrm{t}}_{pq} \leftarrow \boldsymbol{T}^{\mathrm{t}}_{pq} + \boldsymbol{T}^{\mathrm{c}}_{pq}\,. \tag{3.35}$$

On the first step of the simulation, $p$ and $q$ are equal to $j$ and $k$ respectively. These indices result in the visit count $T^{\mathrm{n}}$ of each search grid node within radar coverage being incremented by 1, in addition to the total visit time $T^{\mathrm{t}}$ being incremented by the current revisit time $T^{\mathrm{c}}$. On the second step of the simulation, $p$ and $q$ indicate the respective row and column indices of the search grid nodes which have newly entered radar coverage. From step 3 until the end of the simulation, $p$ and $q$ indicate the respective row and column indices of the search grid nodes which have newly entered coverage and have not been incremented in the previous 2 steps. To update the revisit time search grid, the whole grid is firstly incremented by the current scan time (i.e. the time that has passed since the previous scan) as follows:

$$\boldsymbol{T}^{\mathrm{c}} \leftarrow \boldsymbol{T}^{\mathrm{c}} + \Delta\tau_{\mathrm{r}}\,. \tag{3.36}$$

The current revisit time search nodes that are within the radar grid are then multiplied by either 0 or 1 depending on whether the probability of detection is 0 or greater than 0 respectively. The current revisit time for these nodes is therefore equated as follows:

$$\boldsymbol{T}^{\mathrm{c}}_{jk} \leftarrow \boldsymbol{T}^{\mathrm{c}}_{jk} \circ \lfloor (1 - \boldsymbol{P}_{\mathrm{d}w}) \rfloor\,. \tag{3.37}$$

In order to obtain the average revisit time for the whole search area, the remaining revisit time at the the end of the trajectory needs to be accounted for. As such, the nodes which were covered at both the start and end of the trajectory have their visit counts reduced by 1. This criteria does not include those nodes which were within radar coverage for the full trajectory (i.e. having a visit count equal to 1). Additionally, the nodes within $\boldsymbol{T}^{\mathrm{t}}$ which were not covered at the start/end have their corresponding elements from $\boldsymbol{T}^{\mathrm{r}}$ added, with $\Delta\tau_r$ subtracted from these nodes to account for the start and end step being equivalent. With the remaining revisit time accounted for, the average revisit time $T$ for a given node is then simply the total revisit time divided by the number of visits:

$$T = T^{\mathrm{t}}/T^{\mathrm{n}}\,. \tag{3.38}$$

## 3.3 Optimisation

### 3.3.1 Cost Function

The objective is to find trajectories that simultaneously minimise fuel consumption, maximise the probability of detection within the search grid, and minimise the revisit time within the search grid. In order to achieve this objective, a cost function must be defined for each of the 3 criteria.

The cost for probability of detection is 1 minus the mean probability of detection for a given search grid (recalling equation 3.33). Note that PSO is not a gradient descent method, and so the derivative of the cost formulation has no affect on the optimisation. This cost can be equated as follows:

$$J_D = 1 - \frac{1}{n^{\mathrm{g}}} \sum_{j=1}^{n_y^{\mathrm{g}}} \sum_{k=1}^{n_x^{\mathrm{g}}} \boldsymbol{D}_{jk}\,. \tag{3.39}$$

Similarly, the cost for revisit time is given by the mean revisit time for a given search grid (recalling equation 3.38)

$$J_T = \frac{1}{n^{\mathrm{g}}} \sum_{j=1}^{n_y^{\mathrm{g}}} \sum_{k=1}^{n_x^{\mathrm{g}}} \boldsymbol{T}_{jk}\,. \tag{3.40}$$

For the fuel consumption cost, the cost is simply the mass of the fuel consumed (recalling equation 3.24):

$$J_m = m_{\text{consumed}} = m_{\mathrm{I}} - m_{\mathrm{C}_{n_\tau}}\,. \tag{3.41}$$

The input to the cost function was the UAV trajectory $\boldsymbol{x}$ defined by equation (3.14) in section 3.2.1 where most of the upper and lower values were outlined in either section 3.2.1 or in table 3.1. For a constrained optimisation problem there needs to be limits placed on all inputs. The limits for $\boldsymbol{\tau}$, $\boldsymbol{\Delta\psi}$, and $\boldsymbol{p}_S$ are heuristically chosen to allow for a singular segment to cover either a relatively short or relatively long distance.

However, certain constraints can not be applied to the cost function input and instead must be applied within. For example, in order to ensure that the UAV does not stall, each value for the velocity along a given polynomial must be checked. The constraints that are within the cost function are outlined in equation (3.42). The first of these checks that the airspeed of the UAV at each discrete point along the trajectory neither stalls nor exceeds the maximum airspeed. The second, checks that the thrust is greater than or equal to zero (i.e. no reverse thrust) and less than or equal to the available thrust. The third, checks that the load factor of the UAV is less than the maximum load factor. The final constraint checks that the UAV weight is not less than the minimum required UAV weight at the end of the trajectory. All of these combined are used to determine if a given polynomial is feasible.

$$\begin{aligned} v_{\text{stall}} \leq & v_w \leq v_{\max},\ w = 1,...,k_w\,, \\ 0 \leq & T_w \leq T_{\text{available}},\ w = 1,...,k_w\,, \\ & n_w \leq n_{\max},\ w = 1,...,k_w\,, \\ m_{\text{F}} \leq & m_{\text{C}_{k_w}}\,. \end{aligned} \tag{3.42}$$

Rather than check each discrete point, the maximum or minimum of each value (excluding the minimum load factor and maximum UAV weight) was used as the input to either equation (3.43) or (3.44) depending on whether the upper or lower bounds are being checked. These equations penalize the violation of a given constraint such that the further the value exceeds the bounds, the greater the violation cost. This penalty helps guide the optimisation algorithm towards feasible solutions.

$$c^{\text{l}}(x) = \begin{cases} \varepsilon\left(1 + \frac{x_{\min}-x}{x_{\max}-x_{\min}}\right)^2, & \text{if } x < x_{\min} \\ 0, & \text{otherwise} \end{cases} \tag{3.43}$$

$$c^{\text{u}}(x) = \begin{cases} \varepsilon\left(1 + \frac{x-x_{\max}}{x_{\max}-x_{\min}}\right)^2, & \text{if } x > x_{\max} \\ 0, & \text{otherwise} \end{cases} \tag{3.44}$$

where $c^{\text{l}}(x)$ and $c^{\text{u}}(x)$ are the violation costs for the lower and upper bounds respectively. $\varepsilon$ influences the magnitude of violation cost and was heuristically chosen to be $1e5$.

Recalling the issues that a polynomial in $x$ and $y$ has for changes in heading approximately greater than 45°, an additional violation cost is added. This cost is denoted as $c_\psi$ and is obtained with equations (3.43) and (3.44) where the minimum and maximum values were set to $-45°$ and $45°$ respectively.

To avoid a trajectory appearing within a trajectory, an additional violation cost was added. This cost ensured that the total absolute change in heading was less than 720° which prevented 2 full 360° rotations occurring within the trajectory. The total change in heading was obtained by summing the absolute change in heading for each segment. The cost is denoted by $c_{\Delta\psi}$ and obtained with (3.44) with the condition changed to $x \geq x_{\max}$.

Lastly, for the surveillance of a given area it is required that the whole area is covered. Therefore, an additional violation cost is applied to ensure that every node is visited. This cost is simply the sum of the total unvisited nodes scaled by a factor of $\varepsilon$ and is denoted by $c_R$.

Since these constraints are not applied on the input, there is no absolute guarantee that a trajectory can be obtained that will satisfy them. However, the cost for violating these constraints can be set to a degree that near guarantees that the solution will fall within the constraints (assuming such a trajectory is possible). The total violation cost $J_c$ is then the sum of each violation cost for every polynomial in the trajectory. This cost is equated as follows:

$$J_\mathrm{c} = c_R + c_{\Delta\psi} + c_\psi + c_V^\mathrm{l} + c_V^\mathrm{u} + c_T^\mathrm{l} + c_T^\mathrm{u} + c_n^\mathrm{u} + c_{m_\mathrm{C}}^\mathrm{l}, \tag{3.45}$$

where, for example, $c_V^\mathrm{l}$ is the violation cost for the lower bounds of the UAV airspeed $V$.

The vector of cost functions is given in equation (3.46). Only solutions that do not violate the constraints within the cost function are desired and thus the violation cost is added to the cost vector. This approach ensures that feasible solutions are considered more optimal than non-feasible solutions.

$$\boldsymbol{J} = [J_D, J_T, J_m] + J_\mathrm{c} \tag{3.46}$$

The multi-objective optimisation problem is then formulated as follows:

$$\begin{aligned}
\underset{\boldsymbol{x}}{\text{minimise}} \quad & \boldsymbol{J}(\boldsymbol{x}) \\
\text{subject to} \quad & \tau_\mathrm{min} \leq \tau_i \leq \tau_\mathrm{max},\ i = 1, ..., n_\mathrm{h}, \\
& \Delta\psi_\mathrm{min} \leq \Delta\psi_i \leq \Delta\psi_\mathrm{max},\ i = 1, ..., n_\mathrm{h}, \\
& \boldsymbol{p}_\mathrm{min} \leq \boldsymbol{p}_S \leq \boldsymbol{p}_\mathrm{max}, \\
& 0 \leq \mu_V \leq 1, \\
& \psi_{S_\mathrm{min}} \leq \psi_S \leq \psi_{S_\mathrm{max}}, \\
& 1 \leq k_\tau \leq 1.1579, \\
& h^\mathrm{p}_\mathrm{min} \leq h^\mathrm{p} \leq h^\mathrm{p}_\mathrm{max}
\end{aligned} \tag{3.47}$$

### 3.3.2 Multi-Objective Particle Swarm optimisation

Outlined by Kennedy and Eberhart [69], particle swarm optimisation is based on social behavior of swarms and can be used for global optimisation. For a swarm of $p$ particles, the position of particle $d$ within the variable space is given by $\boldsymbol{x}^d$. At each iteration, the position is updated with a velocity based on its own best position as well as the best position of the swarm. The position update is equated as follows:

$$\boldsymbol{x}_{k+1}^d = \boldsymbol{x}_k^d + \boldsymbol{v}_{k+1}^d, \tag{3.48}$$

where the velocity is updated as follows:

$$\boldsymbol{v}_{k+1}^d = \omega \boldsymbol{v}_k^d + c_1 \boldsymbol{r}_1 \circ (\boldsymbol{p}_k^d - \boldsymbol{x}_k^d) + c_2 \boldsymbol{r}_2 \circ (\boldsymbol{p}_k^g - \boldsymbol{x}_k^d). \tag{3.49}$$

$c_1$ and $c_2$ represent the cognitive learning factor and the social learning factor respectively. In essence, these represent the weighting of the attraction between particle $d$'s best position ($\boldsymbol{p}_k^d$) and the swarm's best position ($\boldsymbol{p}_k^g$), both of which are the best position in the variable space from all previous iterations. $\omega$ is the inertia value which weights the affect of the previous velocity. $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ are vectors of the same length as $x$, containing random numbers

Table 3.2: Multi-objective particle swarm optimisation parameters

| Parameter | Value |
|---|---|
| $c_1$ | 1.5 |
| $c_2$ | 1.5 |
| $\omega$ | 0.3 |
| Maximum iterations | 1000 |
| Swarm size | 3600 |
| Maximum Pareto front size | 250 |
| Mutation probability | 1/length($\boldsymbol{x}$) |

with range $[0,1]$ with uniform distribution.

The original PSO algorithm only optimises for one variable so given the nature of the problem, a multi-objective PSO (MOPSO) algorithm was required. The algorithm selected for in this instance is OMOPSO [96] which was shown to be the most salient [97] of several MOPSO algorithms. Given that more than one cost function is being optimised for, there is more than one solution that could be considered optimal. A solution is considered Pareto-optimal if there exists no alternative solution where any of the costs will be further minimised whilst none of the costs are further maximised. The set of Pareto-optimal solutions is known as the Pareto front. OMOPSO uses crowding distance to reduce the size of the Pareto front and also uses mutation to diversify the swarm search. For the mutation, the swarm is subdivided into three sets of equal size with each subset having a different mutation scheme applied (no mutation, uniform mutation, and non-uniform mutation). The global best used in equation (3.49) is then selected using the particle with the least particles dominated, and in the case that multiple particles meet this criteria, then the particle with the maximum crowding distance is chosen (failing a singular choice, a particle is then chosen randomly from those that meet both criteria).

For high dimensional search spaces, the initialization of the particles can have a significant effect. A method to improve the initialization is outlined by Richards and Ventura [98] which uses centroidal Voronoi tessellations. The parameters used within the OMOPSO algorithm are outlined in table 3.2.

## 3.4 Results

This section outlines the results for two commonly seen scenarios for airborne maritime radar surveillance. The first, scenario A, is a large square area whilst the second, scenario B, is a thin curved coastal strip. For these scenario, the number of heading polynomials $n_\text{h}$ was taken to be 6 and 7 respectively with a grid spacing $\Delta G$ taken to be 2000 m. Furthermore, it is assumed for both scenarios that the radar was stabilised against platform motion.

For each of these scenarios, a Pareto front is obtained with several examples plotted. The Pareto front would allow a trajectory to be selected for a given mission based on further criteria (e.g. minimum revisit times, minimum probability of detection, and total surveillance time required).

The results were obtained with the use of MATLAB used in conjunction with a i7-6700 processor and 16GB of RAM. For each scenario, the simulation was run several times in order to ensure repeatability amongst the solutions. The average execution time for scenario A was 709171 s ( 8.2 days) while for scenario B it was 611845 s ( 7.1 days). Whilst these times show the significant computational cost, this method is intended to be used with predefined search areas where the optimisation is performed once in advance. It is also worth recalling that the persistent surveillance trajectories optimised here are several hours longs, which partially explains the computational cost. The execution time could be improved by using more powerful hardware, a compiled programming language, and/or reducing several of the scenario and optimisation parameters.

### 3.4.1 Scenario A

The scenario being optimised for in this case involves a large rectangular grid, specifically 400 km by 400 km, such that the full radar beam ($\gamma = 360^\circ$) fits within the area. For this scenario, the constraints for $\tau_i$ were heuristically chosen as $\tau_{\min} = 300\,\text{s}$, $\tau_{\max} = 2500\,\text{s}$.

For this search area, constraining the start position to one quadrant of the area has zero effect on the results. As such, the upper and lower limits on $\boldsymbol{p}_S$ were simply set to 200000 m and 0 m respectively. For the same reason, the upper and lower limits for the initial heading $\psi_S$ were set to $\pi/4$ and $-3\pi/4$ (i.e. along the diagonal of the search area). Additionally, and since trajectories with a total absolute change in heading greater than 720° were undesired, the limits of $\Delta\psi$ were heuristically set to $\pi$ and $-\pi/4$ to prevent the trajectory doubling back on itself.

The Pareto front obtained from the optimisation is shown in Fig. 3.5. It was observed that the majority of the trajectories fell under three broad archetypes based on their geometric similarity. Each archetype consists of a series of segments which alternate between a large turn and a small turn (or no turn). The archetypes are differentiated by the number of turning segments which is 2, 3, and 4 for the racetrack, the rounded triangle, and the rounded square respectively. The fact that every trajectory within each archetype shared a common geometric theme is an interesting result as opposed to their being a large variety of differing solutions. Furthermore, the geometrical simplicity of the archetypes would make them more likely to be accepted for practical use over intricate trajectories that are far removed from the current industry standards.

In general, each archetype occupied an area within the Pareto front. Specifically, for probability of detection, revisit time, and fuel consumption, the rounded square offered low to medium values for each. Additionally, the rounded triangle provided medium to high values, whilst the racetrack only offered high values. Three trajectories were selected from the Pareto front to highlight these archetypes as shown in Fig. 3.4.

Trajectory $(a)$ shows a rounded square path which provides low fuel consumption and low revisit times at the cost of low probability of detection. By flying close to the center of the search area, the total distance traveled is minimised. For this trajectory, the UAV flies at an altitude of 12715 m and an airspeed of $128.84\,\mathrm{m\,s^{-1}}$ which is slightly above than the airspeed for maximum endurance. This airspeed further reduces the fuel consumption while also improving the revisit time.

Trajectory $(b)$ shows a rounded triangle path which has both a high probability of detection and low revisit time at the cost of high fuel consumption. The path covers a large distance thus increasing the probability of detection while also maintaining a short distance to the center which lowers the revisit time. With an altitude of 11021 m, the area covered by the radar is nearly maximised at the cost of fuel consumption. However, the UAV flies at an airspeed of $112.31\,\mathrm{m\,s^{-1}}$ which is just below the maximum endurance speed. This speed helps minimise the fuel lost due to the low altitude and large path.

Trajectory $(c)$ shows a large but narrow racetrack path. The narrowness allows for the radar to cover regions of the area to both sides at a given point in time. However, the path must travel nearly the length of the area in order to cover the central regions. This results in a high probability of detection at the cost of higher revisit time and higher fuel consumption. Similar to trajectory $(b)$, the altitude is 11007 m which maximises the area covered by the radar. Conversely, the UAV flies at an airspeed of $93.07\,\mathrm{m\,s^{-1}}$ which is $19.33\,\mathrm{m\,s^{-1}}$ lower than the maximum endurance speed. As a result of this speed, the probability of detection is further increased with both the revisit time and fuel consumption increased.

To highlight a trajectories behavior, the time history of trajectory $(b)$ is shown in Fig. 3.6. Most notably, there are fluctuations in the return segment polynomials which is the main drawback of the $x$ and $y$ polynomial. However, this drawback is accepted due to the need for the UAV to return to the initial position with the same heading as the initial heading.

In order to highlight the performance of these trajectories within this scenario, a baseline trajectory was used for comparison. Based on industry recommendations, a circular trajectory was used as a baseline with the diameter equal to half the length of the square area. This trajectory is shown in Fig. 3.10 and Fig. 3.11. Additionally, the UAV was flown at the maximum endurance speed at an altitude of 11000 m. It was found that 3 trajectories from the Pareto front were better than the baseline trajectory with regards to all 3 cost functions. This few number of trajectories shows that the baseline is not too far from the optimised trajectories, though there is still improvement that can be made. For example, one of these trajectories (not shown) offered a 2.54 % decrease in revisit time, a 2.40 % decrease in fuel consumption, and a 0.16 % increase in probability of detection. Additionally, the Pareto front offers trajectories in any direction in terms of cost (e.g. a trajectory which saves more fuel at the cost of probability of detection).

### 3.4.2 Scenario B

In this scenario, the trajectory is to be optimised for the surveillance of a coastal strip. A curved coastal strip was set up where the curvature was assumed to be circular. For this scenario, it is commonly required for the radar to operate with a sector scan pointing to one side of the UAV. Based on this scenario, the angular position $\delta$ was set to $\pi/2$ rad, the angular width $\gamma$ set to $\pi/3$ rad, and the scan rate set to $\pi/18\,\mathrm{rad\,s^{-1}}$. For this case, the trajectory will simply move around the coastal strip. As a result, several heuristics were employed to reduce the search space and thus execution time. The starting position $\boldsymbol{p}_S$ was constrained to start at some point on the north-west side of the coast with the initial heading $\psi_S$ fixed in the direction of the coastal curvature. The upper and lower bounds of $\tau_i$ and $\Delta\psi_i$ were then set such that there would be two large segments (for turning around the coastal strip) and several smaller segments on either side. The turning segments were heuristically set with the following lower and upper bounds: $\tau_{\min} = 1500\,\mathrm{s}$, $\tau_{\max} = 3500\,\mathrm{s}$, $\Delta\psi_{\min} = -190°$, $\Delta\psi_{\max} = -160°$. Similarly for the smaller segments, the lower and upper bounds were set as $\tau_{\min} = 100\,\mathrm{s}$, $\tau_{\max} = 1000\,\mathrm{s}$, while the bounds for $\Delta\psi_i$ were set such that the value would be between $0°$ and $\pm 0°$.

The Pareto front for this scenario is shown in Fig. 3.8 with three selected trajectories shown in Fig. 3.7. Trajectory $(a)$ provides low fuel consumption and low revisit times at the cost of lower probability of detection. The UAV maintains a close distance to the search area which means a smaller area of the radar beam intersects the area. This close proximity results in a higher revisit time as less search area is covered by the radar at each point along the trajectory. In this trajectory, the UAV flies at an altitude of 110059 m which allows for the radar beam to intersect the surface at closer distances, thus allowing the UAV to maintain a close course. Additionally, the UAV flies at $125.23\,\mathrm{m\,s^{-1}}$ which is well above the maximum endurance speed. The result of this speed is a sacrifice of some fuel for a decrease in revisit time.

Trajectory $(b)$ provides both high probability of detection and low revisit time at the cost of high fuel consumption. By maintaining a distance that allows for the center beam to intersect the search area, the probability of detection is maximised. Furthermore, the UAV flies at a low altitude (11006 m) which further increases the probability of detection by reducing both the distance to the beam center and the range. With an airspeed ($100.61\,\mathrm{m\,s^{-1}}$) less than the airspeed for maximum endurance, there is more time on area for the radar at the cost of fuel consumption.

Trajectory $(c)$ provides high probability of detection by sacrificing fuel consumed and revisit time. The UAV flies at an altitude of 11000 m which provides the maximum probability of detection for the radar. With an airspeed of $92.91\,\mathrm{m\,s^{-1}}$ (well below the maximum endurance speed), more time is spend on each point within area, thus maximising the probability of detection at the cost of fuel.

In terms of path, the biggest difference between trajectory $(a)$, $(b)$, and $(c)$ is the distance between the turns at the tips. Trajectory $(a)$ turns sharply around the tips, to the point where there is significantly less probability of detection at the tips. Conversely, trajectory $(c)$ keeps a distance that allows the radar beam to intersect more of the outer edges of the search area. However, by keeping a far distance from the edges, the flight time is significantly increased which increases the revisit time and the fuel consumed. Trajectory $(b)$ provides a medium between $(a)$ and $(c)$.

The time history of trajectory $(b)$ is shown in Fig. 3.9. Notably, there are larger fluctuations in the return polynomial relative to the fluctuations shown in Fig. 3.6. This difference is largely due to the return polynomial having more curvature relative to the return polynomial in trajectory $(b)$ in scenario A. However, the fluctuation in airspeed for this case is still less than $1\,\mathrm{m\,s^{-1}}$ which has little impact on performance.

The industry recommended trajectory (shown in Fig. 3.12 and Fig. 3.13) for this scenario was such that the center of the beam intersects the center-line of the curved strip. Additionally, circular turns are performed at the ends of the area with the beam still intersecting the center-line. As with scenario A, the UAV was flown at the maximum endurance speed at an altitude of 11000 m. It was found that 97 trajectories from the Pareto front were better than the baseline in all 3 cost functions. This number suggests a great improvement in performance. For example, one of these trajectories offered a 74.33 % decrease in revisit time, a 9.05 % decrease in fuel consumption, and a 26.71 % increase in probability of detection.

In both scenario A and B, none of the trajectories had an altitude above 14500 m. This observation suggests that at higher altitudes, too much fuel is consumed in making up for the lost coverage than would otherwise be saved flying at these high altitudes.

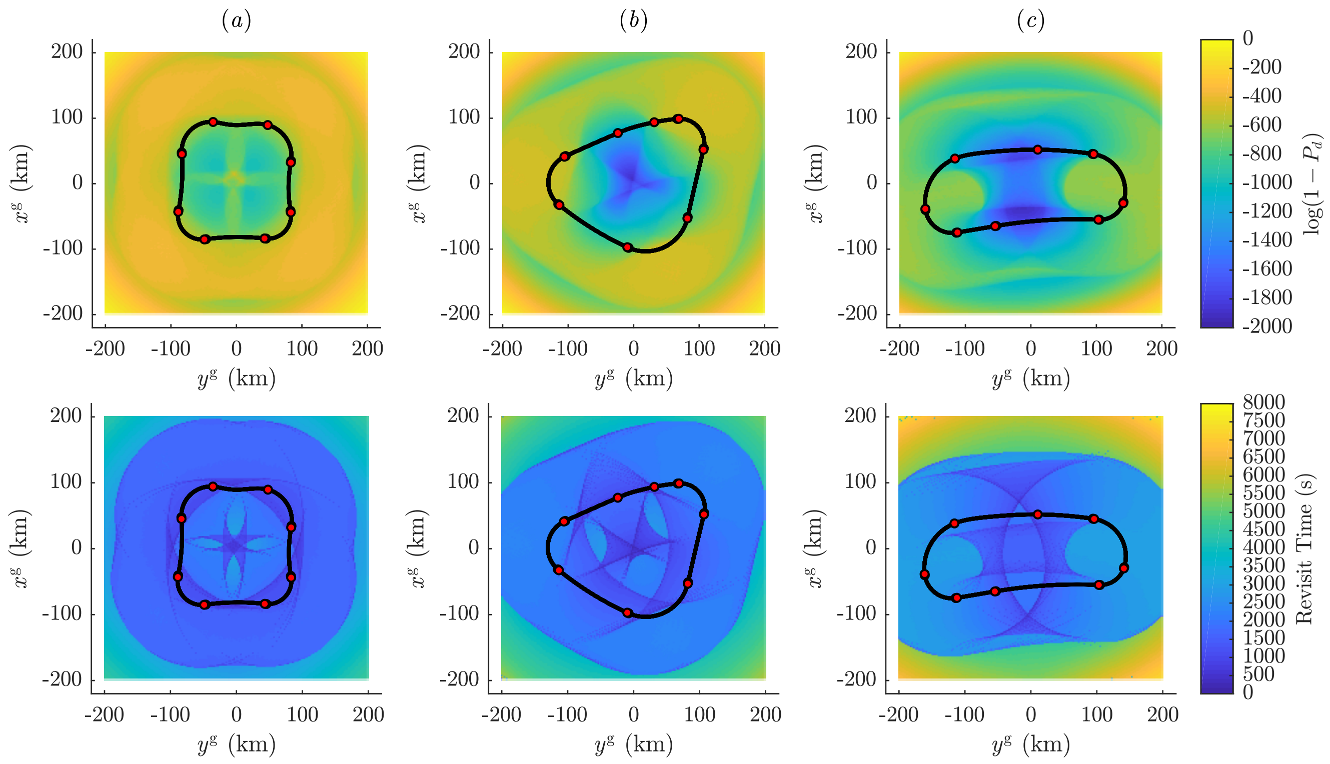


Figure 3.4: Selected Pareto front probability of detection grids (top row) and revisit time grids (bottom row) for scenario A. The black line indicates the UAV's path, while the dots indicate the transition points between polynomials. Note that a lower value in the probability of detection grid means higher probability of detection while a lower value in the revisit time grid means more frequent revisit.

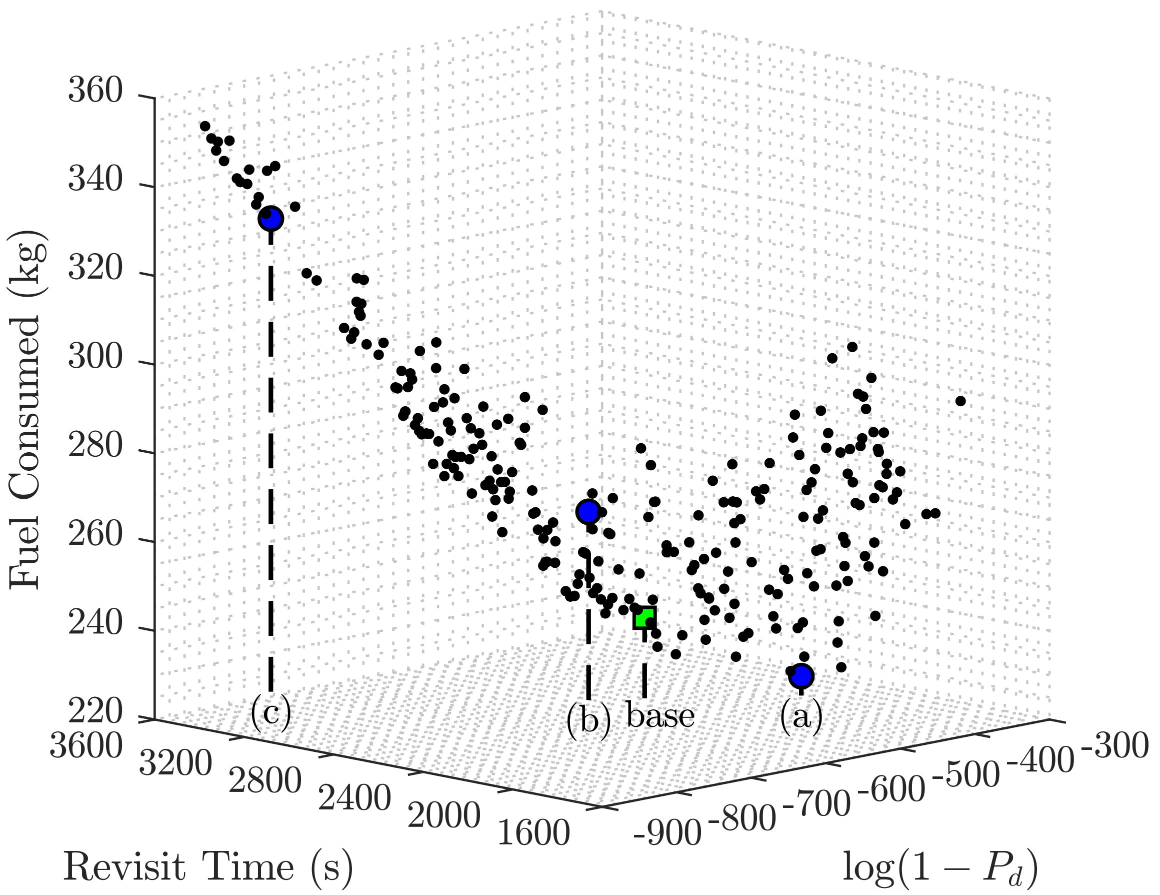


Figure 3.5: Pareto front for scenario A. Trajectories $(a)$, $(b)$, and $(c)$ from Fig. 3.4 are indicated by larger dots with the baseline trajectory indicated by the square.

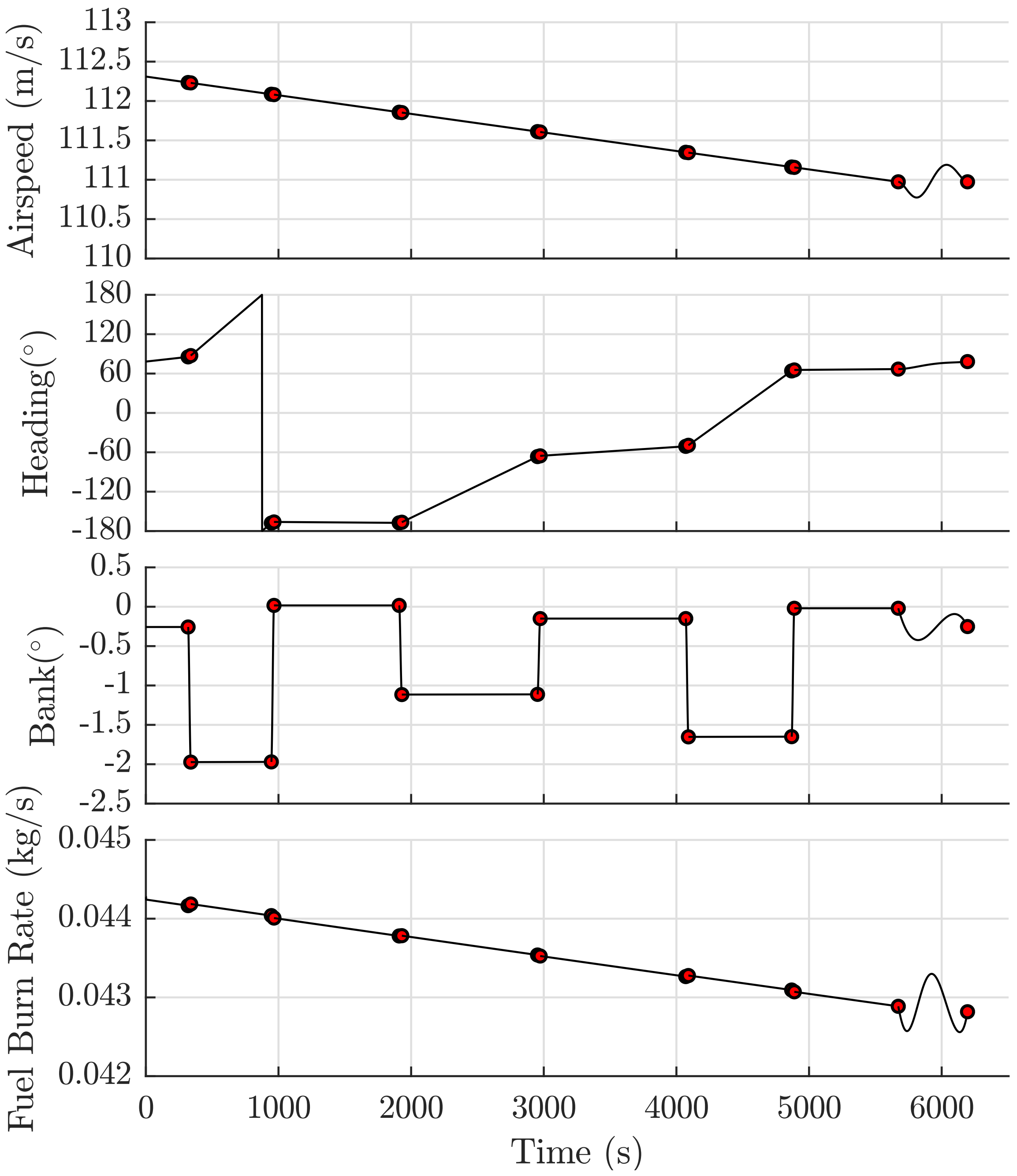


Figure 3.6: Time history for trajectory $(b)$ in scenario A. The dots indicate the end points of each polynomial. Note that the timescale hides the continuous nature of the transition between polynomials. Also note that the apparent discontinuity in heading between $-180°$ and $180°$ is only due to plotting the heading within these limits.

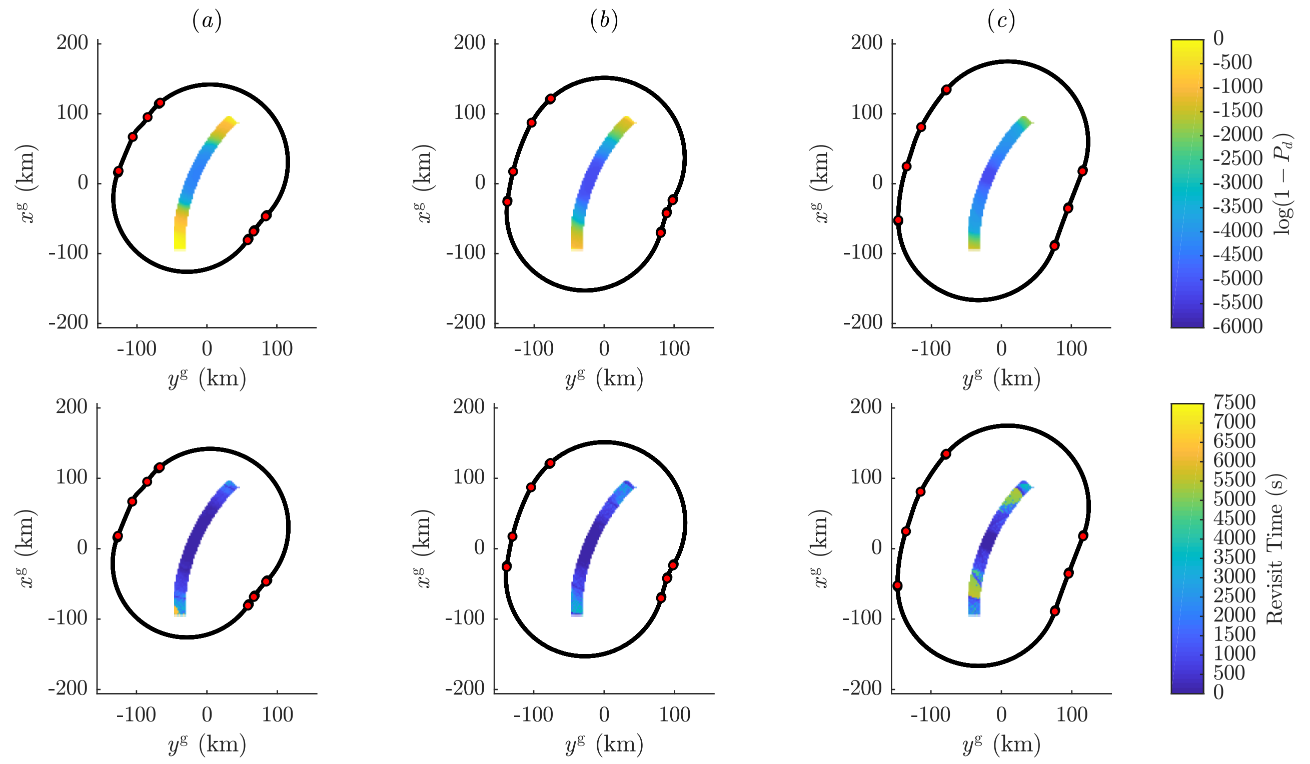


Figure 3.7: Selected Pareto front probability of detection grids (top row) and revisit time grids (bottom row) for scenario B. The black line indicates the UAV's path, while the dots indicate the transition points between polynomials. Note that a lower value in the probability of detection grid means higher probability of detection while a lower value in the revisit time grid means more frequent revisit.

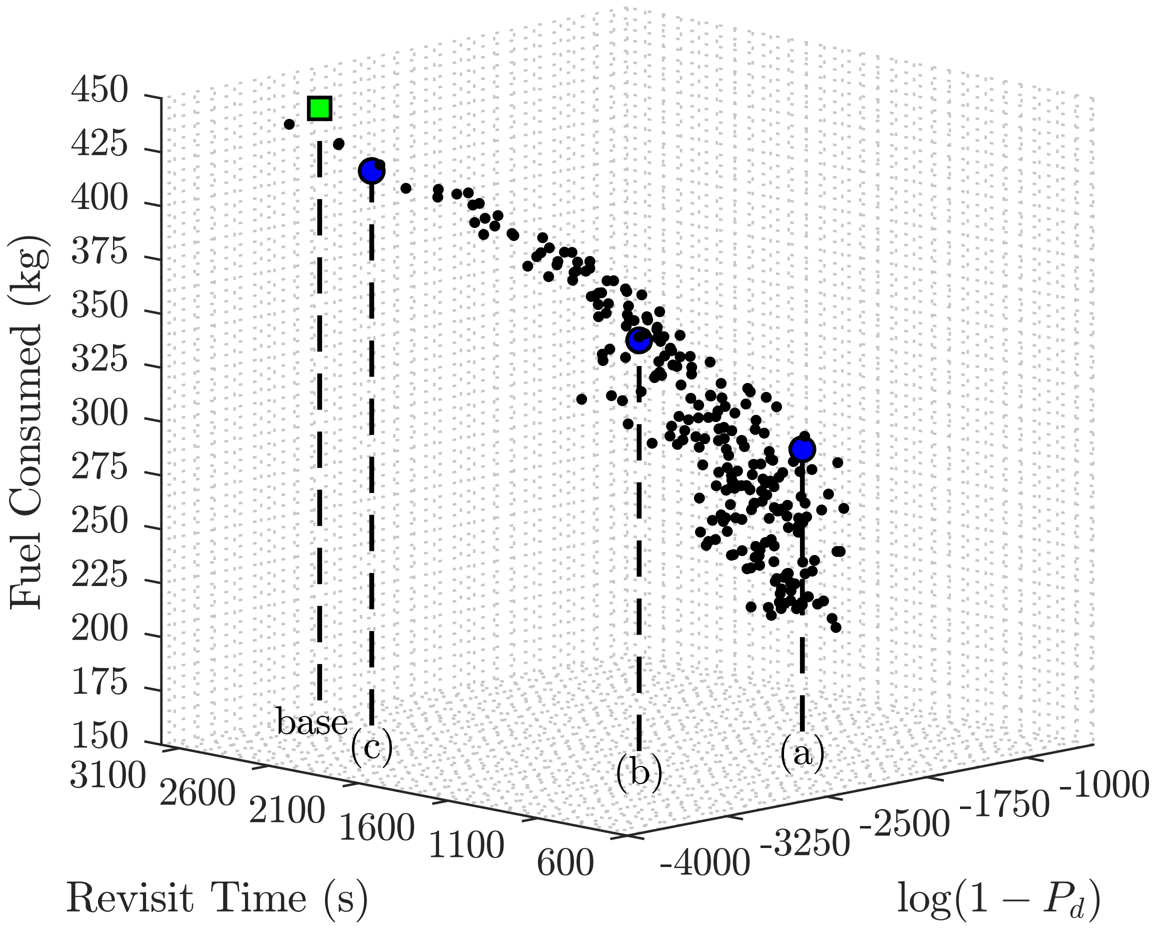


Figure 3.8: Pareto front for scenario B. Selected values shown in Fig. 3.7 are indicated by larger dots with the baseline trajectory indicated by the square.

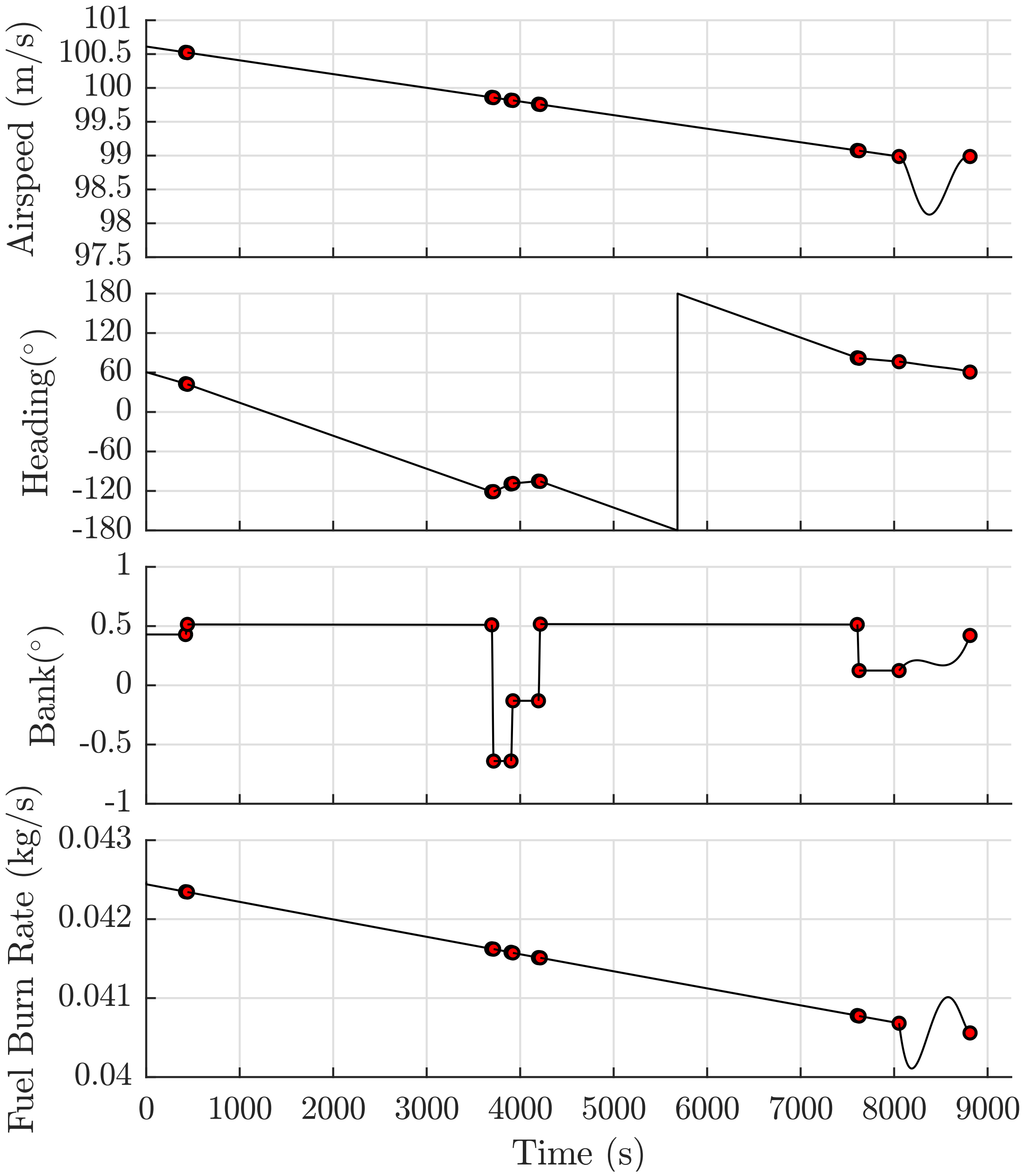


Figure 3.9: Time history for trajectory $(b)$ in scenario B. The dots indicate the end points of each polynomial. Note that the timescale hides the continuous nature of the transition between polynomials. Also note that the apparent discontinuity in heading between $-180°$ and $180°$ is only due to plotting the heading within these limits.

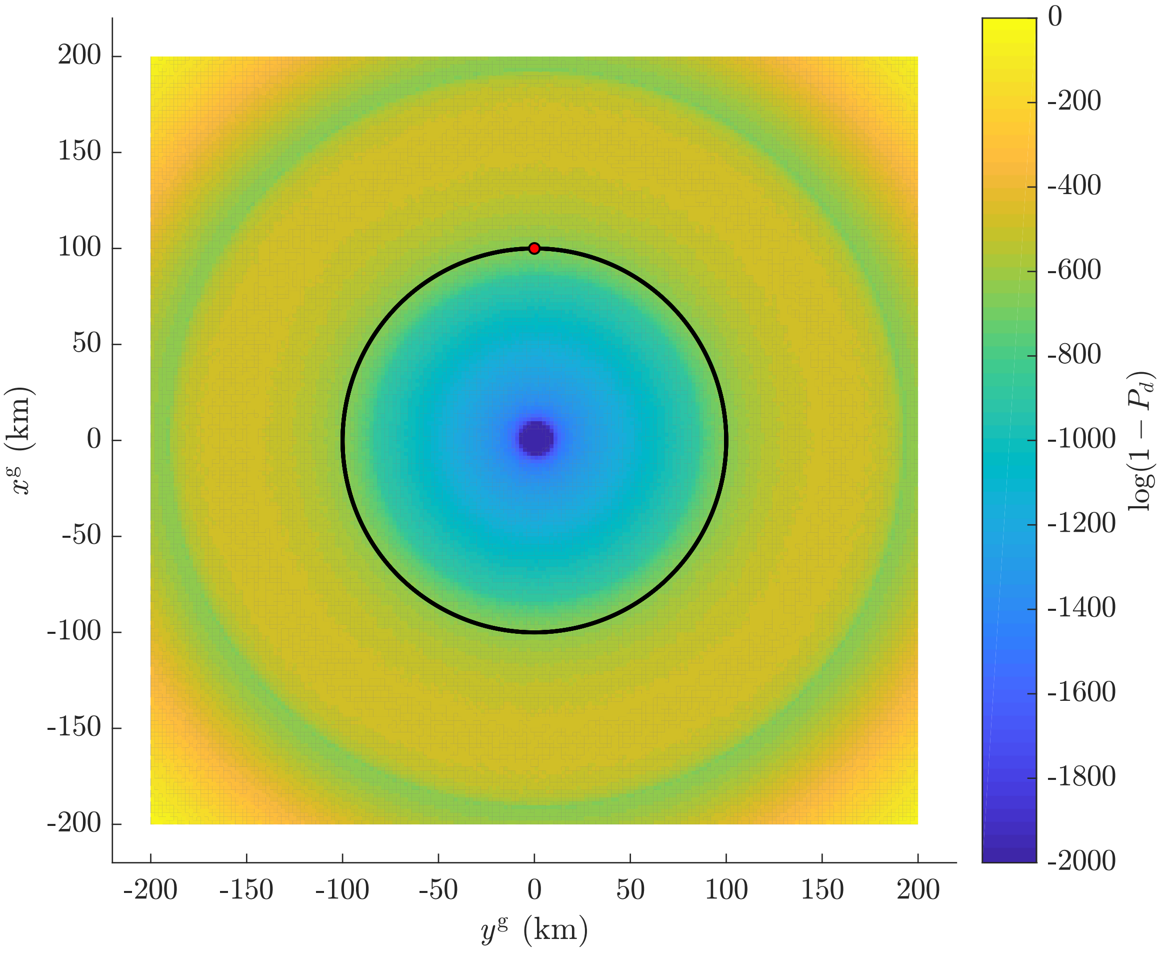


Figure 3.10: The probability of detection grid for the baseline circular trajectory.

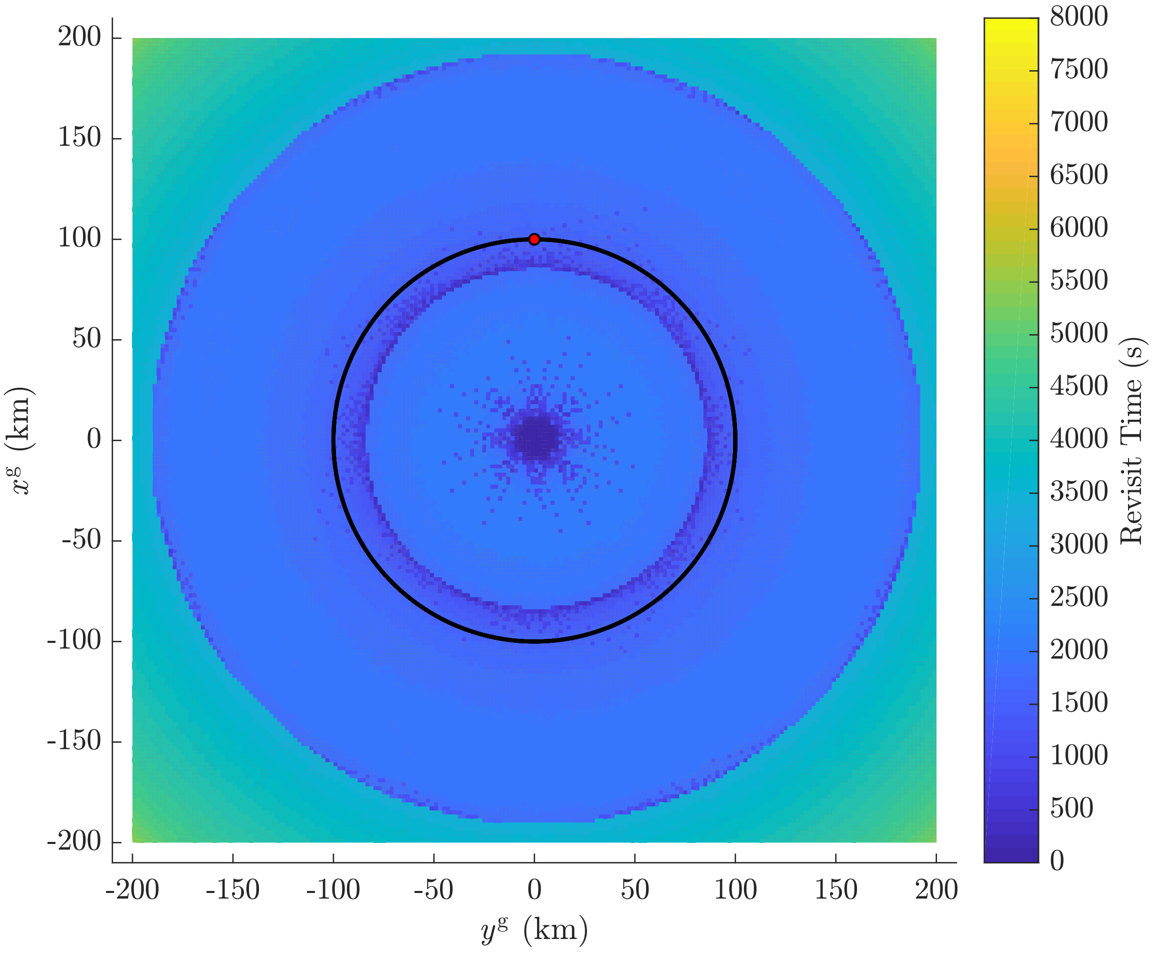


Figure 3.11: The revisit time grid for the baseline circular trajectory.

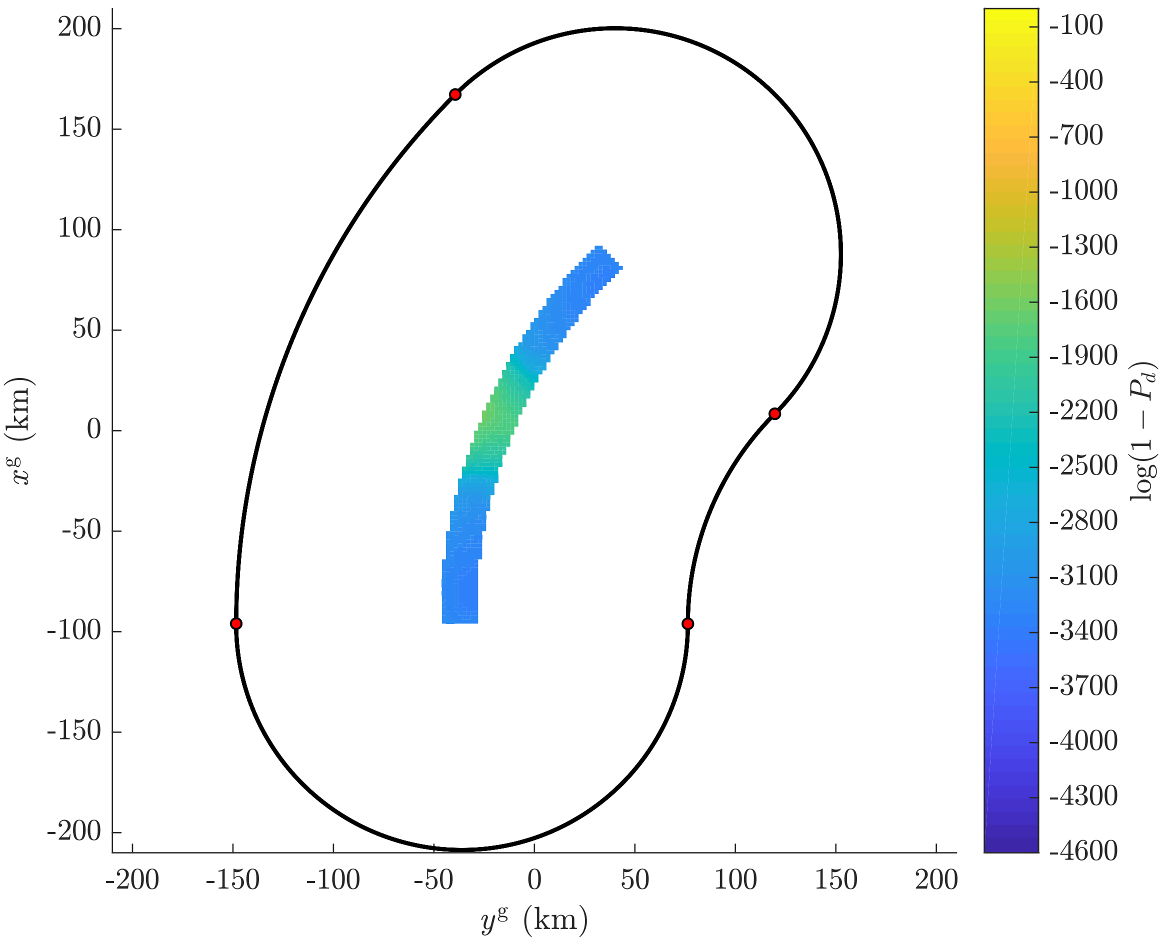


Figure 3.12: The probability of detection grid for the baseline curved area trajectory.

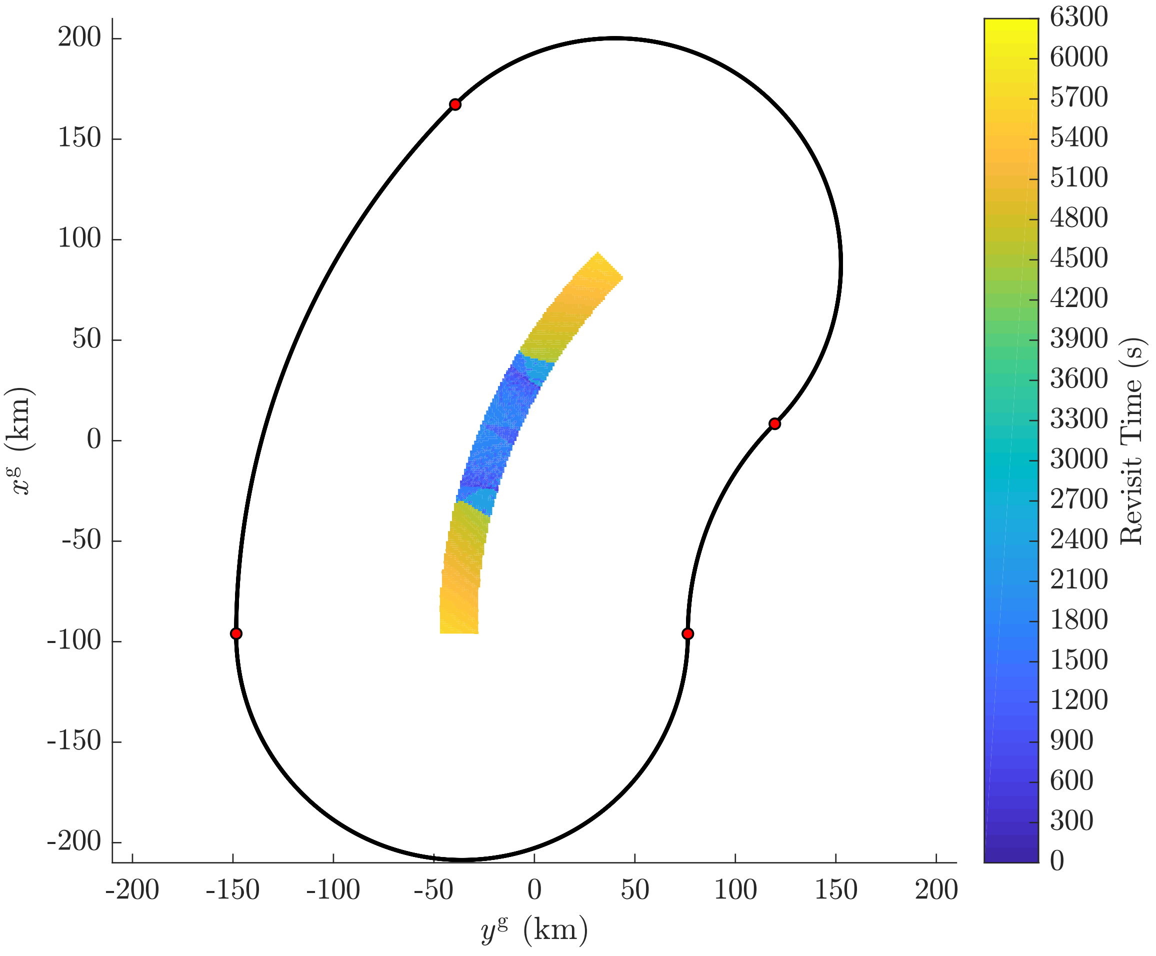


Figure 3.13: The revisit time grid for the baseline curved area trajectory.

## 3.5 Conclusions

This chapter outlines a method for trajectory optimisation for airborne maritime radar wide area persistent surveillance using a polynomial trajectory generation method. The considerations of the dynamics, propulsion, and mission requirements of a fixed-wing UAV, as well as a maritime surveillance radar, provide a method to obtain the fuel consumption, probability of detection, and revisit time for a given trajectory. The polynomial trajectory generation method provides a simple method to produce complex trajectories necessary to obtain the UAV dynamics for the fuel consumption, dynamic limitations of the UAV, and path. By discretising the search area and radar coverage area into grids, a computationally efficient way of obtaining the probability of detection and revisit time for each point in the grid is outlined.

Multi-objective particle swarm optimisation was used in conjunction with the cost function for probability of detection, revisit time, and fuel consumption, resulting in a Pareto front of trajectories that provide several suitable options for various UAV maritime radar surveillance mission requirements. The results are not just in terms of path, but also in terms of the altitude of operation and airspeed of the UAV. Results are obtained for two commonly seen scenarios. For scenario A, the results showed repeated geometrically similar paths within the Pareto front as opposed to a variety of differing paths. Additionally, 3 trajectories were found that were better than the industry recommended baseline in all three cost functions. For scenario B, 97 trajectories were found to be better than the baseline which suggests a significant improvement. For scenario B, the main advantage of the trajectories on the Pareto front is their sharp turns around the corner, increasing the relative time spent covering the long sides of the search area.

# Chapter 4

# Imitation Learning for Radar Operations

## 4.1 Introduction

### 4.1.1 Background

Chapter 1 outlined the need and benefits of a form of artificial intelligence that not only learns from an operator, but acts in the same manner. This branch of AI is known as imitation learning (also known as learning from demonstration, apprenticeship learning, behaviour cloning) which encompasses a wide variety of applications. Early surveys of imitation learning [99]–[102] focus primarily on robotics applications. Specifically, Schaal et al. [99] surveyed the imitation problem to motor learning with consideration to the following problems: movement recognition, pose estimation, pose tracking, body correspondence, coordinate transformation from external to egocentric space, matching of observed against previously learned movement, resolution of redundant degrees-of-freedom that are unconstrained by the observation, suitable movement representations for imitation, and modularization of motor control. In Billard et al. [100], the imitation learning problem is surveyed by separating the approaches into engineering-oriented approaches and biologically-oriented approaches. This survey is weighted heavily towards the robot motor learning problem. Argall et al. [101] attempt to establish a categorical structure to the imitation learning problem, with focus on robotics applications. In this survey, the problem is split into two phases: the demonstration and derivation phases. In the demonstration phase, the method of gathering demonstrations is categorised according to how the demonstration is executed and how the demonstration is recorded. In the derivation phase, the policy derivation methods are categorised according to what is learned: the state–action mapping (mapping function), a model of the world dynamics and/or reward (system model), or a model of action pre and post-conditions (plans). Ijspeert et al. [102] present and review the imitation learning problem with regards to non-linear dynamic systems, with particular regard to motor control and robotics.

However, these early surveys are weighted heavily towards the robotics and motor control domain. Whilst there is overlap between the imitation learning problem in robotics and radar operator imitation (such as acquiring demonstrations and sensory issues), the key difference is the level at which the imitation is applied. For the robotics applications, the imitation is at the low-level control in order to, for example, imitate joint manipulation. Recent advancements in computational power and increased demand for artificial intelligence has seen imitation learning applied to other domains. More recent surveys by Hussein et al. [103] and Bagnell [104] applies greater focus on these other domains which typically deal with high-level operational imitation. Specifically, these surveys focus on the computational methods used to learn policies that solve problems according to demonstrated behaviour. These methods can then be applied to forms of intelligent agents ranging from the aforementioned physical robot to a software agent.

In the robotics domain, imitation learning has been used such that a robotic arm imitates human demonstrations for object manipulation [105]–[107] and ball hitting [108], [109]. For robots with high degree of freedom such as humanoid robots, imitation learning have been applied [110], [111] such that these robots can learn discrete actions such as standing up, and cyclic tasks such as walking. Imitation learning has also been applied to learn low-level control from demonstrations in other platforms such as UAVs [112]–[115] and ground vehicles [116]–[121]. Navigational problems are one example of bridging the gap between the low-level actuator control and the high-level decision making. Imitation learning has been applied to this problem [118], [122], [123] in order to get a robot/vehicle to traverse complex uncertain terrain. More recently, imitation learning has been applied to video games [103], [124], [125].

Within the current literature, there are several reasons for using imitation learning algorithms. One reason is that imitation learning can outperform typical non-imitation machine learning methods. For example, in Guo et al. [126] imitation learning is applied (specifically a deep learning method) to the same Atari benchmark in Bellemare et al. [127] with greater performance achieved. Note that in this case, the demonstrations were from a high-performance non-real-time agent and not a human. Another reasons for the use of imitation learning methods is to try and understand and solve problems where the goals is hard to quantify. In other words, problems which are hard to programme but humans can easily provide demonstrations [108], [114], [128], [129]. Another reason that was previously mentioned in chapter 1 is that imitation learning can provide suitable performance with far less training required than traditional machine learning methods [27]. This reason is particularly useful where running the task itself comes at some form of cost. However, to the author's knowledge, there is no literature on the use of imitation learning as a means to streamline the validation process.

### 4.1.2 Problem Formulation

Imitation learning is somewhat analogous to supervised learning in that in supervised learning the examples are represented with features and labels much like the examples in imitation learning are represented with states and actions. In imitation learning, however, the state-action pairs are often part of a larger collective known as a trajectory. Though, similar to supervised learning, imitation learning aims to obtain a general mapping between the states and actions. Below are several definitions that relate to the imitation learning problem.

**Definition 4.1.** As per Russel and Norvig [130], "An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators".

**Definition 4.2.** A policy is a mapping between any state to an action. In other words, the agent will always know which action to take for a given state if it has a policy.

**Definition 4.3.** A demonstration is a pair of states and actions. The state is a vector of features describing the state at the instance the action was taken. The action is simply the action performed by the demonstrator.

**Definition 4.4.** The process of imitation learning is one by which an agent uses instances of demonstrations in order to learn a policy that performs the given task in the same manner as the demonstrator.

One of the most difficult aspects of any machine learning problem, and imitation learning is no exception, is the the problem representation. There are few problems where all potential states and actions can be accurately accounted for in a machine learning problem. For example, even in a simple task such as driving a remote controlled (RC) car there are countless variables beyond the obvious that may influence the actions taken by the operator. The operator will likely base his inputs to the RC car on the car's current position and any nearby obstacles. However, the operator might also use intuition to factor in the wind speed (if it is particularly strong), or the surface type. The RC car's position and any obstacles are easy to measure and represent within the problem, however, wind speed and surface type are difficult and impractical to measure accurately and their effects just as difficult to represent mathematically. Additionally, there may be environmental information that is irrelevant. In short, sacrifices and considerations have to be made in the problem representation such that the state representation is adequate for training but also efficient with regards to the learning time and any computational requirements.

### 4.1.3 Main Contributions

The contributions of this chapter are as follows:

1. Literature review of imitation learning algorithms and outlining the considerations required for the task of imitating a human radar operator. Suitable algorithms for this problem are then selected.

2. Definition of three typical maritime radar surveillance missions and how these are consequently simulated within an interactive radar model in order to collect demonstrations for the imitation learning algorithm.

## 4.2 Imitation Learning Methods

### 4.2.1 Overview

This sections gives an overview of the different types of imitation learning methods. Figure 4.1 shows a Venn diagram of these different imitation learning methods, outlining the different data sources used by the methods. The purely 'trial and error' methods (i.e. reinforcement learning and optimisation) rely on the agent being able to repeatedly test its actions within the state-space and compare its performance with the operator's performance. The purely 'human expert' methods (i.e. active learning) requires a dedicated demonstrator that can be constantly queried for demonstrations. There are obviously no purely 'observation' methods as this would be equivalent to purely observing the state with no operator input of any form.

From the Venn diagram, transfer learning lies between 'observation' and 'trial and error'. The transfer learning method uses knowledge learnt from other tasks or agents in order to learn a policy for the given task. Apprenticeship learning combines all three sections by using operator demonstrations and observations of the environment in order to learn a reward function. An inner-loop use of reinforcement learning ('trial and error') is then used in order to obtain a policy from that reward function. Between 'human expert' and 'observation' lies the supervised learning and structured learning methods. Structured learning (or structured prediction) encapsulates machine learning methods that learn structured objects rather than scalar values. For example, if the machine learning method is required to output a sentence (i.e. a series of words), then every possible combination of words could form the output, as would be the case in a classic supervised learning problem. However, the output could also be defined as a structure of words, in which each word is mutually dependent on each other word. In this case, structured learning methods can be used. These methods are more of an expansion to the supervised learning methods as they typically use a supervised learning sub-routine. Due to the compounding error induced by the covariate shift [125], [131], supervised learning tends to only achieve acceptable results when large amounts of data are used. A technique to overcome this is to frame the problem as a structured prediction [125], [132], and then query the demonstrator in order to create new training data whilst the agent is performing within the state-space. This technique is known as structured learning. However, structured learning suffers from the same issue as active learning: the demonstrator is required to be available to be queried by the imitation learning algorithm, which is often not possible. Le et al. [133] provide an algorithm that uses a "virtual" expert to provide feedback and thus "smooth" the state-trajectory, negating the use of continuous demonstrator feedback. However, this algorithm is designed for dynamic and continuous state-spaces which are not the types of problems considered here.

Figure 4.2 shows the flow of the imitation learning process with the methods associated with each stage. In particular, the distinction between the ’learning from demonstration’ stage and the ’refining policy stage’ is worth noting. The ’refining policy’ stage is particularly advantageous when there are errors in the demonstrations or their acquisition, or the application of the task significantly differs from the demonstrated task due to environmental or agent differences. However, these types of methods often require the agent to be able to carry out many trials within the state space which can be costly and difficult.

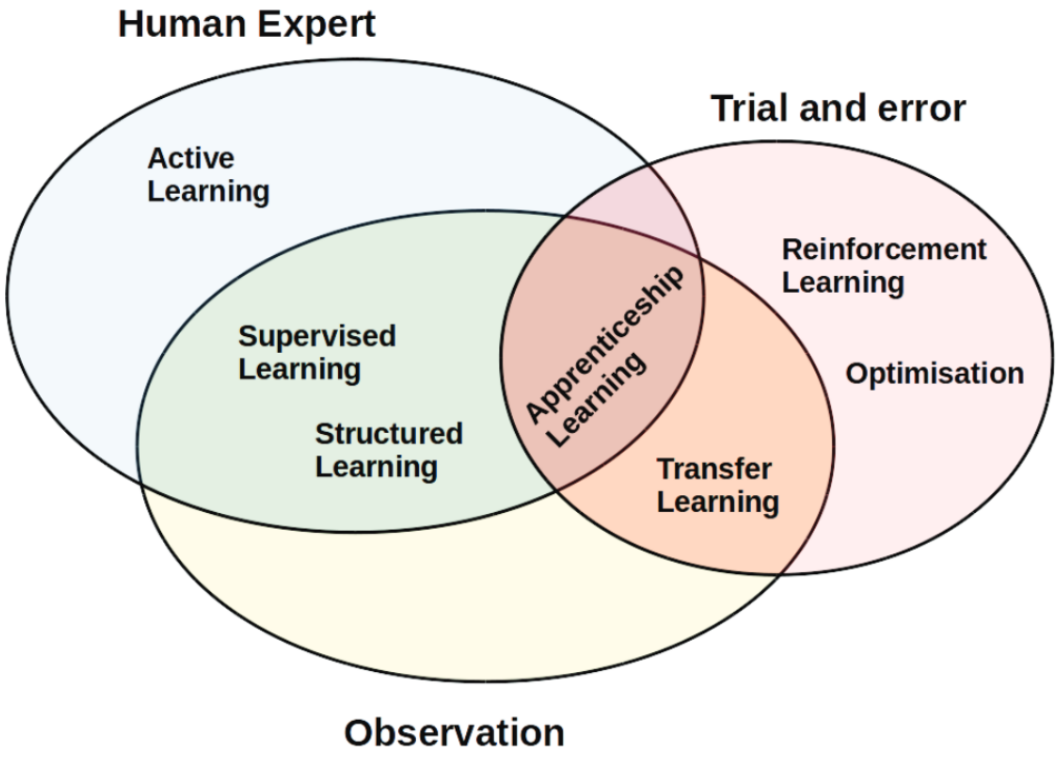


Figure 4.1: A Venn diagram of imitation learning methods [103].

### 4.2.2 Algorithm Selection

In addition to requiring the agent to carry out many trials, the purely ’trial and error’ imitation methods are typically used for refining a policy rather than actual policy determination itself. For these reason, the purely ’trial and error’ methods are not considered for this research. As mentioned previously, active learning and structured learning require the demonstrator to be constantly available to be queried. This is not possible given the setup for the problems considered in this research and, consequently, active learning and structured learning are not considered. Furthermore, since the problem dealt with in this research has not been considered before, transfer learning is not possible since there are no similar tasks and demonstrations to learn from.

By the process of elimination, supervised learning and apprenticeship learning (also known as inverse reinforcement learning) are the two remaining method types that are suitable for the tasks considered in this research.

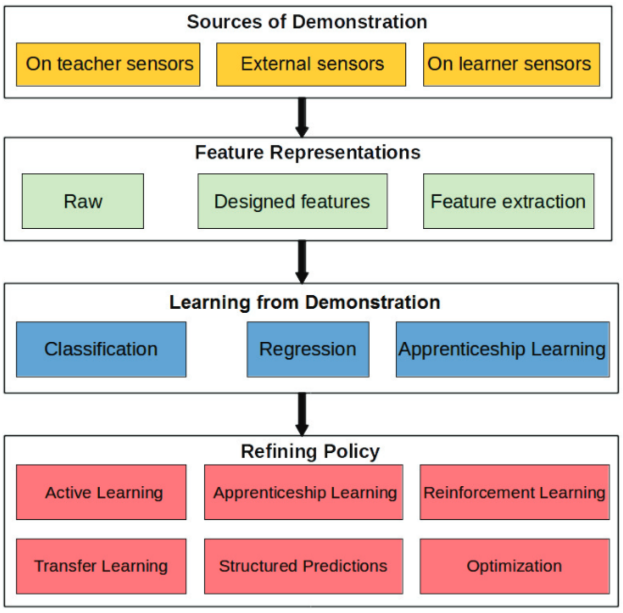


Figure 4.2: A flow chart outlining the stages of imitation learning and the respective methods used. [103].

In addition to whether the algorithm would work with the considered task setup, there are additional features to consider when selecting an algorithm for the use of imitation learning for the ease of qualification and validation processes. As mentioned in chapter 1, the algorithm needs to be transparent in its decision making in order to ensure the trust of the operator. This allows the operator to diagnose where in the process the wrong decision occurred and why that wrong decision was taken.

Additionally, the state-action policy generated by the algorithm is required to be deterministic. This requirement is an extension of the required transparency mentioned above as the policy cannot be transparent if it provides a different action for the exact same state. Furthermore, for defence applications such as the problem considered here, there is often a blanket ban on any non-deterministic algorithms replacing an aspect of operation currently under control by the human radar operator. Therefore, requiring the algorithm to be deterministic meets the requirements for transparency, but primarily allows the algorithm to be considered for operation in the first place.

As mentioned in chapter 1, for the case of radar operator data, obtaining data is a difficult and expensive procedure. Ideally the algorithm would require as few samples as possible in order to reach an operational performance deemed close enough to that of the operator.

### 4.2.3 Supervised Learning

Supervised learning methods map an input state to an output, and are categorized into classification and regression methods. A supervised learning method will learn this mapping between input and output by inferring a function from labelled training data. Ideally, the supervised learning algorithm will produce a function that can generalise for unseen input data.

### Regression

Regression methods are used for problems that deal with a continuous action space, for example a robotic actuator. If the training data consists of independent state values $s_i$ and continuous action values $a_i$ (for $i = 1, \dots, n$, where $n$ is the number of samples in the training data), then the regression algorithm will produce a function mapping any value of $s$ to $y$. From the imitation learning examples mentioned above, some of the problems that deal with low-level actuator control use a regression method. For example, in Muelling et al. [134] a mapping to a continuous action space provides a smooth trajectory given that the state-space is a high degree-of-freedom robot operating in a 3-dimensional space.

However, since the tasks considered here deal with actions in the discrete space, regression methods are not considered.

### Classification

Classification methods are used for problems where the agent's actions can be categorised into a finite set of discrete classes. The training data consists of $n$ independent state value vectors $\boldsymbol{s}_i$ where $\boldsymbol{s} \in \{s_1, \dots, s_m\}$ (with $m$ being the length of the state value vector) and $n$ action classes $a_i$ where $a \in \{a_1, \dots, a_k\}$ (with $k$ being the number of action classes). The classification algorithm then learns from the training data to obtain a mapping between any input state value vector $\boldsymbol{i}$ and an action class $a_i$.

As previously mentioned, these methods are used to deal with problems where the agent's actions can be considered part of a set of discrete classes. A simple example would be a human radar operator observing targets (such as their position and speed) and consequently pressing buttons (such as applying a sector scan on a track). Within the literature, classification methods, of many forms, have seen abundant use for imitation learning. In Chernova and Veloso [120], a Gaussian mixture model is used as a classifier for a corridor navigation problem and a car lane changing problem where both problems have uncertain human demonstrations. In Sammut et al. [112], decision trees are used in order to learn flight controls from a pilot flying an aircraft. Meta classifiers containing neural networks are used by Ross and Bagnell [131] in order to learn how to play certain computer games. In Raza et al. [135], neural network approaches and decision trees are used as classifiers for a multi-agent soccer simulation. The classifiers take input states such as distances from the ball, goal, and teammates in order to obtain actions such as 'go to penalty area' and 'dribble towards goal'.

Whilst there are many forms of classification algorithms, there are certain considerations required for this application of imitation learning, namely the need for transparency. Neural network based approaches (particular deep learning methods) are not well suited to this type of task due to their lack of transparency, and for this reason, were not considered. Decision trees, whilst transparent break each decision into binary options, which is not suited to the problems considered here. Additionally, the structure of the decision tree is similar to that of the Bayesian networks, with Bayesian networks having the flexibility to have any number of inputs and outputs to a decision node. Gaussian mixture models assume the inputs are of a Gaussian distribution which cannot be assumed for the problems considered here. Consequently, Bayesian networks (discussed in chapter 5) were considered for this research due to their transparency and well-suited structure.

### 4.2.4 Apprenticeship Learning

As previously mentioned, another imitation learning approach that is suitable to the problems considered here is apprenticeship learning, otherwise known as inverse reinforcement learning (IRL) [136], [137]. Unsurprisingly, this approach involves reinforcement learning, and so it is necessary to cover the fundamentals of reinforcement learning. At the core of reinforcement learning is the Markov decision process:

**Definition 4.5.** A Markov decision process (MDP) is a discrete-time stochastic process representing a given state-action space. A finite MDP typically consists of a tuple $(S, A, T_{\mathrm{a}}, R_{\mathrm{a}})$, where $S$ is the set of states, $A$ is the set of actions, $T_{\mathrm{a}}$ contains the probabilities of transitioning between states given action $a$, and $R_{\mathrm{a}}$ is the set of rewards obtained from taking action $a$ at state $s$.

In reinforcement learning, the aim is to take the action that maximises the expected reward at each step in the process. In inverse reinforcement learning, the aim is to use operator demonstrations to infer a reward function. The inferred reward function can then be used with reinforcement learning to obtain a policy that imitates the operator. A more complete overview of a Markov decision process, reinforcement learning, and inverse reinforcement learning is provided in chapter 6.

As previously mentioned, there are some problems where the goal can be hard to quantify but easy to obtain expert demonstrations. IRL has been extensively used in various car driving problems such as lane changing [137], [138]. In Lee et al. [139], inverse reinforcement learning is used for the game of Super Mario. And in robotics, IRL has been applied to helicopter aerobatics [128] and robotic manipulators [140].

## 4.3 Task Setup

National fishing areas are sometimes infiltrated by fishing boats performing illegal fishing operations. The monitoring of these fishing areas is therefore required to prevent future illegal operations and intercept ongoing illegal operations. With a surveillance radar onboard an airborne platform, there are several tasks that the human operator does as part of this monitoring process. The first step of the imitation learning work flow is to therefore define these tasks for which the imitation learning algorithms learn to perform. This section outlines three tasks which were selected for testing and analysis that fully encapsulate the maritime radar surveillance process. The first task involves selecting a track from the search area to investigate based on its properties. The second task involves the investigation of the track itself, which results in decisions made on the track and consequently interacting with the track. The final tasks involves the movement towards a track based on the returns from an ISAR image.

The second step in the imitation learning process is to acquire the operator demonstrations. This section also outlines the use of the real-time radar simulation outlined in chapter 2 in order to obtain the scenario and radar information that would be observed by a human operator carrying out a real mission, and also to obtain operator decisions for the given maritime surveillance radar missions. This scenario/radar information and the corresponding operator decisions compromise the state-action pairs for the imitation learning algorithms.

### 4.3.1 Task 1

The first task in the maritime surveillance radar mission starts with the operator being presented with the plan position indicator (PPI) display with several tracks and outlined fishing zones. The operator is then tasked with choosing which tasks to investigate and in what order. The operator observes the position, direction, and transponder information for each track which is displayed next to each track marker as shown in Fig. 4.3. The operator selects a track by clicking the marker with the mouse, and the operator's decision to further investigate this track is saved. Specifically, further track investigation means if the operator decides to apply a sector scan or communicate with the vessel. At the time of saving the data, the information the selected track as well as the remaining tracks is saved as the operator is deciding to select that track from as opposed to the other remaining tracks. Within the GUI, previously selected tracks will appear green to avoid confusion.

It is desirable to investigate the tracks in fishing zones first and then only investigate tracks outside the fishing zone when there are no tracks in the zones left to investigate. Additionally, it is desirable to investigate tracks that are unknown (i.e. the track has no transponder information). If all the remaining tracks are known, it is advantageous to look at the abnormalities in the size and speed of the track. The abnormalities are where the measured size and/or speed differ from the AIS information. Tracks which differ in both size and speed would likely be of greater priority than a track which only differs in either size or speed. If there are no abnormalities, the tracks with certain size and speeds may indicate certain behaviour. For example, if the track is moving fast and moving towards a fishing zone, the operator is more likely to investigate track rather than a slow track.

### 4.3.2 Task 2

The second tasks deals with the track selected for investigation. Given the tasks properties, the operator has to decide whether sector scan and/or communication with the track is required. If only sector scan is required, the decision process stops and the operator returns to select another track. If the operator decides to communicate with the track, the response—or

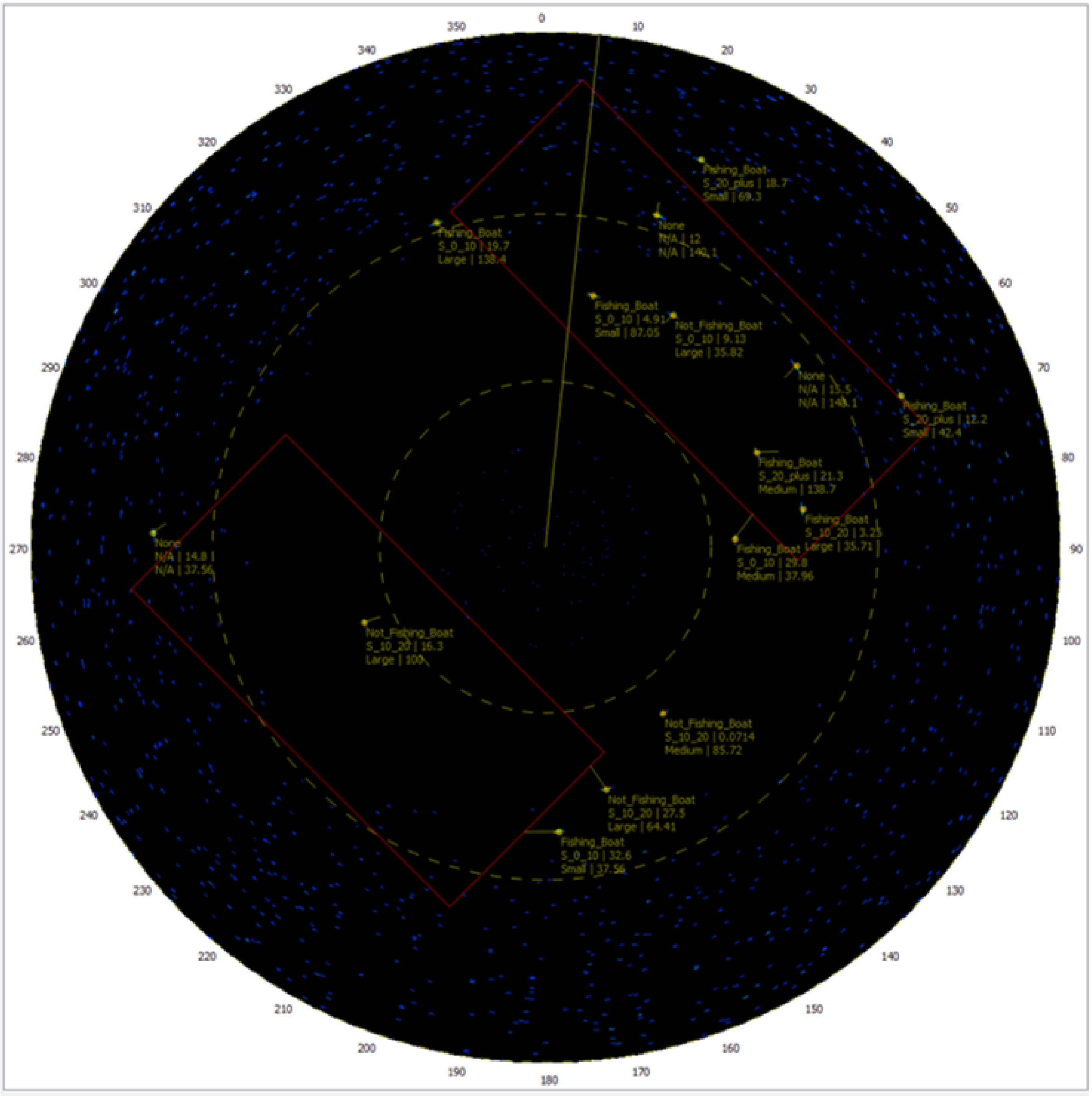


Figure 4.3: An example of the PPI display presented to the operator in task 1.

lack of—is another piece of information used to decide whether even further investigation is required. The vessel's reponse is shown as part of the track marker information (this is shown in Fig 4.4). Specifically, the operator uses vessel's response as well as the track speed and direction in order to decide to move the radar platform towards the track in order to get a better view.

For example, if the operator applies a sector scan to the track but does not communicate, the operator may be wanting to keep an eye on the track without interrogating. If the operator does want to communicate with the track, and the track does not respond, this may indicate the vessel is performing an illegal operation.

In figure 4.4, 'Track Velocity Vector' shows the magnitude and direction of the track's speed, and 'Transponder Vessel Type' shows how the vessel identifies itself. There are 3 vessel types in this scenario: not a fishing boat, fishing boat, or none (in the case of no AIS information). And 'Transponder Typical Speeds' fall into three categories: S_0_10 indicates speeds in the range of 0-10m/s, S_10_20 indicates 10-20m/s, and S_20_plus indicates speeds greater than 20m/s. ‘Transponder Typical Size’ falls into three categories: small, medium, and large. ‘Measured Size’ is calculated based on the SNR and range, and is scaled to represent the RCS of the vessel. The typical values for ‘Measured Size’ are: small indicates an RCS less than 50, medium indicates and RCS greater than 50 and but less than 100, and large indicates an RCS greater than 100.

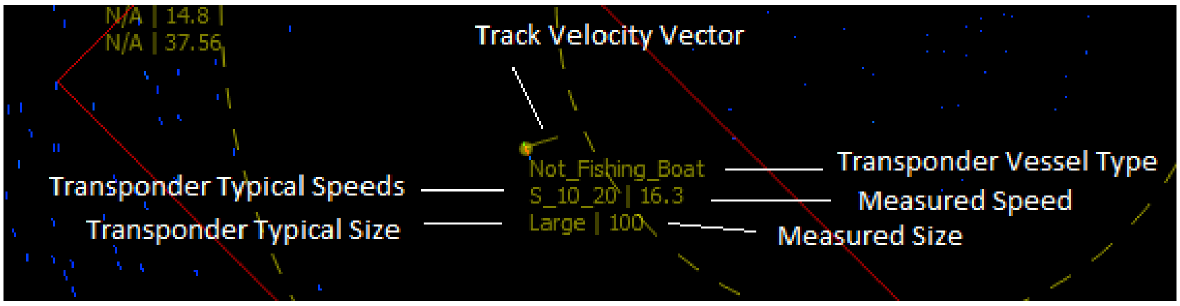


Figure 4.4: An example of the information attached to a track marker for task 2.

## 4.3.3 Task 3

In the current operation of a maritime surveillance mission, the radar platform would be flown by a pilot along a fixed trajectory allowing the radar to cover the search area. The radar operator would then observe the target’s returns in terms of either high resolution radar returns or as a SAR or ISAR image, and if necessary, the operator would request that the pilot move the aircraft in a specific direction towards the target based on the radar output. As mentioned in the section 3, the control of the platform itself must be automated in order to realise the full capability of the autonomous radar operator. Section 3 outlines how to automate and optimise the radar platform trajectory, but the remote radar operator may still require the platform to deviate from the surveillance path in order to get a closer look at a target. In this case, the UAV can fly the fixed surveillance trajectory until decisions are requested by the system to the remote operator, where the operator can, if necessary, send the UAV simple movement commands (e.g. forward, left, right) rather than telling a pilot where to move.

The third task deals with the movement commands requested by the operator based on the radar returns from track. The operator is still considering the single track which they decided to move towards. Since the operator is interacting with the simulation as if it were the actual maritime radar surveillance system, the simulation is therefore required to be real-time. Due to this requirement, an ISAR image generation method in conjunction with an ATR algorithm was avoided. Instead, the ISAR image was represented by a probability matrix in terms of sea state, relative azimuth and elevation between the track and the platform, SNR, number of revisits, and vessel type. In essence, the output probability could be thought of as the output from an ISAR image classifier. A low probability indicates that the observed track is unlikely to be performing any illegal operations whilst a high probability indicates that the track is likely performing an illegal operation. Furthermore, a probability such as

0.5 indicates uncertainty in whether the track is performing illegal operations or not. This probability that the track is performing an illegal activity is shown in the GUI next to the track marker (as shown in Fig. 4.6). Each time the operator requests the platform to either move towards, to the right, or to the left of the track, the probability is updated. In addition to these three movements, the operator can also decide to declare the vessel as performing an illegal operation or to move away from the vessel if the activity is legal. All of the platform controls available to the operator are shown in Fig. 4.7. The platform moves a fixed distance regardless of the movement selected. For clarity, the movement of the platform towards the target can be considered like moving between concentric circles around the target's position as shown in Fig. 4.5.

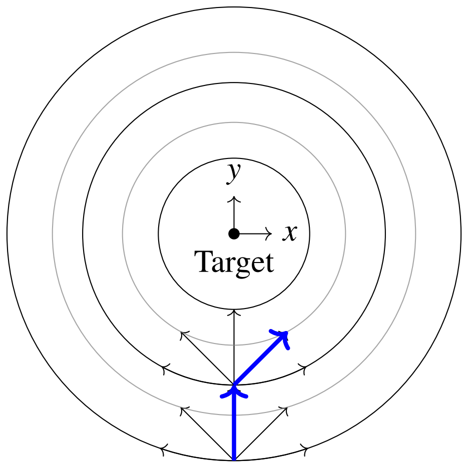


Figure 4.5: Diagram showing two movements made by the operator (in blue) and the respective options available to the operator at the time the decisions were made.

## 4.3.4 Data Collection

In order to make the most of the operator's time, the simulation was setup for fast data collection. Specifically, the scene for task 1 was randomised (within limits) with tracks of varying position, heading, RCS, and vessel type. Once the operator has exhausted all tracks, the scene was reset with a new set of randomised tracks. Additionally, there is a button for the operator to reset the platform position to the starting point if the operator moved the platform in task 3.

Fig. 4.8 shows the data collection process whereby both the scene information available to the operator and the operator's inputs are combined to form the state-action pair and then saved. Due to the COVID-19 pandemic (see chapter ), an actual radar operator was no longer able to use this simulation in order to collect the state-action pairs. Consequently, the author of this thesis acted as an operator based on operational advice provided by Leonardo UK [141].

Chapter 5 and 6 outline how the data collected with this GUI and simulation are used within a Bayesian network framework and a inverse reinforcement learning framework respectively. Additionally, example operator data samples are tabulated in section 5.3.

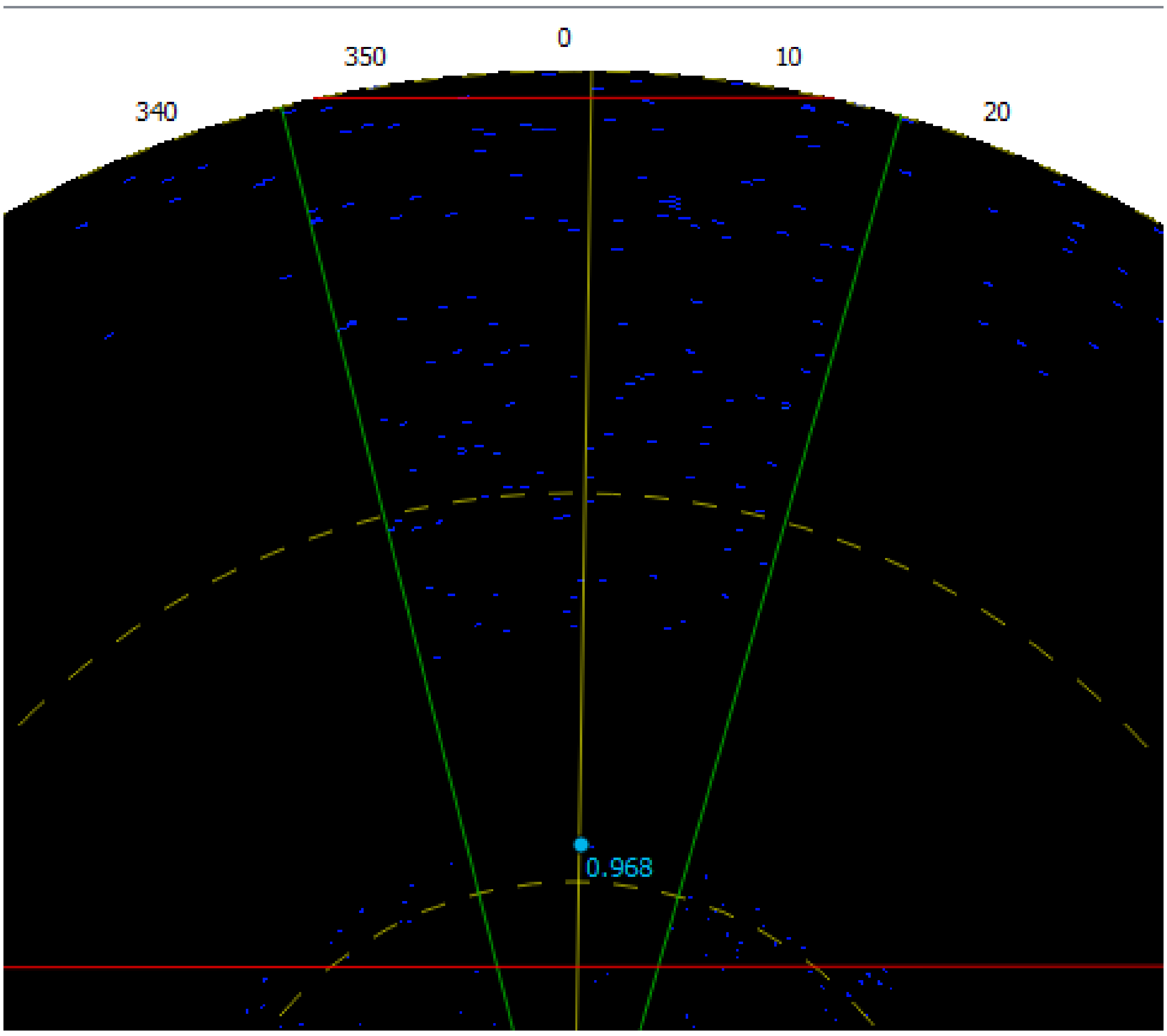


Figure 4.6: An example of the track marker showing the probability that the track is performing an illegal operation. Recalling from 2.2.6, the yellow solid line indicates the current azimuth of the radar boresight, the green lines indicate the current sector scan limits, and the red lines indicate a designated search zone.

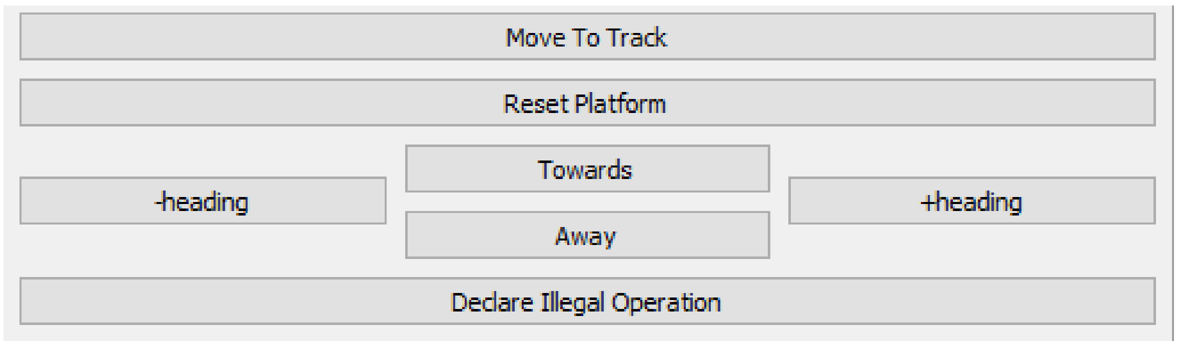


Figure 4.7: The platform controls available to the remote operator.

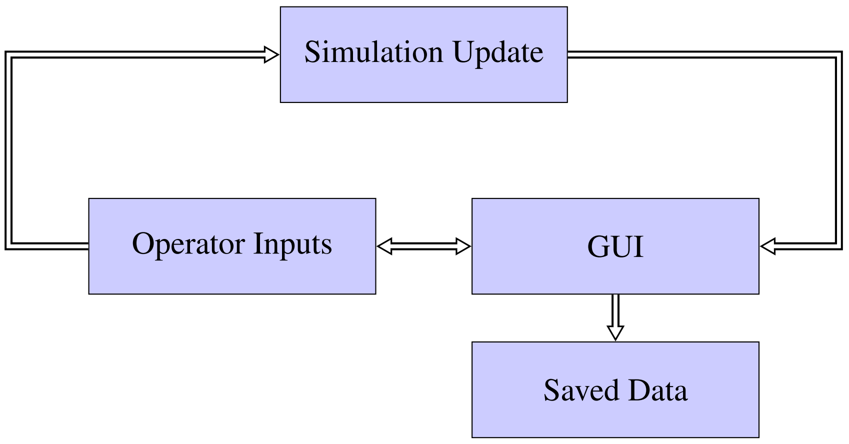


Figure 4.8: Flowchart outlining how and where the operator data is collected.

# Chapter 5

# Bayesian Networks

## 5.1 Introduction

This chapter presents the use of Bayesian networks for learning decisions made by a human radar operator carrying out the typical maritime surveillance missions as outlined in section 4.3. For maritime scenarios, current literature has only focused on using a Bayesian network (BN) for identification and assessment, and often assumes inputs from a generic surveillance sensor rather than a specific sensor such as a radar. Furthermore, in both the maritime surveillance and radar operations domain, there has been no investigation into the use of operator data in order to learn the decisions made throughout the mission. This chapter outlines the maximum likelihood approach used to obtain BN probabilities for the decisions of a given maritime surveillance mission. The BN is then used in place of the operator for interfacing with the simulation in real-time in order to test the suitability of this method.

Bayesian networks, a form of AI, have been used for a wide variety of tasks in both radar operations and in maritime surveillance. For example, investigation has been done on the use of a Bayesian network (BN) for autonomous radar control for polar ice sheet measurements [142]. In terms of maritime surveillance scenarios, a BN has been used for the identification and assessment of maritime objects [143], [144]. Furthermore, a maritime simulation of vessels was used to generate scenarios in which a BN was applied to identify suspicious ships [145].

A BN has also been used to assist a surveillance system operator for maritime situational assessment [146], [147]. Expert knowledge has been used to inform the contextual and behavioural information within a scenario in which a dynamic Bayesian network was used for behavioural based classification of maritime vessels [148]. Additionally, a BN was used for the detection of unusual maritime activity where experts were used to set the network's conditional probability tables [149].

However, while some of the above works leverage expert knowledge, they don't learn from operator data on the scenario nor do they attempt to imitate the operator's decisions. Furthermore, they don't deal with real-time scenarios in which the operator has a sequence of decisions to make where each decision results in more situational information. The use of a BN for this type of imitation task has seen little use [150].

### 5.1.1 Main Contributions

The contributions of this chapter are as follows:

1. The design of a BN in order to imitate an operator performing several maritime surveillance radar tasks.
2. The testing and analysis of the Bayesian Network approach in imitating an operator performing several maritime surveillance radar tasks as outlined in chapter 7.

## 5.2 Bayesian Networks

### 5.2.1 Fundamentals

A Bayesian network [151]–[155] represents a series of connected nodes in the form of a directed acyclic graph (DAG) with each node representing a variable (or event). Additionally, each connection (or edge) represents a conditional dependency between two nodes. It is worth noting Bayesian networks are a form of factor graphs [156] that contain probabilistic models.

Given two events $A$ and $B$, the probability of $A$ occurring given that $B$ has occurred can be written as $P(A|B)$. This probability can be obtained with the used of the conditional probability formula which is stated as:

$$P(A|B) = \frac{P(A,B)}{P(B)} \tag{5.1}$$

Using the chain rule, the conditional probability formula can be expanded to account for any number of events (e.g. $A_1, A_2, ..., A_\text{n}$):

$$P(A_\text{n}|A_{\text{n}-1},\ldots,A_1) = \frac{P(A_\text{n},\ldots,A_1)}{P(A_{\text{n}-1},\ldots,A_1)} \tag{5.2}$$

Equation 5.2 is the fundamental equation within a Bayesian network and allows for probabilities for unknown node states to be obtained given the state of any number of other nodes. In terms of replicating an operator's behaviour, this equation can be used to obtain the probability of an operator performing an action given information known to the operator or even previously made decisions in the decision network.

Table 5.1: List of the example Bayesian network decision nodes and information nodes with their corresponding descriptions.

| Node | Description |
|---|---|
| IZ | Track in zone |
| T | Target has transponder |
| TS | Track's speed |
| MTT | Move to track |

### 5.2.2 Example

A simple example in the radar operator domain would be a node that represents whether or not an operator would move towards a track. This decision could be based on, for example, whether the track in one of the designated zones, whether the track has a transponder or not, and the track's speed. An example Bayesian network of this decision process is shown in Fig. 5.1 with the node descriptions shown in table 5.1. The decision to move towards a track is represented by the node 'MTT' which has two states, 'true' and 'false'. This node is dependant on whether the track has a transponder, represented in the network by node 'T' with states 'true' and 'false'. The 'MTT' node is also dependant on the track's speed, represented by the node 'TS' which has states 0–10 m, 10–20 m, and $\geq$ 20 m which are denoted by 0, 1, and 2 respectively. Additionally, node 'T' is dependant on whether a track is in one of the designated zones which is represented by the 'IZ' node with states 'true' and 'false'.

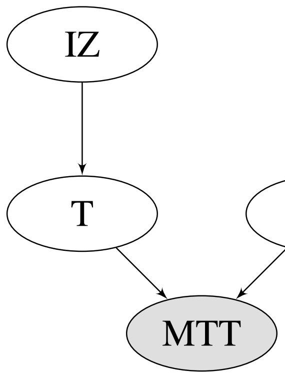

$$\begin{tikzcd}
IZ \arrow[d] & \\
T \arrow[dr] & & TS \arrow[dl] \\
& MTT &
\end{tikzcd}$$

Figure 5.1: An example Bayesian network.

Equation 5.2 can then be used to obtain the probabilities of this network. The following is an example for obtaining the probability of moving towards a track given that the track is within a designated zone.

$$\begin{aligned} P(\text{MTT}=T|\text{IZ}=T) &= \frac{P(\text{MTT}=T,\text{IZ}=T)}{P(\text{IZ}=T)} \\ &= \frac{\sum_{\text{T}\in\{T,F\},\text{TS}\in\{0,1,2\}} P(\text{MTT}=T,\text{T},\text{TS},\text{IZ}=T)}{P(\text{IZ}=T)} \end{aligned} \tag{5.3}$$

It is worth highlighting that the numerator summation is effectively the marginalising of T and TS. Equation 5.4 shows a simplification of the numerator in equation 5.3. This sim-

| | | IZ | |
|---|---|---|---|
| | | *T* | *F* |
| T | *T* | 0.7 | 0.6 |
| | *F* | 0.3 | 0.4 |

Table 5.2: An example conditional probability table for the example Bayesian network. Specifically, this table contains the decision probabilities learned from operator data.

plification is the result of the chain rule of probability calculus (i.e. equation 5.2) as well as reducing probabilities to only include dependant nodes as per the example Bayesian network.

$$\begin{aligned} P(\mathrm{MTT}=T,\mathrm{T},\mathrm{TS},\mathrm{IZ}=T) &= P(\mathrm{MTT}=T|\mathrm{T},\mathrm{TS},\mathrm{IZ}=T)P(\mathrm{T},\mathrm{TS},\mathrm{IZ}=T) \\ &= P(\mathrm{MTT}=T|\mathrm{T},\mathrm{TS})P(\mathrm{T}|\mathrm{TS},\mathrm{IZ}=T)P(\mathrm{TS},\mathrm{IZ}=T) \\ &= P(\mathrm{MTT}=T|\mathrm{T},\mathrm{TS})P(\mathrm{T}|\mathrm{IZ}=T)P(\mathrm{TS}|\mathrm{IZ}=T)P(\mathrm{IZ}=T) \\ &= P(\mathrm{MTT}=T|\mathrm{T},\mathrm{TS})P(\mathrm{T}|\mathrm{IZ}=T)P(\mathrm{TS})P(\mathrm{IZ}=T) \end{aligned} \tag{5.4}$$

The individual probabilities on the right hand side of equation 5.4 are obtained from the conditional tables of each node. These tables are either defined manually or learned from data. An example conditional probability table for this example is shown in 5.2.

### 5.2.3 Bayesian Network Parameters

The nodes within the BN are represented by the vector $\boldsymbol{x}$ of length $m$ with each node $x_i$ having parents denoted by vector $\boldsymbol{u}_i$. In this case, the nodes are assumed to be discrete with the total values for node $x_i$ denoted by $n_i$. Whilst a variable in a Bayesian Network can be taken to be continuous, this requires an assumption on its probability distribution. This distribution can be difficult or impossible to know beforehand, and its assumption can lead to a poor representation of the variable's actual probability. Additionally, if any variable has a continuous probability distribution this can still be approximated with discrete variables.

The data set obtained from the scenario and the operator's decisions is denoted by the $N \times m$ table $\boldsymbol{D}$, where $N$ is the number of data instances, and a given data instance is denoted by the vector $\boldsymbol{d}_i$ of length $m$.

It is often the case that the data set is not complete. In other words, at least one of the data instances $\boldsymbol{d}_i$ has a missing value. For example, if the radar operator does communicate with the vessel, then the vessel reply is not known for that track. In the case where the data set is not complete, it is first important to determine why data is missing. Firstly, if the data being missing is unrelated to the missing data and unrelated to any other variable, then the data is missing completely at random (MCAR). Secondly, in the case where the data being missing is dependent on the data itself, then the data is said to be missing not at random (MNAR).

For the tasks considered in this thesis, the missing data is missing at random (MAR). That is, the fact that the data is missing is unrelated to missing data itself, but it is related to the observed data of other variables. For example, whether or not an operator choses to fuse a track is irrelevant to the whether or not the fuse track data is missing, though it is relevant to whether or not the track is in the zone. An example of missing data within an instance of operator decision data is shown in table 5.10.

There are several reasons why the collected data for a given run might be missing datum. In the case that a decision was made that didn't lead to information being collected (e.g. if the operator opts not to communicate with a track, then no further information is obtained). Or, at the time of the operator making decisions, certain information wasn't available (e.g. a new track appears and the operator makes decisions before all the track information has been obtained). This case is likely to happen when an operator encounters a new track where the available information requires an immediate decision.

The network estimates $\theta$ represent the estimated conditional probabilities for each node. Specifically, the parameter estimate $\theta_{x|\boldsymbol{u}}$ is used to define the probabilities of node $x$ given the values of the node's parents $\boldsymbol{u}$.

In the case where the operator's data is complete (i.e. there are no missing values for each vector $\boldsymbol{d}_i$ within the data set $\boldsymbol{D}$), the empirical distribution can be used to obtain the probability of a given event $x \mid \boldsymbol{u}$ from data set $D$.

$$P_{\boldsymbol{D}}(x \mid \boldsymbol{u}) \triangleq \frac{C(x \mid \boldsymbol{u})}{C(\boldsymbol{u})}, \tag{5.5}$$

where the term $C(x \mid \boldsymbol{u})$ is the number of data instances $\boldsymbol{d}_i$ that satisfy event $x \mid \boldsymbol{u}$.

The likelihood of the network estimates $\theta$ is defined as the probability of observing the data set $\boldsymbol{D}$ under the estimates. Intrinsically, this value can be used as an overall measure of the network's performance in imitating the operator's decisions. For a given setting of $\theta$, the likelihood of observing the data set $\boldsymbol{D}$ is defined as:

$$L(\theta|\boldsymbol{D}) \triangleq \prod_{i=1}^{N} P_\theta(\boldsymbol{d}_i), \tag{5.6}$$

Using the log-likelihood is often more convenient allowing for summations in place of products. The log-likelihood function is the defined as follows:

$$LL(\theta|\boldsymbol{D}) \triangleq \log L(\theta|\boldsymbol{D}) = \sum_{i=1}^{N} \log P_\theta(\boldsymbol{d}_i), \tag{5.7}$$

where $P_\theta(.)$ is the probability distribution induced by the network structure and the network estimates $\theta$.

### 5.2.4 Expectation Maximisation

As mentioned in the previous section, the conditional probability tables which are used to obtain decision probabilities from the Bayesian network can be learned from data. This section outlines the expectation maximisation approach used to learn the tables from operator decision data.

For the complete data set case, the probabilities obtained with empirical distribution (as per equation 5.8) are, by definition, the values of $\theta$ that maximises the likelihood and thus log-likelihood. Therefore, the network estimates $\theta$ that maximise the likelihood are obtained as:

$$\theta_{x|\boldsymbol{u}} = P_{\boldsymbol{D}}(x \mid \boldsymbol{u}) \triangleq \frac{C(x \mid \boldsymbol{u})}{C(\boldsymbol{u})}, \tag{5.8}$$

It is only when data is MNAR that a mechanism needs to be incorporated to account for the missing data. Under either the MAR/MCAR scenario, the maximum likelihood approach can be directly expanded to account for the missing data by using a local search method. The problem described thus far involves maximising a function (in this case, the log-likelihood) which depends on missing data. Therefore, the expectation maximisation (EM) algorithm [152] was chosen as the local search method due to its suitability for these types of problems.

This method starts with an initial set of values for the estimates (denoted by $\theta^0$). Using these initial estimates, the data set is completed by obtaining the probabilities of each possible completion for instances with missing data. Mainly, $\theta^k$ is used to complete the data set with the resulting completed data set used to obtain the new set of parameters $\theta^{k+1}$. This algorithm guarantees that the new set of parameters has a log-likelihood no less than that of the previous set of parameters. As such, this process can be iterated until some convergence criteria is met.

For the complete data case, the counts of each instantiation were summed and divided by the size of the data set. In the incomplete data case, rather than summing the counts, the probabilities of each instantiation being in the completed data set are summed and divided by the data set size. For data instance $\boldsymbol{d}_i$, the probability of a given completion $\boldsymbol{c}_i$ being in the data set is denoted as $P_{\theta^k}(\boldsymbol{c}_i|\boldsymbol{d}_i)$ where the current set of parameters $\theta^k$ is used to obtain the completion probabilities.

For parameter set $\theta^k$, the expected empirical distribution of data set $\boldsymbol{D}$ is defined as:

$$P_{\boldsymbol{D},\theta^k}(\alpha) \triangleq \frac{1}{N} \sum_{\boldsymbol{d}_i,\boldsymbol{c}_i \vDash \alpha} P_{\theta^k}(\boldsymbol{c}_i \mid \boldsymbol{d}_i), \tag{5.9}$$

where $\alpha$ is an event and $\boldsymbol{d}_i, \boldsymbol{c}_i \vDash \alpha$ states that the combination of $\boldsymbol{d}_i$ and $\boldsymbol{c}_i$ satisfies $\alpha$. In this case, the summation is therefore summing over all possible cases $\boldsymbol{d}_i$ and their completions $\boldsymbol{c}_i$

that satisfy the event $\alpha$.

Rather than obtain the expected empirical distribution which can then be used to obtain the set of parameters, it is simpler to iterate over each data set and summing the probability of event $\alpha$ given data instance $\boldsymbol{d}_i$.

$$P_{\boldsymbol{D},\theta^k}(\alpha) = \frac{1}{N}\sum_{i=1}^{N} P_{\theta^k}(\alpha \mid \boldsymbol{d}_i), \tag{5.10}$$

The next step in the EM algorithm is to update the set of parameters $\theta^{k+1}$ which obtains the parameter estimates at step $k+1$ for node $x$ (with parents $\boldsymbol{u}$). This is defined as

$$\theta^{k+1}_{x|\boldsymbol{u}} \triangleq P_{\boldsymbol{D},\theta^k}(x \mid \boldsymbol{u}), \tag{5.11}$$

Using the conditional probability formula in conjunction with equation 5.10, equation 5.11 can be rewritten as follows:

$$\theta^{k+1}_{x|\boldsymbol{u}} = \frac{\sum_{i=1}^{N} P_{\theta^k}(x\boldsymbol{u} \mid \boldsymbol{d}_i)}{\sum_{i=1}^{N} P_{\theta^k}(\boldsymbol{u} \mid \boldsymbol{d}_i)}, \tag{5.12}$$

In practice, the data is looped over with a ‘for’ loop, and within, a counter $c_{x\boldsymbol{u}}$ at each step is incremented by the probability $P_{\theta^k}(x\boldsymbol{u} \mid \boldsymbol{d}_i)$ at step $i$ of the loop. It should also be noted that another counter $c_{\boldsymbol{u}}$ can be obtained by summing $c_{x\boldsymbol{u}}$ for all values $x$. This maximum likelihood expectation maximisation algorithm in outlined in algorithm 2 where $\theta_{\text{mll}}$ represents the algorithm’s parameter estimates that maximise the log-likelihood.

Since the algorithm is a local search, it may converge on different parameter estimates based on the initial parameter estimates $\theta_0$. As such, the algorithm was ran 10 times with the values for $\theta_0$ normalised on the first run and randomised on subsequent runs. The run which provided the highest log-likelihood was then chosen as the final network. Additionally, each run was terminated when the log-likelihood for the network was less than a set tolerance which in this case was heuristically set to $1e-6$. In practice, only unique samples from the data set were used where the values for $c_{x\boldsymbol{u}}$ and $c_{\boldsymbol{u}}$ were scaled according to the number of times that unique samples was within the data set.

**Algorithm 2:** ML Expectation Maximisation

```
Input:
G: Bayesian network structure ($X$ and $\boldsymbol{U}$)
$\theta^0$: inital parameters
$\boldsymbol{D}$: Data set of size $N$
$\eta$: Tolerance for change in log-likelihood value
Output:
$\theta_{\text{mll}}$:
begin
    $k \leftarrow 0$
    $\lambda \leftarrow \infty$
    while $\lambda > \eta$ do
        $c_{x\boldsymbol{u}} \leftarrow 0$
        for $i = 1;\ i < N;\ i \leftarrow i + 1$ do
            foreach $x\boldsymbol{u}$ do
                $c_{x\boldsymbol{u}} \leftarrow c_{x\boldsymbol{u}} + P_{\theta^k}(x\boldsymbol{u} \mid \boldsymbol{d}_i)$
            end
        end
        $c_{\boldsymbol{u}} = \sum_{i=1}^{n_x} c_{x_i\boldsymbol{u}}$
        $\theta^{k+1}_{x|\boldsymbol{u}} = c_{x\boldsymbol{u}} / c_{\boldsymbol{u}}$
        $\lambda = LL(\theta_{k+1} \mid \boldsymbol{D}) - LL(\theta_k \mid \boldsymbol{D})$
        $k \leftarrow k + 1$
    end
    $\theta^{\text{mll}} = \theta^k$
end
```

Table 5.3: Expectation Maximisation Parameters

| Parameter | Value |
|---|---|
| Number of runs, $n_\mathrm{r}$ | 20 |
| Log-likelihood tolerance, $\eta$ | $1e-6$ |

## 5.3 Maritime Radar Surveillance Task Setup

### 5.3.1 Task 1

As mentioned in chapter 4, three tasks representing the stages of a maritime surveillance mission were used to test this method. The first task involves selecting a track from a scene containing many tracks. Since the BN outlined requires discrete inputs, a decision has to be made on how to discretise these inputs. The operator will consider the distance of the tracks to one of the designated zones, and so the inputs to the BN were discretise to the following three binary input nodes:

1. Whether or not there are tracks within any of the zones

2. Whether or not there are tracks in a 10 km border of the zones.

3. Whether or not there are tracks outwith the 10 km border of the zones.

This decision removes all other tracks not in the chosen group from consideration. For example, the operator will likely consider the tracks within the zones first. Consequently, only the information from the remaining tracks is used for the next decision node in the BN. The next decision is based on whether or not any of the remaining tracks have AIS information. Within the BN, the inputs are two binary decision nodes, where the track either does or does not have transponder information (i.e. known or unknown). If there are unknown tracks, the third decision node in the BN is skipped. Note that this has no effect on the probabilistic model, and is in effect the equivalent to unknown tracks having their own input to the third decision node. If there are no unknown tracks, an additional decision node in the BN is used on the available tracks based on whether any of the tracks display abnormalities. In this case, the abnormalities are whether the track speed and/or size does not fall within the typical values ranges obtained from the AIS. Whether there were known or unknown tracks, the final decision node in the BN takes the binary input for whether or not any of the remaining tracks are heading towards the nearest designated zone. This task is represented by the BN shown in Fig. 5.2 with the node descriptions given in table 5.4. Three instances of operator decision data are given for each decision node in tables 5.5, 5.6, 5.6, and 5.8. Note that the white nodes indicate information obtained by the operator while the grey nodes indicate decisions from the operator. The dotted lines indicate that the nodes are not connected within the Bayesian network, but the decision influences the information available to that node. The first decision node TS1 represents the decision to select the tracks within a given region based on the observed information nodes WZ (within zone), W10 (within 10 km), NW (not within 10 km). The next decision node TS2 represents the decision to select tracks based on the observed information nodes K (known) and UK (unknown). The additional decision TS3 represents the decision to further select known tracks based on anomalous information. The observed information nodes for anomalous AIS information ASAS (abnormal size, abnormal speed), ASNS (abnormal size, normal speed), NSAS (normal size, abnormal speed), and NSNS (normal size, normal speed) are determined depending on whether or not the observed size and speed are within the normal operational bounds provided by the AIS. The final decision node TS4 represents the decision to select tracks based on the tracks' directions relative to their respective nearest zone. The observed information zones are T (heading towards the nearest zones) and A (heading away from the nearest zone).

Table 5.4: List of the Bayesian network decision nodes and information nodes for task 1 with their corresponding descriptions.

| Node | Description |
|---|---|
| WZ | Track is in one of the designated zones |
| W10 | Track is within 10 km of one of the designated zones |
| NW | Track is not in one of the designated zones |
| TS1 | Decision to select tracks based on the connecting nodes |
| TS2 | Decision to select tracks based on the connecting nodes |
| TS3 | Decision to select tracks based on the connecting nodes |
| TS4 | Decision to select tracks based on the connecting nodes |
| K | Track is known (i.e. has AIS) |
| UK | Track is unknown (i.e. does not have AIS) |
| ASAS | Track has abnormal size and abnormal speed |
| ASNS | Track has abnormal size and normal speed |
| NSAS | Track has normal size and abnormal speed |
| NSNS | Track has normal size and normal speed |
| A | Track is heading away from its nearest zone |
| T | Track is heading towards its nearest zone |

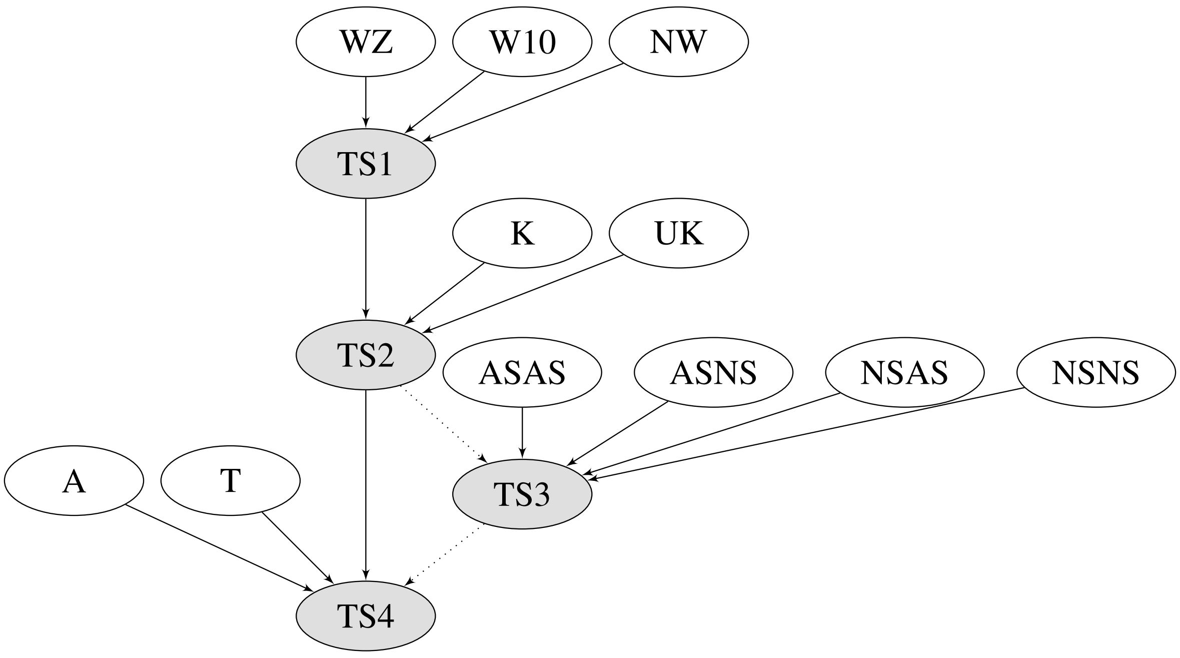


Figure 5.2: Task 1 represented as a Bayesian network. The dotted lines indicate that this node is potentially skipped depending on previous decisions.

Table 5.5: Example radar operator data for the decision to select a set of tracks to further investigate given where they lie relative to one of the designated zones.

| WZ | W10 | NW | TS1 |
|---|---|---|---|
| True | True | True | WZ |
| False | True | True | W10 |
| False | False | True | NW |

Table 5.6: Example radar operator data for the decision to select a set of tracks to further investigate given whether or not they have an AIS.

| TS1 | K | UK | TS2 |
|---|---|---|---|
| WZ | True | True | UK |
| NW | False | True | UK |
| W10 | True | False | K |

Table 5.7: Example radar operator data for the decision to select a set of tracks to further investigate given their speed and size abnormalities.

| TS2 | ASAS | ASNS | NSAS | NSNS | TS3 |
|---|---|---|---|---|---|
| UK | True | True | True | True | ASAS |
| UK | False | False | False | True | NSNS |
| UK | False | True | False | True | ASNS |

Table 5.8: Example radar operator data for the decision to select a set of tracks for further investigation.

| TS2 | TS3 | T | A | TS4 |
|---|---|---|---|---|
| UK | NSAS | True | True | T |
| K | NSAS | False | True | A |
| UK | ASNS | True | False | T |

### 5.3.2 Task 2

Table 5.9: List of Bayesian network decision nodes and information nodes for task 2 with their corresponding descriptions.

| Node | Description |
|---|---|
| IZ | Track is in one of the designated zones |
| AIS | Target has automatic identification system |
| SS | Apply sector scan on track |
| CMV | Communicate with the vessel |
| VR | Vessel's response |
| TD | Track's speed |
| TS | Track's direction |
| MTT | Move towards track |

Table 5.10: Example radar operator data for the decision to apply sector scan and/or communicate with the track.

| AIS | IZ | SS | CMV | VR | TD | TS | MTT |
|---|---|---|---|---|---|---|---|
| True | True | False | False | * | * | * | * |
| False | True | True | True | No AIS | Towards | 10–20 $\mathrm{m\,s^{-1}}$ | True |
| False | True | True | True | Fishing Boat | Towards | 0–10 $\mathrm{m\,s^{-1}}$ | False |

Using the track selected by the first decision process, the operator will further investigate the track. The operator then uses the track's AIS data in conjunction with the zone region the track lies within and decides whether or not to apply a sector scan on that particular track. Additionally, the operator also decides whether or not to communicate with the vessel (in the case where the vessel may have their AIS switched off, for example). The vessel's reply to the operator provides further information about the track's AIS data (e.g. the lack of a vessel reply may indicate suspicious activity). Lastly, based on the vessel's reply, the track direction, and track speed, the operator will make a decision about whether or not to move towards the track in order to get a closer look.

This task is represented by the BN shown in Fig. 5.3 with the nodes summarised in table 5.9. Three instances of radar operator decision data are given in table 5.10. The observed information nodes IZ, AIS, VR, TD, TS represent the track being in one of the fishing zones, the track AIS data, the tracked vessel's reply, the track direction, and the track speed respectively. The state of both AIS and VR indicate whether or not the vessel type is shown as being a fishing boat, not a fishing boat, or that there is no AIS data. The track direction indicates whether or not the track is heading towards the nearest fishing zone. For the BN, the track speed is discretised into three possible states: 0–10 $\mathrm{m\,s^{-1}}$, 10–20 $\mathrm{m\,s^{-1}}$, and above 20 $\mathrm{m\,s^{-1}}$. The operator decision nodes SS, CMV, MTT represent the decision to apply a sector scan to the track, communicate with the track, and move towards the track respectively.

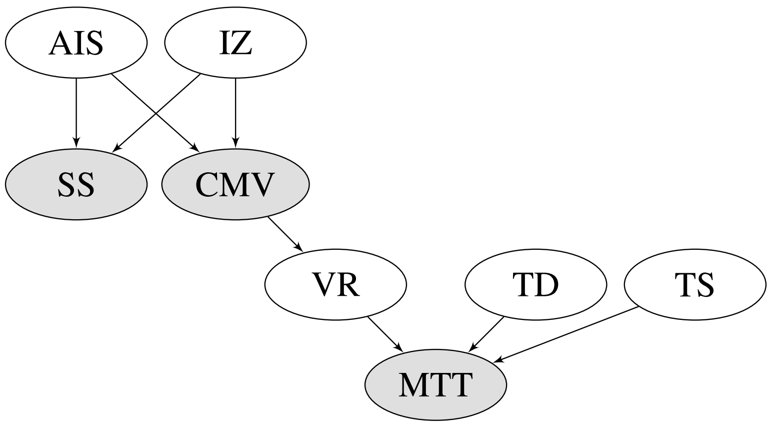


Figure 5.3: Task 2 represented as a Bayesian network.

### 5.3.3 Task 3

The final decision process occurs when the operator decides to move towards the track in order to get a closer look. The operator has the option of moving towards the track, to the left of the track, or to the right of the track. For the BN, this decision is based on the previous two movement directions as well as whether or not the change in probability was positive, negative, or remained the same. Within a real mission, the change in position influences the ISAR image seen by the operator which determines the decision made regarding whether or not the track is likely to be performing an illegal operation.

Task 3 is represented by the BN shown in Fig. 5.4 with the network nodes summarised in table 5.11. Three example instances of operator decision data are shown in table 5.12. The first decision node M represents the move made in terms of left, forward, and right. This decision is based on the information nodes PI1, PD1, PI2, and PD2. PD1 represent the move made at the previous step, and PI1 represents the change in probability at the previous step. Similarly, PD2 and PI2 represent the move made and change in probability two steps ago. For the BN, the upper and lower limits for determining if the probability had changed positively or negatively were 0.05 and $-0.05$ respectively. The final decision node A represents the action performed based on the updated probability I. The action consists of three options: continue moving towards the track, return to the surveillance path and note that the track is not likely to be performing an illegal operation, or declare the track as potentially carrying out an illegal operation. For node I, the probability was discretised into probability bands of 0.05 with a two larger bands of 0.2 in the middle as it was assumed that the operator was likely to continue move towards the track when the uncertainty was high.

Table 5.11: List of Bayesian network decision nodes and information nodes for task 3 with their corresponding descriptions.

| Node | Description |
|---|---|
| PI1 | Change in probability of performing an illegal operation one step previously |
| PD1 | Movement made one step previously |
| PI2 | Change in probability of performing an illegal operation two steps previously |
| PD2 | Movement made two steps previously |
| M | Decision for which movement to make towards the track |
| I | Probability of the target performing an illegal operation |
| A | Action taken based on the newly obtained target information |

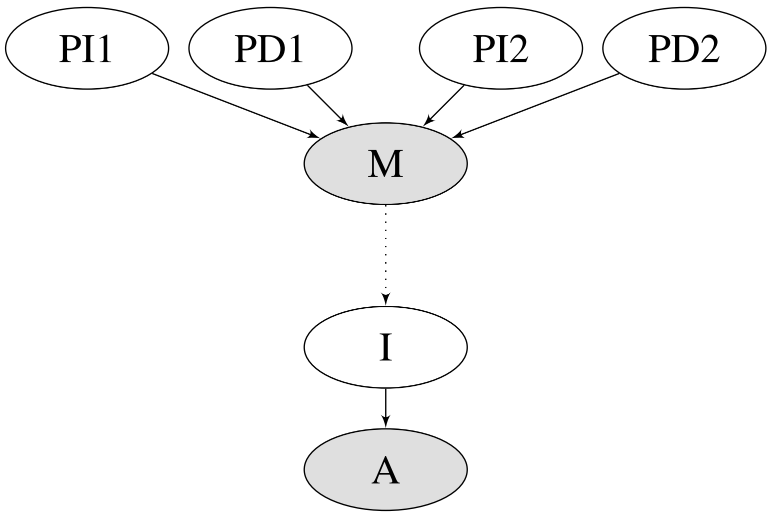


Figure 5.4: Task 3 represented as a Bayesian network.

Table 5.12: Example radar operator data for the movement and action decisions relating to a target potentially performing an illegal operation.

| PD2 | PI2 | PD1 | PI1 | M | I | A |
|---|---|---|---|---|---|---|
| Forward | Increase | Forward | Increase | Forward | 0.7-0.75 | Continue Moving |
| Forward | Decrease | Left | Increase | Left | 0.05-0.1 | Move Away |
| Right | Increase | Right | Increase | Right | 0.95-1.0 | Declare Illegal |

## 5.4 Conclusions

In this chapter, the use of a Bayesian network for imitating a radar operator carrying out a maritime surveillance mission has been outlined. The network's probabilities were learned from operator decision data with the used of a real-time radar simulation for three common surveillance tasks. The results of this network for the three tasks is shown in chapter 7 and compared and contrasted with a different machine learning approach.

# Chapter 6

# Inverse Reinforcement Learning

## 6.1 Introduction

This chapter presents the use of an inverse reinforcement learning (IRL) method for learning decisions made by a human radar operator carrying out a maritime surveillance mission. For radar applications, the current literature has only focused on using IRL methods within specific sub-components of the radar. For example, applied to tracking [157], or as in Krishnamurthy et al. [158], IRL is used to determine if an "enemy" radar is cognitive by considering its sequence of emissions. However, neither of these examples use IRL in a form that leverages radar operator knowledge. Specifically, the examples don't learn from operator data on the scenario nor do they attempt to imitate the operator's decisions.

### 6.1.1 Main Contributions

The contributions of this chapter are as follows:

1. The design of a Markov decision process (MDP) to be used with an IRL method in order to imitate an operator performing several maritime surveillance radar tasks.
2. The testing and analysis of IRL for imitating an operator performing several maritime surveillance radar tasks as outlined in chapter 7.

## 6.2 Inverse Reinforcement Learning Overview

IRL can be considered a fusion of both the supervised learning problem as well as the reinforcement learning problem: a data set of expert actions is provided however, the algorithm still explores the state-action space. Unsurprisingly, inverse reinforcement learning is the inverse of reinforcement learning (RL). Reinforcement learning deals with the optimisation of a control policy given an environment and a reward function. The result is therefore a control policy (i.e. a mapping from states to actions) for the agent that maximises the expected reward when acting within the environment. This process is shown in Fig. 6.1.

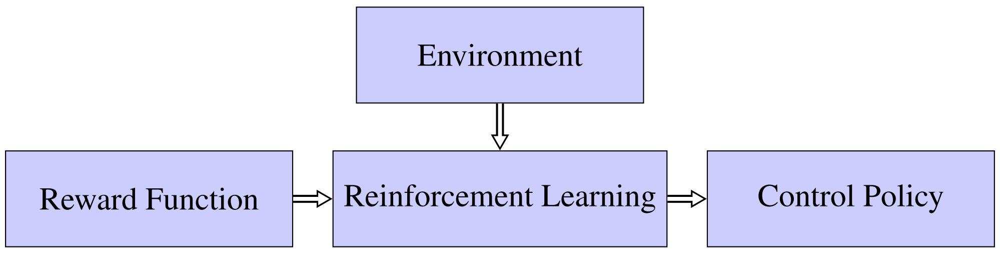


Figure 6.1: Reinforcement learning block diagram.

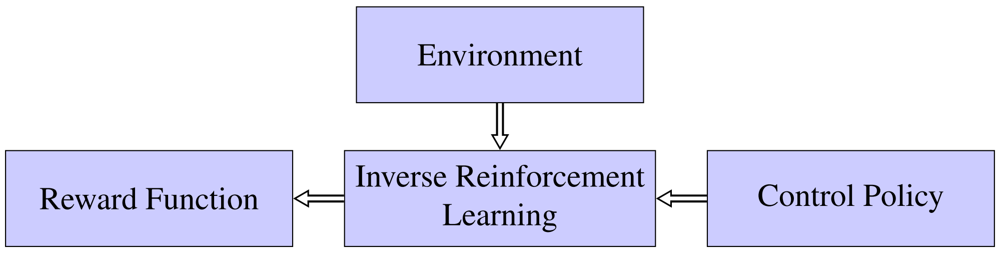


Figure 6.2: Inverse reinforcement learning block diagram.

Conversely, in IRL, the control policy is the input and the reward function is the output. However, it is often the case that the control policy is inferred from sampled trajectories of operator state-action pairs, rather than the control policy passed into the algorithm directly. This process is shown in figure 6.2. Typically, the obtained reward function is then used in a RL sub-routine to obtain a control policy from the reward function, and the control policy is generalised to handle any potential state.

## 6.3 Reinforcement Learning

In order to discuss the fundamentals of IRL, the fundamentals of RL must be discussed beforehand. At the core of a reinforcement learning algorithm is the Markov decision process (MDP) [159]–[162].

### 6.3.1 Markov Decision Process

**Definition 6.1.** A finite-state MDP is comprised of a tuple $(S, A, P(s'|s,a), R, \gamma)$:

- $S$ is a finite state space ($N$ states).
- $A = \{a_1, a_2, ..., a_k\}$ is a finite action space.
- $\{P(s'|s,a)\}$ is a set of transition probabilities where $P(s'|s,a)$ is the probability of transitioning from state $s$ to state $s'$ given action a.
- $R : S \to \mathbb{R}$ is a reward function where $R(s)$ determines the immediate reward at state $s$. $R_{max}$ is the maximum bound for $R$.
- $\gamma \in [0,1)$ is the discount factor which determines the weighting of future rewards. E.g. a discount factor of 0.5 would mean a reward one step into the future is worth half as much as a reward now.

A typical Markov decision process is shown in figure 6.3 with 3 states and 2 actions. One transition probability is shown above its respective action, and one reward is shown above its respective state. For example, the probability of the agent transitioning to state $s_0$ after taking action $a_0$ at state $s_1$ is $P(s_0|s_1,a_0)$. If the agent transitions to $s_0$, a reward of $R(s_0)$ is obtained.

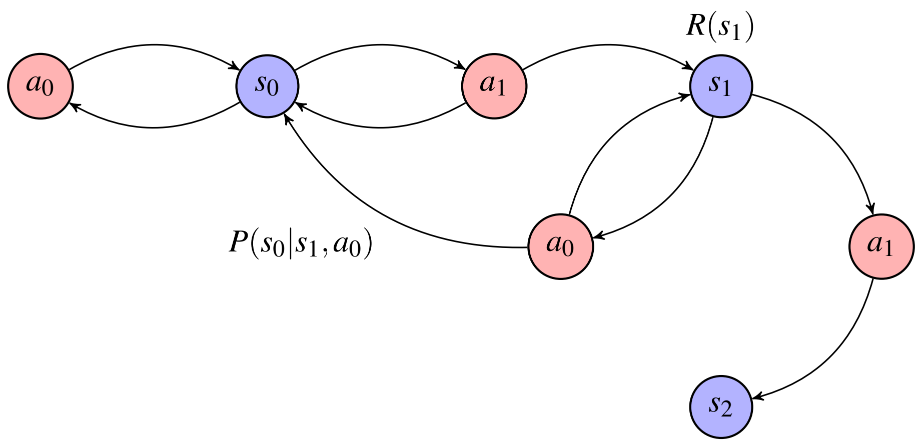


Figure 6.3: Example Markov Decision Process.

A policy within an MDP is defined as $\pi : S \rightarrow A$. In other words, $\pi(s) = a$ describes the policy $\pi$ whereby given a state $s$, an action $a$ is returned.

**Definition 6.2.** The value function for policy $\pi$ is defined as the expected sum of (discounted) rewards from state $s$.

Mathematically, the value function is stated as:

$$V^{\pi}(s) = \mathbf{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t)\right] \tag{6.1}$$

where $\mathbf{E}$ is the expectation function and the value of $t$ is the number of iterations into the future. This can be rewritten by separating the already obtained reward of the current state from the future states:

$$V^{\pi}(s) = R(s) + \mathbf{E}\left[\sum_{t=1}^{\infty} \gamma^t R(s_t)\right] \tag{6.2}$$

By expanding, the above can be written as:

$$V^{\pi}(s) = R(s) + \sum_{s \in S} P(s'|s,\pi(s)) \sum_{t=1}^{\infty} \gamma^t R(s_t) \tag{6.3}$$

A recursive definition of the value function can then be written as:

$$V^{\pi}(s) = R(s) + \gamma \sum_{s' \in S} P(s'|s,\pi(s)) V^{\pi}(s') \tag{6.4}$$

**Definition 6.3.** The Q-function is defined as the expected sum of rewards of policy $\pi$ after taking action $a$ but before the transitional probabilities are considered.

Mathematically the Q-function is formulated in equation 6.5. Note that the Q-function does not consider the optimal policy, and is therefore a function of action $a$.

$$Q^{\pi}(s,a) = R(s) + \gamma \sum_{s' \in S} P(s'|s,a)V^{\pi}(s') \tag{6.5}$$

Equations 6.4 and 6.5 are one form of the Bellman Equations. Note that the optimal policy is denoted as $\pi^*$, and therefore $V^{\pi^*}(s)$ and $Q^{\pi^*}(s,a)$ are the respective value function and Q-function for the optimal policy. Consequently, the value function of the optimal policy at state $s$ will be equal to the maximum Q-function return:

$$V^{\pi^*}(s) = \max_a Q^{\pi^*}(s,a) \tag{6.6}$$

The Bellman Optimality states that for an MDP $M = S, A, P(s'|s,a), R, \gamma$, then a given policy $\pi : S \to A$ is optimal if for all $s \in S$ the following is satisfied:

$$\pi(s) \in arg\ \max_{a \in A} Q^{\pi}(s,a) \tag{6.7}$$

By subbing equation 6.5 into 6.6, the value function for the optimal policy at state $s$ can then be written as:

$$V^{\pi^*}(s) = \max_a \left[ R(s) + \gamma \sum_{s' \in S} P(s'|s,a)V^{\pi^*}(s') \right] \tag{6.8}$$

Since the value function has been defined iteratively, the value for each state in a finite state space can be then be calculated. The optimal policy $\pi^*(s)$ can then be found by taking the action at state $S$ that has the highest value for $Q(s,a)$. A commonly used method for solving the values is known as value iteration outlined in algorithm 3.

**Algorithm 3:** Value Iteration

```
Input:
$\eta$: Tolerance for change in $V(s)$
$(S, A, P(s'|s,a), R, \gamma)$: MDP Tuple
Output:
$V^{\pi^*}(s)$
$\pi^*(s)$
begin
    $V(s) \leftarrow \mathbf{0}$
    $k \leftarrow 0$
    $\lambda \leftarrow \infty$
    while $\lambda > \eta$ do
        foreach $s \in S$ do
            $V_{k+1}(s) \leftarrow \underset{a}{max}\left[R(s) + \gamma \sum_{s' \in S} P(s'|s,a) V_k(s')\right]$
        end
        $\lambda = |\boldsymbol{V}_{k+1}(s) - V_k(s)||$
        $V\pi^*(s) \leftarrow V_{k+1}(s)$
        $k \leftarrow k+1$$
    end
    foreach $s \in S$ do
        $\pi(s) \leftarrow \underset{a \in A}{arg\ max}\left[R(s) + \gamma \sum_{s' \in S} P(s'|s,a) V\pi^*(s')\right]$
    end
end
```

## 6.4 Inverse Reinforcement Learning

Inverse reinforcement learning was initially described by Russell et al. [163] as follows:

**Definition 6.4.** Inverse reinforcement learning:
**Given:** 1) measurements of an agent's behaviour over time, in a variety of circumstances; 2) measurements of the sensory inputs to that agent; 3) a model of the physical environment (including the agent's body).
**Determine:** the reward function that the agent is optimizing.

However, the initial formulation of the inverse reinforcement learning problem was done by Ng and Russell [136]. It is assumed that the expert is trying to optimise an unknown reward function, and IRL uses the expert's sampled decisions to learn this reward function.

One of the primary challenges of IRL is that the problem of obtaining a reward function from observations is ill-posed [136], [137], [164], [165]. In other words, there can be many reward functions that explain the observations. For example, a reward function with all zeros $\boldsymbol{R} = \mathbf{0}$ (or any other constant vector) results in every action producing the same reward which means any policy is optimal. Furthermore, even multiple non-constant vector reward functions could explain the same observations. This is due to the observations usually being in the form of small and finite trajectories and therefore many reward functions can generate policies that fit the observed trajectories. Section 6.4.1 outlines how Ng and Russell address this problem. Inverse reinforcement learning has seen a significant amount of research since the seminal papers. However, this research largely focusses on control problems [128], [140] with, to the author's knowledge, no work done on decision making for mission critical systems such as a maritime surveillance radar.

Another primary challenge of IRL is that the computational complexity grows exponentially with the size of the problem. In particular, the solving an MDP suffers from the curse of dimensionality—its size is exponential to the number of dimensions of the state-space. This curse becomes of particular issue when the problem has an effective discretisation of a continuous state. Additionally, the computational complexity is affected by the length and number of trajectories provided by the operator, and the larger the state-space, the longer and more numerous trajectories that are required in order to maintain the same level of coverage as would be required for a smaller state-space. Section 6.4.2 outlines how Ng and Russell reduced the computational complexity, however, the computational complexity is not of great concern for the the tasks considered here as the learning is done offline.

### 6.4.1 Linear Programming Formulation for Finite State Spaces

Ng and Russell [136] provide a characterisation for a set of reward functions for which a given policy is optimal. Firstly, note that equation 6.4 can be written in vectorial form as follows:

$$\boldsymbol{V}^{\pi} = \boldsymbol{R} + \gamma \boldsymbol{P}^{a} \boldsymbol{V}^{\pi} \tag{6.9}$$

where $\boldsymbol{P}^a$ is an $S \times S$ matrix with element $(s, s')$ indexing $P(s'|s,a)$. $\boldsymbol{V}^{\pi}$ and $\boldsymbol{R}$ are vectors with length $S$ with element $s$ indexing $V^{\pi}(s)$ and $R(s)$ respectively.

For the optimal policy $\pi^*(s)$, the optimal action is $a^*$ for any given state and thus $\pi^*(s) \equiv a^*$. By rearranging equation 6.9 and substituting in the optimal policy $\pi^*$ and the optimal action $a^*$, the following can be obtained:

$$\boldsymbol{V}^{\pi^*} = (\boldsymbol{I} - \gamma \boldsymbol{P}^{a^*})^{-1} \boldsymbol{R} \tag{6.10}$$

Recalling equation 6.7, and using the fact that the optimal action $a^*$ is identical to the optimal policy at state $s$ the following can be said:

$$\begin{aligned}
a^* \equiv \pi^*(s) &\in \underset{a}{arg\ max}\ Q^*(s,a) \qquad \forall s \in S \\
&= \underset{a}{arg\ max}\ \left[R(s) + \gamma \sum_{s'} P(s'|s,a)V^{\pi^*}(s')\right] \qquad \forall s \in S \\
&= \underset{a}{arg\ max}\ \sum_{s'} P(s'|s,a)V^{\pi^*}(s') \qquad \forall s \in S \\
\Leftrightarrow &\sum_{s'} P(s'|s,a^*)V^{\pi^*}(s') \geq \sum_{s'} P(s'|s,a)V^{\pi^*}(s') \qquad \forall s \in S, a \in A \\
\Leftrightarrow &\boldsymbol{P}^{a^*}\boldsymbol{V}^{\pi^*} \succeq \boldsymbol{P}^{a}\boldsymbol{V}^{\pi^*} \qquad \forall a \in A \setminus a^* \\
\Leftrightarrow &\boldsymbol{P}^{a^*}(\boldsymbol{I} - \gamma \boldsymbol{P}^{a^*})^{-1} \succeq \boldsymbol{P}^{a}(\boldsymbol{I} - \gamma \boldsymbol{P}^{a^*})^{-1} \qquad \forall a \in A \setminus a^*
\end{aligned}$$

This can then be rearranged to give the following condition:

$$(\boldsymbol{P}^{a^*} - \boldsymbol{P}^{a})(\boldsymbol{I} - \gamma \boldsymbol{P}^{a^*})^{-1}\boldsymbol{R} \succeq 0 \qquad \forall a \in A \setminus a^* \tag{6.11}$$

This condition holds true if $\pi(s)$ is optimal, however there are many cases (e.g. $\boldsymbol{R}$ being a constant vector) where a solution is degenerate and consequently any policy will be optimal. Furthermore, there are often multiple reward vectors that satisfy the above condition. Ng and Russell applied a further constraint to favour solutions that reduce the total reward by as much as possible when deviating from the optimal policy. This is done by finding the solution (out of those that satisfy 6.11) which maximises the following:

$$\sum_{s} \left( Q^{\pi^*}(s,a^*) - \max_{a \in A \setminus a^*} Q^{\pi^*}(s,a) \right) \tag{6.12}$$

By replacing the *max* inside the brackets with a *min* out of the brackets the above can be rewritten as:

$$maximise \sum_{s} \min_{a \in A \setminus a^*} \left[ Q^{\pi^*}(s,a^*) - Q^{\pi^*}(s,a) \right] \tag{6.13}$$

In a similar manner to the derivation for equation 6.11, this can be further rewritten as:

$$maximise \sum_{s} \min_{a \in A \setminus a^*} \left[ (\boldsymbol{P}^{a^*} - \boldsymbol{P}^{a})(\boldsymbol{I} - \gamma \boldsymbol{P}^{a^*})^{-1}\boldsymbol{R} \right] \tag{6.14}$$

Additional criteria applied by Ng and Russell introduces L1 regularisation in order to favour solutions with small rewards with the intuition being that they are "simpler". Note that for L1 regularisation, $||\boldsymbol{R}||_1 = \sum_i^N |\boldsymbol{R}_i|$ and it is typically applied in the form of a weight decay penalty term i.e. $-\lambda ||\boldsymbol{R}||_1$ where $\lambda$ is adjustable. This then gives the following linear programming formulation to obtain a reward function:

$$maximise \sum_{i=1}^{N} \min_{a \in A \setminus a^*} \left[ (\boldsymbol{P}^{a^*}(i) - \boldsymbol{P}^{a}(i))(\boldsymbol{I} - \gamma \boldsymbol{P}^{a^*})^{-1}\boldsymbol{R} \right] - \lambda ||\boldsymbol{R}||_1 \tag{6.15}$$

$$s.t.\ (\boldsymbol{P}^{a^*} - \boldsymbol{P}^{a})(\boldsymbol{I} - \gamma \boldsymbol{P}^{a^*})^{-1}\boldsymbol{R} \succeq 0 \quad \forall a \in A \setminus a^*, \quad |\boldsymbol{R}_i| \leq R_{max}$$

Where $\boldsymbol{P}_a(i)$ indicates the $i$th row of $\boldsymbol{P}_a$.

## 6.4.2 Large or Infinite State Spaces

For large or infinite state spaces the formulation outlined in the previous section becomes problematic and it is instead easier to use linear approximations in conjunction with feature vectors. The feature vectors $\boldsymbol{\phi}(s)$ are composed of features (or basis functions) $\phi_i(s)$ which describe the current state (e.g. distance to an object). The reward function is then just the linear combination of the features such that:

$$R(s) = \alpha_1\phi_1(s) + \alpha_2\phi_2(s) + ... + \alpha_d\phi_d(s) = \boldsymbol{\alpha} \cdot \boldsymbol{\phi}(s) \tag{6.16}$$

Where $\alpha_i$ are unknown weighting parameters with $|\alpha_i| \leq 1$ .

$V_i^\pi$ is denoted as the value function for the policy $\pi$ when the reward function is $R = \phi_i$. Using the linearity of expectations in conjunction with this definition and equation 6.16 the value function can then also be written as a linear function:

$$V^\pi(s) = \alpha_1 V_1^\pi(s) + \alpha_2 V_2^\pi(s) + ... + \alpha_d V_d^\pi(s) \tag{6.17}$$

It can be recalled from the derivation for equation 6.11 that:

$$\sum_{s'} P(s'|s,a^*)V^{\pi^*}(s') \geq \sum_{s'} P(s'|s,a)V^{\pi^*}(s') \qquad \forall s \in S, a \in A \tag{6.18}$$

Using this, the condition for the optimal policy can be generalised for the linear approximations (note the difference in the action for the transition probabilities):

$$\underset{s' \sim P(s'|s,a^*)}{\boldsymbol{E}} \left[V^{\pi^*}(s')\right] \geq \underset{s' \sim P(s'|s,a)}{\boldsymbol{E}} \left[V^{\pi^*}(s')\right] \quad \forall s \in S,\ \forall a \in A \setminus a^* \tag{6.19}$$

This condition gives rise to two problems in particular. The first problem is that for an infinite state space, the above condition becomes difficult or impossible to check for all possible states. Ng and Russell instead sample a large but finite subset of states $S_0$ and then apply the condition for all $s \in S_0$. The second problem is that the linear approximation prevents any reward function from being expressed such that the policy $\pi$ is optimal. Ng and Russell opt to relax the above condition by introducing a function $p$ that penalises the violation of the condition. In the same vein as 6.15, the linear programming formulation for infinite state spaces is then written as:

$$maximise \sum_{s \in S_0} \underset{a \in A \setminus a^*}{min}\ p\left( \underset{s' \sim P(s'|s,a^*)}{\boldsymbol{E}} [V^{\pi^*}(s')] - \underset{s' \sim P(s'|s,a)}{\boldsymbol{E}} [V^{\pi^*}(s')] \right) \tag{6.20}$$

$$s.t.\ |\alpha_i| \leq 1,\ i = 1, ..., d$$

Where $p(x) = x$ when $x \geq 0$ and $p(x) = 2x$ when $x < 0$. The penalty coefficient of 2 was chosen heuristically by Ng and Russell.

### 6.4.3 Sampled Trajectories

For practical cases, access to the expert policy will not be direct but rather through a sample of trajectories. A single trajectory is, for example, the series of actions that are captured whilst the operator is performing a task. For the sake of simplicity, it is assumed that there is only one start state $s_0$ from a given fixed state distribution $D$. Note however, that this assumption is without loss of generality as $s_0$ can be a “dummy" state with any action from $s_0$ leading to the next state distribution $D$).

Given policy $\pi$, it is necessary to find an estimate for $V^{\pi}(s_0)$ given any values for $\alpha_i$. If there are $M$ trajectories with policy $\pi$ then the average return, if the reward was $R = \phi_i$, is given by $\hat{V}_i^{\pi}(s_0)$. For example, for the case where $M = 1$ and the sequence of states for the trajectory is $(s_0, s_1, s_2, ...)$ then the average $\hat{V}_i^{\pi}(s_0)$ is given by:

$$\hat{V}_i^{\pi}(s_0) = \phi_i(s_0) + \gamma\phi_i(s_1) + \gamma^2\phi_i(s_2) + ... \tag{6.21}$$

More generally $\hat{V}_i^{\pi}(s_0)$ would then be the average over the $M$ trajectories. By recalling equation 6.17 the estimate for $V^{\pi}(s_0)$ is then given by:

$$\hat{V}^{\pi}(s_0) = \alpha_1\hat{V}_1^{\pi}(s_0) + \alpha_2\hat{V}_2^{\pi}(s_0) + ... + \alpha_d\hat{V}_d^{\pi}(s_0) \tag{6.22}$$

Using 6.16 and 6.1, another way of formulating this is as follows:

$$\hat{V}^{\pi}(s_0) = \frac{1}{M}\sum_{m=1}^{M}\sum_{t=0}^{\infty}\gamma^t R(s_t^m) \tag{6.23}$$

At the start of the algorithm [136], [166] the value estimates are calculated for the given set of trajectories provided by the expert. Since it is assumed that the expert will provide optimal policy trajectories, $\hat{V}_i^{\pi^*}(s_0)$ will then be the average of over the given trajectories:

$$\hat{V}^{\pi^*}(s_0) = \alpha_1\hat{V}_1^{\pi^*}(s_0) + \alpha_2\hat{V}_2^{\pi^*}(s_0) + ... + \alpha_d\hat{V}_d^{\pi^*}(s_0) \tag{6.24}$$

The value estimates are also calculated for a “base case" random policy $\pi_1$ which is used as the first entry of $\Pi$. On the $k$th iteration of the algorithm, $\Pi$ will be appended with policy $\pi_{k+1}$.

For the true reward function, the value of the expert policy $V^{\pi^*}(s_0)$ should be at least greater than or equal to any other policy $V^{\pi}(s_0)$ (since it assumed that the expert provides the optimal policy). Thus, the aim of the algorithm is to find a reward function (specifically a setting for the $\alpha_i$ coefficients) that satisfies the following condition:

$$V^{\pi^*}(s_0) \geq V^{\pi}(s_0) \quad \forall \pi \in \Pi \tag{6.25}$$

This requires modification of the previous optimisation function to give a new linear programming formulation:

$$maximise \sum_{\pi \in \Pi} p\left(\hat{V}^{\pi^*}(s_0) - \hat{V}^{\pi}(s_0)\right) \tag{6.26}$$

$$s.t.\ |\alpha_i| \leq 1,\ i = 1, ..., d$$

Where as before:

$$p(x) = \begin{cases} x, & x \geq 0 \\ 2x, & x < 0 \end{cases}$$

This optimisation obtains a reward function (specifically a setting for the $\alpha_i$ coefficients). The rest of the algorithm is then formed of a loop consisting of the above. Explicitly, a new policy $\pi_{k+1}$ is obtained that is optimal for the newly obtained reward function (this can be solved for example using the value iteration algorithm described previously). $\pi_{k+1}$ is then appended to $\Pi$ and the linear programming formulation is solved to obtain a new reward function. The algorithm ends when $\hat{V}^{\pi^*}(s_0) - \hat{V}^{\pi_{k+1}}(s_0) \leq \varepsilon$ where $\varepsilon$ is a set tolerance. This algorithm is further explained in appendix A.

### 6.4.4 Feature Expectations

Abbeel and Ng [137] expand on the previously outlined algorithms but instead focused on obtaining a policy rather than a reward function. However, it is still assumed that an underlying unknown reward function is being optimised by the expert, just that the success of this is not necessary. Abbeel and Ng also expand on the idea of ‘features’ by introducing ‘feature expectations’: recalling 6.16 and 6.1,

$$\begin{aligned} V^{\pi}(s_0) &= \boldsymbol{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t)\right] \\ &= \boldsymbol{E}\left[\sum_{t=0}^{\infty} \gamma^t \boldsymbol{\alpha} \cdot \boldsymbol{\phi}(s_t)\right] \\ &= \boldsymbol{\alpha} \cdot \boldsymbol{E}\left[\sum_{t=0}^{\infty} \gamma^t \boldsymbol{\phi}(s_t)\right] \\ &= \boldsymbol{\alpha} \cdot \mu(\pi) \end{aligned}$$

where feature expectations $\mu$ are then defined as:

$$\mu(\pi) = \boldsymbol{E}\left[\sum_{t=0}^{\infty} \gamma^t \phi(s_t)\right] \tag{6.27}$$

As before, the expert's policy is typically obtained through a set of trajectories. Expanding on 6.23 with the use of 6.27, the feature expectations of the expert can then be estimated with the following:

$$\hat{\mu}(\pi^*) = \frac{1}{M}\sum_{m=1}^{M}\sum_{t=0}^{\infty} \gamma^t \phi(s_t^m) \tag{6.28}$$

The key difference from the previous algorithm is that the aim is to obtain a policy which, in terms of performance on the unknown reward function, is close to that of the expert's. By using feature expectations, the obtained policy will not only mimic the expert's policy but will also be able to perform in the manner of the expert within unseen states. Putting these together, the aim is to obtain a policy $\pi$ which satisfies the following condition:

$$||\mu(\pi^*) - \mu(\pi)||_2 \leq \varepsilon \tag{6.29}$$

The next step is to frame the above into optimisation problem. Starting in a similar manner to 6.25:

$$\begin{aligned} |V^{\pi^*}(s_0) - V^{\pi}(s_0)| &= |\boldsymbol{\alpha} \cdot \mu(\pi^*) - \boldsymbol{\alpha} \cdot \mu(\pi)| \\ &= |\boldsymbol{\alpha} \cdot (\mu(\pi^*) - \mu(\pi))| \\ &\leq ||\boldsymbol{\alpha}||_2 ||\mu(\pi^*) - \mu(\pi)||_2 \\ &\leq ||\mu(\pi^*) - \mu(\pi)||_2 \end{aligned} \tag{6.30}$$

For the above, note that $|\boldsymbol{x}^T\boldsymbol{y}| \leq ||\boldsymbol{x}||_2||\boldsymbol{y}||_2$ and that $||\boldsymbol{x}||_2 \leq ||\boldsymbol{x}||_1 \leq 1$. This shows that for condition 6.29 to hold true then $||\boldsymbol{\alpha}||_2 \leq 1$ must also hold true. Using the first line in the above, the optimisation problem can then be given as:

$$\underset{t,\alpha}{maximise} \quad t \tag{6.31}$$

$$s.t. \quad \boldsymbol{\alpha} \cdot \mu(\pi^*) \geq \boldsymbol{\alpha} \cdot \mu(\pi) + t \quad \forall \pi \in \Pi, \quad ||\boldsymbol{\alpha}||_2 \leq 1$$

The optimisation is trying to obtain a reward function (specifically a setting of the values of $\alpha_i$) such that $V^{\pi^*}(s_0) \geq V^{\pi}(s_0) + t$. In other words, a reward function with which the expert has a greater performance than any previously determined policy by a margin of $t$.

The max-margin approach to inverse reinforcement learning is outlined in algorithm 4 where the concatenation of a policy $\pi$ to the policy set $\Pi$ is donated by $\Pi^\frown\langle\pi\rangle$:

Algorithm 4 returns a set of policies, of which at least one policy whose performance under $R^*$ is at least as good as the expert's performance minus $\varepsilon$. The policies in the set can be

**Algorithm 4:** Inverse Reinforcement Learning: Max-Margin Algorithm

```
Input:
$\Pi$: Empty set of policies
$\varepsilon$: Tolerance for $t$
$\mu^E$: Feature expectations from expert trajectories
$(S, A, P(s'|s,a), \gamma)$: MDP\R Tuple
Output:
$\Pi$
begin
    $\pi^0 \leftarrow$ random initial policy
    $\Pi \leftarrow \Pi^\frown\langle\pi^0\rangle$
    $\mu^0 = \mu(\pi^0)$
    $i \leftarrow 1$
    while $t^i > \varepsilon$ do
        Obtain $t^i$ by solving optimisation 6.31
        Set $\boldsymbol{\alpha}^i$ to be the $\boldsymbol{\alpha}$ that maximises the margin $t$
        Obtain optimal policy $\pi^i$ using algorithm 3 with the MDP where $R = \boldsymbol{\alpha^i} \cdot \phi$
        Append $\pi^i$ to $\Pi$
        $\mu^i = \mu(\pi^i)$
        $i \leftarrow i + 1$
    end
end
```

"mixed" using mixture weights $\lambda_i$ by finding the point closest to $\mu^E$ in the convex closure of $\mu^0, ..., \mu^n$ (where $n$ is the number of policies in $\Pi$). This point, and therefore the policy, can be obtained by solving the following quadratic programming formulation:

$$\min ||\mu^E - \mu||_2 \tag{6.32}$$

$$s.t. \quad \mu = \sum_i^n \lambda_i \mu^i, \quad \lambda_i \geq 0, \quad \sum_i^n \lambda = 1$$

This optimisation problem outlined in equation 6.31 can be equated to the SVM (support vector machine) problem by assigning a label of +1 to the expert's feature expectations $\mu(\pi^*)$ while all the other policies $\mu(\pi)\forall\pi \in \Pi$ can be given a label of -1. The vector $\alpha$ in the context of an SVM is the vector describing the normal of the hyperplane which is to be optimised to obtain the maximum margin hyperplane. The linear SVM problem can be equated as:

$$\underset{\boldsymbol{\alpha}, b}{minimise} \quad \frac{1}{2}||\boldsymbol{w}||^2 \tag{6.33}$$

$$s.t. \quad y_i(\boldsymbol{\alpha}^T \cdot \boldsymbol{x}_i + b) \geq 1 \quad for \quad i = 1, 2, ..., l$$

Note that since this formulation introduces the Euclidean norm, the formulation is no longer linear and as such cannot be solved using linear programming methods. Instead the optimisation problem is solved with quadratic programming methods. A quadratic programming solver provides the solution to the following:

$$\begin{aligned} \underset{x}{minimise} \quad & \frac{1}{2} x^T P x + q^T x \\ s.t. \quad & Gx \preceq h \end{aligned} \tag{6.34}$$

Therefore the SVM problem can be written in the quadratic programming form as follows:

$$\begin{aligned} x &= [\boldsymbol{\alpha}, b]^T \\ P &= \begin{bmatrix} \boldsymbol{I} & 0 \\ 0 & 0 \end{bmatrix} \\ q &= [0, ..., 0]^T \\ G &= \begin{bmatrix} -y_1 \boldsymbol{x}_1^T & -y_1 \\ \vdots & \vdots \\ -y_l \boldsymbol{x}_l^T & -y_l \end{bmatrix} \\ h &= [-1, ..., -1]^T \end{aligned}$$

The weight vector can then be normalised to ensure that the condition $||\boldsymbol{\alpha}||_2 \leq 1$ is satisfied.

An alternative to the max-margin approach is the projection method outlined in algorithm 5. In summary, this method replaces the optimisation step in the max-margin algorithm with a orthogonal projection of the expert's feature expectations onto a line through the previous estimated expectations and the calculated expectations. The full derivation of this algorithm is provided in Abbeel and Ng [137].

**Algorithm 5:** Inverse Reinforcement Learning: Projection Algorithm

```
Input:
$\varepsilon$: Tolerance for $t$
$\mu^E$: Feature expectations from expert trajectories
$(S, A, P(s'|s,a), \gamma)$: MDP\R Tuple
Output:
$\pi^*(s)$
begin
    $\pi^0 \leftarrow$ random initial policy
    $\mu^0 = \mu(\pi^0)$
    $\hat{\mu}^0 = \mu^0$
    $\boldsymbol{\alpha}^0 = \mu^E - \mu^0$
    $i \leftarrow 1$
    while $t^i > \varepsilon$ do
        $\boldsymbol{\alpha}^i = \mu_E - \hat{\mu}^i$
        Obtain optimal policy $\pi^i$ with the MDP where $R = \boldsymbol{\alpha^i} \cdot \phi$
        $\mu^i = \mu(\pi^i)$
        $\hat{\mu}^i = \hat{\mu}^{i-1} + \frac{(\mu^i - \hat{\mu}^{i-1})^T (\mu_E - \hat{\mu}^{i-1})}{(\mu^i - \hat{\mu}^{i-1})^T (\mu^i - \hat{\mu}^{i-1})} (\mu^i - \hat{\mu}^{i-1})$
        $t^i = ||\mu_E - \hat{\mu}^i||_2$
        $i \leftarrow i + 1$
    end
    $\pi^* \leftarrow \pi^i$
end
```

## 6.5 Maritime Radar Surveillance Task Setup

Following on from section 5.3, this section outlines how the maritime radar surveillance tasks are represented within the MDP framework.

### 6.5.1 Task 1

In this example, state $s_1^1$ to $s_1^N$ represents all possible combinations of whether a there are tracks within, within 10 km, or not within 10 km of one of the designated zones. Action $a_{\mathrm{NW}}$, $a_{\mathrm{W10}}$, and $a_{\mathrm{W10}}$ therefore represent the decision to select the tracks that are within one of these states.

State $s_2^{\mathrm{K}}$ and $s_2^{\mathrm{UK}}$ represent whether or not the track has a transponder respectively (i.e. known or unknown track). Consequently, action $a_2^{\mathrm{K}}$ and $a_2^{\mathrm{UK}}$ represents the decisions to select the known or unknown tracks respectively.

State $s_3^1$ to $s_3^N$ represent all possible combinations of whether or not any of the selected tracks have abnormal sizes and whether or not any of the selected tracks have abnormal speeds. These actions are summarised in table 6.1.

State $s_4^1$ to $s_4^N$ represent all possible discrete directions each of the remaining tracks could be heading in. In this example, these discrete directions are taken to be towards or away from one of the designated zones, hence action $a_4^{\mathrm{T}}$ represents the decision to select the tracks

Table 6.1: The actions for all possible track abnormality combinations.

| Action | Description |
|---|---|
| $a_3^{\mathrm{AN}}$ | Select the tracks with abnormal speeds but normal sizes |
| $a_3^{\mathrm{AA}}$ | Select the tracks with abnormal speeds but abnormal sizes |
| $a_3^{\mathrm{NA}}$ | Select the tracks with normal speeds but abnormal sizes |
| $a_3^{\mathrm{NN}}$ | Select the tracks with normal speeds but normal sizes |

heading towards whilst action $a_4^{\mathrm{A}}$ represents the decision to select the tracks away.

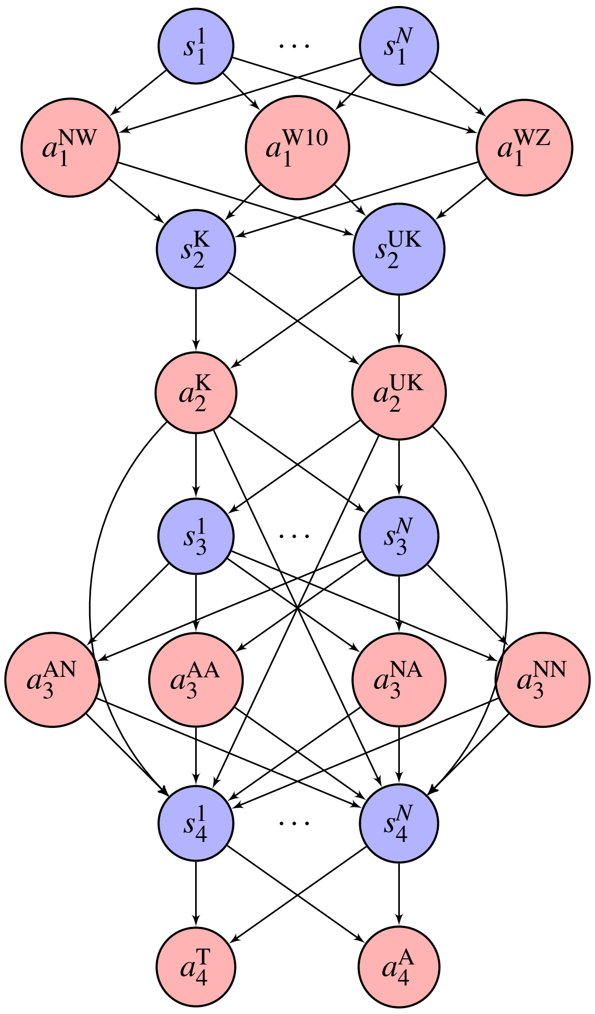


Figure 6.4: Task 1 represented as a MDP.

### 6.5.2 Task 2

State $s_1^1$ to $s_1^N$ represents all possible combinations of whether the selected track's transponder information is anomalous or conforming and whether or not the track is within one of the designated zones. Action $a_1^1$ to $a_1^4$ represent the decisions to further investigate a track with or without anomalous transponder information and whether or not the track is within one of the designated zones. In this context, investigate means to apply sector scan on the track and/or to communicate with the vessel.

State $s_2^1$ to $s_2^N$ represents all possible combinations of the vessel's response, the track's direction, and the track's speed. Correspondingly, action $a_{\mathrm{MT}}$ represents the decision to move towards the track for closer inspection, and $a_{\mathrm{MA}}$ represents the decision to move away from the track and consider another track as per task 1.

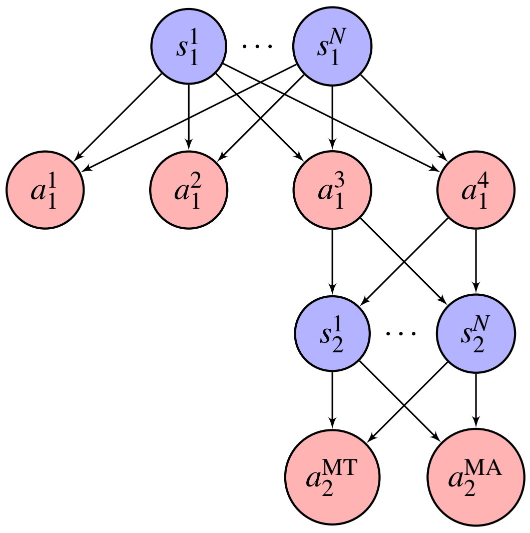


Figure 6.5: Task 2 represented as a MDP.

### 6.5.3 Task 3

State $s_1^1$ to $s_1^N$ represents all possible combinations of the change in probability of the track performing an illegal operation and the change in direction made by the operator on the previous decisions. Consequently, $a_1^1$ to $a_1^N$ represent the discrete direction of movement the operator commands the platform in.

$s_2^1$ to $s_2^N$ represent the discrete probability of detection bins for whether or not the track is performing an illegal operation. The action $a_2^{\mathrm{CM}}$ represents the decision to continue moving towards the track in order to get a better view, action $a_2^{\mathrm{CM}}$ represents the decision to return to the surveillance path and note that the track is not likely to be performing an illegal operation, and action $a_2^{\mathrm{DI}}$ represents the decision to declare the track as potentially carrying out an illegal operation.

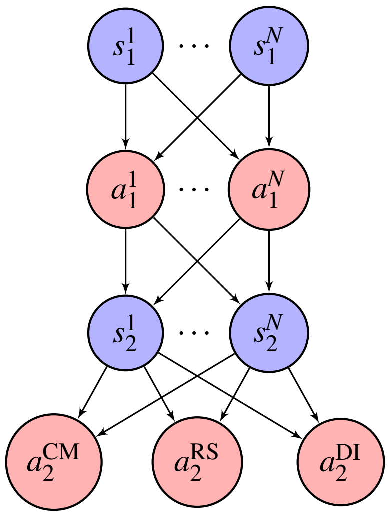


Figure 6.6: Task 3 represented as a MDP.

## 6.6 Conclusions

In this chapter, the use of inverse reinforcement learning algorithms for imitating a radar operator carrying out a maritime surveillance mission has been outlined. The Markov decision networks' policies were learned from operator decision data with the used of a real-time radar simulation for three common surveillance tasks. The results of this network for the three tasks is shown in chapter 7 and compared and contrasted with the previously outlined Bayesian network approach.

# Chapter 7

# Comparison of Methods

## 7.1 Introduction

In this section, the Bayesian network approach is compared against the IRL approach for the tasks outlined in section 4. The comparison is done in terms of their accuracy in decision making relative to that of the radar operator as well as their suitable in terms of qualification for operation. Additionally, different decision discretisation configurations are compared.

Each maritime radar surveillance task was simulated and ran a task-specific number times in order to capture both the information observed by the operator and the operator's decisions. When comparing the decisions made by the chosen algorithm relative to the operator, the recorded operator decision data was split into training data and test data. The reason for this is so the AI agent's decisions are compared against situations it was not trained with. The training data is the data instances used to train the network/policy, with the size of the training data equal to $N$. The size of the test data was fixed at 100 data instances for all tasks. The size of training data (i.e. $N$) was varied and the results for each variation plotted.

For each value of $N$, 20 networks/policies were obtained (with the order of the training data randomised for every network) in order to ascertain a mean accuracy and error for each method. Each of these networks/policies was tested within the simulation against the operator's decisions using the test data. In other words, given the same observed information, the decisions of the AI agent were compared with the decisions previously made by the operator. The accuracy for each decision was simply the number of correct decisions relative to the operator divided by $N$.

Upon testing the different IRL methods outlined in section 6, there was no apparent difference in the policies learned for the tasks considered here. Consequently, the projection method was selected as it did not require the additional step of policy selection.

A study [167] showed that a system must have a correctness of at least 95% to be useable by operators. Consequently, this number was used to determine if a task was successfully learned by the chosen algorithm.

### 7.1.1 Main Contributions

The contributions of this chapter are as follows:

1. Firstly, two algorithms are compared in their efficacy in imitating a human radar operator.
2. The accuracy of each decision for each task are compared and contrasted.
3. Several different task variable discretisation configurations are outlined with their accuracies compared and contrasted.
4. The sources of error in the AI decision making process are then discussed.
5. Lastly, the two algorithms are then discussed in terms of how well they meet the system qualification requirements.

## 7.2 Results for Each Decision in Each Task

For the Bayesian network approach, the results of task 1, task 2, and task 3 in terms of accuracy and error margins are shown in Fig. 7.1, 7.3, 7.5 respectively. And for the IRL approach, the results of task 1, task 2, and task 3 in terms of accuracy and error margins are shown in Fig. 7.2, 7.4, 7.6 respectively. These graphs shown the mean accuracy of each decision node for a given number of data instances with error bars shown representing 1 standard error. Note that the difference between the two approaches is presented in section 7.4.

### 7.2.1 Task 1

For task 1, the decision node with the lowest initial accuracy is the selection of tracks based on anomalous AIS information. This is due to the fact that this node is only used when the AIS data is known, and consequently data for this node is often missing. Additionally, if the scene has few tracks available, the track is occasionally selected before the fourth decision is required. As such, with few data instances, the AI agent performs the fourth decision poorly, but with high data instances is able to perform accurately. It is also unsurprising that decisions further down the process required more data to achieve the same performance. Data would only be seen for nodes further down the network if previous decisions were made that led to those nodes. As such, any missions with a long sequence of decisions will require significantly more operator data than a mission with a comparable number of decisions but performed in parallel. However, both approaches still provided high levels of accuracy for this mission.

The decision to select tracks with a transponder (known) or without a transponder (unknown)

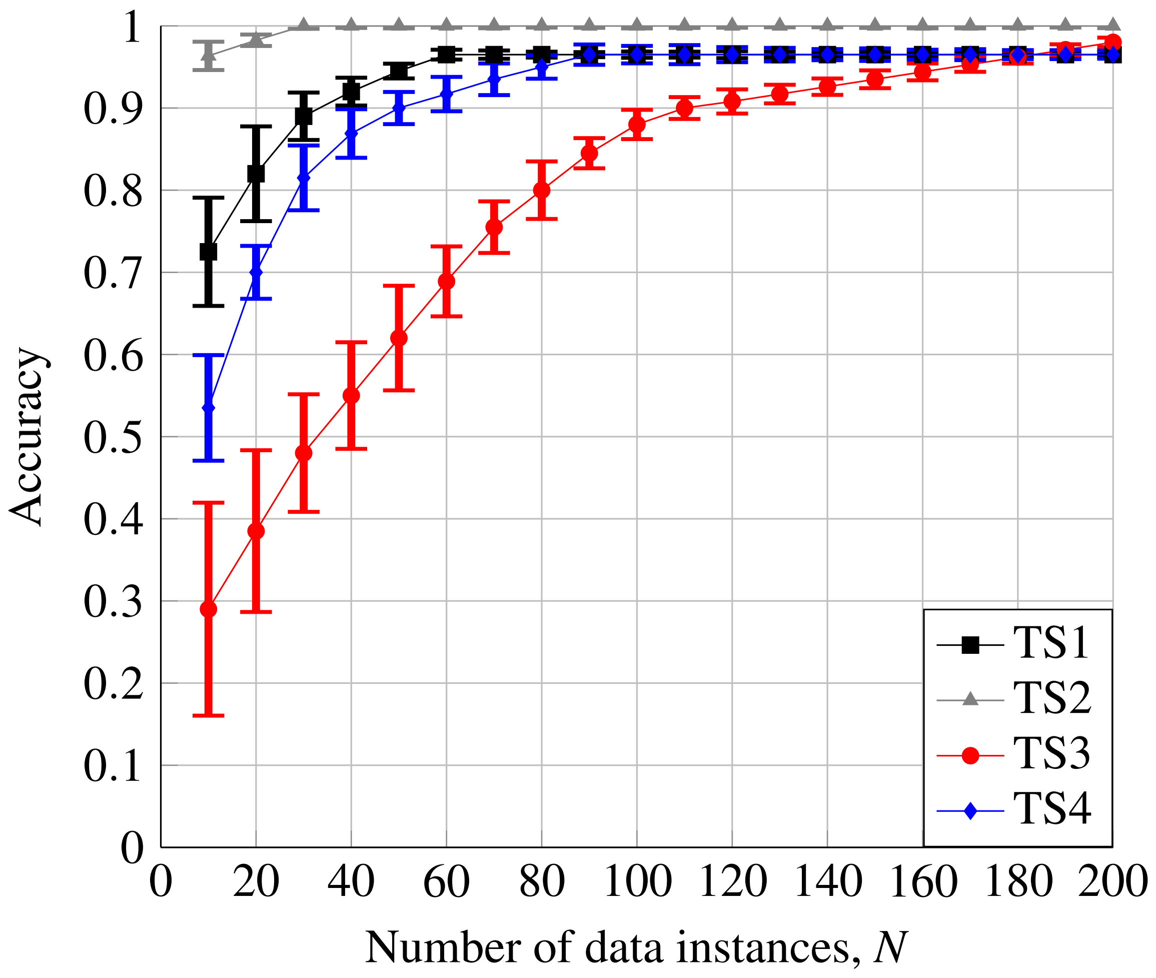


Figure 7.1: Accuracy and error in each Bayesian network decision node for task 1.

has the highest accuracy as well as requiring few data instances to reach this accuracy. This is, recalling from section 5.3, due to the decision only having two options, as well as the fact that the operator, baring any errors, will always select the unknown tracks. Specifically, both approaches were able to achieve an accuracy of 0.9625 after only 10 data instances.

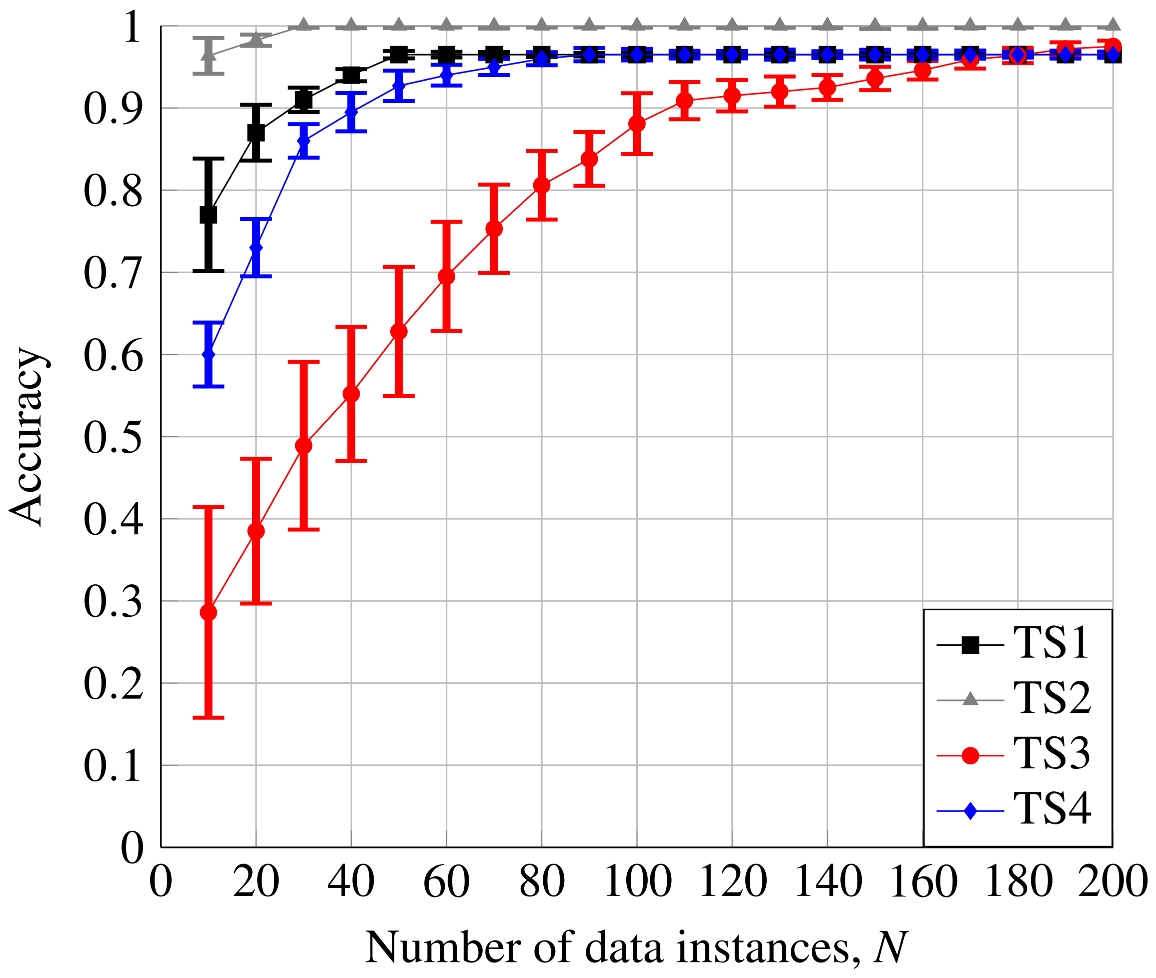


Figure 7.2: Accuracy and error in each IRL decision for task 1.

### 7.2.2 Task 2

For task 2, it is surprising that the node furthest down the decision process performs slightly better than the other decisions. However, this is due to the fact that this decision is more deterministic than the other two decisions for the operator. In other words, the operator will likely only move towards the track when the track is in specific and defined situations.

The sector scan (SS) decision initially performs worse than the CMV decision in both the Bayesian network and the IRL approach, despite being connected to the same nodes. This is likely due to to the operator being more deterministic in the CMV decision than the SS decision, as the SS decision takes far less effort and has far less impact on the operational environment than communicating with the vessel.

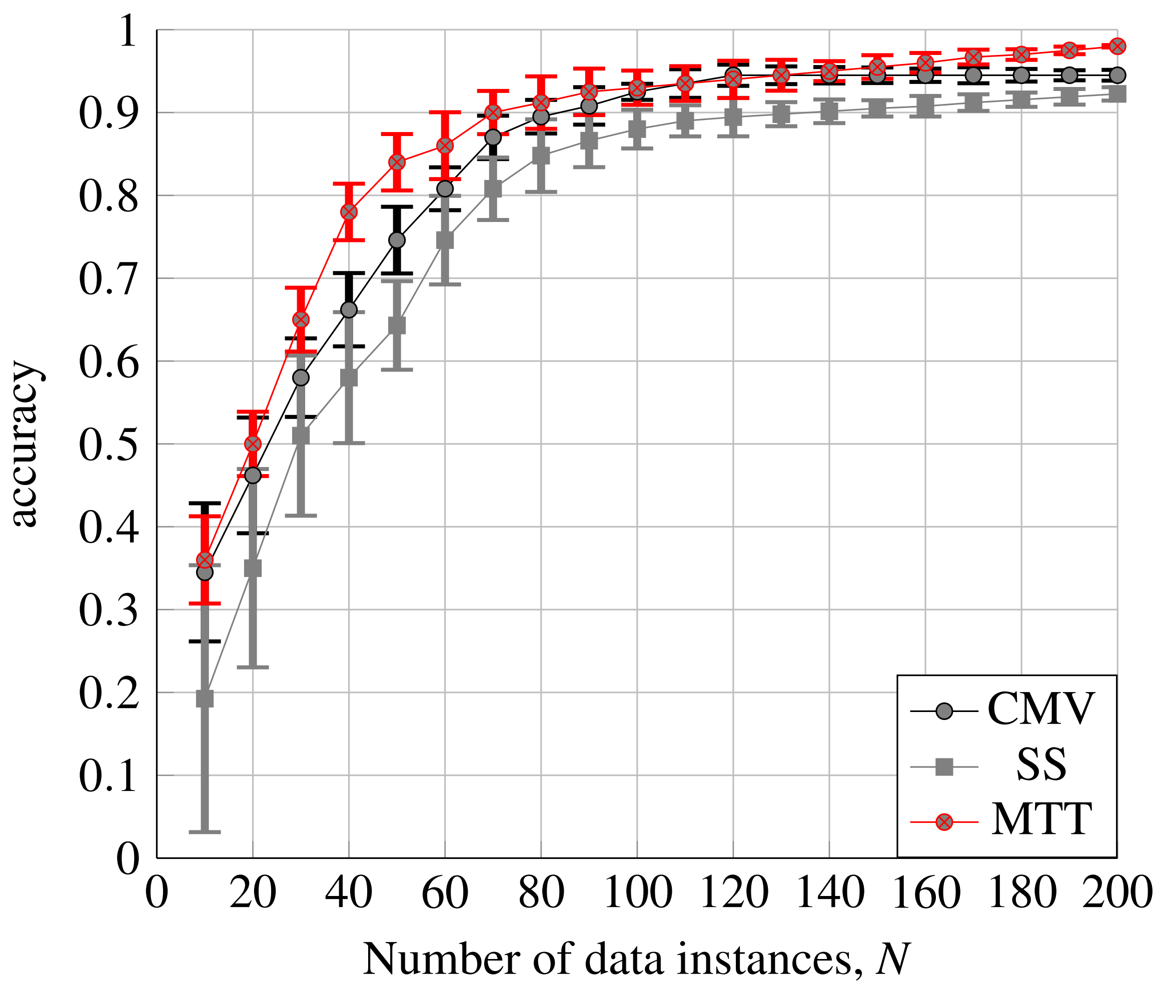


Figure 7.3: Accuracy and error in each Bayesian network decision node for task 2.

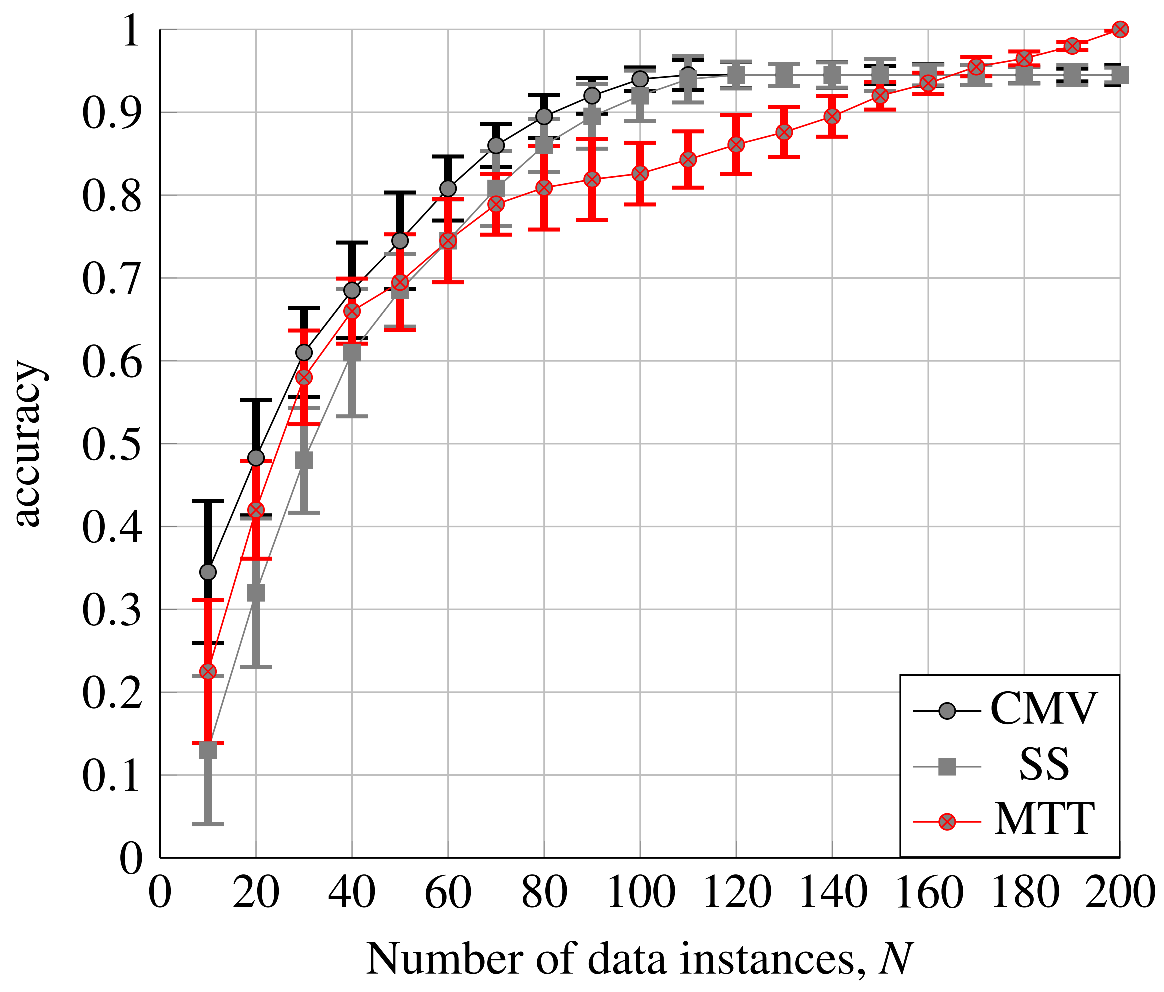


Figure 7.4: Accuracy and error in each IRL decision for task 2.

### 7.2.3 Task 3

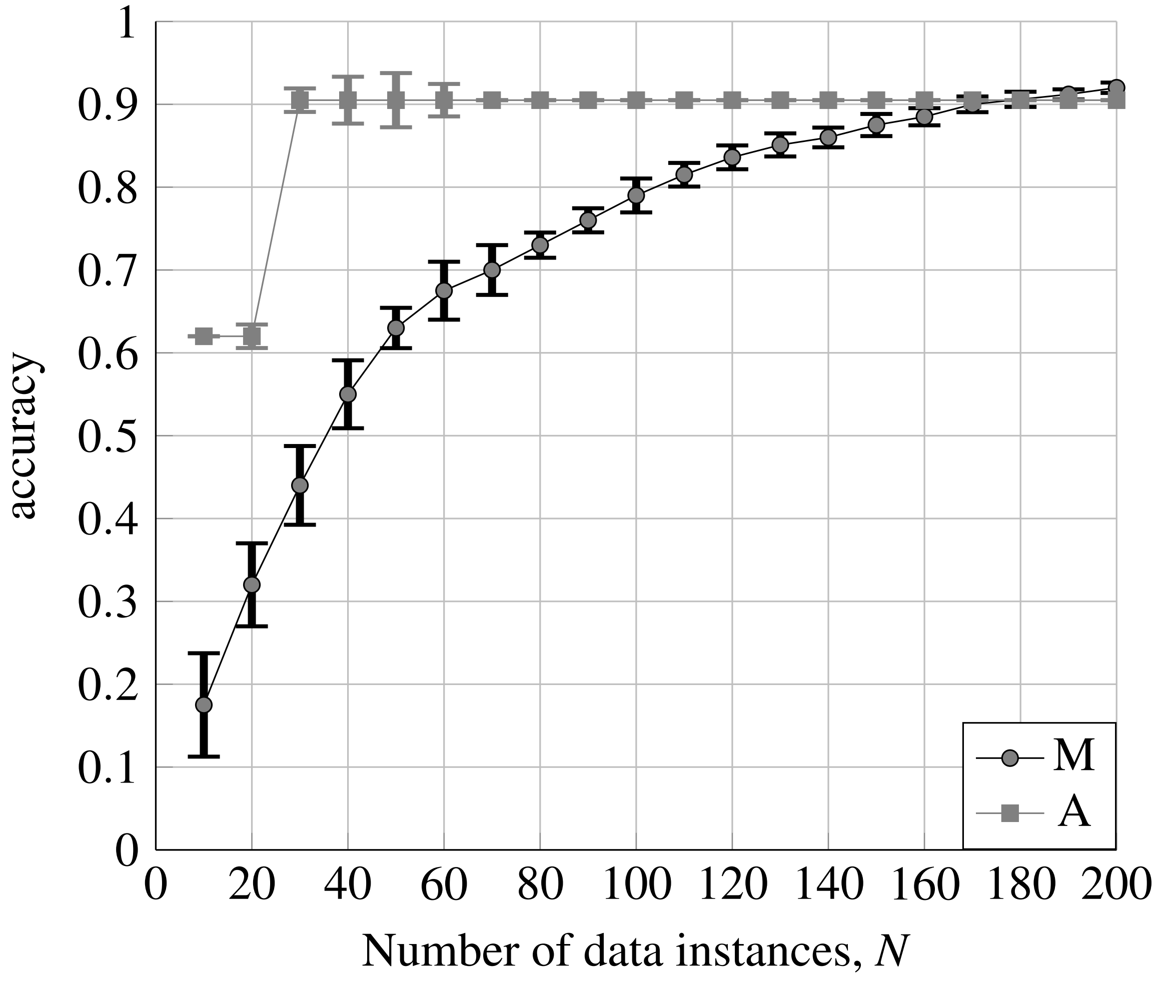


Figure 7.5: Accuracy and error in each Bayesian network decision node for task 3.

For task 3, initially the Action decision has a higher accuracy, but at a greater number of data instances the Move decision is higher. This is likely due to the more stochastic nature of the move decision which results in a reduced accuracy due to the deterministic manner of the AI.

Additionally, the Action decision plateaus around an accuracy of 0.9 with few data instances. This is likely due the decision being simplistic in terms of there being only one input that the operator considers, i.e. the probability of the target performing an illegal operation. As a consequence of there only being one input for the operator to consider, the way this input is discretised for the given imitation learning approach will greatly affect how many data instances are required to achieve a given accuracy as well as the limit of the decision accuracy.

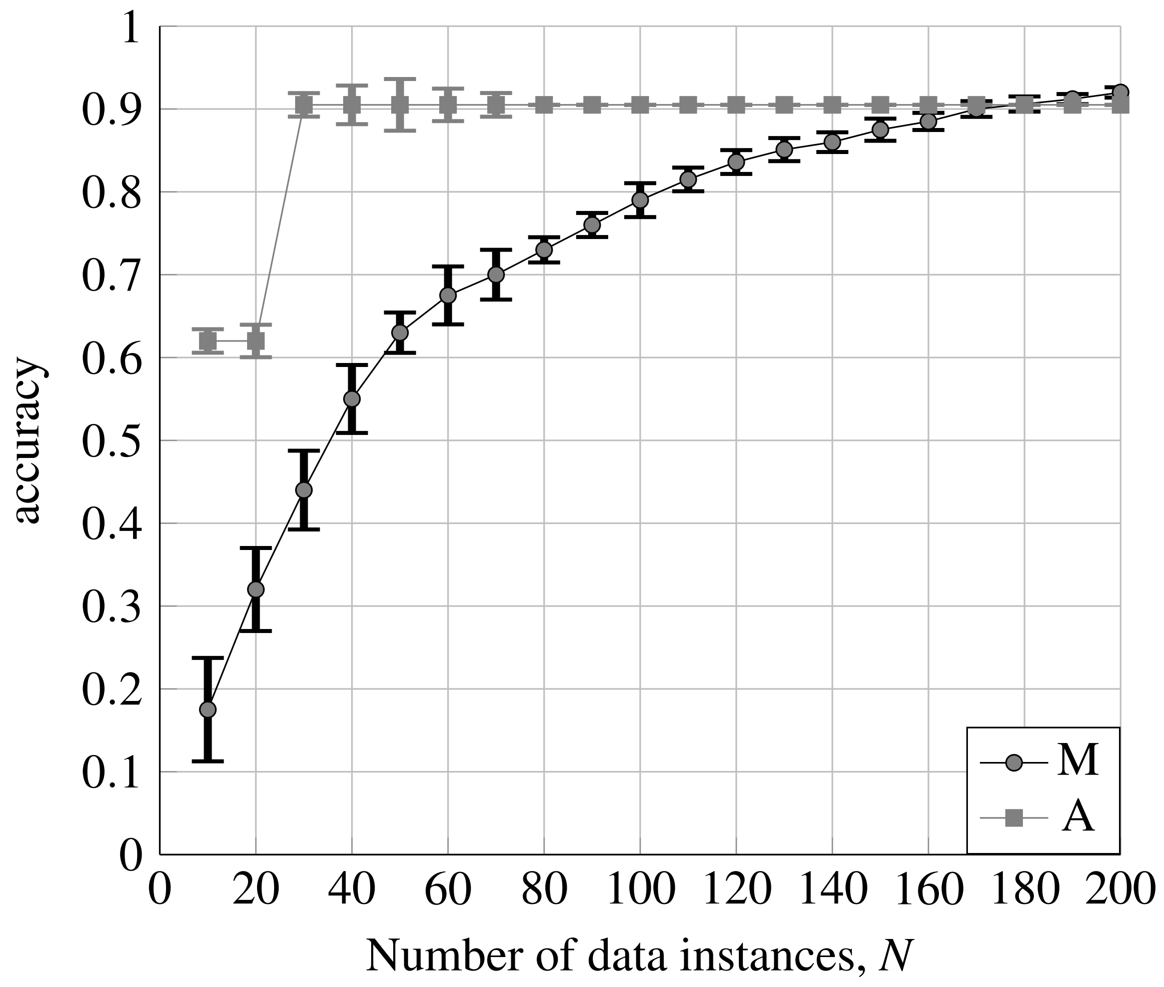


Figure 7.6: Accuracy and error in each IRL decision for task 3.

## 7.3 Results for Each Decision for Each Discretisation

In this section, different discretisation configurations are compared for each decision in each task. Note that in task 1, the decision to select known or unknown tracks can only have one form of discretisation, so there are no results in this section for this decision.

For the decisions that involve a track's distance to the nearest designated zone, the following three different discretisation configurations were used:

- Discretisation 1
  - track is within one of the designated zones
  - track is within 10 km of one of the designated zones
  - track is outwith 10 km of any of the designated zones
- Discretisation 2:
  - track is within one of the designated zones
  - track is within 5 km of one of the designated zones

  - track is within 10 km of one of the designated zones
  - track is outwith 10 km of any of the designated zones
- Discretisation 3:
  - track is within one of the designated zones
  - track is within 3 km of one of the designated zones
  - track is within 6 km of one of the designated zones
  - track is within 10 km of one of the designated zones
  - track is outwith 10 km of any of the designated zones

For the decisions that involve a track's abnormalities, the following three different discretisation configurations were used:

- Discretisation 1:
  - track measured speed is within or not within $\pm 15\,\mathrm{m\,s^{-1}}$ of the transponder value
  - track measured size is within or not within $\pm 100\,\mathrm{m^2}$ of the transponder value
- Discretisation 2:
  - track measured speed is within $\pm 7.5\,\mathrm{m\,s^{-1}}$ of the transponder value, track measured speed is not within $\pm 7.5\,\mathrm{m\,s^{-1}}$ but is within $\pm 15\,\mathrm{m\,s^{-1}}$ of the transponder value, and track measured speed is not within $\pm 15\,\mathrm{m\,s^{-1}}$ of the transponder value
  - track measured size is within $\pm 50\,\mathrm{m^2}$ of the transponder value, track measured size is not within $\pm 50\,\mathrm{m^2}$ but is within $\pm 100\,\mathrm{m^2}$ of the transponder value, and track measured size is not within $\pm 100\,\mathrm{m^2}$ of the transponder value
- Discretisation 3:
  - track measured speed is within $\pm 5\,\mathrm{m\,s^{-1}}$ of the transponder value, track measured speed is not within $\pm 5\,\mathrm{m\,s^{-1}}$ but is within $\pm 10\,\mathrm{m\,s^{-1}}$ of the transponder value, track measured speed is not within $\pm 10\,\mathrm{m\,s^{-1}}$ but is within $\pm 15\,\mathrm{m\,s^{-1}}$ of the transponder value, and track measured speed is not within $\pm 15\,\mathrm{m\,s^{-1}}$ of the transponder value
  - track measured size is within $\pm 30\,\mathrm{m^2}$ of the transponder value, track measured size is not within $\pm 30\,\mathrm{m^2}$ but is within $\pm 60\,\mathrm{m^2}$ of the transponder value, track measured size is not within $\pm 60\,\mathrm{m^2}$ but is within $\pm 100\,\mathrm{m^2}$ of the transponder value, and track measured size is not within $\pm 100\,\mathrm{m^2}$ of the transponder value

For the decisions that involve a track's heading towards the nearest designated zones, the following three different discretisation configurations were used:

- Discretisation 1:

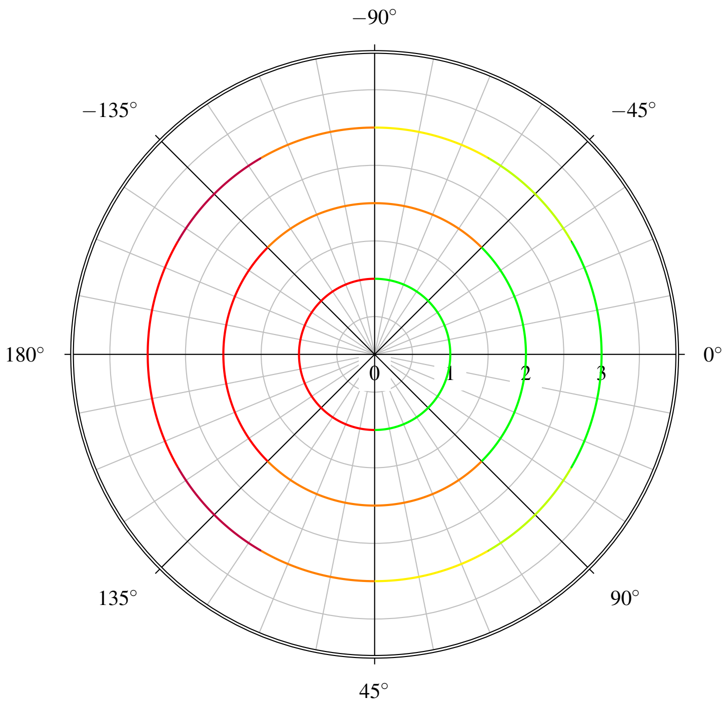


Figure 7.7: A polar plot showing the discretisation of the track's heading relative to its nearest zone. The range represents which discretisation configuration is being used, and the colour represents which discretisation band the heading is in.

  - the track's heading is within ±90° in the direction the nearest designated zone (as indicated with green in Fig. 7.7)
  - the track's heading is outwith ±90° in the direction the nearest designated zone (as indicated with red in Fig. 7.7)

- Discretisation 2:
  - the track's heading is within ±45° in the direction of the nearest designated zone (as indicated with green in Fig. 7.7)
  - the track's heading is between 45° and 135° or between −45° and −135° in the direction of the nearest designated zone (as indicated with orange in Fig. 7.7)
  - the track's heading is outwith ±135° in the direction of the nearest designated zone (as indicated with red in Fig. 7.7)
- Discretisation 3:
  - the track's heading is within ±30° in the direction of the nearest designated zone (as indicated with green in Fig. 7.7)
  - the track's heading is between 30° and 60° or between −30° and −60° in the direction of the nearest designated zone (as indicated with light green in Fig. 7.7)
  - the track's heading is between 60° and 90° or between −60° and −90° in the direction of the nearest designated zone (as indicated with yellow in Fig. 7.7)

- the track's heading is between $90^\circ$ and $120^\circ$ or between $-90^\circ$ and $-120^\circ$ in the direction of the nearest designated zone (as indicated with orange in Fig. 7.7)
- the track's heading is between $120^\circ$ and $150^\circ$ or between $-120^\circ$ and $-150^\circ$ in the direction of the nearest designated zone (as indicated with purple in Fig. 7.7)
- the track's heading is outwith $\pm 150^\circ$ in the direction of the nearest designated zone (as indicated with red in Fig. 7.7)

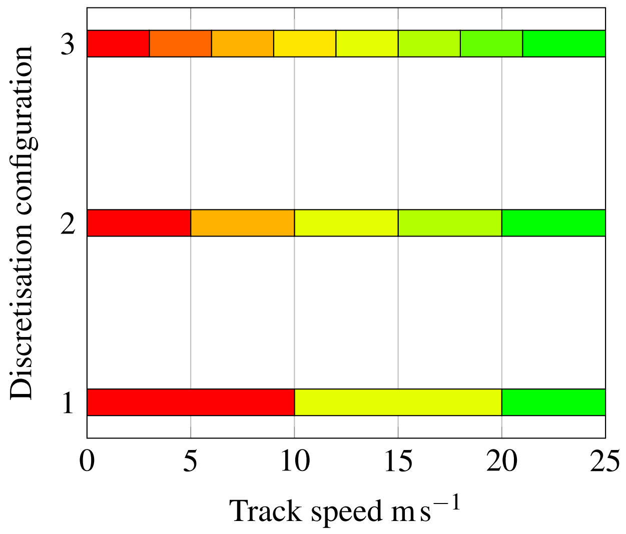


Figure 7.8: A chart showing the three discretisation configurations for the target's speed.

For the decisions that involve a track's speed, the following three different discretisation configurations were used:

- Discretisation 1: the track's speed falls within one of the following ranges [0-10, 10-20, $\geq$ 20]$\mathrm{m\,s^{-1}}$ (as indicated with the red to green scale in Fig. 7.8)
- Discretisation 2: the track's speed falls within one of the following ranges [0-5, 5-10, 10-15, 15-20, $\geq$ 20]$\mathrm{m\,s^{-1}}$ (as indicated with the red to green scale in Fig. 7.8)
- Discretisation 3: the track's speed falls within one of the following ranges [0-3, 3-6, 6-9, 9-12, 12-15, 15-18, 18-20, $\geq$ 20] $\mathrm{m\,s^{-1}}$ (as indicated with the red to green scale in Fig. 7.8)

For the decision that involves the direction of movement towards a track based on its probability of performing an illegal operation, the change in probability was discretised into the following ranges [$\leq -0.05$, $-0.05$-0.05, $\geq 0.05$]. The three different discretisation configurations used were as follows:

- Discretisation 1: Only the previous movement direction and change in probability was used
- Discretisation 2: The previous two movement directions and changes in probability were used

- Discretisation 3: The previous three movement directions and changes in probability were used

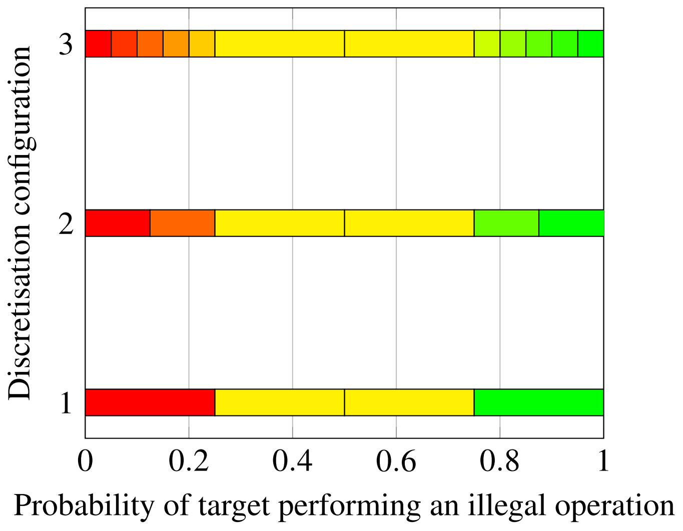


Figure 7.9: A chart showing the three discretisation configurations for the probability of a target performing an illegal operation.

For the decision that involves the action to take given a track's probability of performing an illegal operation, the following three different discretisation configurations were used:

- Discretisation 1: the track's probability is in one of the following ranges [0-0.25, 0.25-0.5, 0.5-0.75, 0.75-1.0] (as indicated with the red to green scale in Fig. 7.9)
- Discretisation 2: the track's probability is in one of the following ranges [0.125-0.25, 0.125-0.25, 0.25-0.5, 0.5-0.75, 0.75-0.875, 0.875-1.0] (as indicated with the red to green scale in Fig. 7.9)
- Discretisation 3: the track's probability is in one of the following ranges [0-0.05, 0.05-0.1, 0.1-0.15, 0.15-0.2, 0.2-0.25, 0.25-0.5, 0.5-0.75, 0.75-0.8, 0.8-0.85, 0.85-0.9, 0.9-0.95, 0.95-1.0] (as indicated with the red to green scale in Fig. 7.9)

### 7.3.1 Task 1 - Decision 1

For this decision, the discretisation with wider ranges (discretisation 1) is able to obtain a higher initial accuracy but immediately levels off with an accuracy below the desired 0.95. The discretisation with narrower ranges (discretisation 3) is has a accuracy lower than that of discretisation 1 and discretisation 2 until around 90 data instances are used, at which point this discretisation is able to achieve a high accuracy of 0.99. In short, the narrow discretisation ranges allow for high accuracy only when a large number of data instances are provided whereas wide discretisation ranges allow for the highest initial accuracy but this accuracy sees no improvement with more data instances. Given that there are few real-world missions in which operator data and scenario data for these mission can be obtained, it is important that the discretisation is adjusted after each new data set in order to maximise the imitation accuracy. Alternatively, or in addition, the use of a simulation such as the one

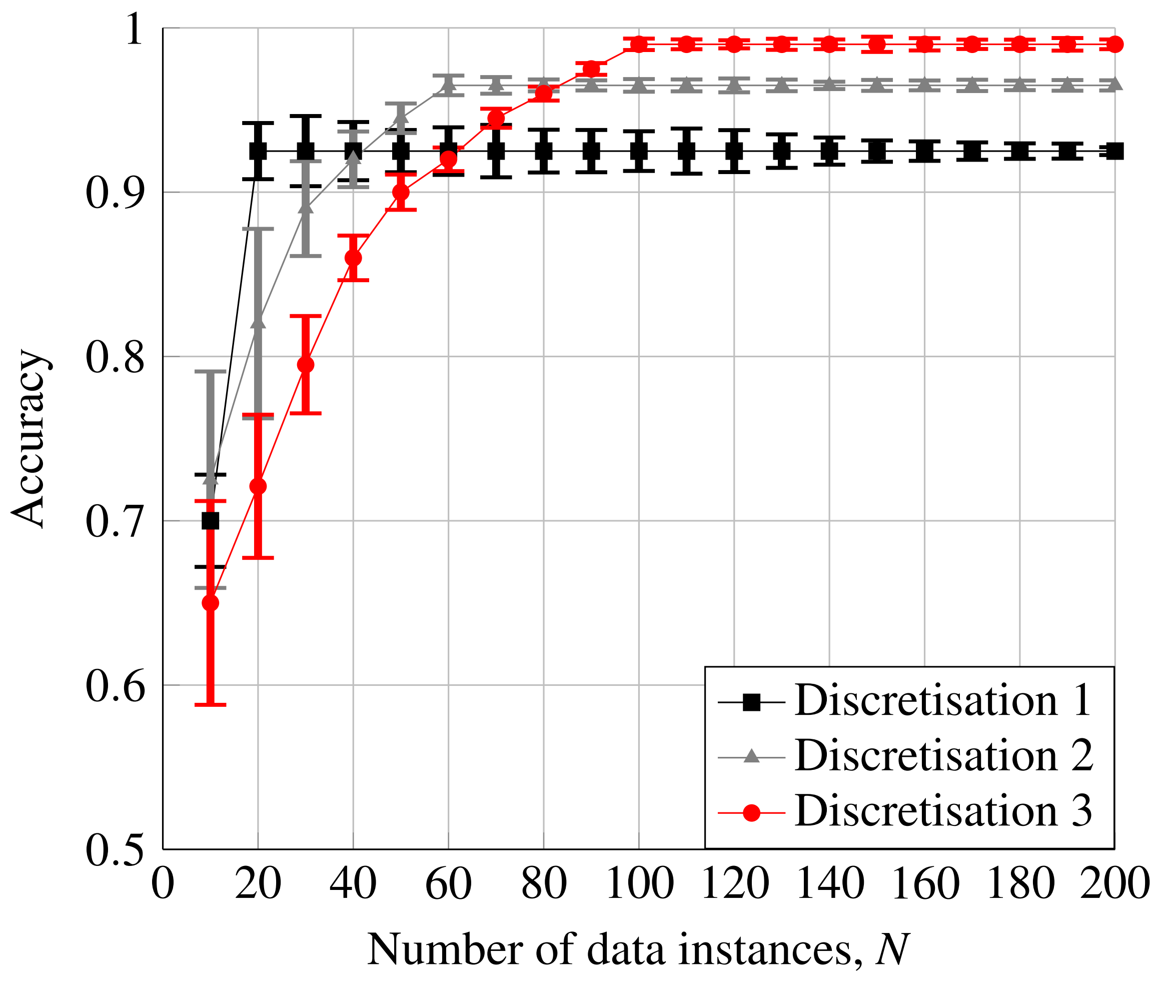


Figure 7.10: Accuracy and error in the Bayesian network approach of each discretisation for task 1 decision 1.

outlined in this thesis (see chapter 2) could be used to increase the imitation accuracy by obtaining operator decision data on realistic simulated scenario data.

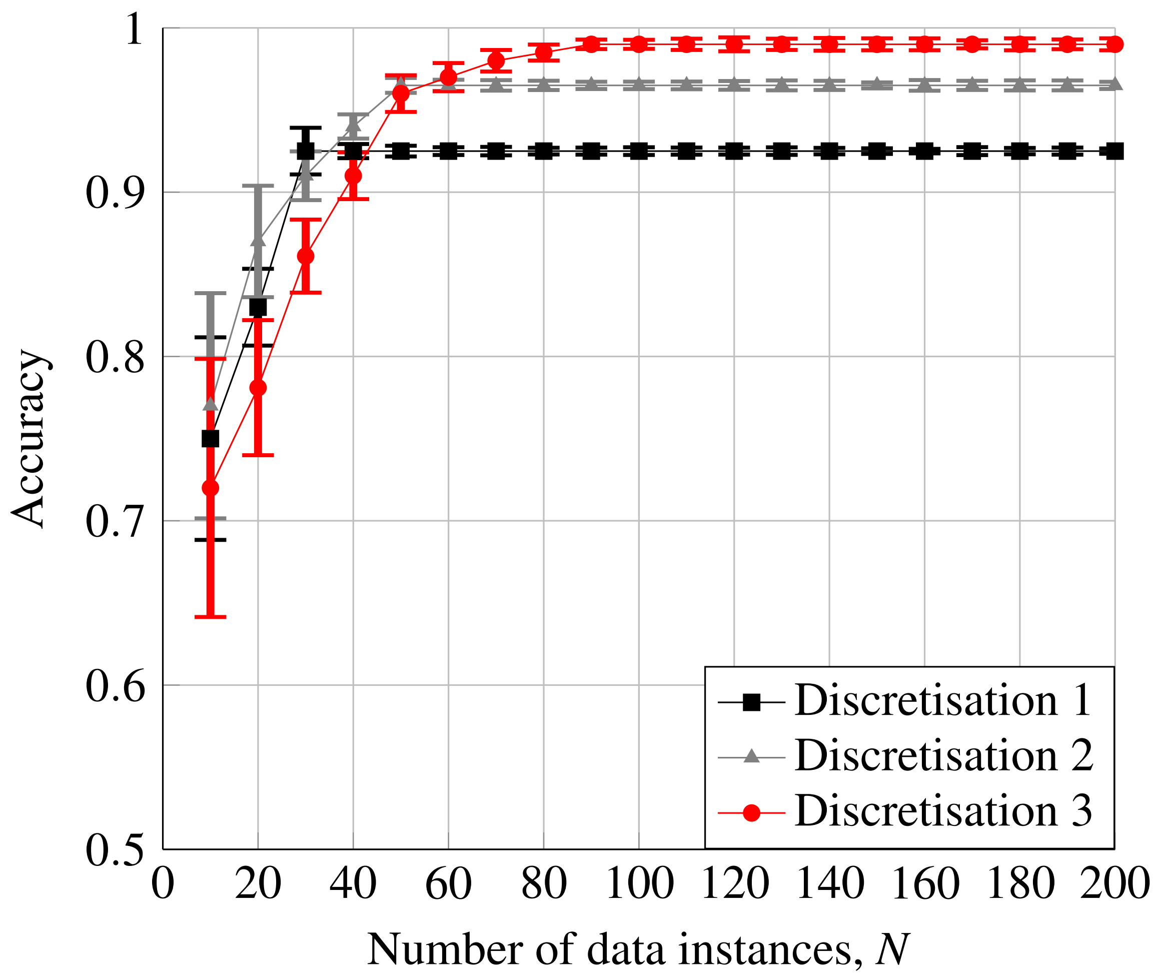


Figure 7.11: Accuracy and error in the IRL approach of each discretisation for task 1 decision 1.

### 7.3.2 Task 1 - Decision 3

For this decision, the discretisation with wider ranges (discretisation 1) is able to obtain a high accuracy of 0.955 with as few data instances as 40. Furthermore, it is only after 180 data instances that a discretisation with narrower ranges is able to achieve a higher accuracy. After 200 data instances, the discretisation with the narrowest ranges (discretisation 3) achieves the lowest accuracy of the three discretisation types with an accuracy of 0.84. With more data instances the accuracy for discretisation 3 might improve beyond the other two due to there being more resolution in the data, but the resolution might not be the limiting source of error.

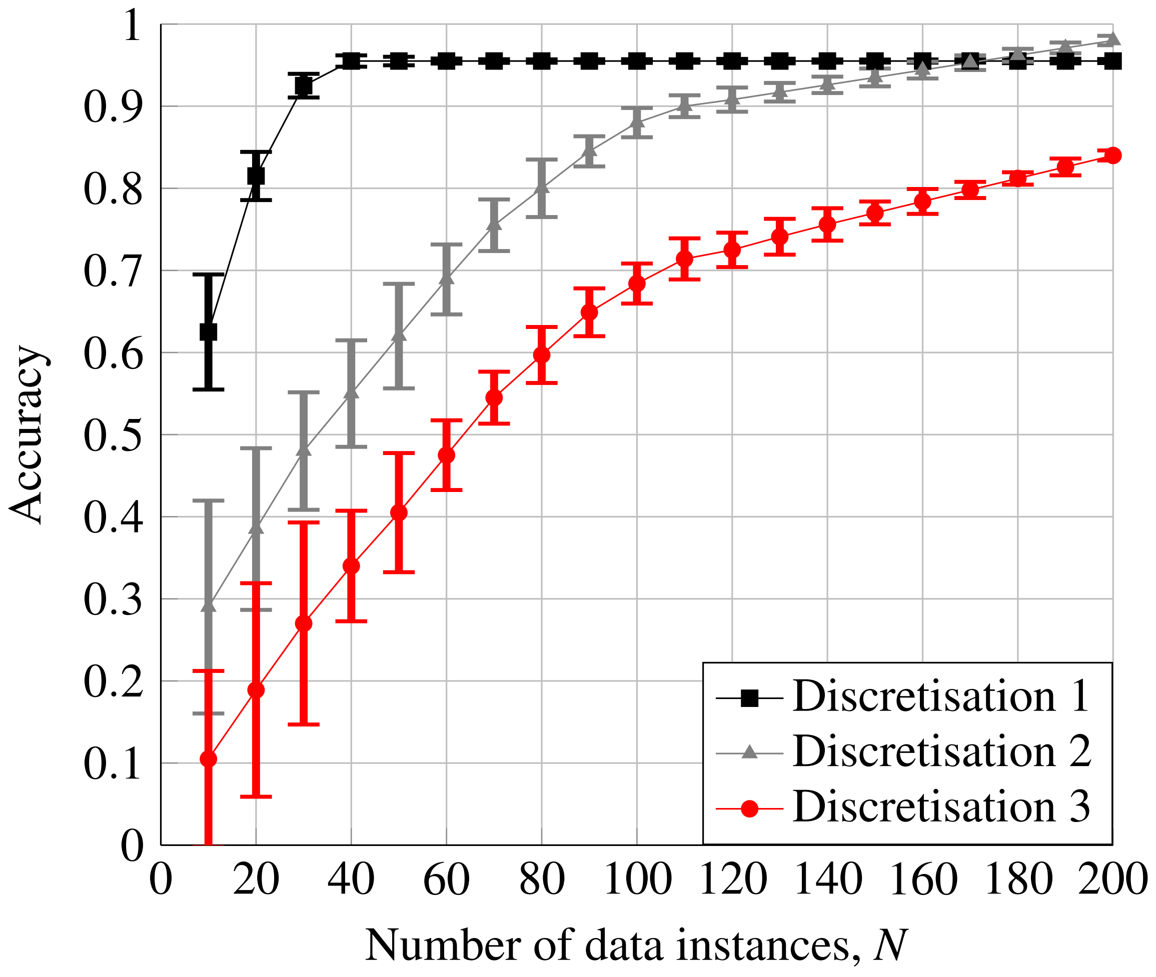


Figure 7.12: Accuracy and error in the Bayesian network approach of each discretisation for task 1 decision 3.

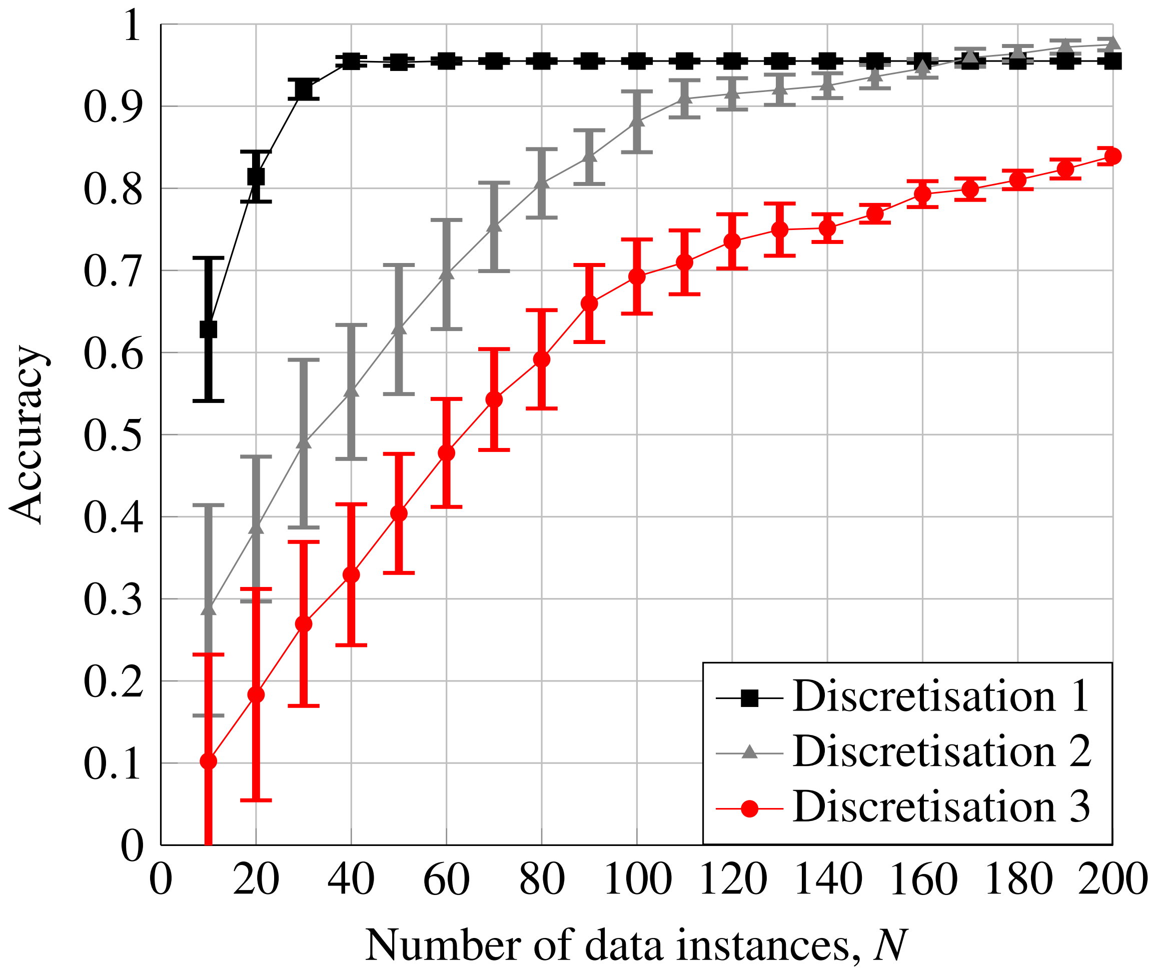


Figure 7.13: Accuracy and error in the IRL approach of each discretisation for task 1 decision 3.

### 7.3.3 Task 1 - Decision 4

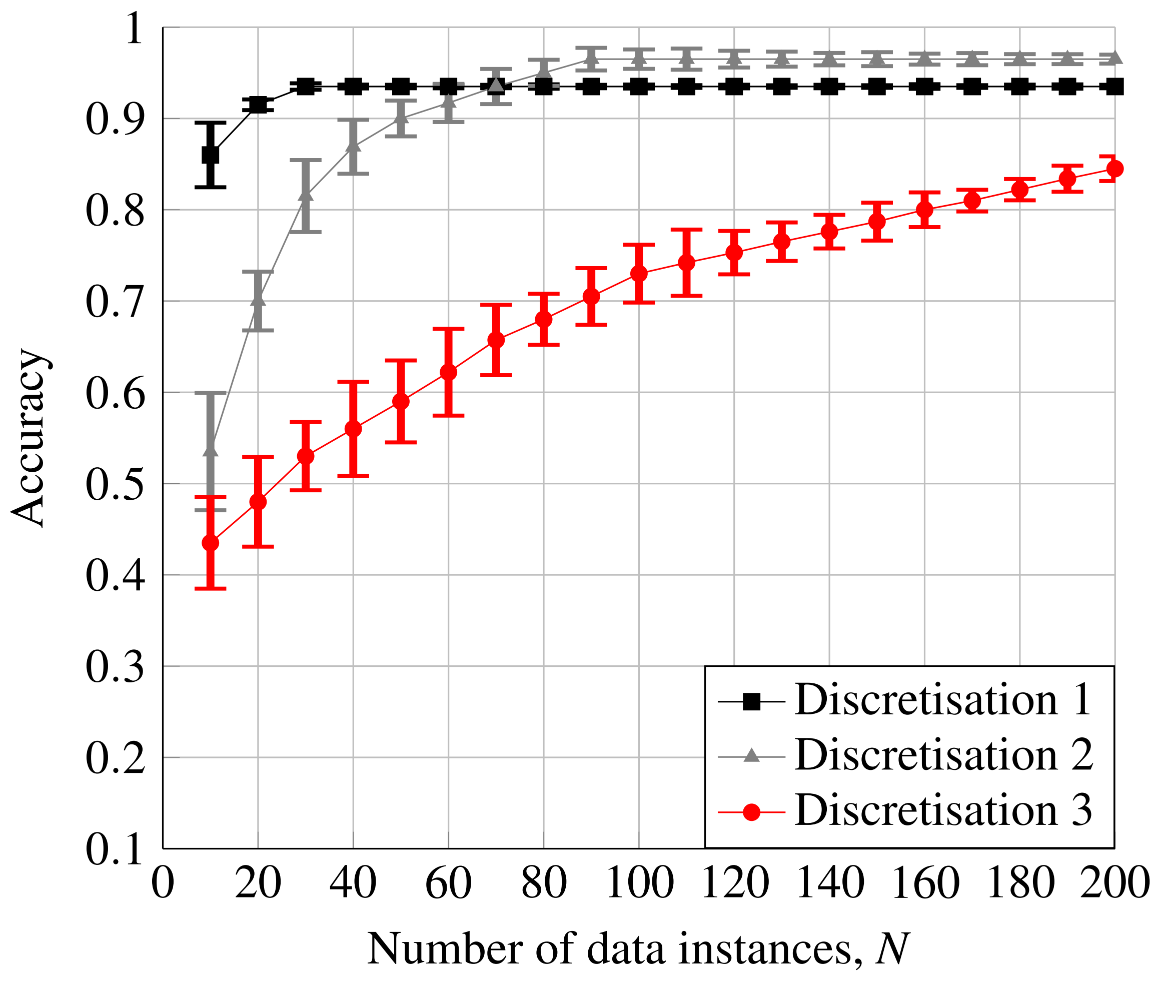


Figure 7.14: Accuracy and error in the Bayesian network approach of each discretisation for task 1 decision 4.

Similar to task 1 decision 3, the discretisation with the narrowest ranges (discretisation 3) has the lowest accuracy after 200 data instances, and the discretisation with the widest ranges has the highest initial accuracy. However, discretisation 2 is able to surpass discretisation 1 after 80 data instances in the case of the Bayesian network or 60 data instances in the case of IRL. Additionally, in both the Bayesian network and IRL, the accuracy of discretisation 1 never exceeds 0.935 which is below the required threshold.

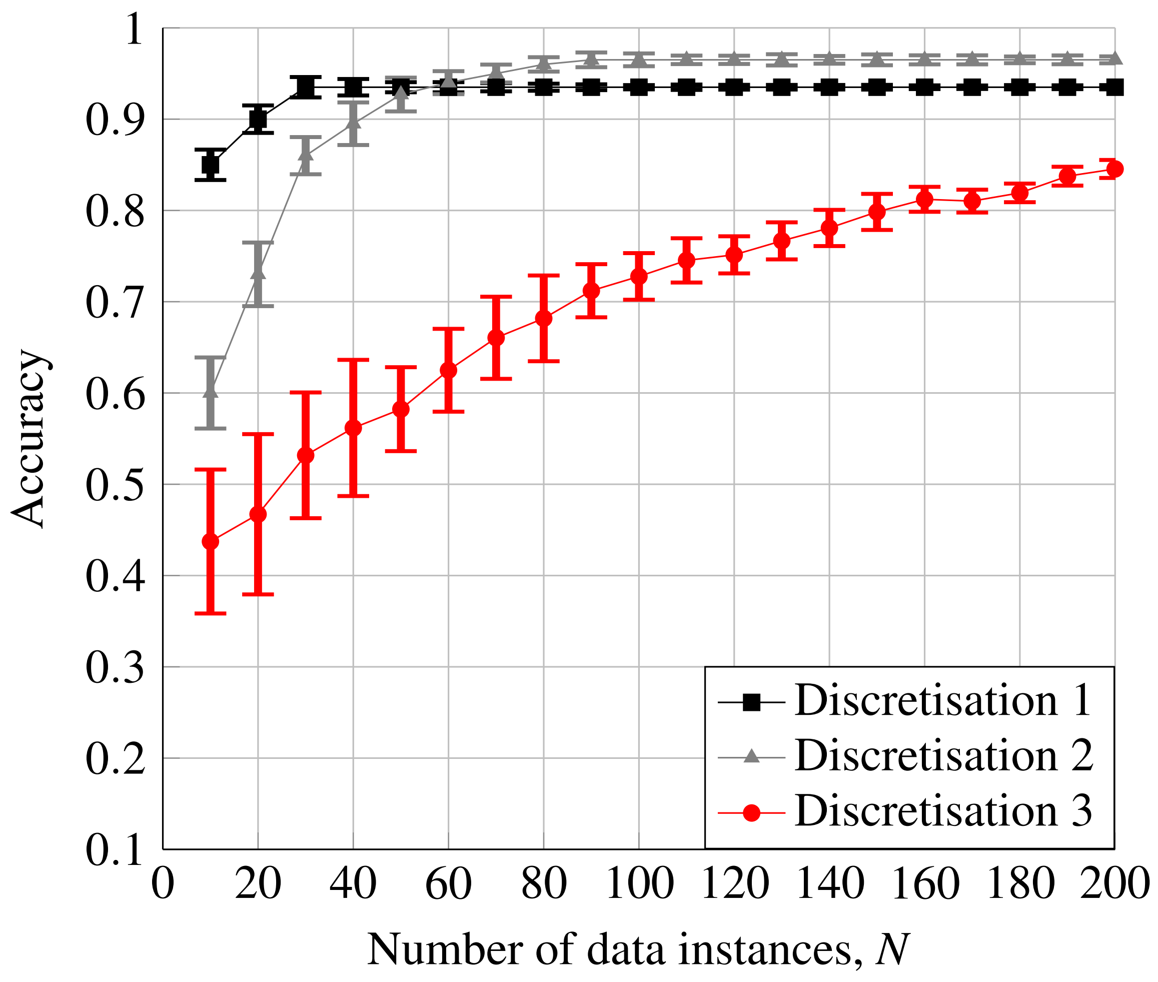


Figure 7.15: Accuracy and error in the IRL approach of each discretisation for task 1 decision 4.

### 7.3.4 Task 2 - Decision 1

For this decision, discretisation 1 levels off at an accuracy of 0.88 and discretisation 2 levels off at an accuracy of 0.935. Neither of these accuracies is greater than the required 0.95. After 190 data instances, discretisation 3 reaches an accuracy greater than that of discretisation 2 and surpasses the required accuracy of 0.95. At 200 data instances discretisation 3 achieves an accuracy of 0.98 in the Bayesian network case and an accuracy of 0.995 in the IRL case.

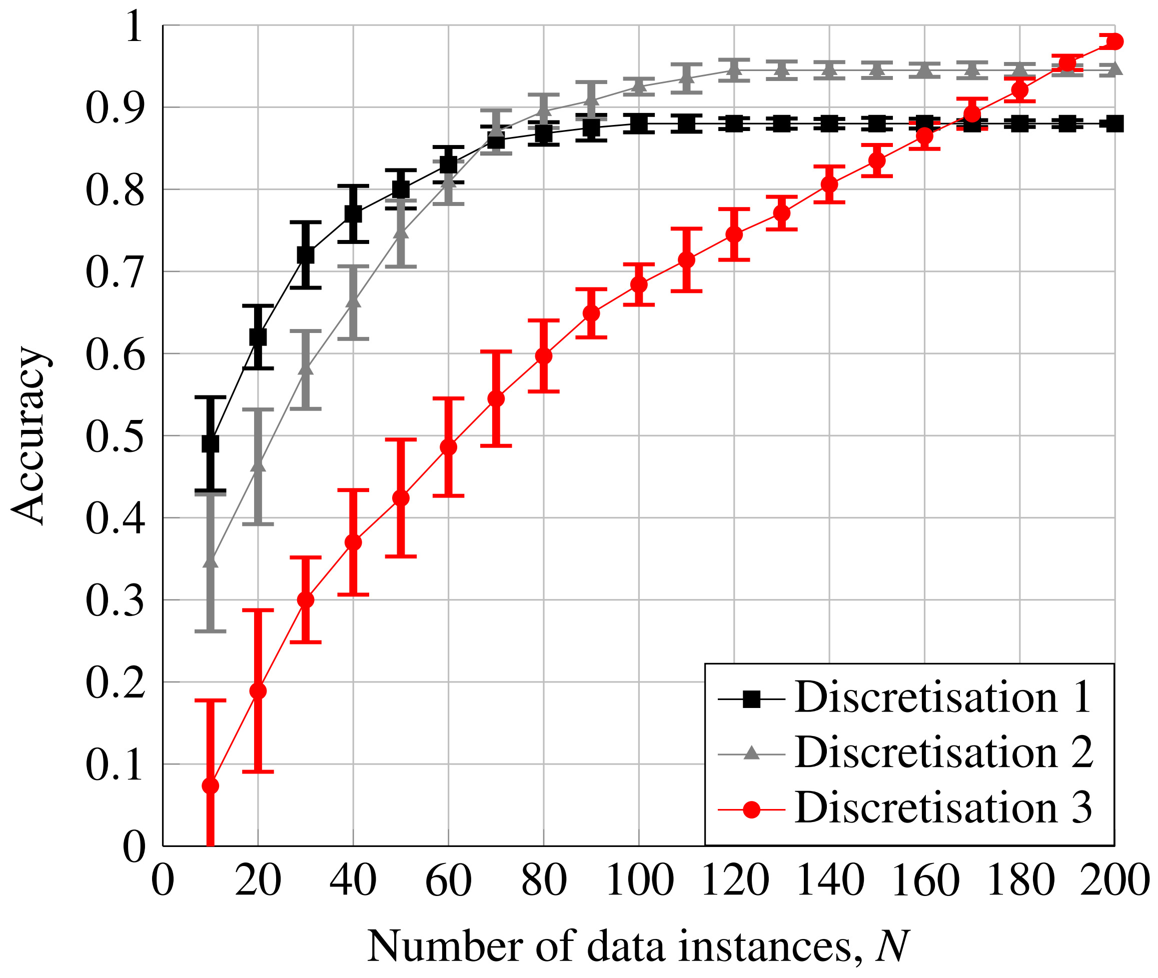


Figure 7.16: Accuracy and error in the Bayesian network approach of each discretisation for task 2 decision 1.

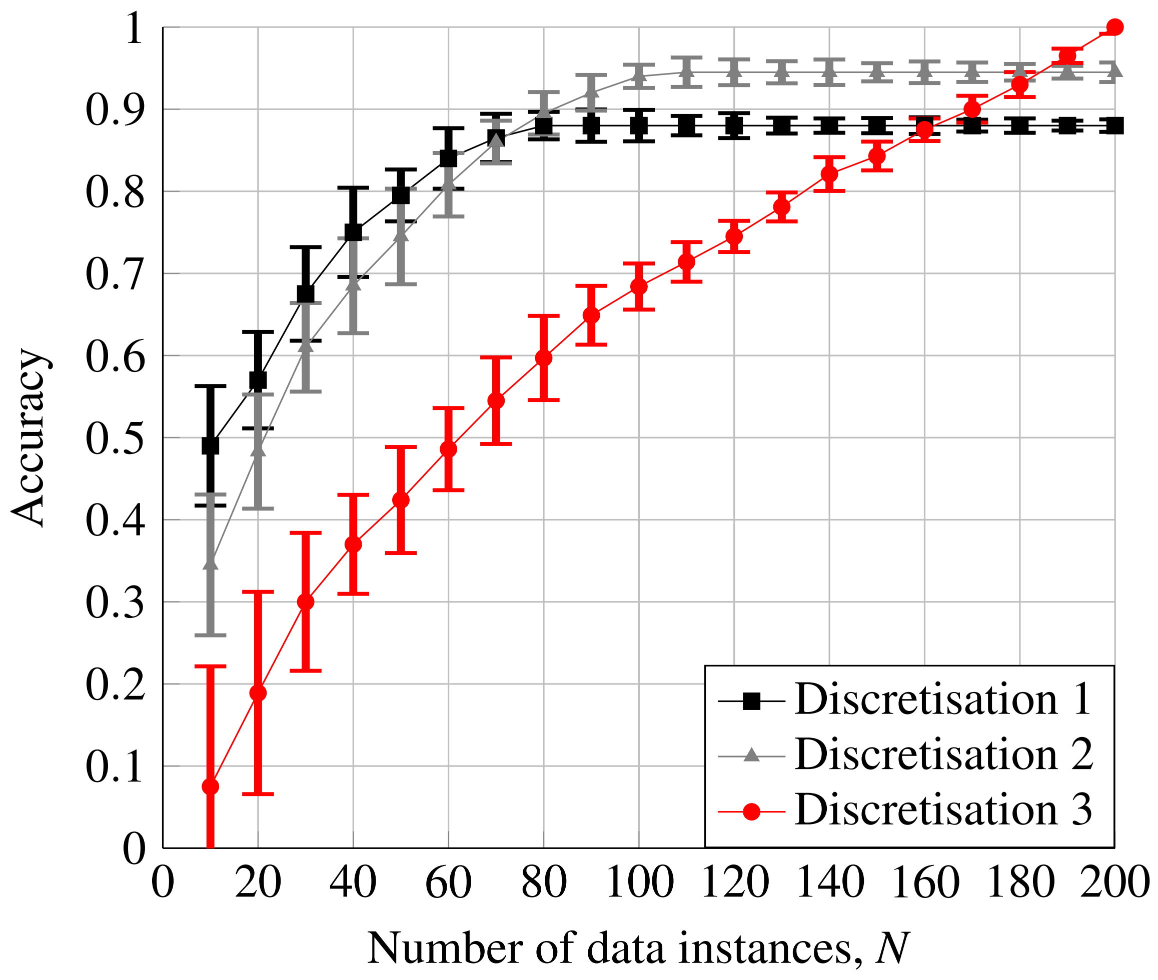


Figure 7.17: Accuracy and error in the IRL approach of each discretisation for task 2 decision 1.

### 7.3.5 Task 2 - Decision 2

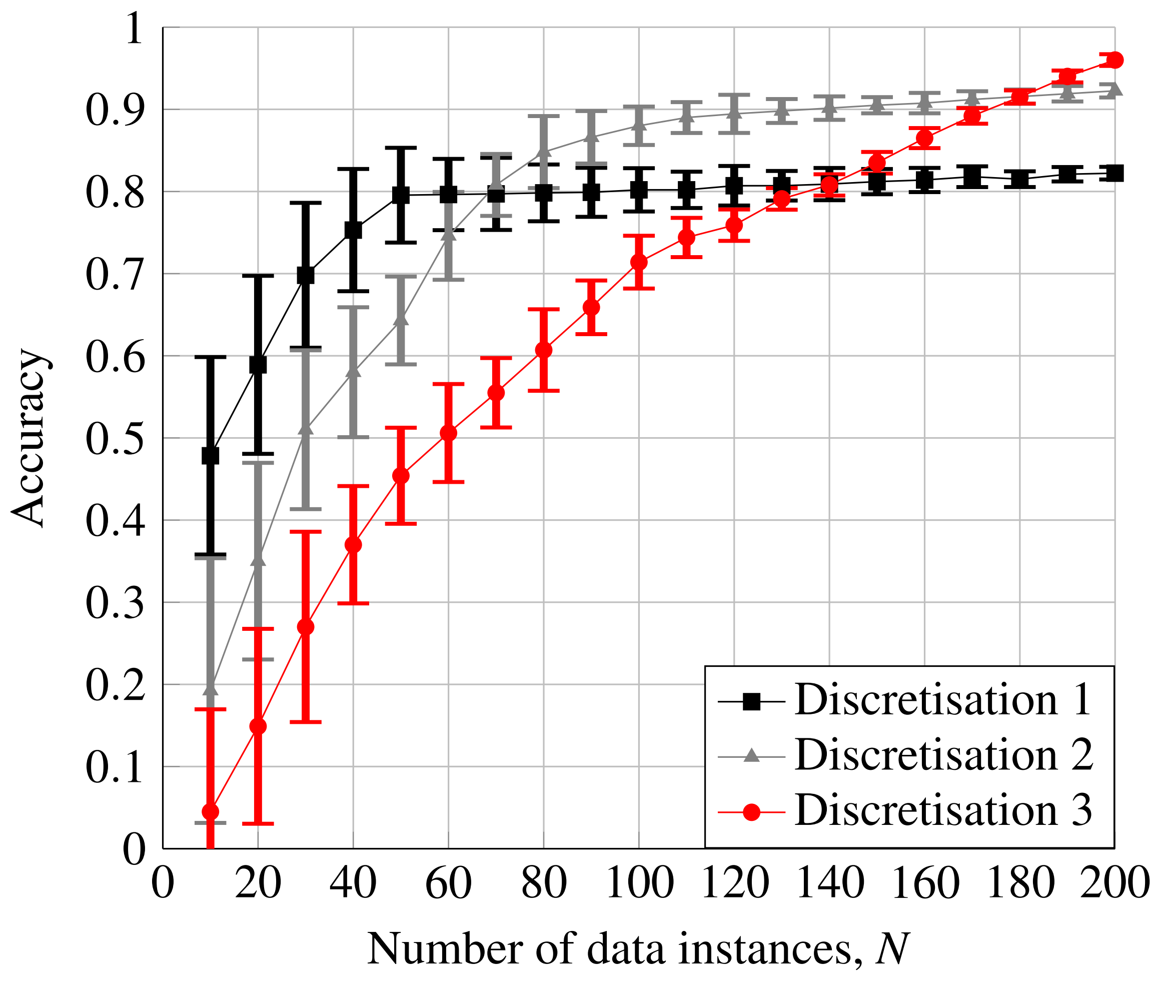


Figure 7.18: Accuracy and error in the Bayesian network approach of each discretisation for task 2 decision 2.

Similar to task 2 decision 1, only discretisation 3 is able to surpass the required accuracy with an accuracy of 0.995 and 0.96 achieved in the case of IRL and the Bayesian network respectively.

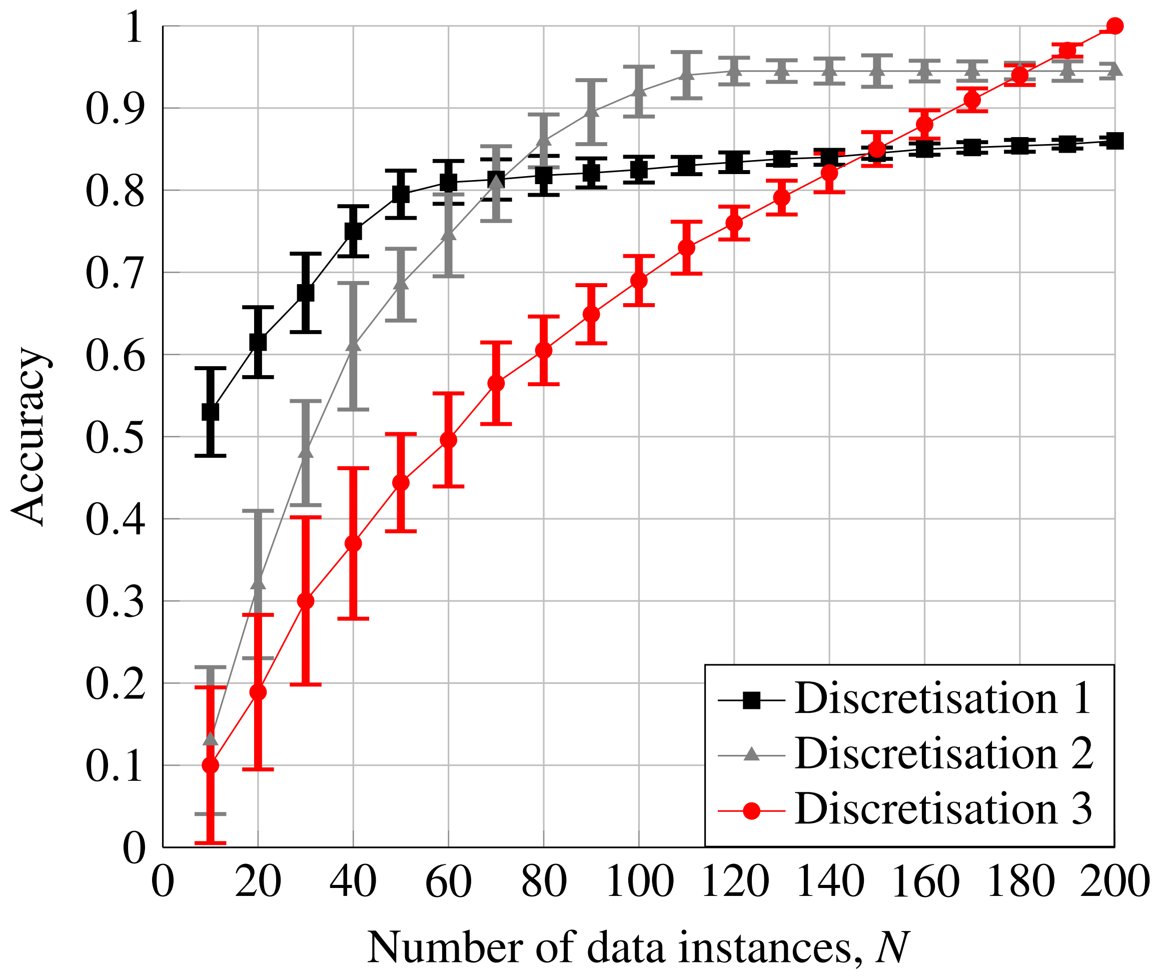


Figure 7.19: Accuracy and error in the IRL approach of each discretisation for task 2 decision 2.

### 7.3.6 Task 2 - Decision 3

In this task, both discretisation 2 and 3 are able to surpass the required accuracy of 0.95 in both the Bayesian network and IRL cases. However, discretisation 2 was able to achieve the required accuracy in 140 data instances in the Bayesian network case and 170 data instances in the IRL case. Additionally, even at 200 data instances, the accuracy in discretisation 3 is only equal to that of discretisation 2 (with an accuracy of 0.98 in the Bayesian network case and 0.995 in the IRL case).

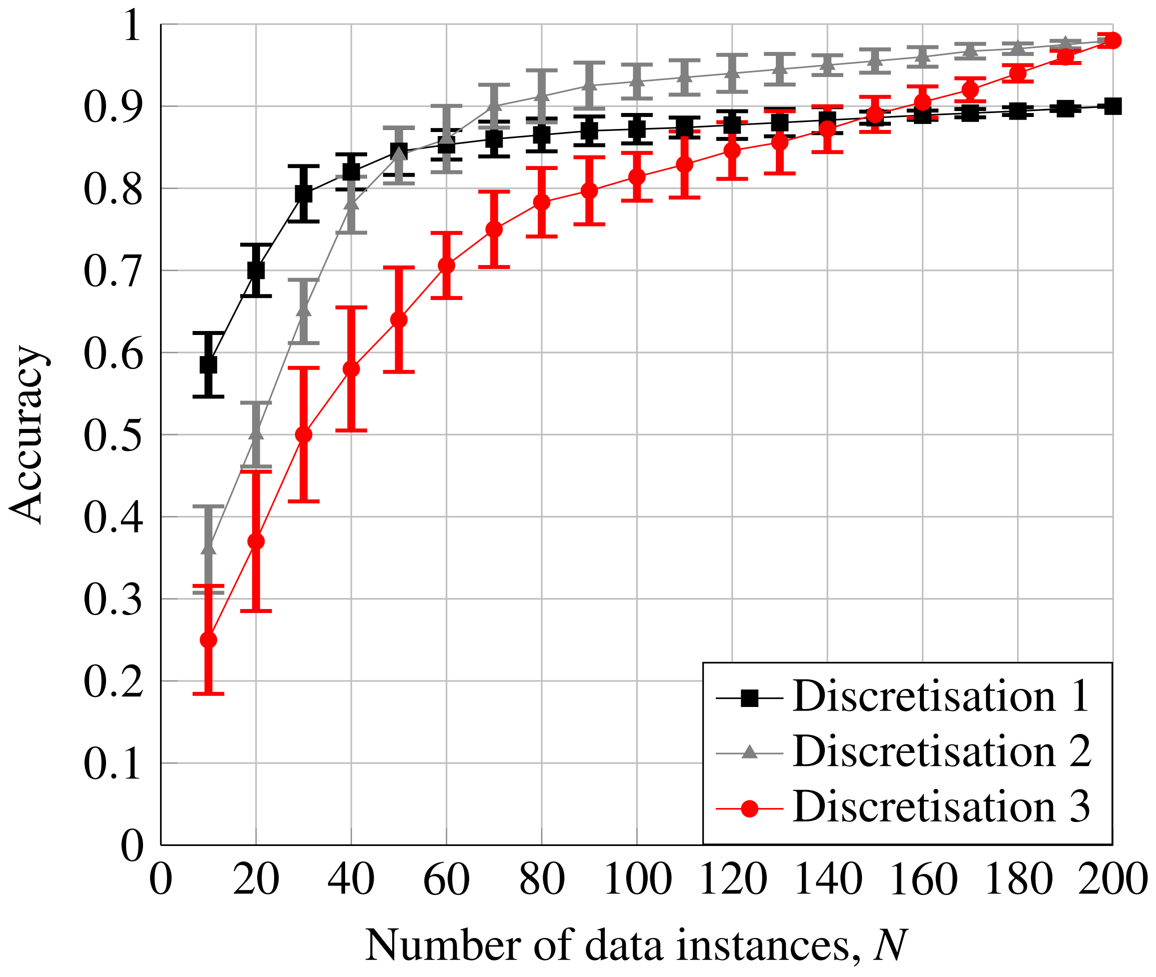


Figure 7.20: Accuracy and error in the Bayesian network approach of each discretisation for task 2 decision 3.

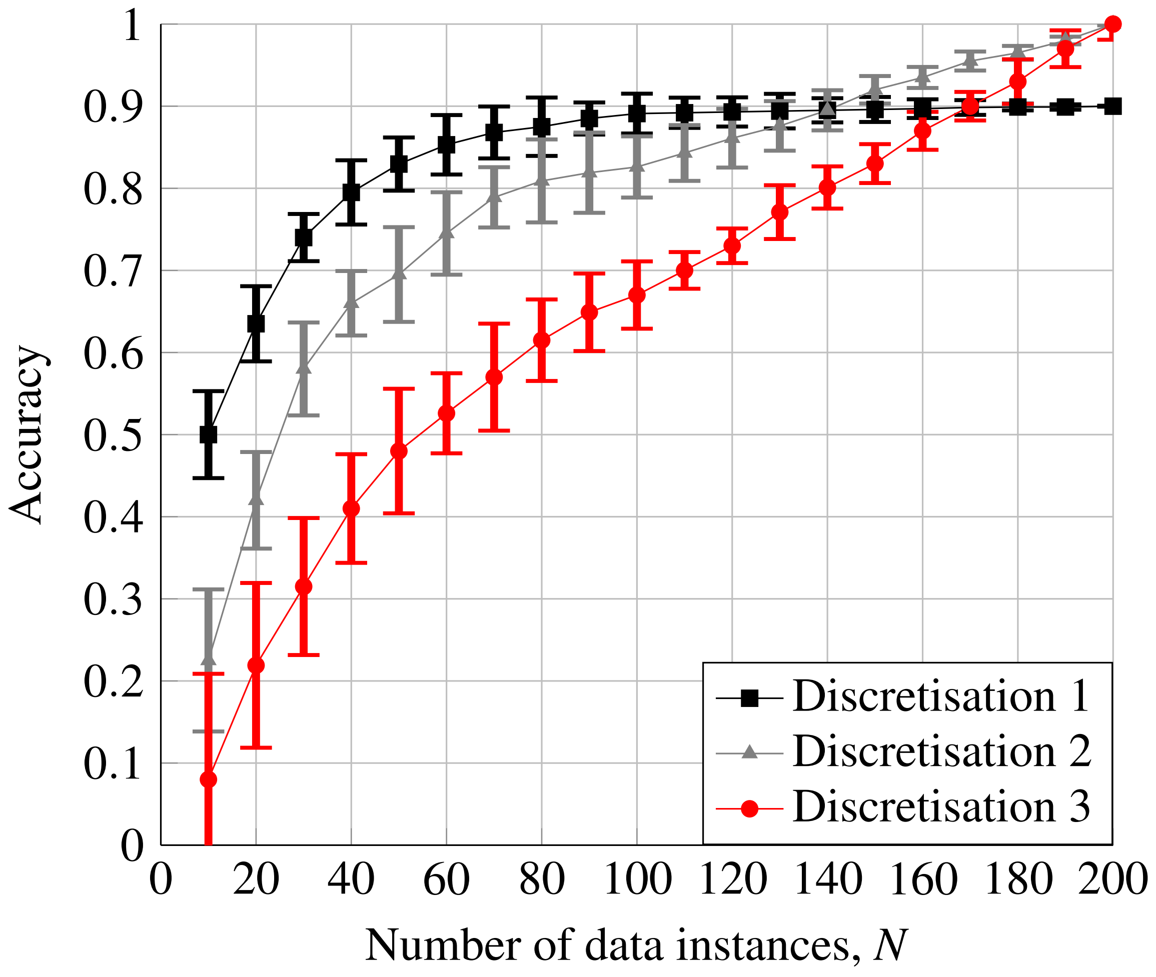


Figure 7.21: Accuracy and error in the IRL approach of each discretisation for task 2 decision 3.

### 7.3.7 Task 3 - Decision 1

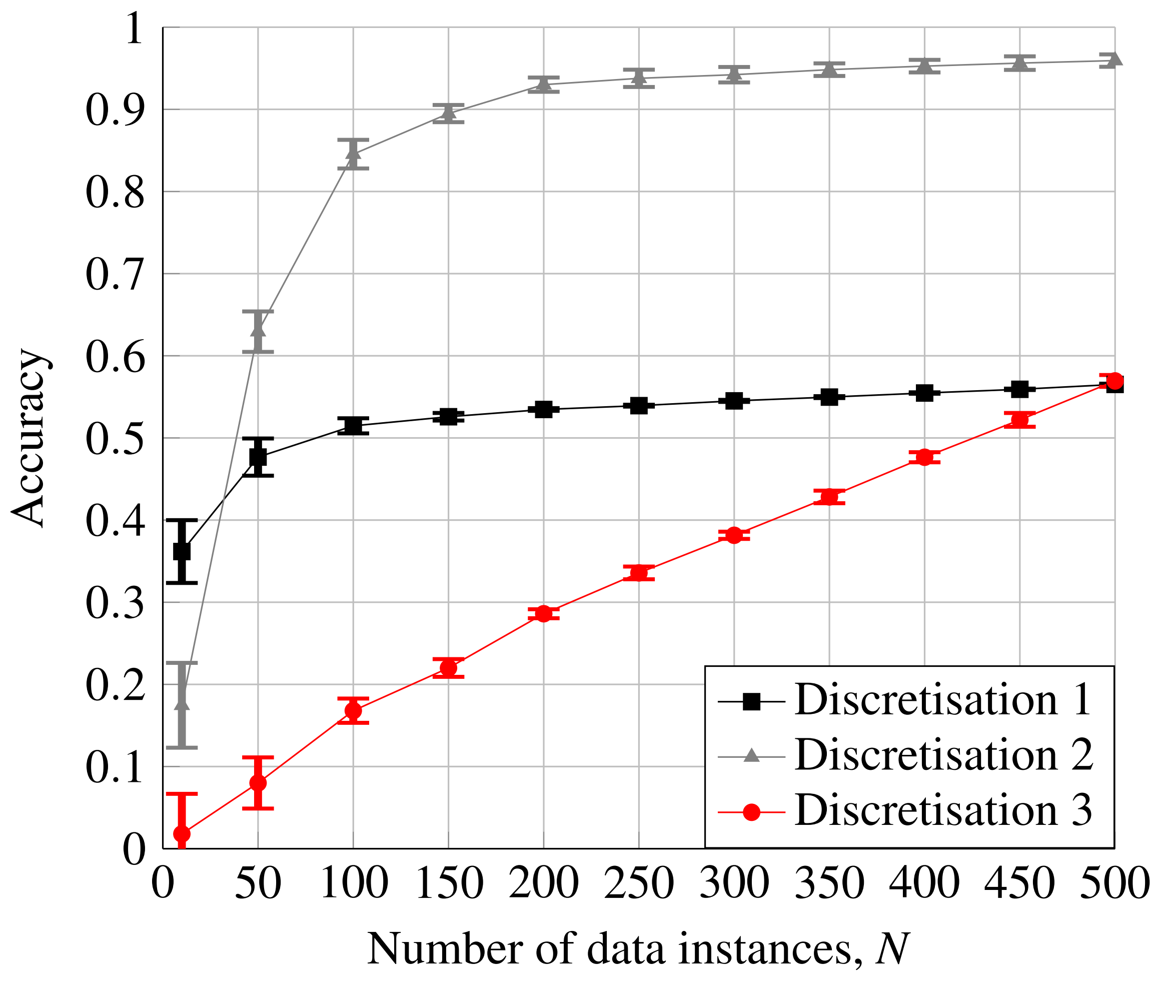


Figure 7.22: Accuracy and error in the Bayesian network approach of each discretisation for task 3 decision 1.

In discretisation 1, where only the previous movement direction and change in probability were used, the decision accuracy levelled off at an accuracy of 0.57 in both the IRL and Bayesian network case. In discretisation 3, by using the previous three movement directions and the corresponding changes in probability, an accuracy of only 0.5 is achieved in the Bayesian network case after 500 data instances. However, even after 500 data instances the accuracy is still increasing. With significantly more data, discretisation 3 might not even reach the same level of accuracy as discretisation 2 if the operator does not factor in what happened three decisions ago.

By using only the information from only the previous two decisions, the required accuracy of 0.95 was able to be achieved after 400 data instances in both the IRL case and the Bayesian network case.

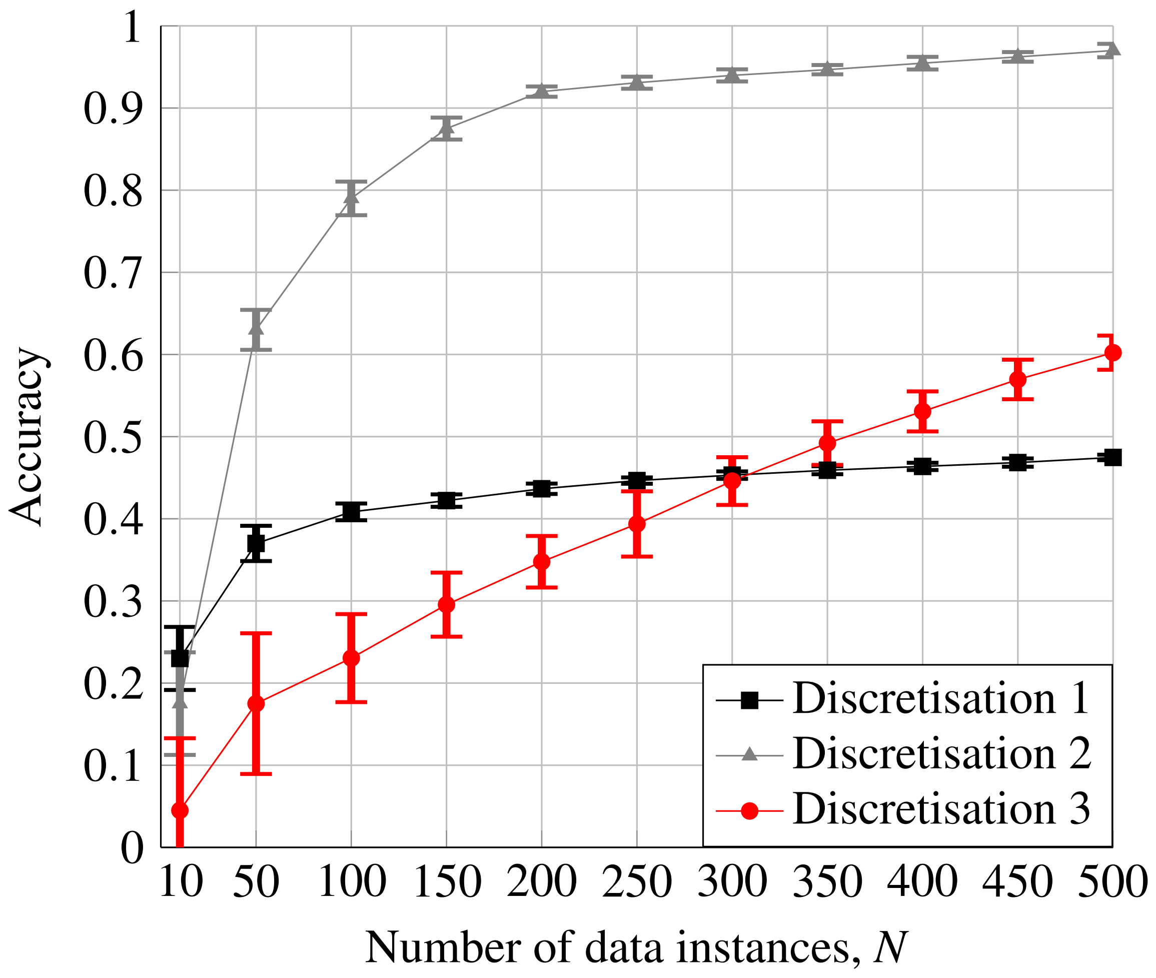


Figure 7.23: Accuracy and error in the IRL approach of each discretisation for task 3 decision 1.

### 7.3.8 Task 3 - Decision 2

In this decision, all three discretisation configurations levelled off at 60 data instances or less. The operator is only likely to make the decision to declare the target as performing an illegal operation when they are fully confident, and consequently this decision is highly deterministic resulting in each of the discretisation configurations levelling off with few data instances.

Furthermore, due to the operator only taking actions when they are completely confident in the decision then this requires narrow discretisation ranges that only discretisation 3 provides. Consequently, discretisation 3 is able to achieve an accuracy of 1.0 in both the IRL and Bayesian network case, whereas discretisation 1 and 2 do not achieve the required accuracy of 0.95.

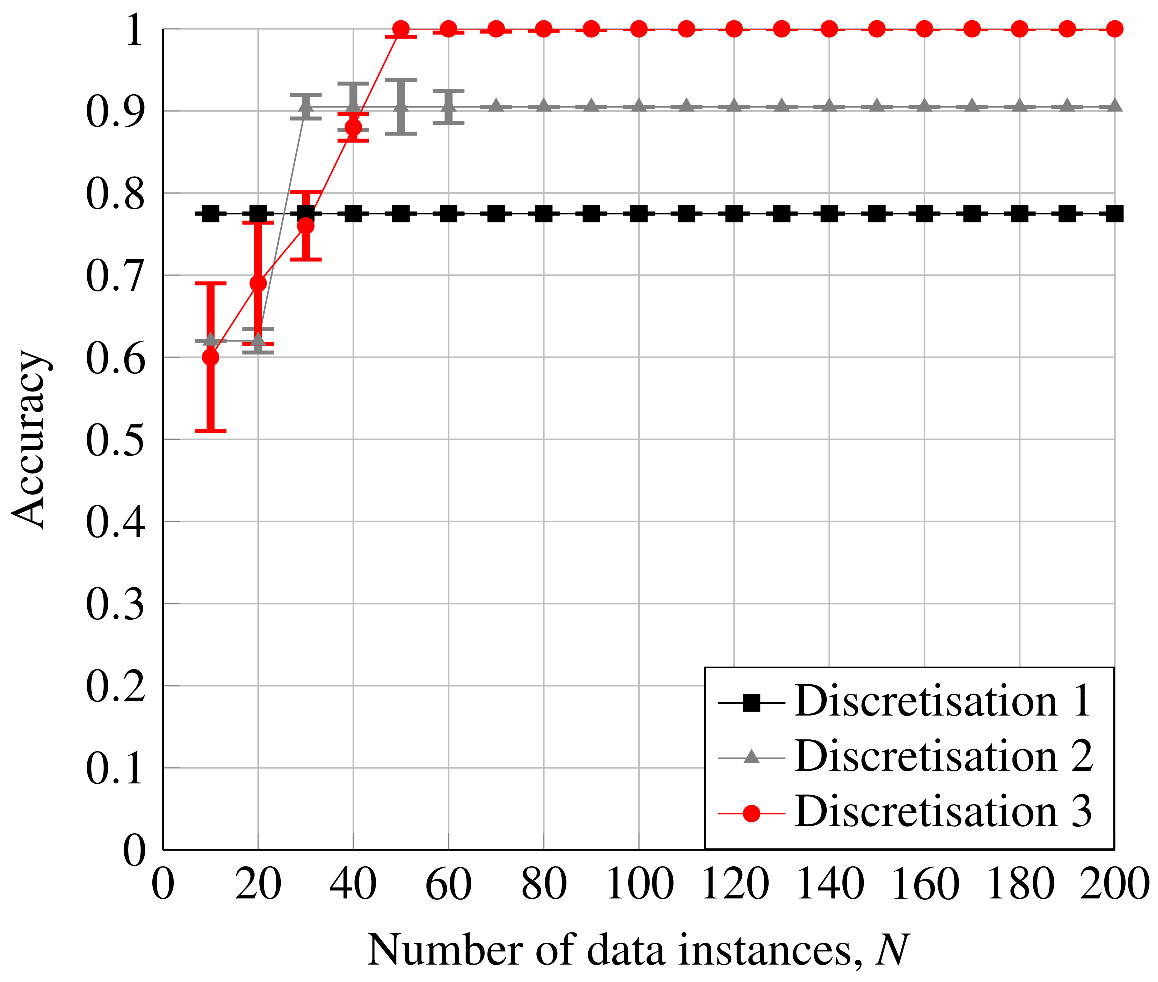


Figure 7.24: Accuracy and error in the Bayesian network approach of each discretisation for task 3 decision 2.

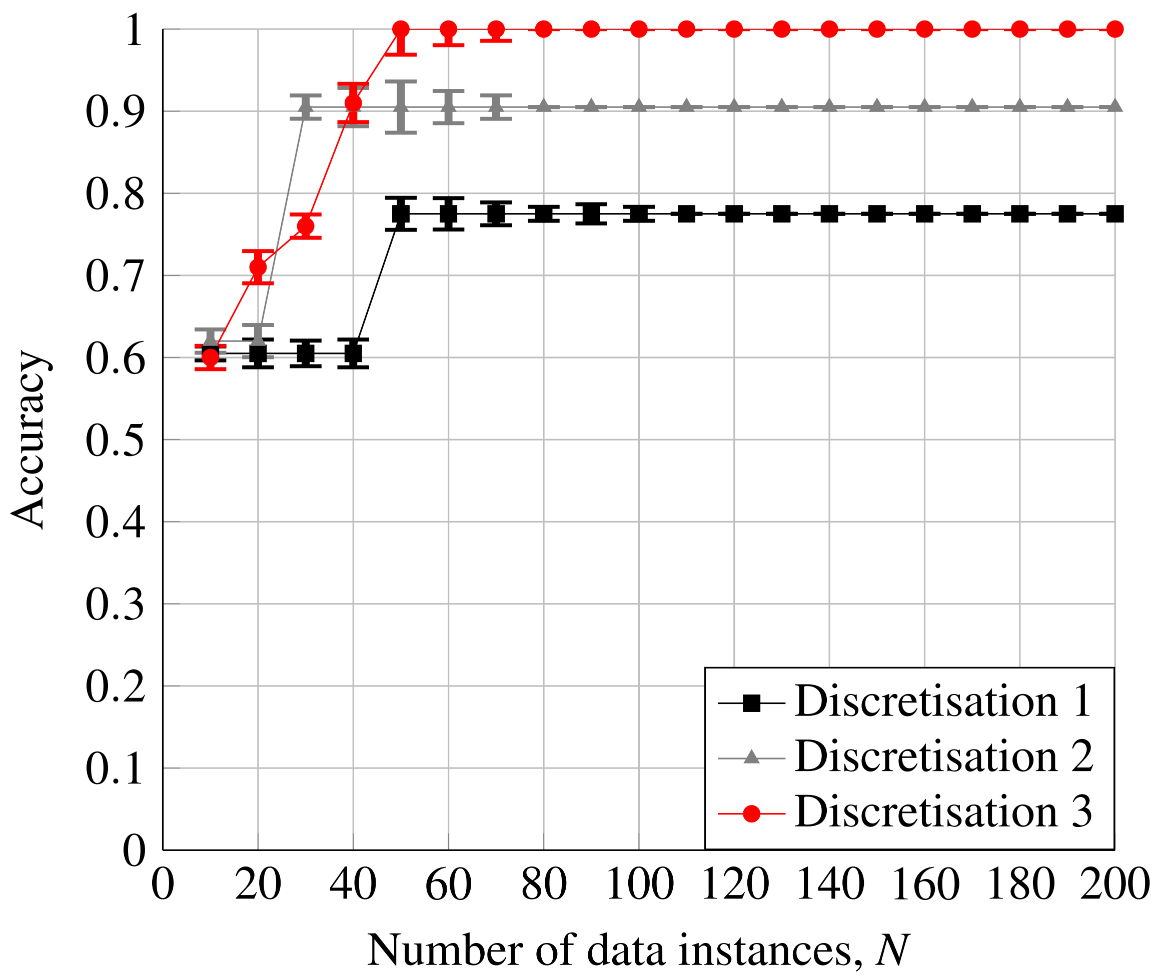


Figure 7.25: Accuracy and error in the IRL approach of each discretisation for task 3 decision 2.

### 7.3.9 Task 3 - Learned Trajectories

A sample of the trajectories learned by AI agent is shown in Fig. 7.26. Note that the target is shown as a black star, the start point of the trajectory is shown by the green circle, and the point at which the agent performs decision 2 is shown as a red circle. The black dots represent the positions along the trajectory. Lastly, the trajectories have been normalised such the target and radar platform start in the same position for each run.

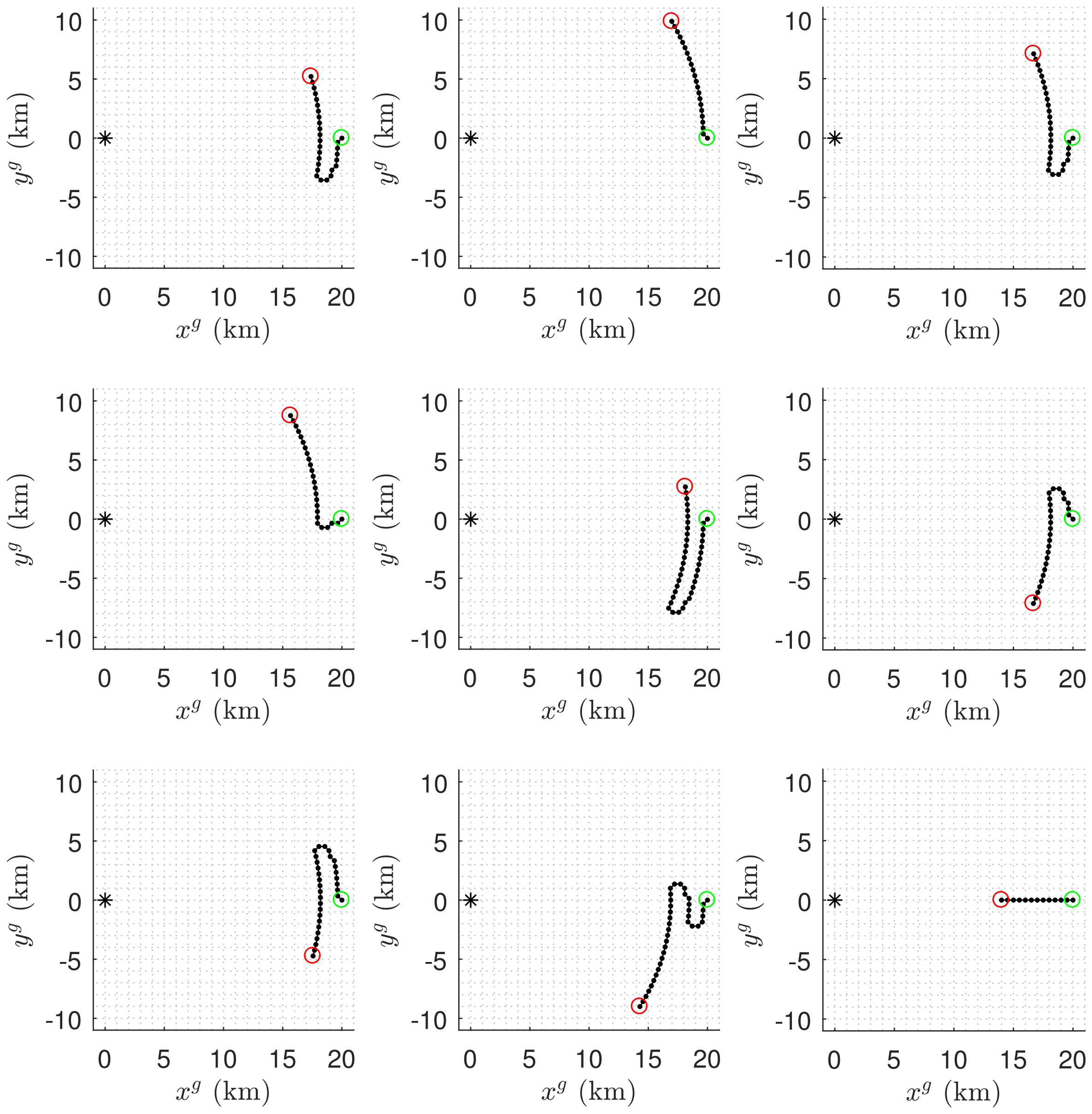


Figure 7.26: A sample of the trajectories performed by the AI agent (specifically, the Bayesian network approach) in task 3.

## 7.4 Comparison

In this section, the differences in accuracy between the Bayesian network approach and the IRL approach are shown for each discretisation configuration. A negative difference occurs when the mean accuracy of the IRL approach is greater than the Bayesian network approach and vice versa.

### 7.4.1 Task 1

In decision 1, the IRL approach performs slightly better than the Bayesian Network approach. At the number of data instances in which both approaches were greater than 0.95, the accuracy was the same for discretisation 1 and 2. For discretisation 3, the IRL approach is able to achieve an accuracy of 0.96 after 50 data instances whereas the Bayesian network approach achieved the same accuracy after 80 data instances.

Due to decision 2 being highly deterministic, both approaches were able to achieve the same accuracy with the same data instances.

In decision 3, the difference between the two approaches was negligible for all three discretisation configurations.

In decision 4, the IRL approach achieved a slightly higher accuracy for discretisation 2 at a lower number of data instances. However, at the number of data instances at which IRL outperforms the Bayesian network, the IRL accuracy is below the required accuracy of 0.95 except at 70 data instances. Furthermore, at 70 data instances, the difference in accuracy between the two approaches falls well within the standard error of both approaches.

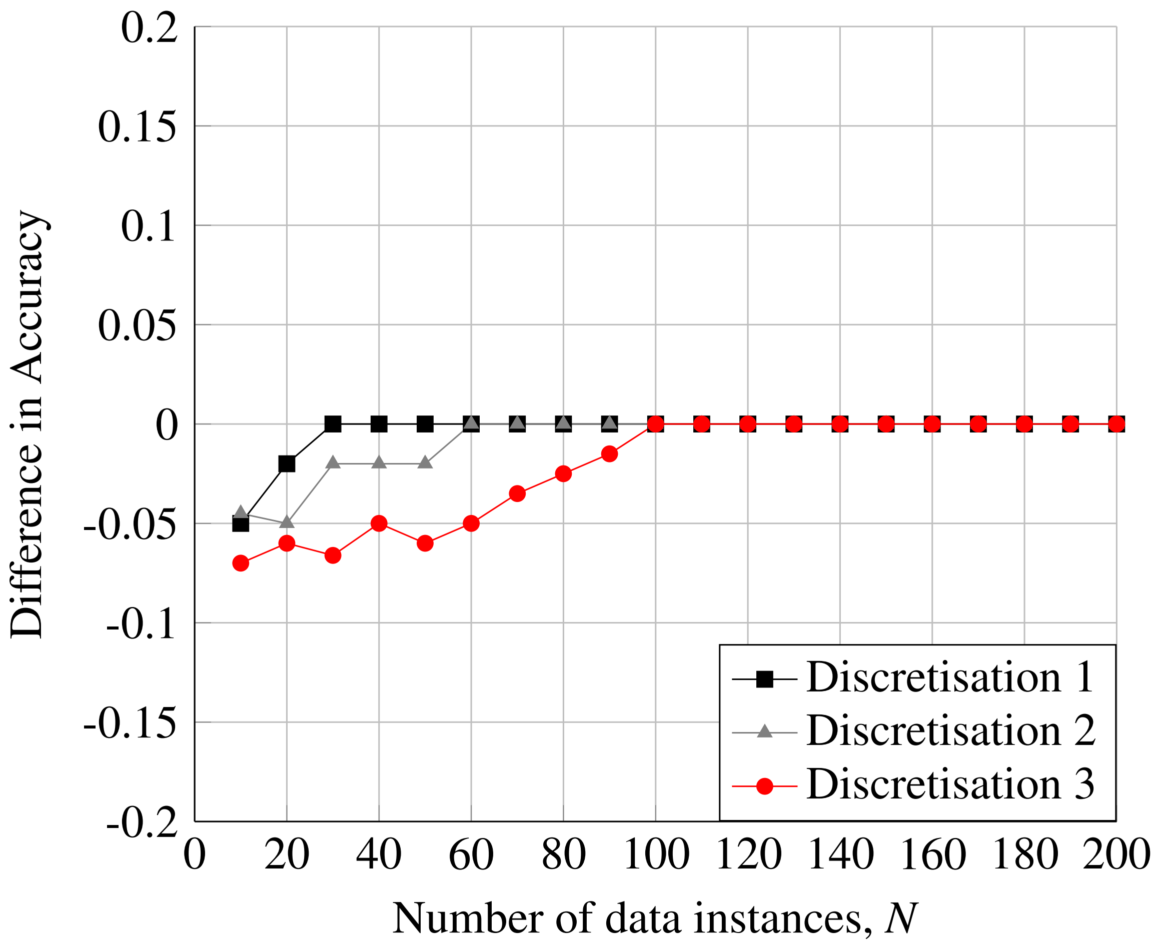


Figure 7.27: Difference in accuracy between the Bayesian network and IRL approaches for each discretisation configuration in task 1 decision 1.

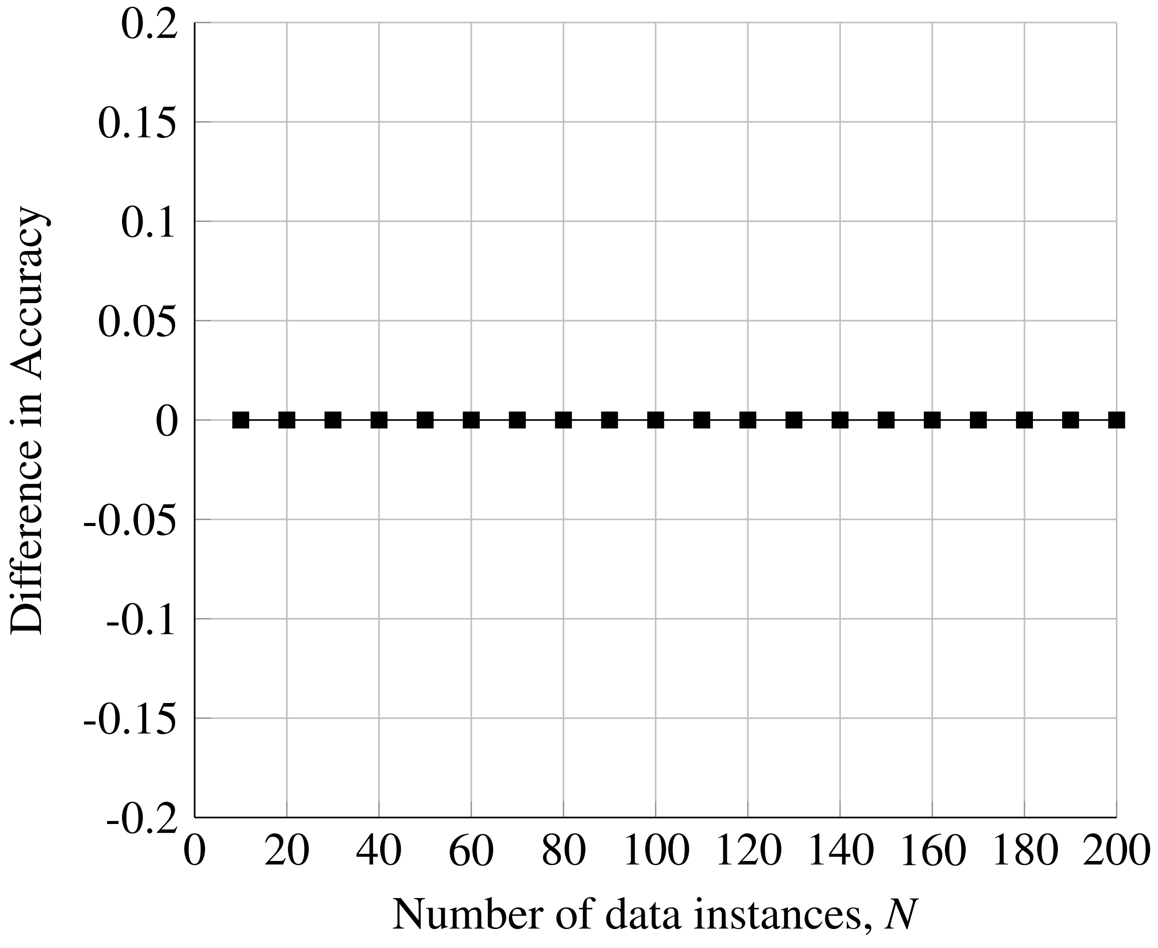


Figure 7.28: Difference in accuracy between the Bayesian network and IRL approaches in task 1 decision 2.

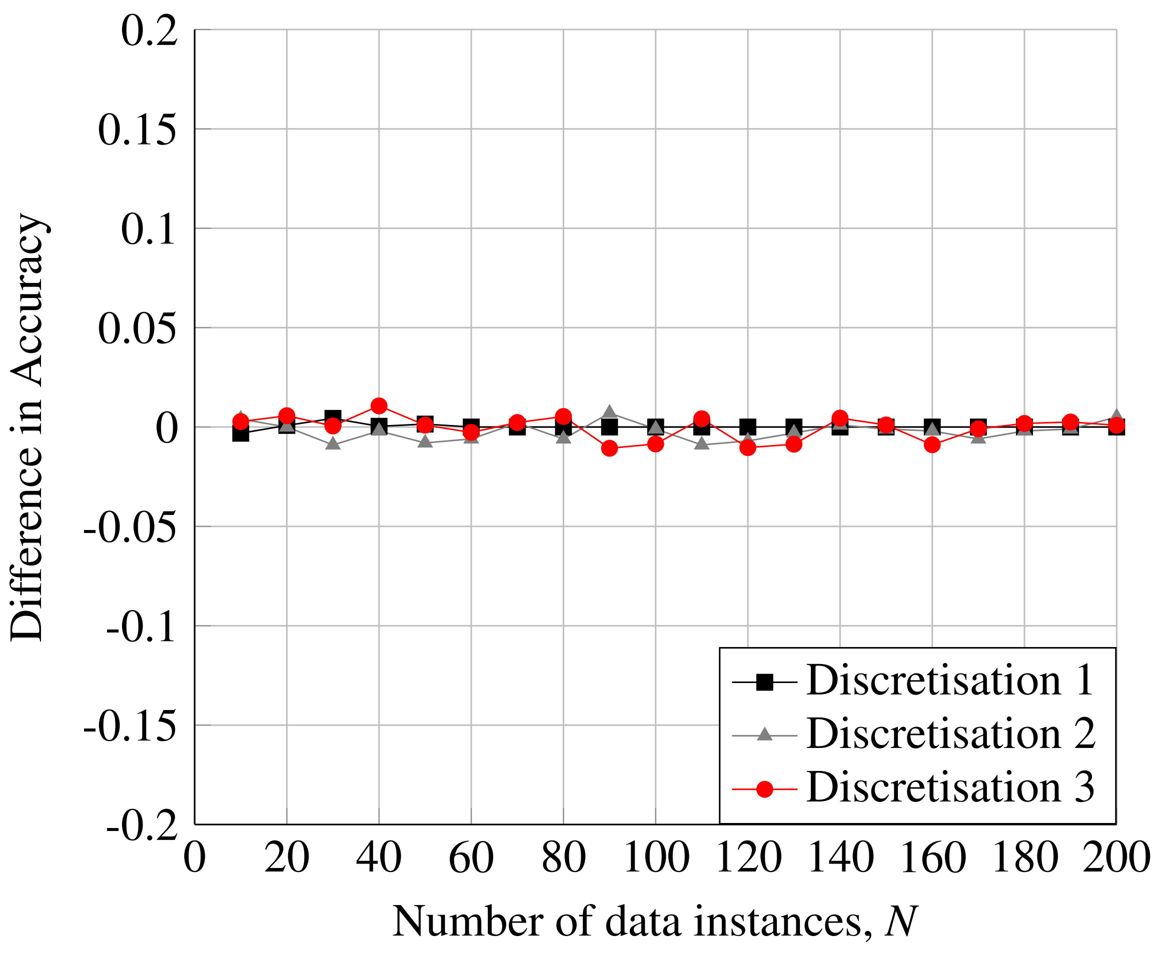


Figure 7.29: Difference in accuracy between the Bayesian network and IRL approaches for each discretisation configuration in task 1 decision 3.

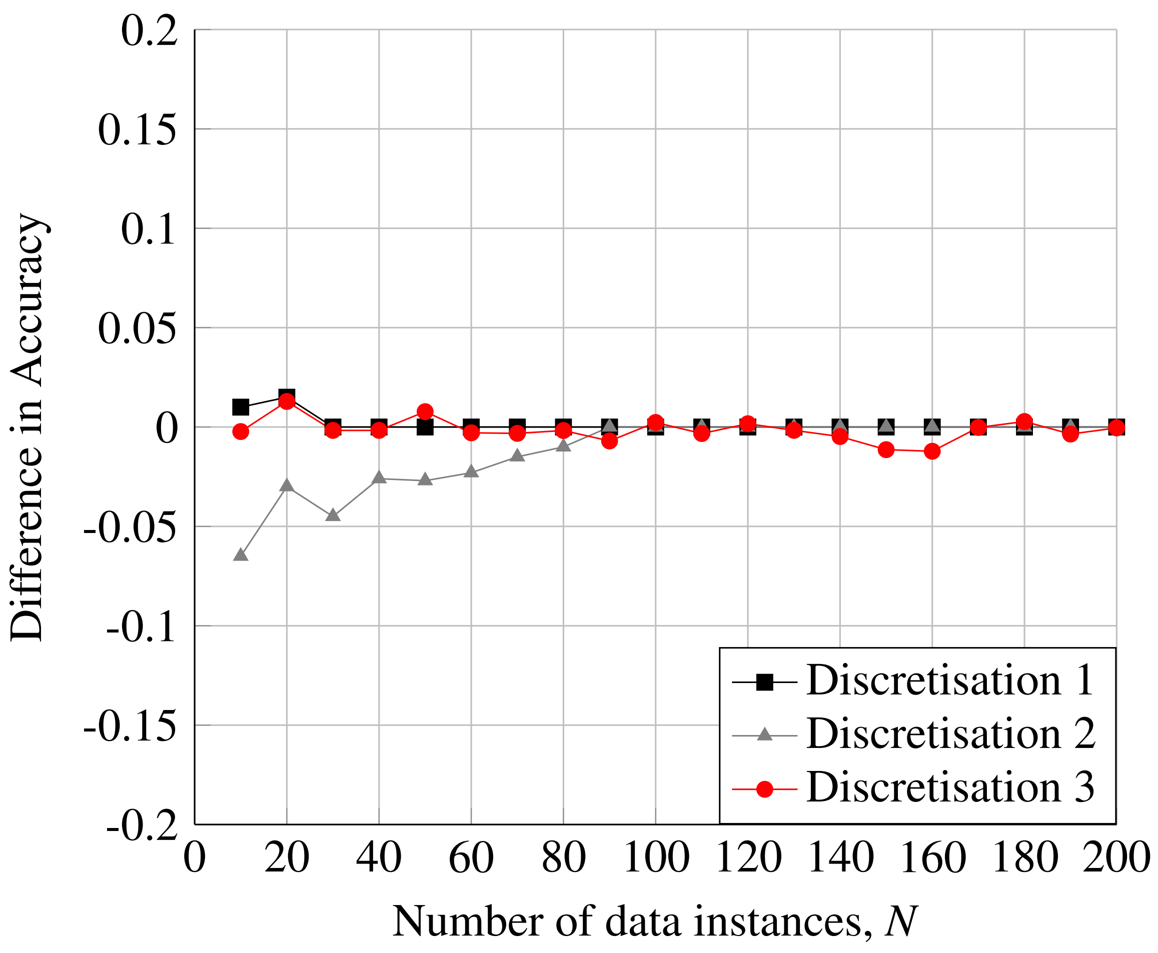


Figure 7.30: Difference in accuracy between the Bayesian network and IRL approaches for each discretisation configuration in task 1 decision 4.

### 7.4.2 Task 2

In decision 1, there is no significant difference between the two approaches after 40 data instances for any discretisation with neither approach surpassing the required accuracy of 0.95 in the first 40 data instances. Specifically, a maximum accuracy of 0.77 was achieved by the Bayesian network at 40 data instances for discretisation 1.

For decision 2, the IRL approach achieves a slightly greater accuracy on average for each discretisation. However, for this decision, discretisation 1 and 2 never surpass the required accuracy of 0.95 for either the IRL approach or the Bayesian network approach. For discretisation 3, the required accuracy is surpassed at 200 and 190 data instances for the Bayesian network and IRL approach respectively.

For decision 3, the Bayesian network approach initially outperforms the IRL approach, but with more data instances the IRL approach achieves a greater accuracy for all three discretisation configurations. However, the difference when more data instances were used is within the standard error of both approaches.

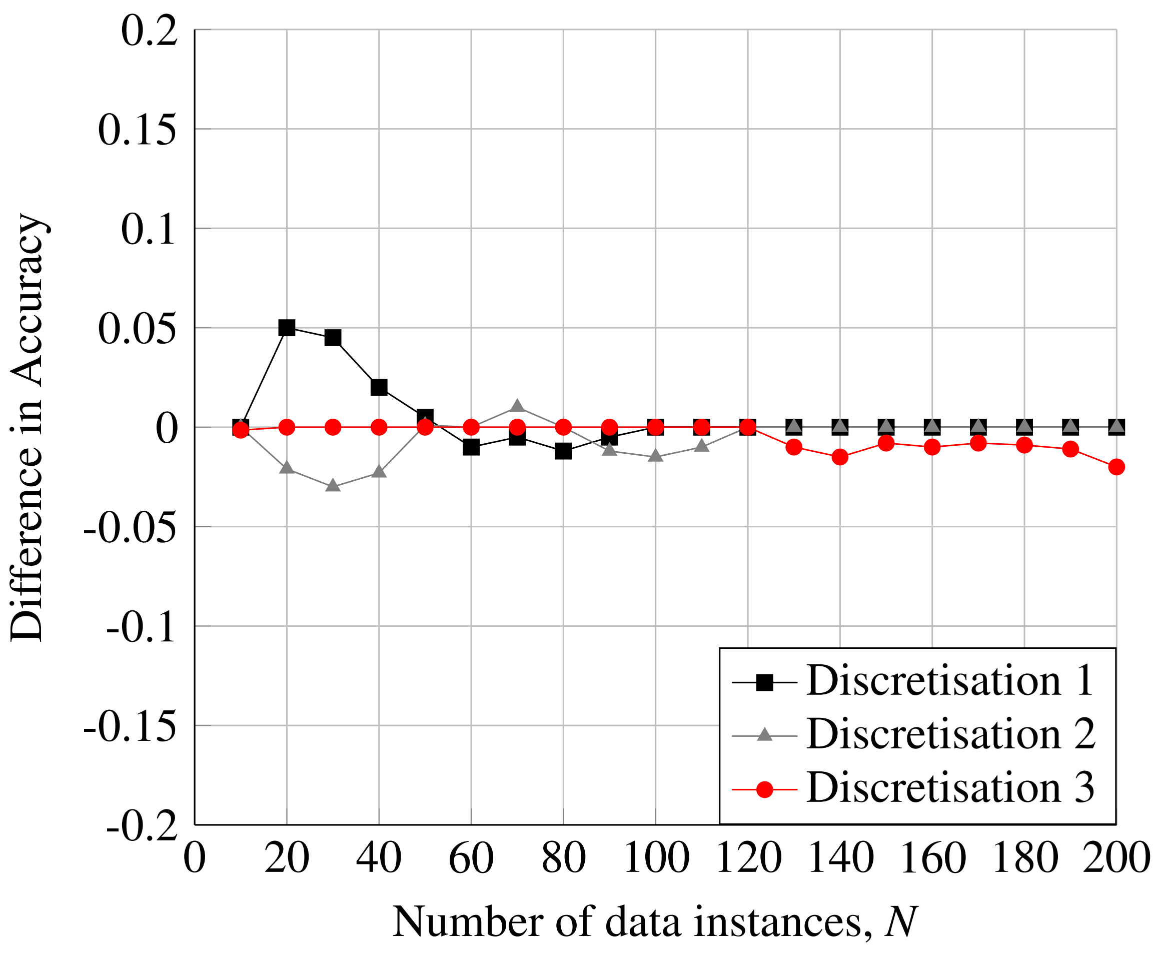


Figure 7.31: Difference in accuracy between the Bayesian network and IRL approaches for each discretisation configuration in task 2 decision 1.

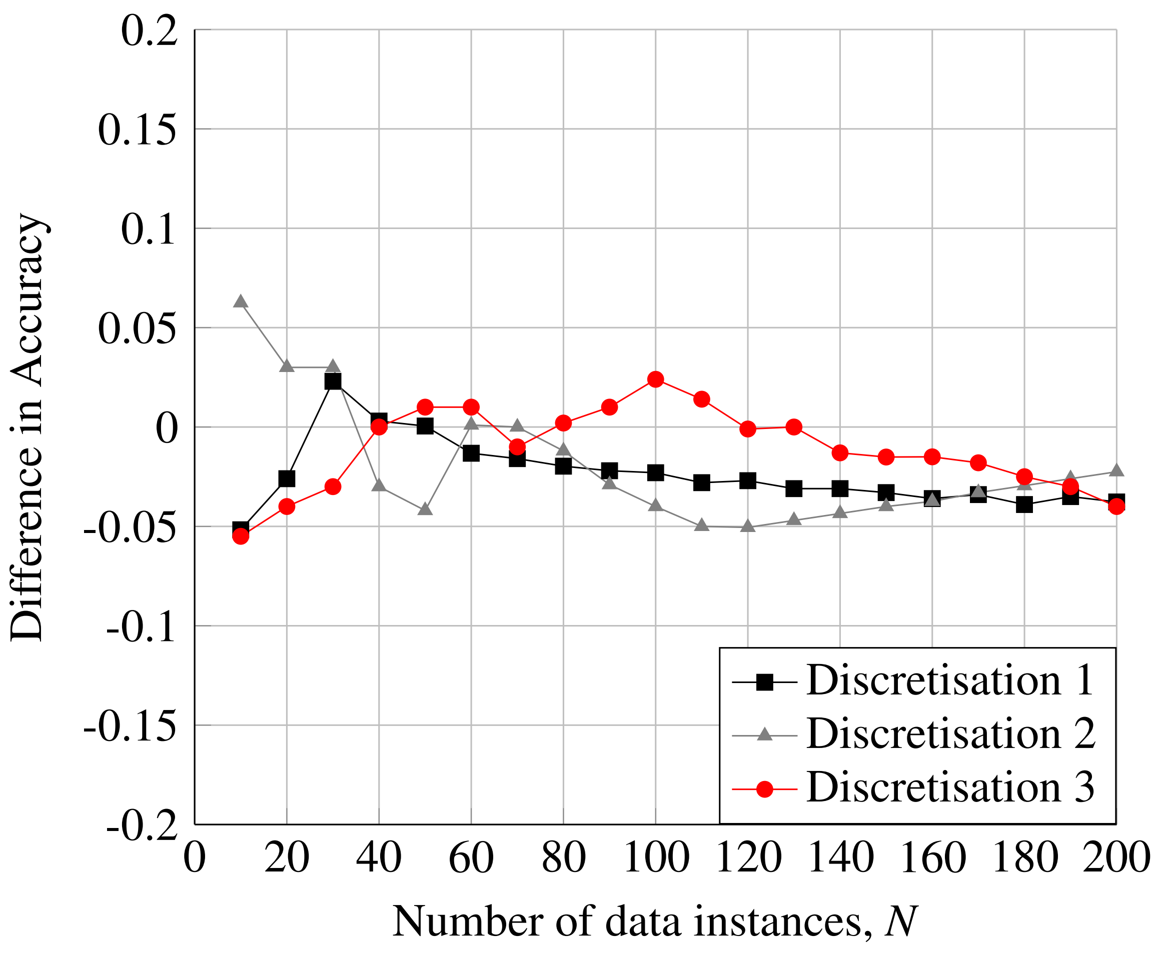


Figure 7.32: Difference in accuracy between the Bayesian network and IRL approaches for each discretisation configuration in task 2 decision 2.

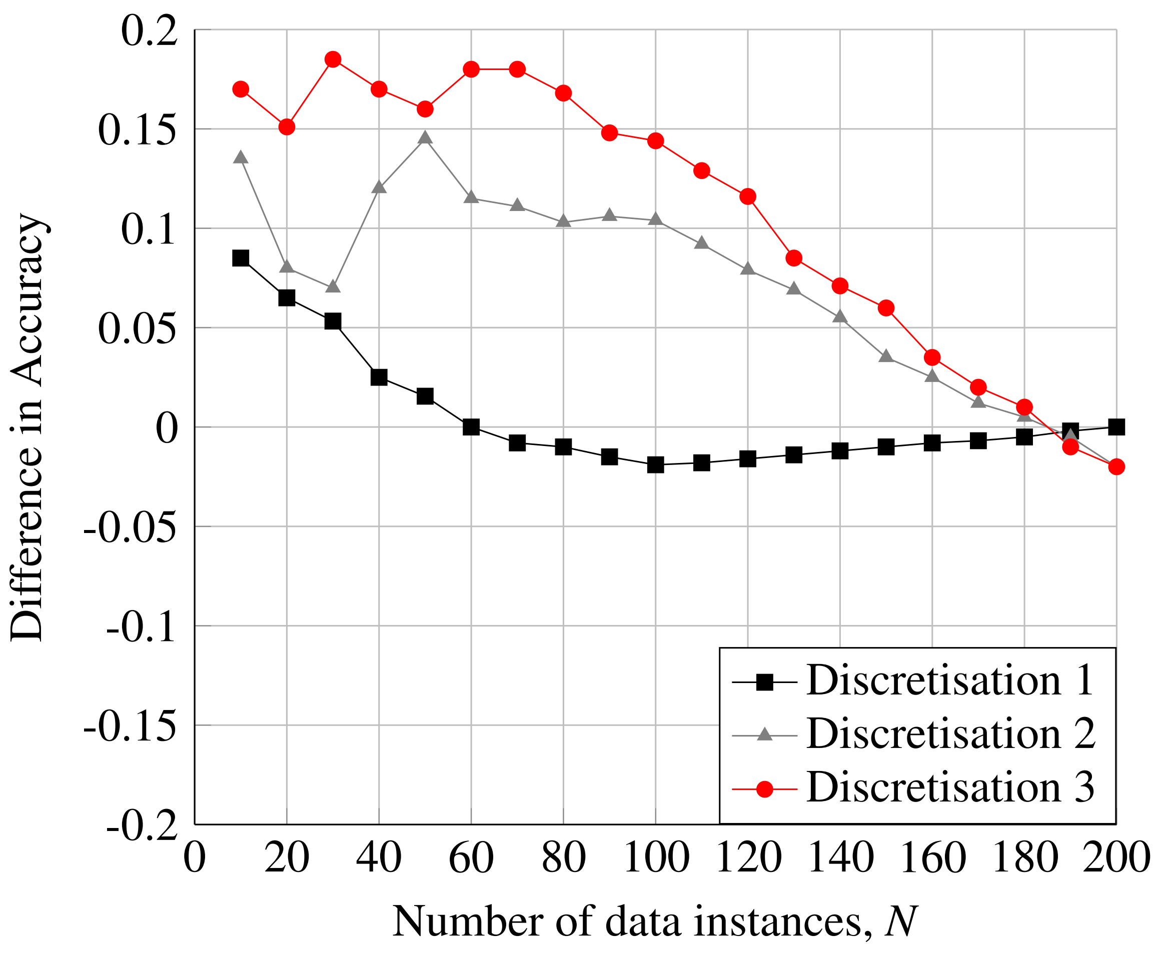


Figure 7.33: Difference in accuracy between the Bayesian network and IRL approaches for each discretisation configuration in task 2 decision 3.

### 7.4.3 Task 3

In task 3 decision 1, the difference between the two approaches initially seems significant. However, the biggest differences occur for discretisation 1 and 3 in which the accuracy never exceeds 0.6, far below the required accuracy of 0.95. Additionally, the accuracy in discretisation 2 surpasses the required accuracy after 400 data instances in both the IRL and Bayesian network case. After 400 data instances the difference between the two approaches is within the standard error of both approaches.

For decision 2, there is no significant difference between the IRL approach and the Bayesian network approach for discretisation 2 and 3. For discretisation 1, the Bayesian network initially has a greater accuracy, however neither approach exceeds an accuracy of 0.775 for discretisation 1.

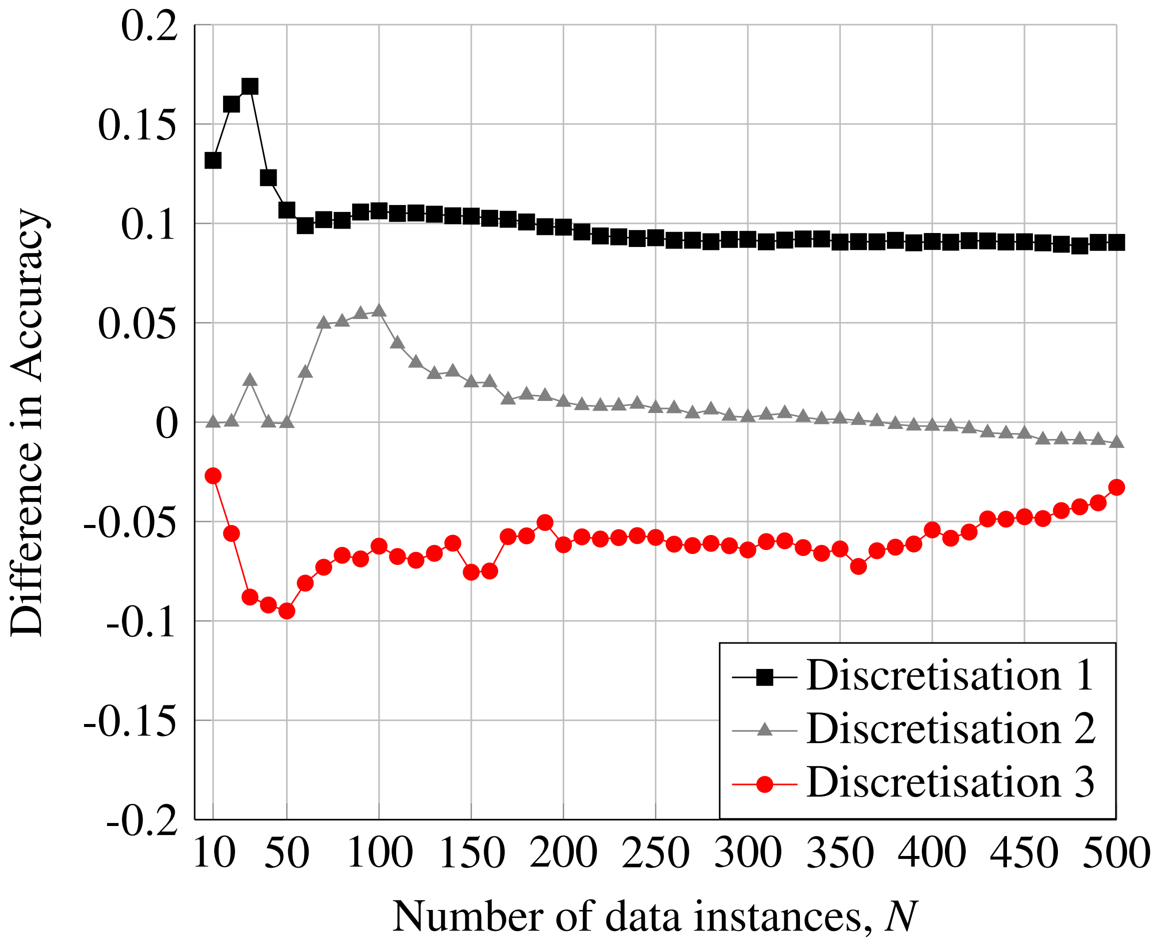


Figure 7.34: Difference in accuracy between the Bayesian network and IRL approaches for each discretisation configuration in task 3 decision 1.

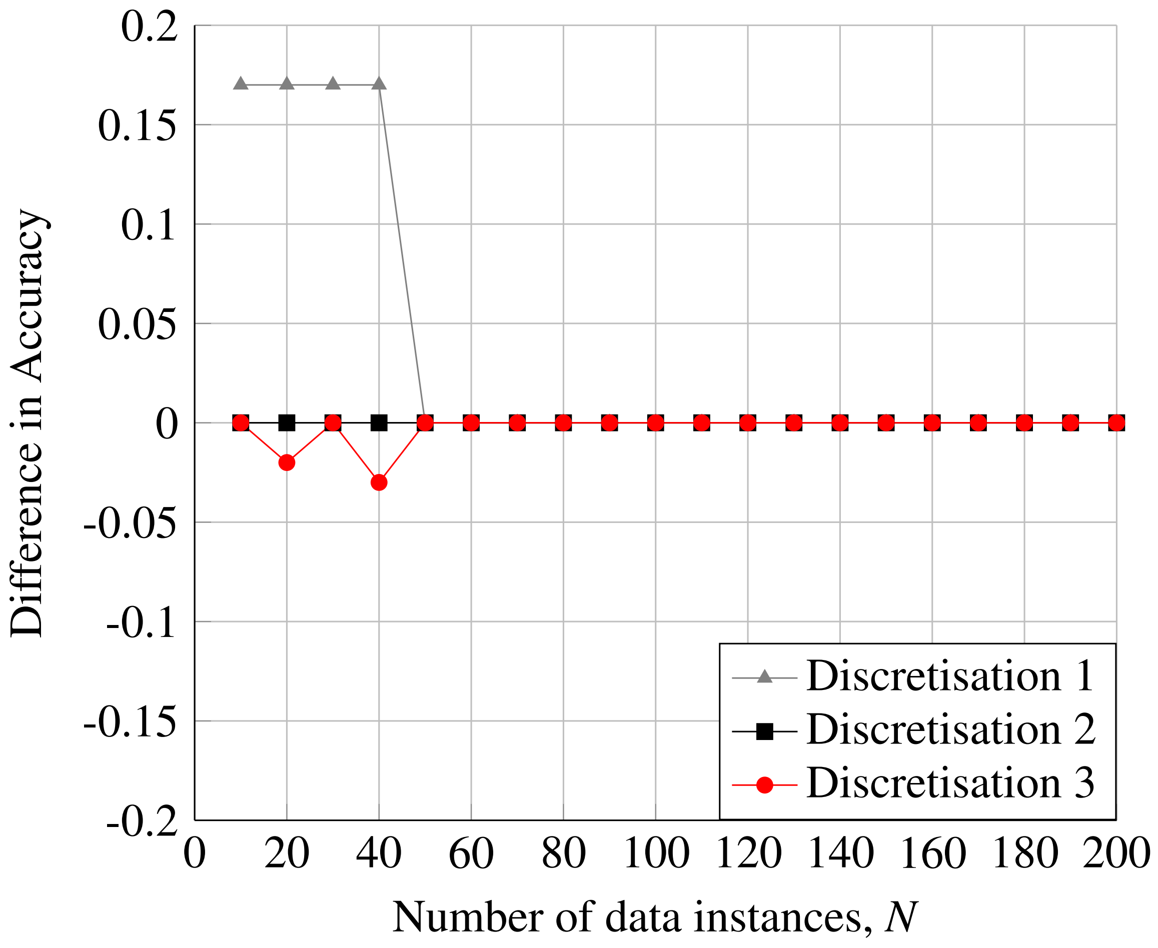


Figure 7.35: Difference in accuracy between the Bayesian network and IRL approaches for each discretisation configuration in task 3 decision 2.

## 7.5 Sources of Error

For each task considered, several sources of error became apparent when comparing the operator's decisions to both of the algorithm's decisions.

Firstly, the largest source of error was discretisation size used. For task 1 decision 3, moving between the three discretisation configurations meant an additional discretisation bin was added with each discretisation. In other words, discretisation 1 had 2 ranges each for abnormal size and abnormal speed, whilst there was 3 and 4 ranges each for discretisation 2 and 3 respectively. From Fig. 7.12 and Fig. 7.13, using discretisation 1 allows for an accuracy greater than the required 0.95 after only 40 data instances whilst discretisation 2 and 3 had an accuracy of around 0.55 and 0.35 respectively. In summary, by adding an additional bin, the accuracy after only 40 data instances dropped by around 0.4 and then dropped a further 0.2 with one more bin.

On the other hand, in certain discretisation configurations the ranges were too wide and did not sufficiently capture the operator's process. For example, in Fig. 7.16 and Fig. 7.17, only by using the discretisation configuration with the narrowest ranges was the AI agent able to surpass the required 0.95 accuracy. However, by increasing the number of discretisation ranges, the more training data the AI agent requires to achieve the required accuracy.

Another source of error is superfluous data. As shown in Fig. 7.12 and Fig. 7.13, discretisation 3 follows the same learning rate as discretisation 2 but has a constant negative offset in accuracy. This is likely due to the increase in data resolution offered by discretisation 3 providing no benefit to the AI agent's decision accuracy whilst requiring more data instances to reach the same accuracy.

A lack of information used during the decision making process is also a source of error. For example, if the operator made a decision to move towards a track based on whether its velocity was decreasing, then this would not be factored into the decision. However, there is an obvious trade-off between adding more variables for the AI agent to act upon and the number of data instances required to obtain sufficient performance.

Sensor error is another source of error in the decision making process. Whilst the operator and the AI agent are viewing the same sensor data, fluctuations in the radar returns can result in the data saved when the decision was made not being fully representative of the information the operator made the decision on. For example, if the operator makes a decision not to investigate a track as there is no abnormalities in the track's size or speed, but at the time the decision was made there was a spike in the radar return resulting in a track appearing to have a larger estimated size.

Another source of error is errors associated with the operator. The operator does not know the exact information of the full scenario. For example, when viewing the PPI display and deciding which track to further investigate, the operator might select a track that is slightly further away from one of the zones than another track is. Alternatively, the operator might miss a track entirely. Lastly, the operator is human and is subject to human error. Even under perfect circumstances, the operator will still make errors.

Continuing, the stochastic nature of the operator in certain tasks results in the accuracy plateauing at the same value for both approaches. Whilst this may seem to offer lower performance than other tasks, the Bayesian network approach, for example, is capable of learning the probabilities for these decisions, but it is desirable for any form of AI to be deterministic rather than stochastic. In other words, the AI will perform the same action given the same inputs, and as such, the accuracy is reduced by the AI agent operating in a deterministic manner.

## 7.6 Qualification of System

Beyond the decision accuracy of the AI agent, there is also the AI agent's decision making transparency to consider. As mentioned in the chapter , the initial plan was to use actual radar operator from Leonardo MW Ltd to interact with the GUI in order to obtain data. This plan would also have included questions posed to the operator on how well they trusted the autonomous system and how transparent they found the autonomous system. Specifically, a comparison would have been done between the operator's perceived trust and transparency in each algorithm. Results would include the percentage of data instances in which the operator overrode each algorithm. The results would also have included a measure of the operator's understanding of the algorithm's decisions, based on the questions posed.

However, in the absence of an actual radar operator there are still some points to be made with regards to algorithm transparency and trust. Fig. 7.36 shows how a Bayesian network can be visualised to the benefit of an operator. This visualisation allows the operator to inspect each decision's probabilities to see why a decision was made as well as the Bayesian network's learned probability of making each decision. Conversely, there is no concept of decision uncertainty or decision probability in the IRL methods used. The IRL algorithm offers a policy and a reward function which are not as obvious to the operator as to how confident the AI agent is in making the decision. This is a significant drawback in terms of transparency to the operator, and would ultimately make the system qualification difficult. Furthermore, if it is intended to move maritime surveillance radar autonomy towards level 3 (as mentioned in section 1.1.5), then the operator is still required to respond to a "request to intervene". In which case, it is essential that the radar operator understands the autonomous system's operation. Consequently, the Bayesian network visualisation provides a significant advantage over the IRL approach in terms of increasing radar autonomy.

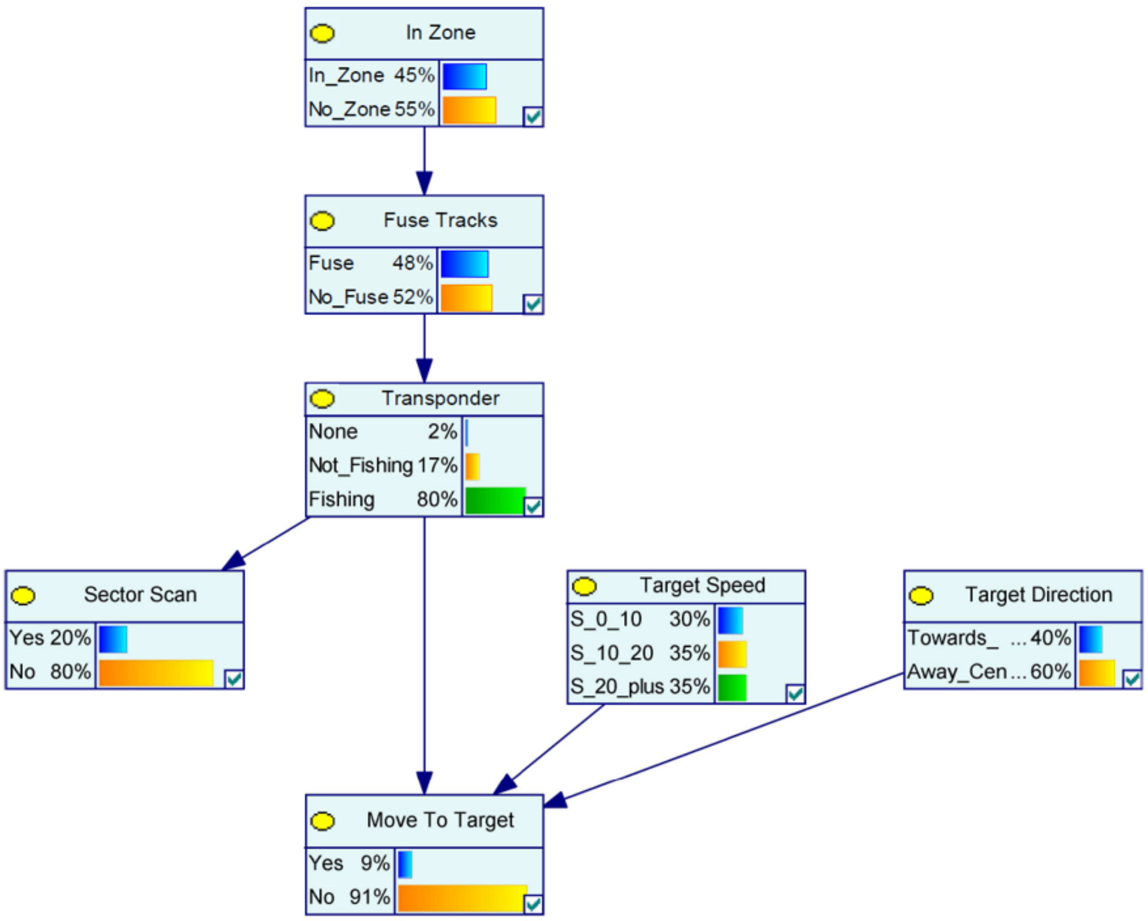


Figure 7.36: An example of how a Bayesian network can be visualised.

## 7.7 Conclusions

This chapter outlines the results of two methods of imitation learning tested within a maritime radar simulation against an operator's decisions given the same observational information. The results show the high accuracy and low error margin of both methods indicating their usefulness in imitating a human radar operator. For the majority of tasks, the Bayesian network approach and the IRL approach had negligible difference in accuracy, however, the Bayesian network offers greater transparency to the operator which is a significant advantage for operational use.

The sources of decision accuracy error are discussed and it is shown that the biggest limitation of the decision accuracy is not the chosen algorithm but the task variable discretisation ranges. Discretisation configurations with few but wide ranges typically allow for a high accuracy to be achieved with few data instances but then little or no improvement is seen with more data instances. However, discretisation configurations with many narrow ranges typically allow for the highest accuracy but only when a significant number of data instances are used. Specifically, a multiple roughly in the range of 1.9 to 4.5 data instances was required to achieve the same accuracy as the maximum accuracy of discretisation 1 for the discretisation that achieved the greatest accuracy. This multiple translates to the number of missions that the radar and operator would need to carry out to achieve the same accuracy.

# Chapter 8

# Conclusions and Further Work

## 8.1 Conclusions

The aim of this research was to improve the autonomy of maritime surveillance missions. By considering all aspects of the surveillance mission, several algorithms were developed and implemented to move the current operation of a surveillance mission towards full autonomy. The full mission was split into two aspects: the autonomy of the radar platform and the autonomy of the radar operator.

A simple and computationally efficient maritime radar simulation was developed in conjunction with a graphical user interface to allow an operator to control a radar system in a similar manner to an actual system in maritime surveillance scenarios. The radar was modelled to to receive power returns from both targets and sea clutter, and from the power returns, detections were obtained, refined, and passed to the tracker to form target tracks. Both the refined detections and the tracks were displayed on the operator's interface, with the operator able to select tracks to obtain more information on that track. Lastly, the operator had the ability to input guidance commands to the simulated platform which allowed for movement towards a specific target.

To address autonomy of the first mission aspect, the maritime surveillance radar simulation was used to determine the optimal radar platform trajectories for persistent surveillance. A polynomial trajectory generation method was derived to provide a simple method of producing the complex trajectories necessary to obtain the UAV dynamics for the fuel consumption, dynamic limitations of the UAV, and flight path. By discretising the search area and radar coverage area into grids, a computationally efficient way of obtaining the probability of detection and revisit time for each point in the search area was derived. This formulation was then used with a multi-objective particle swarm optimisation algorithm to obtain persistent radar platform trajectories that maximise the probability of detection, minimise the revisit time, and minimise the fuel consumption. In the two scenarios considers, several trajectories were found that outperformed the industry recommended baseline in all three cost functions.

The developed simulation and operator interface was further used to address the autonomy of the second mission aspect: the decision making. The typical tasks that an operator would perform during a maritime surveillance radar mission were outlined and data was collected from a human operator performing these tasks. Methods of imitation learning were then investigated as a way introducing autonomy into mission critical radar operations due to their ability to perform the tasks in the same manner as the current operational standard. Two methods of imitation learning were then implemented and their efficacy in imitating a human radar operator compared. The results show the high accuracy and low error margin of both methods indicating their usefulness in performing the maritime surveillance tasks in the same manner as a human operator. Whilst there was little difference between the accuracy of both methods, the Bayesian network offers greater operator transparency which is a significant advantage for operational use. The sources of error in the algorithms' decision accuracy were then discussed and it was concluded that the biggest limitation in accuracy is not the chosen algorithm, but the way in which the task is discretised.

## 8.2 Further Work

### 8.2.1 Trajectory Optimisation

The obvious direction for further work on the outlined trajectory optimisation algorithm is to consider multiple UAVs operating with one or more designated search areas. The optimisation formulation could then be expanded to include a coupling of the multiple UAVs' trajectories.

The trajectory optimisation work considered here dealt with areas of uniform importance. However, certain missions deal with search areas where sub-regions are known to be more likely to contain targets. One such example is a missing person search. By pre-initialising the radar coverage matrices at the sub-regions of interest, the outlined algorithm could be used to determine trajectories for these missions.

Another scenario to consider is the surveillance of a moving vessel, for example a ship under threat as it passes through a particular area. In this scenario, the surveillance around the moving vessel is of more importance than at points further away. Consequently, the weighting of the radar coverage closer to the ship at a specific time along the moving vessel's trajectory would be higher.

Whilst the trajectory optimisation algorithm naturally converges to solutions with smooth trajectories as this minimises the fuel consumption cost, it would be beneficial to introduce a radar coverage 'smoothness' metric into the radar costs. For search areas that are more complex than a square or curved area, such as a specific coastline, there is a trade-off between minimising the fuel consumption and keeping the radar's field of view centred on the coastline. By introducing a radar coverage smoothness metric, the algorithm would converge to trajectories preferred by the operator in both the fuel consumption cost and the radar coverage costs.

This work outlines algorithms for autonomy for both the aircraft and the radar, and consequently, another useful avenue of research is to use the trajectory algorithm optimisation to assist the operator in the task outlined in section 4.3.3. Since the radar operator is not an aircraft pilot, the trajectory optimisation algorithm could be used to inform the operator which aircraft directions would lead to minimising the time spent investigating the track as well as maximising the probability of determining the track's operational intentions. This can then be extended to the situation where the radar operator may wish to investigate several tracks in succession. For example, if there are several boats in a designated zone which are not responding to operator communication, then the operator may wish to investigate each track in succession. Consequently, determining the trajectory that minimises the time to investigate all tracks is vital for the success of the surveillance mission.

For constant surveillance, there must be a hand-over from one UAV to another. Another factor in the optimisation problem could be determining the coordinates of this hand-over and how this affects fuel consumption and radar coverage.

Due to the computational time required to obtain optimised trajectories, another avenue of research would be to introduce machine learning techniques to reduce the required computational time. One such example is to use Kriging techniques to approximate each objective function [168]. Similarly, research could be done to adapt existing optimised trajectories to other search areas by exploiting the similarities between search areas.

### 8.2.2 Autonomous Decision Making

One of the biggest effects on the algorithm's decision accuracy is how the task is defined for the algorithm. From chapter 7, the way in which each task is discretised has a major effect on the accuracy relative to the number of data samples. The accuracy of the algorithms relative to the number of available data samples could be improved by introducing an optimisation algorithm that determines the best discretisation configuration for a given number of data samples.

Two algorithms were considered for improving the autonomy in the radar operator decision making. However, an interesting alternative to inverse reinforcement learning is Generative Adversarial Imitation Learning (GAIL) [169], which is akin to generative adversarial networks (GANs) [170]. The main advantage of this method is that it removes the computationally expensive reinforcement learning inner loop to obtain the policy.

Another consideration for future work is the use of Dynamic Bayesian Networks [171], [172]. Dynamic Bayesian Networks allow for the relation of data between time-steps, for example, changes in direction or velocity. These networks typically require more data samples to achieve an acceptable accuracy, but for the latter stages of the AI learning process, these networks would allow for further refinement of the decision making and could result in an improvement in accuracy. Additionally, these type of networks might reduce the errors associated with instantaneous sensor measurements.

One potential improvement on the IRL algorithm employed for task 3 is to consider partially observable Markov decision processes [173]. A partially observable Markov decision process assumes an underlying Markov decision process but that the agent cannot directly observe each state and instead observes each state through noisy observations. Each action taken has a stochastic result upon which the agent receives a noisy observation. However, the factors that go into getting the probability of a target being illegal or not are complex (e.g. SNR, vessel type, angles and distances, sea state). Each of these would have to be factored into the MDP states which would greatly increase the size of the MDP (for example, a state for the current sea state and whether the target is at a certain distance, angle, SNR). If all these factors are known, this could technically be done in IRL for POMDP, but the size of the underlying MDP would make learning on few samples difficult. For later stages of the learning process where more data samples are available, IRL for POMDP could be used to further increase the decision making accuracy.

### 8.2.3 Remote Operation

From the literature on UAV remote operation, one of the most significant issues faced by the remote operators is the sensory isolation from the aircraft [174]. Pilots typically use ambient visual input, kinaesthetic, vestibular information to aid their understanding of the aircraft and their decision making. Additionally, current radar operators will use their view of the local environment when making decisions. This view includes the weather (such as rain or wind), the sea-state, and any objects within visual range. Consequently, remote radar operators would suffer similar sensory isolation issues as the remote UAV operators. Investigation is required into how useful this information is to the operator, whether a remote operator suffers if this information were to be sent over a data-link in some form, and how to tie this information into an autonomous system.

### 8.2.4 Additional Topics of Research

A typical maritime surveillance mission is performed using one radar operator, however there may be multiple operators performing the same mission on a regular basis. Consequently, there may be operator data from multiple operators of varying experience. By using the methods outlined by Beliaev [175], data obtained from multiple operators with varying expertise can be used to determine a single optimal policy that can outperform even the best operator. Additionally, the work in Beliaev can be used to determine the expertise level of each operator which could be useful as a training tool for new operators.

In the future, there may be surveillance mission which require multiple UAVs performing simultaneous surveillance missions. It would then be necessary to consider the cooperation between operators [176].

### 8.2.5 Learning without an Operator

The chosen algorithms for imitation learning can be easily adapted to continue learning without an operator which is a useful feature when moving from management-by-consent to management-by-exception.

"In uncharted territory—where one would expect learning to be most beneficial—an agent must be able to learn from its own experience." [161]. It is therefore important that for future missions, where the agent is performing with management-by-exception, the agent is able to learn by itself from its own experiences.

Another benefit to replacing a remote operator with an AI agent is the reduced data rates in uplink and downlink, potentially removing a bottleneck in the whole radar system. This benefit arises as it would no longer be necessary to send the radar data to a remote operator as the AI agent can process the data onboard. Furthermore, sending data to and from the radar and the remote operator could potentially introduce latency into the system [6]. Onboard decision making removes this latency which, for missions where fast decision making is required, is of great importance as any delay could cause significant operational failures.

The use of active learning could be used during the transition from management-by-consent to management-by-exception where, if the AI agent finds itself in an unknown situation, the agent can query the operator for a decision and use that decision for learning. This is similar to the DaGGER approach [125].

# Appendix A

# Inverse Reinforcement Learning

Equation 6.20 and 6.26 outline the inverse reinforcement learning algorithms initially derived by Ng and Russell [136]. These equations provide the fundamentals to IRL, and understanding it allows for the equations derived by Abbeel and Ng [137] to be understood more easily. Therefore, this section outlines an alternative form of these equations that may help in their understanding.

## A.1 Value Matrix

$$\underline{\boldsymbol{F}}_{n_f,n_s} = \begin{pmatrix} V_{1,1} & V_{1,2} & \cdots & V_{1,n_s} \\ V_{2,1} & V_{2,2} & \cdots & V_{2,n_s} \\ \vdots & \vdots & \cdots & \vdots \\ V_{n_f,1} & V_{n_f,2} & \cdots & V_{n_f,n_s} \end{pmatrix} \tag{A.1}$$

where $n_f$ is the number of feature and $n_s$ is the number of states. Therefore, the value matrix can be obtained as follows:

$$V = \underline{\boldsymbol{F}}^T \cdot \boldsymbol{\alpha} \tag{A.2}$$

## A.2 IRL MDP Linear Programming

For reference, equation 6.20 is rewritten below:

$$maximise \sum_{s \in S_0} \underset{a \in A \backslash a^*}{min}\, p\left( \underset{s' \sim P(s'|s,a^*)}{\boldsymbol{E}}[V^{\pi^*}(s')] - \underset{s' \sim P(s'|s,a)}{\boldsymbol{E}}[V^{\pi^*}(s')] \right) \tag{A.3}$$

$$s.t.\ |\alpha_i| \leq 1,\ i = 1, ..., d$$

Where $p(x) = x$ when $x \geq 0$ and $p(x) = 2x$ when $x < 0$. The penalty coefficient of 2 was chosen heuristically by Ng and Russell.

This can be re-written as follows:

$$\begin{aligned}
&\textit{maximise} \sum_{s \in S} \min_{a \in A \backslash a^*} p(\underline{\boldsymbol{v}} \cdot \boldsymbol{\alpha}) \\
&\textit{maximise} \sum_{s \in S} \min_{a \in A \backslash a^*} \min(\underline{\boldsymbol{v}} \cdot \boldsymbol{\alpha}, 2\underline{\boldsymbol{v}} \cdot \boldsymbol{\alpha}) \\
&\textit{maximise} \sum_{i=1}^{n_s} \min(\underline{\boldsymbol{v}}_{i,1} \cdot \boldsymbol{\alpha}, 2\underline{\boldsymbol{v}}_{i,1} \cdot \boldsymbol{\alpha}, \underline{\boldsymbol{v}}_{i,2} \cdot \boldsymbol{\alpha}, 2\underline{\boldsymbol{v}}_{i,2} \cdot \boldsymbol{\alpha}, \cdots, \underline{\boldsymbol{v}}_{i,n_a-1} \cdot \boldsymbol{\alpha}, 2\underline{\boldsymbol{v}}_{i,n_a-1} \cdot \boldsymbol{\alpha})
\end{aligned} \tag{A.4}$$

Given the following two matrices:

$$\underline{\boldsymbol{v}}_{n_s,n_a-1,n_f} = \begin{pmatrix}
\begin{pmatrix} \Delta V_{1,1,1} & \Delta V_{1,2,1} & \cdots & \Delta V_{1,n_a-1,1} \\ \Delta V_{2,1,1} & \Delta V_{2,2,1} & \cdots & \Delta V_{2,n_a-1,1} \\ \vdots & \vdots & \cdots & \vdots \\ \Delta V_{n_s,1,1} & \Delta V_{n_s,2,1} & \cdots & \Delta V_{n_s,n_a-1,1} \end{pmatrix} \\
\begin{pmatrix} \Delta V_{1,1,2} & \Delta V_{1,2,2} & \cdots & \Delta V_{1,n_a-1,2} \\ \Delta V_{2,1,2} & \Delta V_{2,2,2} & \cdots & \Delta V_{2,n_a-1,2} \\ \vdots & \vdots & \cdots & \vdots \\ \Delta V_{n_s,1,2} & \Delta V_{n_s,2,2} & \cdots & \Delta V_{n_s,n_a-1,2} \end{pmatrix} \\
\vdots \\
\begin{pmatrix} \Delta V_{1,1,n_f} & \Delta V_{1,2,n_f} & \cdots & \Delta V_{1,n_a-1,n_f} \\ \Delta V_{2,1,n_f} & \Delta V_{2,2,n_f} & \cdots & \Delta V_{2,n_a-1,n_f} \\ \vdots & \vdots & \cdots & \vdots \\ \Delta V_{n_s,1,n_f} & \Delta V_{n_s,2,n_f} & \cdots & \Delta V_{n_s,n_a-1,n_f} \end{pmatrix}
\end{pmatrix} \tag{A.5}$$

where $\Delta V_{i,j,k=:} = P(i|a^*) \times \underline{\boldsymbol{F}}^T - P(i|j) \times \underline{\boldsymbol{F}}^T$ for state $i$, action $j$, and feature $k$.

$$\underline{\boldsymbol{v}} \cdot \boldsymbol{\alpha} = \begin{pmatrix} \Delta V_{1,1} & \Delta V_{1,2} & \cdots & \Delta V_{1,n_a-1} \\ \Delta V_{2,1} & \Delta V_{2,2} & \cdots & \Delta V_{2,n_a-1} \\ \vdots & \vdots & \cdots & \vdots \\ \Delta V_{n_s,1} & \Delta V_{n_s,2} & \cdots & \Delta V_{n_s,n_a-1} \end{pmatrix} \tag{A.6}$$

where $\Delta V_{i,j} = \Delta V_{i,j,1}\alpha_1 + \Delta V_{i,j,2}\alpha_2 + \cdots + \Delta V_{i,j,n_f}\alpha_{n_f}$. Equation 6.26 can be restated by using the the matrices outlined above:

## A.3 IRL MDP Linear Programming - Sampled Trajectories

For reference, equation 6.26 is rewritten below:

$$\textit{maximise} \sum_{\pi \in \Pi} p\left(\hat{V}^{\pi^*}(s_0) - \hat{V}^{\pi}(s_0)\right) \tag{A.7}$$

$$s.t.\ |\alpha_i| \leq 1,\ i = 1, ..., d$$

where as before:

$$p(x) = \begin{cases} x, & x \geq 0 \\ 2x, & x < 0 \end{cases}$$

This can be rewritten as follows:

$$\begin{aligned} &\max_{\boldsymbol{\alpha}} \sum_{\pi \in \Pi} p(\boldsymbol{v}^{\pi} \cdot \boldsymbol{\alpha}) \\ &\max_{\boldsymbol{\alpha}} \sum_{\pi \in \Pi} \min(\boldsymbol{v}^{\pi} \cdot \boldsymbol{\alpha}, 2\boldsymbol{v}^{\pi} \cdot \boldsymbol{\alpha}) \\ &\max_{\boldsymbol{\alpha}} [\min(\boldsymbol{v}^{\pi_1} \cdot \boldsymbol{\alpha}, 2\boldsymbol{v}^{\pi_1} \cdot \boldsymbol{\alpha}) + ... + \min(\boldsymbol{v}^{\pi_\Pi} \cdot \boldsymbol{\alpha}, 2\boldsymbol{v}^{\pi_\Pi} \cdot \boldsymbol{\alpha})] \end{aligned} \tag{A.8}$$

Equation A.8 can then be written in linear programming format as follows:

$$\begin{aligned} \underset{\boldsymbol{\alpha}}{\text{maximize}} \quad & \sum_{\pi \in \Pi} t_{\pi} \\ \text{subject to} \quad & \boldsymbol{v}^{\pi} \cdot \boldsymbol{\alpha} \leq t_{\pi}, \quad \pi \in \Pi, \\ & 2\boldsymbol{v}^{\pi} \cdot \boldsymbol{\alpha} \leq t_{\pi}, \quad \pi \in \Pi, \\ & |\alpha_i| \leq 1, \quad i = 1, \ldots, d \end{aligned} \tag{A.9}$$

# Bibliography

[1] F. Jiang, Y. Jiang, H. Zhi, *et al.*, *Artificial intelligence in healthcare: Past, present and future*, 2017. DOI: 10.1136/svn-2017-000101.

[2] R. Abduljabbar, H. Dia, S. Liyanage, and S. A. Bagloee, *Applications of artificial intelligence in transport: An overview*, 2019. DOI: 10.3390/su11010189.

[3] A. Bahrammirzaee, "A comparative survey of artificial intelligence applications in finance: Artificial neural networks, expert system and hybrid intelligent systems," *Neural Computing and Applications*, 2010, ISSN: 09410643. DOI: 10.1007/s00521-010-0362-z.

[4] R. Parasuraman and V. Riley, "Humans and automation: Use, misuse, disuse, abuse," *Human Factors*, 1997, ISSN: 00187208. DOI: 10.1518/001872097778543886.

[5] N. Sarter, D. Woods, and C. Billings, "Automation Suprises," in *Handbook of Human Factors & Ergonomics*, G. Salvendy, Ed., 2nd Editio, New York: Wiley, 1997, pp. 1926–1943.

[6] J. McCarley and C. Wickens, "Human factors implications of UAVs in the national airspace," *Vasa*, 2005.

[7] K. W. Williams, "A Summary of Unmanned Aircraft Accident / Incident Data: Human Factors Implications," Tech. Rep., 2004.

[8] P. Eisenstein, *Safety groups want FTC, state probes of Tesla's Autopilot system-and its marketing efforts*, 2019. [Online]. Available: https://www.cnbc.com/2019/07/26/safety-groups-want-ftc-state-probes-of-teslas-autopilot-system.html (visited on 12/03/2020).

[9] T. Hitchens, *AI Slays Top F-16 Pilot In DARPA Dogfight Simulation*, 2020. [Online]. Available: https://breakingdefense.com/2020/08/ai-slays-top-f-16-pilot-in-darpa-dogfight-simulation (visited on 12/02/2020).

[10] R. Hopcroft, E. Burchat, and J. Vince, "Unmanned Aerial Vehicles for Maritime Patrol: Human Factors Issues," DSTO Defence Science and Technology Organisation, Victoria, Australia, Tech. Rep., 2006, pp. 1–43. [Online]. Available: apps.dtic.mil/sti/pdfs/ADA454918.pdf.

[11] Y. Lim, A. Gardi, R. Sabatini, *et al.*, *Avionics Human-Machine Interfaces and Interactions for Manned and Unmanned Aircraft*, 2018. DOI: 10.1016/j.paerosci.2018.05.002.

[12] H. A. Ruff, S. Narayanan, and M. H. Draper, “Human Interaction with Levels of Automation and Decision-Aid Fidelity in the Supervisory Control of Multiple Simulated Unmanned Air Vehicles,” *Presence: Teleoperators and Virtual Environments*, vol. 11, no. 4, 2002. DOI: `10.1162/105474602760204264`.

[13] M. Mouloua, R. Gilson, E. Daskarolis-Kring, *et al.*, “Ergonomics of UAV/UCAV Mission Success: Considerations for Data Link, Control, and Display Issues,” in *Proceedings of the Human Factors and Ergonomics Society 45th Annual Meeting*, Minneapolis, USA, 2001, pp. 144–148. DOI: `10.1177/154193120104500231`.

[14] V. Riley, “Operator Reliance on Automation: Theory and Data,” in *Automation and Human Performance: Theory and Applications*, R. Parasuraman and M. Mouloua, Eds., Lawrence Erlbaum Associates, 1996, pp. 19–35, ISBN: 9780805816167.

[15] W. Olson, “Identifying and Mitigating the Risks of Cockpit Automation,” *Air Command Staff and College Wright Flyer Paper No. 14*, 2001.

[16] R. Parasuraman, R. Molloy, M. Mouloua, and B. Hilburn, “Monitoring of automated systems,” in *Automation and Human Performance: Theory and Applications*, R. Parasuraman and M. Mouloua, Eds., Hillsdale, New Jersey: Lawrence Erlbaum Associates, 1996, pp. 91–115.

[17] P. Sparaco, “A330 Crash to Spur Changes at Airbus,” *Aviation Week and Space Technology*, vol. 141, no. 6, pp. 20–22, 1994.

[18] International Nuclear Safety Advisory Group, “The Chernobyl Accident: Updating of INSAG-1,” IAEA, Vienna, Tech. Rep., 1992.

[19] F. Lerch and M. Prictula, “How Do We Trust Machine Advice? Designing and using human-computer interfaces and knowledge based systems,” in *Proceedings of the Third International Conference on Human-Computer Interaction*, Amsterdam: Elsevier, 1989, pp. 411–419.

[20] S. R. Dixon, C. D. Wickens, and D. Chang, “Comparing Quantitative Model Predictions to Experimental Data in Multiple-UAV Flight Control,” in *Proceedings of the Human Factors and Ergonomics Society 47th Annual Meeting*, 2003. DOI: `10.1177/154193120304700122`.

[21] R. Parasuraman, R. Molloy, and I. L. Singh, “Performance Consequences of Automation-Induced “Complacency”,” *The International Journal of Aviation Psychology*, vol. 3, 1993. DOI: `10.1207/s15327108ijap0301_1`.

[22] J. B. F. van Erp, “Controlling Unmanned Vehicles: The Human Factors Solution,” in *RTO SCI Symposium on "Warfare Automation: Procedures and Techniques for Unmanned Vehicles"*, Ankara, Turkey, 2000.

[23] P. J. Durlach, “Change Blindness and its Implications for Complex Monitoring and Control Systems Design and Operator Training,” *Human-Computer Interaction*, vol. 19, no. 4, pp. 423–451, 2004. DOI: `10.1207/s15327051hci1904_10`.

[24] R. Parasuraman, T. B. Sheridan, and C. D. Wickens, “A model for types and levels of human interaction with automation,” *IEEE Transactions on Systems, Man, and*

*Cybernetics Part A:Systems and Humans.*, 2000, ISSN: 10834427. DOI: 10.1109/3468.844354.

[25] SAE International, “Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles,” Tech. Rep., 2018, pp. 1–35. [Online]. Available: https://saemobilus.sae.org/content/j3016_201806.

[26] U.S. Department of Transportation, “Automated Vehicles Comprehensive Plan,” Washington DC, Tech. Rep., 2021. [Online]. Available: https://www.transportation.gov/av/avcp.

[27] Y. Pan, C. A. Cheng, K. Saigol, *et al.*, “Imitation learning for agile autonomous driving,” *The International Journal of Robotics Research*, vol. 39, no. 2-3, pp. 286–302, 2020. DOI: 10.1177/0278364919880273.

[28] N. B. Sarter and D. D. Woods, “How in the World Did We Ever Get into That Mode? Mode Error and Awareness in Supervisory Control,” *Human Factors*, vol. 37, no. 1, pp. 5–19, 1995. DOI: 10.1518/001872095779049516.

[29] Leonardo MW Ltd, “Private Communication Regarding Increased Radar System Performance and the Effect on the Operator,” 2020.

[30] ——, *Private Communication Regarding the Qualification of Algorithms for Autonomous Operations*, 2016.

[31] S. Legg and M. Hutter, “A Collection of Definitions of Intelligence,” *Frontiers in Artificial Intelligence and Applications*, vol. 157, pp. 17–24, 2007. arXiv: arXiv:0706.3639.

[32] D. L. Poole, A. Mackworth, and R. G. Goebel, *Computational Intelligence: A Logical Approach*. New York, NY, USA: Oxford University Press, 1998, pp. 1–7, ISBN: 978-0195102703.

[33] T. Mitchell, *Machine Learning*. New York: McGraw-Hill, 1997, ISBN: 9780070428072.

[34] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA: MIT Press, 2016, p. 1, ISBN: 9781491925614.

[35] H. Tran, “A Survey of Machine Learning and Data Mining Techniques used in Multimedia System,” 2019. DOI: 10.13140/RG.2.2.20395.49446/1.

[36] K. He, X. Zhang, S. Ren, and J. Sun, “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, ISBN: 9781467383912. DOI: 10.1109/ICCV.2015.123. arXiv: 1502.01852.

[37] S. Chen and H. Wang, “SAR Target Recognition Based on Deep Learning,” *2014 International Conference on Data Science and Advanced Analytics (DSAA)*, pp. 541–547, 2014.

[38] S. Haykin, “Cognitive radar: a way of the future,” *IEEE Signal Processing Magazine*, 2006, ISSN: 1053-5888. DOI: 10.1109/msp.2006.1593335.

[39] K. L. Bell, C. J. Baker, G. E. Smith, *et al.*, "Cognitive Radar Framework for Target Detection and Tracking," *IEEE Journal on Selected Topics in Signal Processing*, 2015, ISSN: 19324553. DOI: 10.1109/JSTSP.2015.2465304.

[40] S. Haykin, "Cognitive dynamic systems," *International Journal of Cognitive Informatics and Natural Intelligence*, 2011. DOI: 10.4018/jcini.2011100103.

[41] E. Selvi, R. M. Buehrer, A. Martone, and K. Sherbondy, "On the use of Markov Decision Processes in cognitive radar: An application to target tracking," in *2018 IEEE Radar Conference, RadarConf 2018*, 2018. DOI: 10.1109/RADAR.2018.8378616.

[42] A. Brown and D. Anderson, "Trajectory Optimization for High-Altitude Long-Endurance UAV Maritime Radar Surveillance," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 56, no. 3, pp. 2406–2421, 2020. DOI: 10.1109/TAES.2019.2949384.

[43] ——, "Imitating Radar Operator Decisions for Maritime Surveillance Missions Using Bayesian Networks," in *2019 International Radar Conference (RADAR)*, Toulon, France, 2019, pp. 1–6. DOI: doi:10.1109/RADAR41533.2019.171229.

[44] D. Anderson, *MAVERIC: Modelling of Autonomous Vehicles using Robust, Intelligent Computing*. [Online]. Available: https://www.gla.ac.uk/media/media_480053_en.pdf (visited on 06/21/2018).

[45] D. Anderson and K. Carson, "Integrated variable-fidelity modeling for remote sensing system design," in *Technologies for Optical Countermeasures VI*, 2009, ISBN: 9780819477897. DOI: 10.1117/12.832633.

[46] The Qt Company, *Qt Creator 4.10.2 Based on Qt 5.13.2*. [Online]. Available: https://www.qt.io.

[47] M. A. Richards, J. A. Scheer, and W. A. Holm, "Swerling 1 Target Model," in *Principles of Modern Radar: Basic Principles*, SciTech Publishing, Inc., 2010, ch. 3.3.8.2, pp. 107–108, ISBN: 978-1-891121-52-4.

[48] P. Williams, H. Cramp, and K. Curtis, "Experimental study of the radar cross section of maritime targets," *IEE Journal of Electronic Circuits and Systems*, vol. 2, no. 4, pp. 121–136, 1978. DOI: 10.1049/ij-ecs.1978.0026.

[49] Leonardo MW Ltd, *Leonardo Surveillance Radar Capability Brochure*, 2018. [Online]. Available: https://www.leonardocompany.com/documents/20142/3152033/Surveillance+Radar+Capability+Brochure+%7B%5C%%7D28mm08771%7B%5C%%7D29_LQ.pdf.

[50] M. A. Richards, J. A. Scheer, and W. A. Holm, *Principles of Modern Radar: Basic Principles*, 1st. Scitech Publishing, 2010, ISBN: 978-1891121524.

[51] D. K. Barton, *Radar System Analysis and Modeling*. Artech House Publishers, 2004, ISBN: 978-1580536813.

[52] B. R. Mahafza, *Radar Systems Analysis and Design Using MATLAB*, 3rd. Chapman and Hall/CRC, 2013, ISBN: 978-1439884959.

[53] M. I. Skolnik, *Introduction to Radar Systems*, 3rd ed. McGraw-Hill, 2001, ISBN: 9780071181891.

[54] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960. DOI: 10.1115/1.3662552.

[55] B. Ristic, S. Arulampalam, and N. Gordon, *Beyond The Kalman Filter: Particle Filters for Tracking Applications*. Artech House, 2004, ISBN: 9781580536318.

[56] R. Schubert, E. Richter, and G. Wanielik, "Comparison and Evaluation of Advanced Motion Models for Vehicle Tracking," in *Proceedings of the 11th International Conference on Information Fusion, FUSION 2008*, 2008, ISBN: 9783000248832. DOI: 10.1109/ICIF.2008.4632283.

[57] Z. Goraj, A. Frydrychewicz, R. Świtkiewicz, *et al.*, "High altitude long endurance unmanned aerial vehicle of a new generation - a design challenge for a low cost, reliable and high performance aircraft," *Buletin Of The Polish Academy of Sciences, Technical Sciences*, vol. 52, no. 3, pp. 173–194, 2004.

[58] Leonardo MW Ltd, *Private Communication Regarding Radar Surveillance Trajectories*, 2018.

[59] J. Hall and D. Anderson, "Reactive route selection from pre-calculated trajectories - application to micro-UAV path planning," *The Aeronautical Journal*, vol. 115, no. 1172, pp. 635–640, 2011. DOI: 10.1017/S0001924000006321.

[60] X.-S. Yang, "Biology-Derived Algorithms in Engineering Optimization," in *Handbook of Bioinspired Algorithms and Applications*, Olarius and Zomaya, Eds., Chapman and Hall/CRC, 2005, ch. 32. DOI: 10.1201/9781420035063.ch32.

[61] X.-s. Yang, *Nature-Inspired Metaheuristic Algorithms*. Luniver Press, 2008, ISBN: 978-1-905986-10-1.

[62] X.-S. Yang and S. Deb, "Engineering Optimisation by Cuckoo Search," *International Journal of Mathematical Modelling and Numerical Optimisation*, vol. 1, no. 4, pp. 330–343, 2010. DOI: 10.1504/IJMMNO.2010.035430. arXiv: 1005.2908.

[63] J. Arora, *Introduction to Optimum Design*. 2012, ISBN: 9780123813756. DOI: 10.1016/C2009-0-61700-1.

[64] K. Deb, *Optimisation for Engineering Design: Algorithms and Examples*, 2nd ed. New Dehli: PHI Learning Private Limited, 1995, p. 330, ISBN: 978-81-203-4678-9.

[65] C. Blum and A. Roli, "Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison," *ACM Computing Surveys*, vol. 35, no. 3, pp. 268–308, 2003. DOI: 10.1145/937503.937505.

[66] V. Gazi and K. M. Passino, "Stability Analysis of Social Foraging Swarms," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 34, no. 1, pp. 539–557, 2004. DOI: 10.1109/TSMCB.2003.817077.

[67] X. S. Yang, “Harmony Search as a Metaheuristic Algorithm,” in *Studies in Computational Intelligence, vol 191*, Springer, Ed., Berlin, Heidelberg, 2009, ISBN: 9783642001840. DOI: 10.1007/978-3-642-00185-7_1.

[68] D. E. Goldberg and J. H. Holland, *Genetic Algorithms and Machine Learning*, 1988. DOI: 10.1023/A:1022602019183.

[69] J. Kennedy and R. Eberhart, “Particle Swarm Optimization,” in *IEEE International Conference on Neural Networks*, vol. 4, Perth, Australia, 1995, pp. 1942–1948. DOI: 10.1109/ICNN.1995.488968.

[70] D. Delahaye, S. Puechmorel, P. Tsiotras, and E. Feron, “Mathematical Models for Aircraft Trajectory Design: A Survey,” in *Lecture Notes in Electrical Engineering*, vol. 290, 2014, pp. 205–247. DOI: 10.1007/978-4-431-54475-3_12.

[71] K. Bousson and T. A. Gamerio, “A Quintic Spline Approach to 4D Trajectory Generation for Unmanned Aerial Vehicles,” *International Review of Aerospace Engineering (IREASE)*, vol. 8, no. 1, pp. 1–9, 2015. DOI: 10.15866/irease.v8i1.4780.

[72] I. D. Cowling, O. A. Yakimenko, J. F. Whidborne, and A. K. Cooke, “A Prototype of an Autonomous Controller for a Quadrotor UAV,” in *European Control Conference*, Kos, Greece, 2007, pp. 4001–4008. DOI: 10.23919/ECC.2007.7068316.

[73] A. Barrientos, P. Gutiérrez, and J. Colorado, “Advanced UAV Trajectory Generation: Planning and Guidance,” in *Aerial Vehicles*, T. M. Lam, Ed., InTech, 2009, pp. 55–82. DOI: 10.5772/6467.

[74] L. Babel, “Three-dimensional Route Planning for Unmanned Aerial Vehicles in a Risk Environment,” *Journal of Intelligent & Robotic Systems*, vol. 71, no. 2, pp. 255–269, 2013. DOI: 10.1007/s10846-012-9773-7.

[75] B. Salamat and A. M. Tonello, “Stochastic Trajectory Generation Using Particle Swarm Optimization for Quadrotor Unmanned Aerial Vehicles (UAVs),” *Aerospace*, vol. 4, no. 2, 2017. DOI: 10.3390/aerospace4020027.

[76] J. L. Foo, J. S. Knutzon, J. H. Oliver, and E. H. Winer, “Three Dimensional Path Planning of Unmanned Aerial Vehicles using Particle Swarm Optimization,” in *11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Portsmouth, Virginia, 2006, pp. 1–10. DOI: 10.2514/6.2006-6995.

[77] J. Wang, L. Liu, T. Long, and Z. Wang, “Three-Dimensional Constrained UAV Path Planning using Modified Particle Swarm Optimization with Digital Pheromones,” in *3rd International Conference on Engineering Optimization*, Rio de Janeiro, Brazil, 2012.

[78] J. Karimi and S. H. Pourtakdoust, “A real-time algorithm for variable-objective motion planning over terrain and threats,” *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, vol. 229, no. 6, pp. 1043–1056, 2015. DOI: 10.1177/0954410014543807.

[79] E. Besada-Portas, L. De La Torre, J. M. De La Cruz, and B. De Andrés-Toro, “Evolutionary Trajectory Planner for Multiple UAVs in Realistic Scenarios,” *IEEE Transac-*

*tions on Robotics*, vol. 26, no. 4, pp. 619–634, 2010. DOI: 10.1109/TRO.2010.2048610.

[80] M. B. Pellazar, "Vehicle Route Planning with Constraints using Genetic Algorithms," in *National Aerospace and Electronics Conference*, Dayton, Ohio, 1994, pp. 111–118. DOI: 10.1109/NAECON.1994.333010.

[81] J. L. Foo, J. Knutzon, V. Kalivarapu, *et al.*, "Path Planning of Unmanned Aerial Vehicles using B-Splines and Particle Swarm Optimization," *Journal of Aerospace Computing, Information, and Communication*, vol. 6, no. 4, pp. 271–290, 2009. DOI: 10.2514/1.36917.

[82] R. Hassan, B. Cohanim, O. de Weck, and G. Venter, "A Comparison of Particle Swarm Optimization and the Genetic Algorithm," in *46th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, 2005, ISBN: 978-1-62410-065-9. DOI: 10.2514/6.2005-1897.

[83] M. Bagherian and A. Alos, "3D UAV trajectory planning using evolutionary algorithms: A comparison study," *The Aeronautical Journal*, vol. 119, no. 1220, pp. 1271–1285, 2015. DOI: 10.1017/S0001924000011246.

[84] E. Besada-Portas, L. De La Torre, A. Moreno, and J. L. Risco-Martín, "On the performance comparison of multi-objective evolutionary UAV path planners," *Information Sciences*, vol. 238, pp. 111–125, 2013. DOI: 10.1016/j.ins.2013.02.022.

[85] A. A. Heidari, A. Afghan-Toloee, and R. A. Abbaspour, "Path Planning of an Autonomous Mobile Multi-Sensor Platform in a 3D Environment Using Newtonian Imperialist Competitive Optimization Method," *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XL-1/W3, pp. 191–196, 2013. DOI: 10.5194/isprsarchives-XL-1-W3-191-2013.

[86] Y. Fu, M. Ding, C. Zhou, and H. Hu, "Route Planning for Unmanned Aerial Vehicle (UAV) on the Sea Using Hybrid Differential Evolution and Quantum-Behaved Particle Swarm Optimization," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 43, no. 6, pp. 1451–1465, 2013. DOI: 10.1109/TSMC.2013.2248146.

[87] H. Ergezer and K. Leblebicioglu, "3D Path Planning for Multiple UAVs for Maximum Information Collection," *Journal of Intelligent & Robotic Systems*, vol. 73, no. 1, pp. 737–762, 2014. DOI: 10.1007/s10846-013-9895-6.

[88] ——, "Path Planning for UAVs for Maximum Information Collection," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 49, no. 1, pp. 502–520, 2013. DOI: 10.1109/TAES.2013.6404117.

[89] S. Perez-Carabaza, E. Besada-Portas, J. A. Lopez-Orozco, and J. M. de la Cruz, "A Real World Multi-UAV Evolutionary Planner for Minimum Time Target Detection," in *Genetic and Evolutionary Computation Conference*, 2016, pp. 981–988. DOI: 10.1145/2908812.2908876.

[90] ——, “Ant colony optimization for multi-UAV minimum time search in uncertain domains,” *Applied Soft Computing Journal*, vol. 62, pp. 789–806, 2018. DOI: 10.1016/j.asoc.2017.09.009.

[91] J. Li, J. Chen, P. Wang, and C. Li, “Sensor-Oriented Path Planning for Multiregion Surveillance with a Single Lightweight UAV SAR,” *Sensors*, vol. 18, no. 2, p. 548, 2018. DOI: 10.3390/s18020548.

[92] Office of the Under Secretary of Defence, “UAV Annual Report, FY 1997,” Washington DC, Tech. Rep., 1997, pp. 22, 23, 32. [Online]. Available: http://www.dtic.mil/dtic/tr/fulltext/u2/a336710.pdf.

[93] Y. Zeng and R. Zhang, “Energy-Efficient UAV Communication with Trajectory Optimization,” *IEEE Transactions on Wireless Communications*, vol. 16, no. 6, pp. 3747–3760, 2017. DOI: 10.1109/TWC.2017.2688328.

[94] J. B. Russell, “Standard atmospheres,” in *Performance & Stability of Aircraft*, Butterworth-Heinemann, 1996, ch. 1.5, pp. 15–17, ISBN: 0 340 63170 8.

[95] M. E. Eshelby, *Aircraft Performance: Theory and Practice*. Butterworth-Heinemann, 2000, ISBN: 978-0340758977.

[96] M. R. Sierra and C. A. Coello Coello, “Improving PSO-based multi-objective optimization using crowding, mutation and $\varepsilon$-dominance,” in *Evolutionary Multi-Criterion Optimization*, Guanajuato, Mexico, 2005, pp. 505–519. DOI: 10.1007/978-3-540-31880-4_35.

[97] J. Durillo, J. García-Nieto, A. J. Nebro, *et al.*, “Multi-Objective Particle Swarm Optimizers: An Experimental Comparison,” in *Evolutionary Multi-Criterion Optimization*, Nantes, France, 2009, pp. 495–509. DOI: 10.1007/978-3-642-01020-0_39.

[98] M. Richards and D. Ventura, “Choosing a Starting Configuration for Particle Swarm Optimization,” in *IEEE International Joint Conference on Neural Networks*, vol. 3, 2004, pp. 2309–2312. DOI: 10.1109/IJCNN.2004.1380986.

[99] S. Schaal, A. Ijspeert, and A. Billard, “Computational approaches to motor learning by imitation,” *Philosophical Transactions of the Royal Society B: Biological Sciences*, 2003, ISSN: 09628436. DOI: 10.1098/rstb.2002.1258.

[100] A. Billard, S. Calinon, R. Dillmann, and S. Schall, “Robot programming from demonstration,” in *Handbook of Robotics*, New York: Springer, 2008, ch. 59. DOI: 10.1007/978-3-540-30301-5_60.

[101] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, “A survey of robot learning from demonstration,” *Robotics and Autonomous Systems*, vol. 57, no. 5, pp. 469–483, 2009, ISSN: 09218890. DOI: 10.1016/j.robot.2008.10.024.

[102] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, *et al.*, *Dynamical movement primitives: Learning attractor models formotor behaviors*, 2013. DOI: 10.1162/NECO_a_00393.

[103] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne, “Imitation Learning: A Survey of Learning Methods,” *ACM Computing Surveys (CSUR)*, vol. 50, no. 2, 2017. DOI: 10.1145/3054912.

[104] J. A. Bagnell, “An Invitation to Imitation,” Tech. Rep., 2015.

[105] S. Calinon and A. Billard, “Incremental learning of gestures by imitation in a humanoid robot,” in *HRI 2007 - Proceedings of the 2007 ACM/IEEE Conference on Human-Robot Interaction - Robot as Team Member*, 2007, ISBN: 1595936173. DOI: 10.1145/1228716.1228751.

[106] T. Asfour, P. Azad, F. Gyarfas, and R. Dillmann, “Imitation learning of dual-arm manipulation tasks in humanoid robots,” *International Journal of Humanoid Robotics*, 2008, ISSN: 02198436. DOI: 10.1142/S0219843608001431.

[107] C. Finn, T. Yu, T. Zhang, *et al.*, “One-shot Visual Imitation via Meta-Learning,” in *Conference on Robot Learning, CoRL 2017*, 2017, ISBN: 9781510855144. arXiv: 1703.03400.

[108] K. Mülling, J. Kober, O. Kroemer, and J. Peters, “Learning to select and generalize striking movements in robot table tennis,” *International Journal of Robotics Research*, 2013, ISSN: 02783649. DOI: 10.1177/0278364912472380.

[109] K. Muelling, O. Kroemer, C. H. Lampert, and B. Schölkopf, “Movement Templates for Learning of Hitting and Batting,” in *Springer Tracts in Advanced Robotics*, 2014. DOI: 10.1007/978-3-319-03194-1_3.

[110] E. Berger, H. Ben Amor, D. Vogt, and J. Bernhard, “Towards a Simulator for Imitation Learning with Kinesthetic Bootstrapping,” in *Workshop Proceedings of Intl. Conf. on Simulation, Modeling and Programming for Autonomous Robots (SIMPAR)*, 2008, pp. 163–173.

[111] C. Yang, K. Yuan, S. Heng, *et al.*, “Learning Natural Locomotion Behaviors for Humanoid Robots Using Human Bias,” *IEEE Robotics and Automation Letters*, 2020, ISSN: 23773766. DOI: 10.1109/LRA.2020.2972879.

[112] C. Sammut, S. Hurst, D. Kedzier, and D. Michie, “Learning to Fly,” *Proceedings of the Ninth International Workshop on Machine Learning*, pp. 385–393, 1992. DOI: 10.1016/B978-1-55860-247-2.50055-3.

[113] M. Bain and C. Sammut, “A Framework for Behavioural Cloning,” *Machine Intelligence*, vol. 15, 1995.

[114] P. Abbeel, A. Coates, M. Quigley, and A. Y. Ng, “An application of reinforcement learning to aerobatic helicopter flight,” in *Advances in Neural Information Processing Systems*, 2007, ISBN: 9780262195683. DOI: 10.7551/mitpress/7503.003.0006.

[115] D. Gandhi, L. Pinto, and A. Gupta, “Learning to fly by crashing,” in *IEEE International Conference on Intelligent Robots and Systems*, 2017, ISBN: 9781538626825. DOI: 10.1109/IROS.2017.8206247. arXiv: 1704.05588.

[116] D. a. Pomerleau, “Alvinn: An autonomous land vehicle in a neural network,” *Advances in Neural Information Processing Systems 1*, pp. 305–313, 1989.

[117] D. A. Pomerleau, "Efficient Training of Artificial Neural Networks for Autonomous Navigation," *Neural Computation*, 1991, ISSN: 0899-7667. DOI: `10.1162/neco.1991.3.1.88`.

[118] N. Ratliff, D. Bradley, J. A. Bagnell, and J. Chestnutt, "Boosting structured prediction for imitation learning," in *Advances in Neural Information Processing Systems*, 2007, ISBN: 9780262195683. DOI: `10.7551/mitpress/7503.003.0149`.

[119] M. Ollis, W. H. Huang, and M. Happold, "A Bayesian approach to imitation learning for robot navigation," in *IEEE International Conference on Intelligent Robots and Systems*, 2007, ISBN: 1424409128. DOI: `10.1109/IROS.2007.4399220`.

[120] S. Chernova and M. Veloso, "Confidence-based policy learning from demonstration using Gaussian mixture models," in *Proceedings of the International Conference on Autonomous Agents*, 2007, ISBN: 9788190426275. DOI: `10.1145/1329125.1329407`.

[121] D. Silver, J. A. Bagnell, and A. Stentz, "High performance outdoor navigation from overhead data using imitation learning," in *Robotics: Science and Systems*, 2009, ISBN: 9780262513098. DOI: `10.15607/rss.2008.iv.034`.

[122] ——, "Learning from demonstration for autonomous navigation in complex unstructured terrain," in *International Journal of Robotics Research*, 2010. DOI: `10.1177/0278364910369715`.

[123] A. Hussein, E. Elyan, M. M. Gaber, and C. Jayne, "Deep imitation learning for 3D navigation tasks," *Neural Computing and Applications*, 2018, ISSN: 09410643. DOI: `10.1007/s00521-017-3241-z`.

[124] C. Thurau, C. Bauckhage, and G. Sagerer, "Imitation learning at all levels of game-AI," in *Proc. Int. Conf. Computer Games, Artificial Intelligence, Design and Education*, 2004.

[125] S. Ross, G. J. Gordon, and J. A. Bagnell, "A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning," *AISTATS*, vol. 15, pp. 627–635, 2011, ISSN: <null>. arXiv: `1011.0686`. [Online]. Available: `http://arxiv.org/abs/1011.0686`.

[126] X. Guo, S. Singh, H. Lee, *et al.*, "Deep learning for real-time Atari game play using offline Monte-Carlo tree search planning," in *Advances in Neural Information Processing Systems*, 2014.

[127] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The Arcade Learning Environment: An evaluation platform for general agents," *Journal of Artificial Intelligence Research*, 2013, ISSN: 10769757. DOI: `10.1613/jair.3912`. arXiv: `1207.4708`.

[128] P. Abbeel, A. Coates, and A. Y. Ng, "Autonomous Helicopter Aerobatics through Apprenticeship Learning," *The International Journal of Robotics Research*, vol. 29, no. 13, pp. 1608–1639, 2010, ISSN: 0278-3649, 1741-3176. DOI: `10.1177/02783649103`

[129] M. Zucker, N. Ratliff, M. Stolle, *et al.*, "Optimization and learning for rough terrain legged locomotion," *International Journal of Robotics Research*, 2011, ISSN: 02783649. DOI: `10.1177/0278364910392608`.

[130] S. Russel and P. Norvig, *Artificial intelligence—a modern approach 3rd Edition*. 2012, ISBN: 0136042597. DOI: `10.1017/S0269888900007724`. arXiv: `9809069v1 [arXiv:gr-qc]`.

[131] S. Ross and J. A. Bagnell, "Efficient Reductions for Imitation Learning," *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*, vol. 9, pp. 661–668, 2010, ISSN: 15324435.

[132] H. Daumé, J. Langford, and D. Marcu, "Search-based structured prediction," *Machine Learning*, 2009, ISSN: 08856125. DOI: `10.1007/s10994-009-5106-x`.

[133] H. M. Le, A. Kang, Y. Yue, and P. Carr, "Smooth imitation learning for online sequence prediction," in *33rd International Conference on Machine Learning, ICML 2016*, 2016, ISBN: 9781510829008. arXiv: `1606.00968`.

[134] K. Mülling, J. Kober, O. Kroemer, and J. Peters, "Learning to select and generalize striking movements in robot table tennis," *The International Journal of Robotics Research*, vol. 32, no. 3, pp. 263–279, 2013, ISSN: 0278-3649, 1741-3176. DOI: `10.1177/0278364912472380`. [Online]. Available: `http://ijr.sagepub.com/content/32/3/263%7B%5C%%7D5Cnhttp://ijr.sagepub.com/content/32/3/263.full.pdf`.

[135] S. Raza, S. Haider, and M. A. Williams, "Teaching coordinated strategies to soccer robots via imitation," in *2012 IEEE International Conference on Robotics and Biomimetics, ROBIO 2012 - Conference Digest*, 2012, ISBN: 9781467321273. DOI: `10.1109/ROBIO.2012.6491170`.

[136] A. Ng and S. Russell, "Algorithms for inverse reinforcement learning," *Proceedings of the Seventeenth International Conference on Machine Learning*, vol. 0, pp. 663–670, 2000, ISSN: 00029645. DOI: `10.2460/ajvr.67.2.323`. arXiv: `arXiv:1011.1669v3`.

[137] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," *Proceedings of the 21st International Conference on Machine Learning (ICML)*, pp. 1–8, 2004. DOI: `10.1145/1015330.1015430`. arXiv: `1206.5264`.

[138] S. Sharifzadeh, I. Chiotellis, R. Triebel, and D. Cremers, "Learning to Drive using Inverse Reinforcement Learning and Deep Q-Networks," in *NIPS 2016 Workshop on Deep Learning for Action and Interaction*, 2016.

[139] G. Lee, M. Luo, F. Zambetta, and X. Li, "Learning a Super Mario Controller from Examples of Human Play," in *Proceedings of the 2014 IEEE Congress on Evolutionary Computation, CEC 2014*, 2014, pp. 1–8, ISBN: 9781479914883. DOI: `10.1109/CEC.2014.6900246`.

[140] C. Finn, S. Levine, and P. Abbeel, "Guided Cost Learning: Deep Inverse Optimal Control via Policy Optimization," *ICML 2016*, vol. 48, no. 2000, 2016. arXiv: `1603.00448`. [Online]. Available: `http://arxiv.org/abs/1603.00448`.

[141] Leonardo MW Ltd, "Private Communication Regarding Typical Operator Operation During a Maritime Surveillance Radar Mission," 2017.

[142] S. Sivashanmugam and C. Tsatsoulis, "A Bayesian Network for Autonomous Sensor Control during Polar Ice Sheet Measurements," in *IEEE International Geoscience and Remote Sensing Symposium*, 2004, pp. 101–104.

[143] M. Krüger, J. Ziegler, and K. Heller, "A Generic Bayesian Network for Identification and Assessment of Objects in Maritime Surveillance," in *15th International Conference on Information Fusion*, 2012, ISBN: 978-0-9824438-5-9.

[144] R. N. Carvalho, R. Haberlin, P. C. G. Costa, *et al.*, "Modeling a Probabilistic Ontology for Maritime Domain Awareness," in *14th International Conference on Information Fusion*, 2011, ISBN: 978-0-9824438-2-8.

[145] P. C. Costa, K. Laskey, K. C. Chang, *et al.*, "High-level information Fusion with Bayesian Semantics," in *CEUR Workshop Proceedings*, 2012.

[146] Y. Fischer and J. Beyerer, "Modeling of Expert Knowledge for Maritime Situation Assessment," *International Journal on Advances in Systems and Measurements*, vol. 6, no. 3&4, pp. 245–259, 2013.

[147] Y. Fischer, A. Reiswich, and J. Beyerer, "Modeling and Recognizing Situations of Interest in Surveillance Applications," in *IEEE International Inter-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support (CogSIMA)*, 2014, pp. 209–215.

[148] J. J. Dabrowski, J. P. de Villiers, and C. Beyers, "Context-based behaviour modelling and classification of marine vessels in an abalone poaching situation," *Engineering Applications of Artificial Intelligence*, vol. 64, pp. 95–111, 2017. DOI: `10.1016/j.engappai.2017.06.005`.

[149] F. Fooladvandi, C. Brax, P. Gustavsson, and M. Fredin, "Signature-based activity detection based on Bayesian networks acquired from expert knowledge," *12th International Conference on Information Fusion*, 2009.

[150] R. Parra and L. Garrido, "Bayesian Networks for Micromanagement Decision Imitation in the RTS Game Starcraft," in *Advances in Computational Intelligence*, 2012, pp. 433–443, ISBN: 978-3-642-37797-6. DOI: `10.1007/978-3-642-37798-3_38`.

[151] R. E. Neapolitan, *Learning Bayesian Networks*. Prentice Hall, 2003, ISBN: 978-0130125347.

[152] A. Darwiche, *Modeling and reasoning with Bayesian networks*. Cambridge University Press, 2009, ISBN: 978-0-521-88438-9.

[153] J. Pearl, *Probabilistic Reasoning in Intelligent Systems*, 2nd ed. San Mateo, California, 1988, ISBN: 0-934613-73-7.

[154] F. V. Jensen and T. D. Nielsen, *Bayesian Networks and Decision Graphs*, 2nd ed. Springer, 2007, ISBN: 978-0-387-68281-5.

[155] M. I. Jordan, Ed., *Learning In Graphical Models*. The MIT Press, 1999, ISBN: 9780262600323.

[156] H. A. Loeliger, "An introduction to Factor Graphs," *IEEE Signal Processing Magazine*, 2004, ISSN: 10535888. DOI: `10.1109/MSP.2004.1267047`.

[157] W. Xue, P. Kolaric, J. Fan, *et al.*, "Inverse Reinforcement Learning in Tracking Control Based on Inverse Optimal Control," *IEEE Transactions on Cybernetics*, 2021, ISSN: 21682275. DOI: `10.1109/TCYB.2021.3062856`.

[158] V. Krishnamurthy, D. Angley, R. Evans, and B. Moran, "Identifying Cognitive Radars - Inverse Reinforcement Learning Using Revealed Preferences," *IEEE Transactions on Signal Processing*, 2020, ISSN: 19410476. DOI: `10.1109/TSP.2020.3013516`.

[159] R. Bellman, "The Theory of Dynamic Programming," *Bulletin of the American Mathematical Society*, 1954, ISSN: 02730979. DOI: `10.1090/S0002-9904-1954-09848-8`.

[160] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley-Interscience, 1994, ISBN: 978-0471727828.

[161] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, Second Edi. MIT Press, 2018, ISBN: 978-0262039246.

[162] M. Lapan, *Deep Reinforcement Learning Hand-On*, 1st. Packt Publishing, 2018, ISBN: 978-1788834247.

[163] S. Russell, "Learning agents for uncertain environments (extended abstract)," *Proceedings of the 11th Annual Conference on Computational Learning Theory (COLT)*, pp. 101–103, 1998. DOI: `10.1145/279943.279964`.

[164] S. Zhifei and E. M. Joo, "A survey of inverse reinforcement learning techniques," *International Journal of Intelligent Computing and Cybernetics*, 2012, ISSN: 1756378X. DOI: `10.1108/17563781211255862`.

[165] S. Arora and P. Doshi, *A survey of inverse reinforcement learning: Challenges, methods and progress*, 2021. DOI: `10.1016/j.artint.2021.103500`. arXiv: `1806.06877`.

[166] J. Choi and K.-E. Kim, "Inverse Reinforcement Learning in Partially Observable Environments," *Journal of Machine Learning Research*, vol. 12, no. August, pp. 691–730, 2011, ISSN: 1532-4435.

[167] W. B. Rouse, *Design for Success: A Human Centered Approach for Designing Successful Products and Systems*. New York: Wiley, 1991, ISBN: 9780471524830.

[168] T. Chugh, Y. Jin, K. Miettinen, *et al.*, "A Surrogate-Assisted Reference Vector Guided Evolutionary Algorithm for Computationally Expensive Many-Objective Optimization," *IEEE Transactions on Evolutionary Computation*, 2018, ISSN: 1089778X. DOI: `10.1109/TEVC.2016.2622301`.

[169] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Advances in Neural Information Processing Systems*, 2016. arXiv: `1606.03476`.

[170] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, *et al.*, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, 2014.

[171] Y. Wang, Y. Sun, J. Y. Li, and S. T. Xia, "Air defense threat assessment based on dynamic Bayesian network," in *2012 International Conference on Systems and In-*

*formatics, ICSAI 2012*, 2012, ISBN: 9781467301992. DOI: `10.1109/ICSAI.2012.6223112`.

[172] H. Sun, X. Xie, T. Sun, and L. Zhang, "Threat assessment method of warships formation air defense based on DBN under the condition of small sample data missing," *Xi Tong Gong Cheng Yu Dian Zi Ji Shu/Systems Engineering and Electronics*, 2019, ISSN: 1001506X. DOI: `10.3969/j.issn.1001-506X.2019.06.18`.

[173] T. Smith and R. Simmons, "Heuristic Search Value Iteration for POMDPs," in *UAI '04: Proceedings of the 20th conference on Uncertainty in artificial intelligence*, Banff, Canada, 2004, pp. 520–527.

[174] M. H. Draper, H. A. Ruff, D. W. Repperger, and L. G. Lu, "Multi-Sensory Interface Concepts Supporting Turbulence Detection by UAV Controllers," in *Proceedings of the First Human Performance, Situation Awareness & Automation Conference*, 2000, pp. 107–112.

[175] M. Beliaev, A. Shih, S. Ermon, *et al.*, "Imitation Learning by Estimating Expertise of Demonstrators," *arXiv preprint*, 2022. DOI: `https://doi.org/10.48550/arXiv.2202.01288`.

[176] H. M. Le, Y. Yue, P. Carr, and P. Lucey, "Coordinated multi-agent imitation learning," in *34th International Conference on Machine Learning, ICML 2017*, 2017, ISBN: 9781510855144. arXiv: `1703.03121`.